

NBC

Version 1.2.1 r5

Generated by Doxygen 1.8.2

Mon Feb 18 2013 01:01:44

## Contents

<b>1 NBC Programmer's Guide</b>	<b>1</b>
<b>2 Introduction</b>	<b>1</b>
<b>3 The NBC Language</b>	<b>2</b>
3.1 Lexical Rules . . . . .	2
3.1.1 Comments . . . . .	2
3.1.2 Whitespace . . . . .	3
3.1.3 Numerical Constants . . . . .	3
3.1.4 String Constants . . . . .	3
3.1.5 Identifiers and Keywords . . . . .	4
3.2 Program Structure . . . . .	5
3.2.1 Threads . . . . .	5
3.2.2 Subroutines . . . . .	6
3.2.3 Macro Functions . . . . .	7
3.2.4 Data Segments . . . . .	8
3.3 The Preprocessor . . . . .	14
3.3.1 #include . . . . .	14
3.3.2 #define . . . . .	15
3.3.3 ## (Concatenation) . . . . .	15
3.3.4 Conditional Compilation . . . . .	16
3.3.5 #import . . . . .	16
3.3.6 #download . . . . .	16
3.4 Compiler Tokens . . . . .	16
3.5 Expression Evaluator . . . . .	17
3.6 Statements . . . . .	18
3.6.1 Assignment Statements . . . . .	18
3.6.2 Math Statements . . . . .	20
3.6.3 Logic Statements . . . . .	27
3.6.4 Bit Manipulation Statements . . . . .	28
3.6.5 Comparison Statements . . . . .	30
3.6.6 Control Flow Statements . . . . .	31
3.6.7 syscall . . . . .	33
3.6.8 Timing Statements . . . . .	48
3.6.9 Array Statements . . . . .	49
3.6.10 String Statements . . . . .	50

3.6.11	Scheduling Statements . . . . .	53
3.6.12	Input Statements . . . . .	57
3.6.13	Output Statements . . . . .	57
3.6.14	Compile-time Statements . . . . .	58
<b>4</b>	<b>TXGPacket</b>	<b>60</b>
<b>5</b>	<b>Module Documentation</b>	<b>60</b>
5.1	Comparison Constants . . . . .	60
5.1.1	Detailed Description . . . . .	61
5.1.2	Macro Definition Documentation . . . . .	61
5.2	NXT Firmware Modules . . . . .	62
5.2.1	Detailed Description . . . . .	62
5.3	Input module . . . . .	63
5.3.1	Detailed Description . . . . .	63
5.4	Input module constants . . . . .	64
5.4.1	Detailed Description . . . . .	64
5.4.2	Macro Definition Documentation . . . . .	64
5.5	Sensor types and modes . . . . .	66
5.5.1	Detailed Description . . . . .	66
5.6	Output module . . . . .	67
5.6.1	Detailed Description . . . . .	67
5.7	Output module constants . . . . .	68
5.7.1	Detailed Description . . . . .	68
5.8	Command module . . . . .	69
5.8.1	Detailed Description . . . . .	69
5.9	Command module constants . . . . .	70
5.9.1	Detailed Description . . . . .	70
5.9.2	Macro Definition Documentation . . . . .	70
5.10	Comm module . . . . .	72
5.10.1	Detailed Description . . . . .	72
5.11	Button module . . . . .	73
5.11.1	Detailed Description . . . . .	73
5.12	IOCtrl module . . . . .	74
5.12.1	Detailed Description . . . . .	74
5.13	Loader module . . . . .	75
5.13.1	Detailed Description . . . . .	75
5.14	Sound module . . . . .	76

5.14.1 Detailed Description . . . . .	76
5.15 Ui module . . . . .	77
5.15.1 Detailed Description . . . . .	77
5.16 Low Speed module . . . . .	78
5.16.1 Detailed Description . . . . .	78
5.17 Display module . . . . .	79
5.17.1 Detailed Description . . . . .	79
5.18 HiTechnic API Functions . . . . .	80
5.18.1 Detailed Description . . . . .	86
5.18.2 Macro Definition Documentation . . . . .	87
5.19 SuperPro analog output mode constants . . . . .	117
5.19.1 Detailed Description . . . . .	117
5.19.2 Macro Definition Documentation . . . . .	117
5.20 SuperPro LED control constants . . . . .	119
5.20.1 Detailed Description . . . . .	119
5.20.2 Macro Definition Documentation . . . . .	119
5.21 SuperPro digital pin constants . . . . .	120
5.21.1 Detailed Description . . . . .	120
5.21.2 Macro Definition Documentation . . . . .	120
5.22 SuperPro Strobe control constants . . . . .	121
5.22.1 Detailed Description . . . . .	121
5.22.2 Macro Definition Documentation . . . . .	121
5.23 MindSensors API Functions . . . . .	122
5.23.1 Detailed Description . . . . .	132
5.23.2 Macro Definition Documentation . . . . .	133
5.24 Codatex API Functions . . . . .	181
5.24.1 Detailed Description . . . . .	181
5.24.2 Macro Definition Documentation . . . . .	181
5.25 Dexter Industries API Functions . . . . .	184
5.25.1 Detailed Description . . . . .	185
5.25.2 Macro Definition Documentation . . . . .	185
5.26 Microinfinity API Functions . . . . .	192
5.26.1 Detailed Description . . . . .	192
5.26.2 Macro Definition Documentation . . . . .	192
5.27 RIC Macro Wrappers . . . . .	194
5.27.1 Detailed Description . . . . .	195
5.27.2 Macro Definition Documentation . . . . .	195

5.28 NXT firmware module names . . . . .	199
5.28.1 Detailed Description . . . . .	199
5.28.2 Macro Definition Documentation . . . . .	199
5.29 NXT firmware module IDs . . . . .	201
5.29.1 Detailed Description . . . . .	201
5.29.2 Macro Definition Documentation . . . . .	201
5.30 Miscellaneous NBC/NXC constants . . . . .	203
5.30.1 Detailed Description . . . . .	203
5.30.2 Macro Definition Documentation . . . . .	203
5.31 Third-party NXT devices . . . . .	204
5.31.1 Detailed Description . . . . .	204
5.32 Standard-C API functions . . . . .	205
5.32.1 Detailed Description . . . . .	205
5.33 A simple 3D graphics library . . . . .	206
5.33.1 Detailed Description . . . . .	207
5.33.2 Macro Definition Documentation . . . . .	207
5.34 Output module functions . . . . .	212
5.34.1 Detailed Description . . . . .	214
5.34.2 Macro Definition Documentation . . . . .	214
5.35 Input module functions . . . . .	226
5.35.1 Detailed Description . . . . .	227
5.35.2 Macro Definition Documentation . . . . .	228
5.36 LowSpeed module functions . . . . .	238
5.36.1 Detailed Description . . . . .	239
5.36.2 Macro Definition Documentation . . . . .	239
5.37 Low level LowSpeed module functions . . . . .	246
5.37.1 Detailed Description . . . . .	246
5.37.2 Macro Definition Documentation . . . . .	246
5.38 Display module functions . . . . .	250
5.38.1 Detailed Description . . . . .	252
5.38.2 Macro Definition Documentation . . . . .	252
5.39 Sound module functions . . . . .	265
5.39.1 Detailed Description . . . . .	265
5.39.2 Macro Definition Documentation . . . . .	266
5.40 Command module functions . . . . .	271
5.40.1 Detailed Description . . . . .	273
5.40.2 Macro Definition Documentation . . . . .	273

5.41	Button module functions . . . . .	285
5.41.1	Detailed Description . . . . .	285
5.41.2	Macro Definition Documentation . . . . .	285
5.42	Ui module functions . . . . .	289
5.42.1	Detailed Description . . . . .	290
5.42.2	Macro Definition Documentation . . . . .	290
5.43	Comm module functions . . . . .	296
5.43.1	Detailed Description . . . . .	300
5.43.2	Macro Definition Documentation . . . . .	301
5.44	Direct Command functions . . . . .	327
5.44.1	Detailed Description . . . . .	328
5.44.2	Macro Definition Documentation . . . . .	328
5.45	System Command functions . . . . .	337
5.45.1	Detailed Description . . . . .	338
5.45.2	Macro Definition Documentation . . . . .	338
5.46	IOCtrl module functions . . . . .	347
5.46.1	Detailed Description . . . . .	347
5.46.2	Macro Definition Documentation . . . . .	347
5.47	Loader module functions . . . . .	348
5.47.1	Detailed Description . . . . .	349
5.47.2	Macro Definition Documentation . . . . .	349
5.48	cstdlib API . . . . .	357
5.48.1	Detailed Description . . . . .	357
5.48.2	Macro Definition Documentation . . . . .	357
5.49	cmath API . . . . .	358
5.49.1	Detailed Description . . . . .	358
5.49.2	Macro Definition Documentation . . . . .	358
5.50	Property constants . . . . .	359
5.50.1	Detailed Description . . . . .	359
5.50.2	Macro Definition Documentation . . . . .	359
5.51	Variable type constants . . . . .	360
5.51.1	Detailed Description . . . . .	360
5.51.2	Macro Definition Documentation . . . . .	360
5.52	Array operation constants . . . . .	363
5.52.1	Detailed Description . . . . .	363
5.52.2	Macro Definition Documentation . . . . .	363
5.53	System Call function constants . . . . .	365

5.53.1 Detailed Description . . . . .	366
5.53.2 Macro Definition Documentation . . . . .	366
5.54 Line number constants . . . . .	372
5.54.1 Detailed Description . . . . .	372
5.54.2 Macro Definition Documentation . . . . .	372
5.55 Time constants . . . . .	374
5.55.1 Detailed Description . . . . .	375
5.55.2 Macro Definition Documentation . . . . .	375
5.56 Mailbox constants . . . . .	379
5.56.1 Detailed Description . . . . .	379
5.56.2 Macro Definition Documentation . . . . .	379
5.57 VM state constants . . . . .	381
5.57.1 Detailed Description . . . . .	381
5.57.2 Macro Definition Documentation . . . . .	381
5.58 Fatal errors . . . . .	382
5.58.1 Detailed Description . . . . .	382
5.58.2 Macro Definition Documentation . . . . .	382
5.59 General errors . . . . .	384
5.59.1 Detailed Description . . . . .	384
5.59.2 Macro Definition Documentation . . . . .	384
5.60 Communications specific errors . . . . .	385
5.60.1 Detailed Description . . . . .	385
5.60.2 Macro Definition Documentation . . . . .	385
5.61 Remote control (direct commands) errors . . . . .	386
5.61.1 Detailed Description . . . . .	386
5.61.2 Macro Definition Documentation . . . . .	386
5.62 Program status constants . . . . .	387
5.62.1 Detailed Description . . . . .	387
5.62.2 Macro Definition Documentation . . . . .	387
5.63 Command module IOMAP offsets . . . . .	388
5.63.1 Detailed Description . . . . .	388
5.63.2 Macro Definition Documentation . . . . .	388
5.64 IOCtrl module constants . . . . .	390
5.64.1 Detailed Description . . . . .	390
5.65 PowerOn constants . . . . .	391
5.65.1 Detailed Description . . . . .	391
5.65.2 Macro Definition Documentation . . . . .	391

5.66	IOCtrl module IOMAP offsets	392
5.66.1	Detailed Description	392
5.66.2	Macro Definition Documentation	392
5.67	Loader module constants	393
5.67.1	Detailed Description	393
5.67.2	Macro Definition Documentation	393
5.68	Loader module IOMAP offsets	394
5.68.1	Detailed Description	394
5.68.2	Macro Definition Documentation	394
5.69	Loader module error codes	395
5.69.1	Detailed Description	395
5.69.2	Macro Definition Documentation	395
5.70	Loader module function constants	399
5.70.1	Detailed Description	399
5.70.2	Macro Definition Documentation	399
5.71	Sound module constants	403
5.71.1	Detailed Description	403
5.72	SoundFlags constants	404
5.72.1	Detailed Description	404
5.72.2	Macro Definition Documentation	404
5.73	SoundState constants	405
5.73.1	Detailed Description	405
5.73.2	Macro Definition Documentation	405
5.74	SoundMode constants	406
5.74.1	Detailed Description	406
5.74.2	Macro Definition Documentation	406
5.75	Sound module IOMAP offsets	407
5.75.1	Detailed Description	407
5.75.2	Macro Definition Documentation	407
5.76	Sound module miscellaneous constants	408
5.76.1	Detailed Description	408
5.76.2	Macro Definition Documentation	408
5.77	Tone constants	409
5.77.1	Detailed Description	410
5.77.2	Macro Definition Documentation	410
5.78	Button module constants	415
5.78.1	Detailed Description	415

5.79 Button name constants . . . . .	416
5.79.1 Detailed Description . . . . .	416
5.79.2 Macro Definition Documentation . . . . .	416
5.80 ButtonState constants . . . . .	418
5.80.1 Detailed Description . . . . .	418
5.80.2 Macro Definition Documentation . . . . .	418
5.81 Button module IOMAP offsets . . . . .	419
5.81.1 Detailed Description . . . . .	419
5.81.2 Macro Definition Documentation . . . . .	419
5.82 Ui module constants . . . . .	420
5.82.1 Detailed Description . . . . .	420
5.83 CommandFlags constants . . . . .	421
5.83.1 Detailed Description . . . . .	421
5.83.2 Macro Definition Documentation . . . . .	421
5.84 UIState constants . . . . .	423
5.84.1 Detailed Description . . . . .	423
5.84.2 Macro Definition Documentation . . . . .	423
5.85 UIButton constants . . . . .	425
5.85.1 Detailed Description . . . . .	425
5.85.2 Macro Definition Documentation . . . . .	425
5.86 BluetoothState constants . . . . .	426
5.86.1 Detailed Description . . . . .	426
5.86.2 Macro Definition Documentation . . . . .	426
5.87 VM run state constants . . . . .	427
5.87.1 Detailed Description . . . . .	427
5.87.2 Macro Definition Documentation . . . . .	427
5.88 Ui module IOMAP offsets . . . . .	428
5.88.1 Detailed Description . . . . .	428
5.88.2 Macro Definition Documentation . . . . .	428
5.89 NBC Input port constants . . . . .	430
5.89.1 Detailed Description . . . . .	430
5.89.2 Macro Definition Documentation . . . . .	430
5.90 NBC sensor type constants . . . . .	431
5.90.1 Detailed Description . . . . .	431
5.90.2 Macro Definition Documentation . . . . .	431
5.91 NBC sensor mode constants . . . . .	434
5.91.1 Detailed Description . . . . .	434

5.91.2 Macro Definition Documentation . . . . .	434
5.92 Input field constants . . . . .	436
5.92.1 Detailed Description . . . . .	436
5.92.2 Macro Definition Documentation . . . . .	436
5.93 Input port digital pin constants . . . . .	437
5.93.1 Detailed Description . . . . .	437
5.93.2 Macro Definition Documentation . . . . .	437
5.94 Color sensor array indices . . . . .	438
5.94.1 Detailed Description . . . . .	438
5.94.2 Macro Definition Documentation . . . . .	438
5.95 Color values . . . . .	439
5.95.1 Detailed Description . . . . .	439
5.95.2 Macro Definition Documentation . . . . .	439
5.96 Color calibration state constants . . . . .	440
5.96.1 Detailed Description . . . . .	440
5.96.2 Macro Definition Documentation . . . . .	440
5.97 Color calibration constants . . . . .	441
5.97.1 Detailed Description . . . . .	441
5.97.2 Macro Definition Documentation . . . . .	441
5.98 Input module IOMAP offsets . . . . .	442
5.98.1 Detailed Description . . . . .	442
5.98.2 Macro Definition Documentation . . . . .	442
5.99 Constants to use with the Input module's Pin function . . . . .	445
5.99.1 Detailed Description . . . . .	445
5.99.2 Macro Definition Documentation . . . . .	445
5.100 Output port constants . . . . .	446
5.100.1 Detailed Description . . . . .	446
5.100.2 Macro Definition Documentation . . . . .	446
5.101 PID constants . . . . .	447
5.101.1 Detailed Description . . . . .	447
5.101.2 Macro Definition Documentation . . . . .	447
5.102 Output port update flag constants . . . . .	449
5.102.1 Detailed Description . . . . .	449
5.102.2 Macro Definition Documentation . . . . .	449
5.103 Tachometer counter reset flags . . . . .	451
5.103.1 Detailed Description . . . . .	451
5.103.2 Macro Definition Documentation . . . . .	451

5.104Output port mode constants . . . . .	452
5.104.1 Detailed Description . . . . .	452
5.104.2 Macro Definition Documentation . . . . .	452
5.105Output port option constants . . . . .	453
5.105.1 Detailed Description . . . . .	453
5.105.2 Macro Definition Documentation . . . . .	453
5.106Output regulation option constants . . . . .	454
5.106.1 Detailed Description . . . . .	454
5.106.2 Macro Definition Documentation . . . . .	454
5.107Output port run state constants . . . . .	455
5.107.1 Detailed Description . . . . .	455
5.107.2 Macro Definition Documentation . . . . .	455
5.108Output port regulation mode constants . . . . .	456
5.108.1 Detailed Description . . . . .	456
5.108.2 Macro Definition Documentation . . . . .	456
5.109Output field constants . . . . .	457
5.109.1 Detailed Description . . . . .	457
5.109.2 Macro Definition Documentation . . . . .	458
5.110Output module IOMAP offsets . . . . .	462
5.110.1 Detailed Description . . . . .	462
5.110.2 Macro Definition Documentation . . . . .	462
5.111LowSpeed module constants . . . . .	465
5.111.1 Detailed Description . . . . .	465
5.112LSSState constants . . . . .	466
5.112.1 Detailed Description . . . . .	466
5.112.2 Macro Definition Documentation . . . . .	466
5.113LSChannelState constants . . . . .	467
5.113.1 Detailed Description . . . . .	467
5.113.2 Macro Definition Documentation . . . . .	467
5.114LSMode constants . . . . .	468
5.114.1 Detailed Description . . . . .	468
5.114.2 Macro Definition Documentation . . . . .	468
5.115LSErrorType constants . . . . .	469
5.115.1 Detailed Description . . . . .	469
5.115.2 Macro Definition Documentation . . . . .	469
5.116Low speed module IOMAP offsets . . . . .	470
5.116.1 Detailed Description . . . . .	470

5.116.2 Macro Definition Documentation . . . . .	470
5.117LSNoRestartOnRead constants . . . . .	472
5.117.1 Detailed Description . . . . .	472
5.117.2 Macro Definition Documentation . . . . .	472
5.118Standard I2C constants . . . . .	474
5.118.1 Detailed Description . . . . .	474
5.118.2 Macro Definition Documentation . . . . .	474
5.119LEGO I2C address constants . . . . .	475
5.119.1 Detailed Description . . . . .	475
5.119.2 Macro Definition Documentation . . . . .	475
5.120Ultrasonic sensor constants . . . . .	476
5.120.1 Detailed Description . . . . .	476
5.120.2 Macro Definition Documentation . . . . .	476
5.121LEGO temperature sensor constants . . . . .	478
5.121.1 Detailed Description . . . . .	478
5.121.2 Macro Definition Documentation . . . . .	478
5.122E-Meter sensor constants . . . . .	480
5.122.1 Detailed Description . . . . .	480
5.122.2 Macro Definition Documentation . . . . .	480
5.123I2C option constants . . . . .	481
5.123.1 Detailed Description . . . . .	481
5.123.2 Macro Definition Documentation . . . . .	481
5.124Display module constants . . . . .	482
5.124.1 Detailed Description . . . . .	483
5.124.2 Macro Definition Documentation . . . . .	483
5.125DisplayExecuteFunction constants . . . . .	487
5.125.1 Detailed Description . . . . .	487
5.125.2 Macro Definition Documentation . . . . .	487
5.126Drawing option constants . . . . .	489
5.126.1 Detailed Description . . . . .	489
5.126.2 Macro Definition Documentation . . . . .	490
5.127Font drawing option constants . . . . .	492
5.127.1 Detailed Description . . . . .	492
5.127.2 Macro Definition Documentation . . . . .	492
5.128Display flags . . . . .	494
5.128.1 Detailed Description . . . . .	494
5.128.2 Macro Definition Documentation . . . . .	494

5.129Display contrast constants . . . . .	495
5.129.1 Detailed Description . . . . .	495
5.129.2 Macro Definition Documentation . . . . .	495
5.130Text line constants . . . . .	496
5.130.1 Detailed Description . . . . .	496
5.130.2 Macro Definition Documentation . . . . .	496
5.131Display module IOMAP offsets . . . . .	498
5.131.1 Detailed Description . . . . .	498
5.131.2 Macro Definition Documentation . . . . .	498
5.132Comm module constants . . . . .	501
5.132.1 Detailed Description . . . . .	501
5.133Miscellaneous Comm module constants . . . . .	502
5.133.1 Detailed Description . . . . .	502
5.133.2 Macro Definition Documentation . . . . .	502
5.134Joystick message constants . . . . .	504
5.134.1 Detailed Description . . . . .	504
5.134.2 Macro Definition Documentation . . . . .	505
5.135Bluetooth State constants . . . . .	508
5.135.1 Detailed Description . . . . .	508
5.135.2 Macro Definition Documentation . . . . .	508
5.136Data mode constants . . . . .	509
5.136.1 Detailed Description . . . . .	509
5.136.2 Macro Definition Documentation . . . . .	509
5.137Bluetooth state status constants . . . . .	510
5.137.1 Detailed Description . . . . .	510
5.137.2 Macro Definition Documentation . . . . .	510
5.138Remote connection constants . . . . .	511
5.138.1 Detailed Description . . . . .	511
5.138.2 Macro Definition Documentation . . . . .	511
5.139Bluetooth hardware status constants . . . . .	513
5.139.1 Detailed Description . . . . .	513
5.139.2 Macro Definition Documentation . . . . .	513
5.140Hi-speed port constants . . . . .	514
5.140.1 Detailed Description . . . . .	514
5.141Hi-speed port flags constants . . . . .	515
5.141.1 Detailed Description . . . . .	515
5.141.2 Macro Definition Documentation . . . . .	515

5.142Hi-speed port state constants . . . . .	516
5.142.1 Detailed Description . . . . .	516
5.142.2 Macro Definition Documentation . . . . .	516
5.143Hi-speed port SysCommHSControl constants . . . . .	517
5.143.1 Detailed Description . . . . .	517
5.143.2 Macro Definition Documentation . . . . .	517
5.144Hi-speed port baud rate constants . . . . .	518
5.144.1 Detailed Description . . . . .	518
5.144.2 Macro Definition Documentation . . . . .	518
5.145Hi-speed port UART mode constants . . . . .	520
5.145.1 Detailed Description . . . . .	520
5.145.2 Macro Definition Documentation . . . . .	520
5.146Hi-speed port data bits constants . . . . .	521
5.146.1 Detailed Description . . . . .	521
5.146.2 Macro Definition Documentation . . . . .	521
5.147Hi-speed port stop bits constants . . . . .	522
5.147.1 Detailed Description . . . . .	522
5.147.2 Macro Definition Documentation . . . . .	522
5.148Hi-speed port parity constants . . . . .	523
5.148.1 Detailed Description . . . . .	523
5.148.2 Macro Definition Documentation . . . . .	523
5.149Hi-speed port combined UART constants . . . . .	524
5.149.1 Detailed Description . . . . .	524
5.149.2 Macro Definition Documentation . . . . .	524
5.150Hi-speed port address constants . . . . .	525
5.150.1 Detailed Description . . . . .	525
5.150.2 Macro Definition Documentation . . . . .	525
5.151Device status constants . . . . .	527
5.151.1 Detailed Description . . . . .	527
5.151.2 Macro Definition Documentation . . . . .	527
5.152Comm module interface function constants . . . . .	528
5.152.1 Detailed Description . . . . .	528
5.152.2 Macro Definition Documentation . . . . .	528
5.153Comm module status code constants . . . . .	531
5.153.1 Detailed Description . . . . .	531
5.153.2 Macro Definition Documentation . . . . .	531
5.154Comm module IOMAP offsets . . . . .	532

5.154.1 Detailed Description . . . . .	533
5.154.2 Macro Definition Documentation . . . . .	533
5.155RCX constants . . . . .	537
5.155.1 Detailed Description . . . . .	537
5.156RCX output constants . . . . .	538
5.156.1 Detailed Description . . . . .	538
5.156.2 Macro Definition Documentation . . . . .	538
5.157RCX output mode constants . . . . .	539
5.157.1 Detailed Description . . . . .	539
5.157.2 Macro Definition Documentation . . . . .	539
5.158RCX output direction constants . . . . .	540
5.158.1 Detailed Description . . . . .	540
5.158.2 Macro Definition Documentation . . . . .	540
5.159RCX output power constants . . . . .	541
5.159.1 Detailed Description . . . . .	541
5.159.2 Macro Definition Documentation . . . . .	541
5.160RCX IR remote constants . . . . .	542
5.160.1 Detailed Description . . . . .	542
5.160.2 Macro Definition Documentation . . . . .	542
5.161RCX and Scout sound constants . . . . .	544
5.161.1 Detailed Description . . . . .	544
5.161.2 Macro Definition Documentation . . . . .	544
5.162Scout constants . . . . .	545
5.162.1 Detailed Description . . . . .	545
5.163Scout light constants . . . . .	546
5.163.1 Detailed Description . . . . .	546
5.163.2 Macro Definition Documentation . . . . .	546
5.164Scout sound constants . . . . .	547
5.164.1 Detailed Description . . . . .	547
5.164.2 Macro Definition Documentation . . . . .	547
5.165Scout sound set constants . . . . .	550
5.165.1 Detailed Description . . . . .	550
5.165.2 Macro Definition Documentation . . . . .	550
5.166Scout mode constants . . . . .	551
5.166.1 Detailed Description . . . . .	551
5.166.2 Macro Definition Documentation . . . . .	551
5.167Scout motion rule constants . . . . .	552

5.167.1 Detailed Description . . . . .	552
5.167.2 Macro Definition Documentation . . . . .	552
5.168Scout touch rule constants . . . . .	553
5.168.1 Detailed Description . . . . .	553
5.168.2 Macro Definition Documentation . . . . .	553
5.169Scout light rule constants . . . . .	554
5.169.1 Detailed Description . . . . .	554
5.169.2 Macro Definition Documentation . . . . .	554
5.170Scout transmit rule constants . . . . .	555
5.170.1 Detailed Description . . . . .	555
5.170.2 Macro Definition Documentation . . . . .	555
5.171Scout special effect constants . . . . .	556
5.171.1 Detailed Description . . . . .	556
5.171.2 Macro Definition Documentation . . . . .	556
5.172RCX and Scout source constants . . . . .	557
5.172.1 Detailed Description . . . . .	557
5.172.2 Macro Definition Documentation . . . . .	558
5.173RCX and Scout opcode constants . . . . .	561
5.173.1 Detailed Description . . . . .	562
5.173.2 Macro Definition Documentation . . . . .	562
5.174HiTechnic/mindsensors Power Function/IR Train constants . . . . .	569
5.174.1 Detailed Description . . . . .	569
5.175Power Function command constants . . . . .	570
5.175.1 Detailed Description . . . . .	570
5.175.2 Macro Definition Documentation . . . . .	570
5.176Power Function channel constants . . . . .	571
5.176.1 Detailed Description . . . . .	571
5.176.2 Macro Definition Documentation . . . . .	571
5.177Power Function mode constants . . . . .	572
5.177.1 Detailed Description . . . . .	572
5.177.2 Macro Definition Documentation . . . . .	572
5.178PF/IR Train function constants . . . . .	573
5.178.1 Detailed Description . . . . .	573
5.178.2 Macro Definition Documentation . . . . .	573
5.179IR Train channel constants . . . . .	574
5.179.1 Detailed Description . . . . .	574
5.179.2 Macro Definition Documentation . . . . .	574

5.180Power Function output constants . . . . .	575
5.180.1 Detailed Description . . . . .	575
5.180.2 Macro Definition Documentation . . . . .	575
5.181Power Function pin constants . . . . .	576
5.181.1 Detailed Description . . . . .	576
5.181.2 Macro Definition Documentation . . . . .	576
5.182Power Function single pin function constants . . . . .	577
5.182.1 Detailed Description . . . . .	577
5.182.2 Macro Definition Documentation . . . . .	577
5.183Power Function CST options constants . . . . .	578
5.183.1 Detailed Description . . . . .	578
5.183.2 Macro Definition Documentation . . . . .	578
5.184Power Function PWM option constants . . . . .	580
5.184.1 Detailed Description . . . . .	580
5.184.2 Macro Definition Documentation . . . . .	580
5.185HiTechnic device constants . . . . .	582
5.185.1 Detailed Description . . . . .	582
5.185.2 Macro Definition Documentation . . . . .	582
5.186HiTechnic IRSeeker2 constants . . . . .	584
5.186.1 Detailed Description . . . . .	584
5.186.2 Macro Definition Documentation . . . . .	584
5.187HiTechnic IRReceiver constants . . . . .	586
5.187.1 Detailed Description . . . . .	586
5.187.2 Macro Definition Documentation . . . . .	586
5.188HiTechnic Color2 constants . . . . .	587
5.188.1 Detailed Description . . . . .	587
5.188.2 Macro Definition Documentation . . . . .	587
5.189HiTechnic Angle sensor constants . . . . .	589
5.189.1 Detailed Description . . . . .	589
5.189.2 Macro Definition Documentation . . . . .	589
5.190HiTechnic Barometric sensor constants . . . . .	591
5.190.1 Detailed Description . . . . .	591
5.190.2 Macro Definition Documentation . . . . .	591
5.191HiTechnic Prototype board constants . . . . .	592
5.191.1 Detailed Description . . . . .	592
5.191.2 Macro Definition Documentation . . . . .	592
5.192HiTechnic Prototype board analog input constants . . . . .	594

5.192.1 Detailed Description . . . . .	594
5.192.2 Macro Definition Documentation . . . . .	594
5.193HiTechnic SuperPro constants . . . . .	595
5.193.1 Detailed Description . . . . .	596
5.193.2 Macro Definition Documentation . . . . .	596
5.194HiTechnic SuperPro analog input index constants . . . . .	601
5.194.1 Detailed Description . . . . .	601
5.194.2 Macro Definition Documentation . . . . .	601
5.195HiTechnic SuperPro analog output index constants . . . . .	602
5.195.1 Detailed Description . . . . .	602
5.195.2 Macro Definition Documentation . . . . .	602
5.196HiTechnic PIR sensor constants . . . . .	603
5.196.1 Detailed Description . . . . .	603
5.196.2 Macro Definition Documentation . . . . .	603
5.197MindSensors device constants . . . . .	604
5.197.1 Detailed Description . . . . .	605
5.197.2 Macro Definition Documentation . . . . .	605
5.198MindSensors DIST-Nx constants . . . . .	607
5.198.1 Detailed Description . . . . .	607
5.198.2 Macro Definition Documentation . . . . .	607
5.199MindSensors PSP-Nx constants . . . . .	609
5.199.1 Detailed Description . . . . .	609
5.199.2 Macro Definition Documentation . . . . .	609
5.200MindSensors PSP-Nx button set 1 constants . . . . .	611
5.200.1 Detailed Description . . . . .	611
5.200.2 Macro Definition Documentation . . . . .	611
5.201MindSensors PSP-Nx button set 2 constants . . . . .	612
5.201.1 Detailed Description . . . . .	612
5.201.2 Macro Definition Documentation . . . . .	612
5.202MindSensors nRLink constants . . . . .	613
5.202.1 Detailed Description . . . . .	613
5.202.2 Macro Definition Documentation . . . . .	613
5.203MindSensors ACCL-Nx constants . . . . .	615
5.203.1 Detailed Description . . . . .	615
5.203.2 Macro Definition Documentation . . . . .	615
5.204MindSensors ACCL-Nx sensitivity level constants . . . . .	618
5.204.1 Detailed Description . . . . .	618

5.204.2 Macro Definition Documentation . . . . .	618
5.205 MindSensors PFMate constants . . . . .	619
5.205.1 Detailed Description . . . . .	619
5.205.2 Macro Definition Documentation . . . . .	619
5.206 PFMate motor constants . . . . .	621
5.206.1 Detailed Description . . . . .	621
5.206.2 Macro Definition Documentation . . . . .	621
5.207 PFMate channel constants . . . . .	622
5.207.1 Detailed Description . . . . .	622
5.207.2 Macro Definition Documentation . . . . .	622
5.208 MindSensors NXTServo constants . . . . .	623
5.208.1 Detailed Description . . . . .	623
5.209 MindSensors NXTServo registers . . . . .	624
5.209.1 Detailed Description . . . . .	624
5.209.2 Macro Definition Documentation . . . . .	624
5.210 MindSensors NXTServo position constants . . . . .	627
5.210.1 Detailed Description . . . . .	627
5.210.2 Macro Definition Documentation . . . . .	627
5.211 MindSensors NXTServo quick position constants . . . . .	628
5.211.1 Detailed Description . . . . .	628
5.211.2 Macro Definition Documentation . . . . .	628
5.212 MindSensors NXTServo servo numbers . . . . .	629
5.212.1 Detailed Description . . . . .	629
5.212.2 Macro Definition Documentation . . . . .	629
5.213 MindSensors NXTServo commands . . . . .	630
5.213.1 Detailed Description . . . . .	630
5.213.2 Macro Definition Documentation . . . . .	630
5.214 MindSensors NXTHID constants . . . . .	632
5.214.1 Detailed Description . . . . .	632
5.215 MindSensors NXTHID registers . . . . .	633
5.215.1 Detailed Description . . . . .	633
5.215.2 Macro Definition Documentation . . . . .	633
5.216 MindSensors NXTHID modifier keys . . . . .	634
5.216.1 Detailed Description . . . . .	634
5.216.2 Macro Definition Documentation . . . . .	634
5.217 MindSensors NXTHID commands . . . . .	636
5.217.1 Detailed Description . . . . .	636

5.217.2 Macro Definition Documentation . . . . .	636
5.218 MindSensors NXTPowerMeter constants . . . . .	637
5.218.1 Detailed Description . . . . .	637
5.219 MindSensors NXTPowerMeter registers . . . . .	638
5.219.1 Detailed Description . . . . .	638
5.219.2 Macro Definition Documentation . . . . .	638
5.220 MindSensors NXTPowerMeter commands . . . . .	640
5.220.1 Detailed Description . . . . .	640
5.220.2 Macro Definition Documentation . . . . .	640
5.221 MindSensors NXTSumoEyes constants . . . . .	641
5.221.1 Detailed Description . . . . .	641
5.221.2 Macro Definition Documentation . . . . .	641
5.222 MindSensors NXTLineLeader constants . . . . .	642
5.222.1 Detailed Description . . . . .	642
5.223 MindSensors NXTLineLeader registers . . . . .	643
5.223.1 Detailed Description . . . . .	643
5.223.2 Macro Definition Documentation . . . . .	643
5.224 MindSensors NXTLineLeader commands . . . . .	645
5.224.1 Detailed Description . . . . .	645
5.224.2 Macro Definition Documentation . . . . .	645
5.225 MindSensors NXTNumericPad constants . . . . .	647
5.225.1 Detailed Description . . . . .	647
5.226 MindSensors NXTNumericPad registers . . . . .	648
5.226.1 Detailed Description . . . . .	648
5.226.2 Macro Definition Documentation . . . . .	648
5.227 MindSensors NXTTouchPanel constants . . . . .	649
5.227.1 Detailed Description . . . . .	649
5.228 MindSensors NXTTouchPanel registers . . . . .	650
5.228.1 Detailed Description . . . . .	650
5.228.2 Macro Definition Documentation . . . . .	650
5.229 MindSensors NXTTouchPanel commands . . . . .	651
5.229.1 Detailed Description . . . . .	651
5.229.2 Macro Definition Documentation . . . . .	651
5.230 Codatex device constants . . . . .	652
5.230.1 Detailed Description . . . . .	652
5.231 Codatex RFID sensor constants . . . . .	653
5.231.1 Detailed Description . . . . .	653

5.231.2 Macro Definition Documentation . . . . .	653
5.232 Codatex RFID sensor modes . . . . .	654
5.232.1 Detailed Description . . . . .	654
5.232.2 Macro Definition Documentation . . . . .	654
5.233 Dexter Industries device constants . . . . .	655
5.233.1 Detailed Description . . . . .	655
5.234 Dexter Industries GPS sensor constants . . . . .	656
5.234.1 Detailed Description . . . . .	656
5.234.2 Macro Definition Documentation . . . . .	656
5.235 Dexter Industries IMU sensor constants . . . . .	658
5.235.1 Detailed Description . . . . .	658
5.235.2 Macro Definition Documentation . . . . .	658
5.236 Dexter Industries IMU Gyro register constants . . . . .	660
5.236.1 Detailed Description . . . . .	660
5.236.2 Macro Definition Documentation . . . . .	660
5.237 Dexter Industries IMU Gyro control register 1 constants . . . . .	664
5.237.1 Detailed Description . . . . .	664
5.237.2 Macro Definition Documentation . . . . .	664
5.238 Dexter Industries IMU Gyro control register 2 constants . . . . .	666
5.238.1 Detailed Description . . . . .	666
5.238.2 Macro Definition Documentation . . . . .	666
5.239 Dexter Industries IMU Gyro control register 3 constants . . . . .	668
5.239.1 Detailed Description . . . . .	668
5.239.2 Macro Definition Documentation . . . . .	668
5.240 Dexter Industries IMU Gyro control register 4 constants . . . . .	669
5.240.1 Detailed Description . . . . .	669
5.240.2 Macro Definition Documentation . . . . .	669
5.241 Dexter Industries IMU Gyro control register 5 constants . . . . .	670
5.241.1 Detailed Description . . . . .	670
5.241.2 Macro Definition Documentation . . . . .	670
5.242 Dexter Industries IMU Gyro FIFO control register onstants . . . . .	672
5.242.1 Detailed Description . . . . .	672
5.242.2 Macro Definition Documentation . . . . .	672
5.243 Dexter Industries IMU Gyro status register constants . . . . .	673
5.243.1 Detailed Description . . . . .	673
5.243.2 Macro Definition Documentation . . . . .	673
5.244 Dexter Industries IMU Accelerometer register constants . . . . .	674

5.244.1 Detailed Description . . . . .	674
5.244.2 Macro Definition Documentation . . . . .	674
5.245Dexter Industries IMU Accelerometer status register constants . . . . .	677
5.245.1 Detailed Description . . . . .	677
5.245.2 Macro Definition Documentation . . . . .	677
5.246Dexter Industries IMU Accelerometer mode control register constants . . . . .	678
5.246.1 Detailed Description . . . . .	678
5.246.2 Macro Definition Documentation . . . . .	678
5.247Dexter Industries IMU Accelerometer interrupt latch reset register constants . . . . .	679
5.247.1 Detailed Description . . . . .	679
5.247.2 Macro Definition Documentation . . . . .	679
5.248Dexter Industries IMU Accelerometer control register 1 constants . . . . .	680
5.248.1 Detailed Description . . . . .	680
5.248.2 Macro Definition Documentation . . . . .	680
5.249Dexter Industries IMU Accelerometer control register 2 constants . . . . .	682
5.249.1 Detailed Description . . . . .	682
5.249.2 Macro Definition Documentation . . . . .	682
5.250Microinfinity device constants . . . . .	683
5.250.1 Detailed Description . . . . .	683
5.251Microinfinity CruizCore XG1300L sensor constants . . . . .	684
5.251.1 Detailed Description . . . . .	684
5.251.2 Macro Definition Documentation . . . . .	684
5.252Microinfinity CruizCore XG1300L . . . . .	686
5.252.1 Detailed Description . . . . .	686
5.252.2 Macro Definition Documentation . . . . .	686
5.253Data type limits . . . . .	687
5.253.1 Detailed Description . . . . .	687
5.253.2 Macro Definition Documentation . . . . .	687
5.254Graphics library begin modes . . . . .	689
5.254.1 Detailed Description . . . . .	689
5.254.2 Macro Definition Documentation . . . . .	689
5.255Graphics library actions . . . . .	690
5.255.1 Detailed Description . . . . .	690
5.255.2 Macro Definition Documentation . . . . .	690
5.256Graphics library settings . . . . .	692
5.256.1 Detailed Description . . . . .	692
5.256.2 Macro Definition Documentation . . . . .	692

---

5.257 Graphics library cull mode . . . . .	693
5.257.1 Detailed Description . . . . .	693
5.257.2 Macro Definition Documentation . . . . .	693
<b>6 File Documentation</b>	<b>694</b>
6.1 NBCAPIDocs.h File Reference . . . . .	694
6.1.1 Detailed Description . . . . .	694
6.2 NBCCommon.h File Reference . . . . .	694
6.2.1 Detailed Description . . . . .	735
6.2.2 Macro Definition Documentation . . . . .	735

<b>Index</b>	<b>881</b>
--------------	------------

## 1 NBC Programmer's Guide

February 17, 2013

by John Hansen

- [Introduction](#)
- [The NBC Language](#)

## 2 Introduction

### Introduction

NBC stands for NeXT Byte Codes. It is a simple language for programming the LEGO MINDSTORMS NXT product. The NXT has a byte-code interpreter (provided by LEGO), which can be used to execute programs. The NBC compiler translates a source program into LEGO NXT byte-codes, which can then be executed on the NXT itself. Although the preprocessor and format of NBC programs are similar to assembly, NBC is not a general-purpose assembly language - there are many restrictions that stem from limitations of the LEGO byte-code interpreter.

Logically, NBC is defined as two separate pieces. The NBC language describes the syntax to be used in writing programs. The NBC Application Programming Interface (API) describes the system functions, constants, and macros that can be used by programs. This API is defined in a special file known as a "header file" which is automatically included at the beginning of any NBC program.

This document describes both the NBC language and the NBC API. In short, it provides the information needed to write NBC programs. Since there are different interfaces for NBC, this document does not describe how to use any specific NBC implementation (such as the command-line compiler or Bricx Command Center). Refer to the documentation provided with the NBC tool, such as the NBC User Manual, for information specific to that implementation.

For up-to-date information and documentation for NBC, visit the NBC website at <http://bricxcc.sourceforge.net/nbc/>.

### 3 The NBC Language

#### The NBC Language

This section describes the NBC language itself. This includes the lexical rules used by the compiler, the structure programs, statements, and expressions, and the operation of the preprocessor.

Unlike some assembly languages, NBC is a case-sensitive language. That means that the identifier "xYz" is not the same identifier as "Xyz". Similarly, the subtract statement begins with the keyword "sub" but "suB", "Sub", or "SUB" are all just valid identifiers - not keywords.

- [Lexical Rules](#)
- [Program Structure](#)
- [The Preprocessor](#)
- [Compiler Tokens](#)
- [Expression Evaluator](#)
- [Statements](#)

#### 3.1 Lexical Rules

##### Lexical Rules

The lexical rules describe how NBC breaks a source file into individual tokens. This includes the way comments are written, then handling of whitespace, and valid characters for identifiers.

- [Comments](#)
- [Whitespace](#)
- [Numerical Constants](#)
- [String Constants](#)
- [Identifiers and Keywords](#)

##### 3.1.1 Comments

###### Comments

Three forms of comments are supported in NBC. The first form (traditional C comments) begin with '/\*' and end with '\*/'. These comments are allowed to span multiple lines, but they cannot be nested.

```
/* this is a comment */  
/* this is a two  
line comment */  
  
/* another comment...  
/* trying to nest...  
   ending the inner comment...*/  
this text is no longer a comment! */
```

The second form of comments supported in NXBC begins with '//' and continues to the end of the current line. These are sometimes known as C++ style comments.

```
// a single line comment
```

The third form of comments begins with ; and ends with a newline. This form is the traditional assembly language style comments.

```
; another single line comment
```

As you might guess, the compiler ignores comments. Their only purpose is to allow the programmer to document the source code.

### 3.1.2 Whitespace

#### Whitespace

Whitespace consists of all spaces, tabs, and newlines. It is used to separate tokens and to make a program more readable. As long as the tokens are distinguishable, adding or subtracting whitespace has no effect on the meaning of a program. For example, the following lines of code both have the same meaning:

```
set x,2
set x, 2
```

Generally, whitespace is ignored outside of string constants and constant numeric expressions. However, unlike in C, NBC statements may not span multiple lines. Aside from pre-processor macros invocations, each statement in an NBC program must begin and end on the same line.

```
add x, x, 2 // okay
add x,
      // error
x, 2      // error

set x, (2*2)+43-12 // okay
set x, 2 * 2 // error (constant expression contains whitespace)
```

The exception to this rule is if you end a line with the \' character which makes the NBC parser continue the current statement on the next line just like with preprocessor macros.

```
add x, \
x, 2 // okay
```

### 3.1.3 Numerical Constants

#### Numerical Constants

Numerical constants may be written in either decimal or hexadecimal form. Decimal constants consist of one or more decimal digits. Decimal constants may optionally include a decimal point along with one or more decimal digits following the decimal point. Hexadecimal constants start with 0x or 0X followed by one or more hexadecimal digits.

```
set x, 10 // set x to 10
set x, 0x10 // set x to 16 (10 hex)
mov f, 1.5 // set f to 1.5
```

### 3.1.4 String Constants

#### String Constants

String constants in NBC are delimited with either single or double quote characters. NBC represents a string as an array of bytes, with the last byte in the array being a zero. The final zero byte is generally referred to as the null terminator.

```
TextOut(0, LCD_LINE1, 'testing')
```

### 3.1.5 Identifiers and Keywords

#### Identifiers and Keywords

Identifiers are used for variable, task, function, and subroutine names. The first character of an identifier must be an upper or lower case letter or the underscore ('\_'). Remaining characters may be letters, numbers, and underscores.

A number of tokens are reserved for use in the NBC language itself. These are called keywords and may not be used as identifiers. A complete list of keywords appears below:

add	sub	neg	mul
div	mod	and	or
xor	not	cmp	tst
index	replace	arrsize	arrbuild
arrsubset	arrinit	mov	set
flatten	unflatten	numtostr	strtonum
strcat	strsubset	strtoarr	arrrtostr
jmp	brcmp	brtst	syscall
stop	exit	exitto	acquire
release	subcall	subret	setin
setout	getin	getout	wait
gettick	thread	endt	subroutine
follows	precedes	segment	ends
typedef	struct	db	byte
sbyte	ubyte	dw	word
sword	uword	dd	dword
sdword	udword	long	slong
ulong	void	mutex	waitv
call	return	abs	sign
strindex	strreplace	strlen	shl
shr	sizeof	compchk	compif
compelce	compend		isconst
asl	asr	lsl	lsr
rotl	rotr	start	stopthread
priority	cmnt	fmtnum	compchktype
float	wait2	sqrt	waitv
arrop	acos	asin	atan
ceil	exp	floor	tan
tanh	cos	cosh	log
log10	sin	sinh	trunc
frac	atan2	pow	muldiv
acosd	asind	atand	tand
tanhd	cosd	coshd	sind
sinhd	atan2d	addrf	exittovar
subcallvar	stopthreadvar	startvar	jmpabsvar
brcmpabsvar	brtstabsvar	callvar	setclump

setlabel			
----------	--	--	--

## 3.2 Program Structure

### Program Structure

An NBC program is composed of code blocks and global variables in data segments. There are two primary types of code blocks: thread and subroutines. Each of these types of code blocks has its own unique features and restrictions, but they share a common structure.

A third type of code block is the preprocessor macro function. This code block type is used throughout the NBC API. Macro functions are the only type of code block, which use a parameter passing syntax similar to what you might see in a language like C or Pascal.

Data segment blocks are used to define types and to declare variables. An NBC program can have zero or more data segments, which can be placed either outside of a code block or within a code block. Regardless of the location of the data segment, all variables in an NBC program are global.

- [Threads](#)
- [Subroutines](#)
- [Macro Functions](#)
- [Data Segments](#)

### 3.2.1 Threads

#### Threads

The NXT implicitly supports multi-threading, thus an NBC thread directly corresponds to an NXT thread. Threads are defined using the `thread` keyword with the following syntax:

```
thread name
  // the thread's code is placed here
endt
```

The name of the thread may be any legal identifier. A program must always have at least one thread. If there is a thread named "main" then that thread will be the thread that is started whenever the program is run. If none of the threads are named "main" then the very first thread that the compiler encounters in the source code will be the main thread. The maximum number of threads supported by the NXT is 256.

The body of a thread consists of a list of statements and optional data segments. Threads may be started by scheduling dependant threads using the [precedes](#) or [follows](#) statements. You may also start a thread using the [start](#) statement. With the standard NXT firmware threads cannot be stopped by another thread. The only way to stop a thread is by stopping all threads using the [stop](#) statement or by a thread stopping on its own via the [exit](#) and [exitto](#) statements. Using the NBC/NBC enhanced firmware you can also stop another thread using the [stopthread](#) statement.

```
thread main
  precedes waiter, worker
  // thread body goes here
  // finalize this thread and schedule the threads in the
  // specified range to execute
  exit // all dependants are automatically scheduled
endt

thread waiter
  // thread body goes here
  // exit
  // exit is optional due to smart compiler finalization
```

```

endt

thread worker
    precedes waiter
    // thread body goes here
    exit // only one dependent - schedule it to execute
endt

```

- endt

### 3.2.1.1 endt

End thread

A thread definition ends with the endt keyword.

```

thread main
    // thread body goes here
endt

```

### 3.2.2 Subroutines

Subroutines

Subroutines allow a single copy of some code to be shared between several different callers. This makes subroutines much more space efficient than macro functions. Subroutines are defined using the subroutine keyword with the following syntax:

```

subroutine name
    // body of subroutine
    return // subroutines must end with a return statement
ends

```

A subroutine is just a special type of thread that is designed to be called explicitly by other threads or subroutines. Its name can be any legal identifier. Subroutines are not scheduled to run via the same mechanism that is used with threads. Instead, subroutines and threads execute other subroutines by using the [call](#) statement (described in the [Statements](#) section).

```

thread main
    // body of main thread goes here
    call mySub // compiler handles subroutine return address
    exit // finalize execution (details handled by the compiler)
endt

subroutine mySub
    // body of subroutine goes here
    return // compiler handles the subroutine return address
ends

```

You can pass arguments into and out of subroutines using global variables. If a subroutine is designed to be used by concurrently executing threads then calls to the subroutine must be protected by acquiring a mutex prior to the subroutine call and releasing the mutex after the call.

You can also call a thread as a subroutine using a slightly different syntax. This technique is required if you want to call a subroutine which executes two threads simultaneously. The [subcall](#) and [subret](#) statements must be used instead of [call](#) and [return](#). You also must provide a global variable to store the return address as shown in the sample code below.

```

thread main
    // thread body goes here
    acquire ssMutex
    call SharedSub // automatic return address
    release ssMutex
    // calling a thread as a subroutine

```

```

subcall AnotherSub, anothersub_returnaddress
exit
endt

subroutine SharedSub
// subroutine body goes here
return // return is required as the last operation
ends

thread AnotherSub
// threads can be subroutines too
subret anothersub_returnaddress // manual return address
endt

```

After the subroutine completes executing, it returns back to the calling routine and program execution continues with the next statement following the subroutine call. The maximum number of threads and subroutines supported by the NXT firmware is 256.

- [ends](#)

### 3.2.2.1 ends

End subroutine or data segment.

A subroutine definition ends with the ends keyword.

```

subroutine doStuff
// subroutine body goes here
ends

```

A data segment also ends with the ends keyword.

```

dseg segment
// data definitions go here
dseg ends

```

### 3.2.3 Macro Functions

#### Macro Functions

It is often helpful to group a set of statements together into a single function, which can then be called as needed. NBC supports macro functions with arguments. Values may be returned from a macro function by changing the value of one or more of the arguments within the body of the macro function.

Macro functions are defined using the following syntax:

```

#define name(argument_list) \
// body of the macro function \
// last line in macro function body has no '\' at the end

```

Please note that the newline escape character ('\') must be the very last character on the line. If it is followed by any whitespace or comments then the macro body is terminated at that point and the next line is not considered to be part of the macro definition.

The argument list may be empty, or it may contain one or more argument definitions. An argument to a macro function has no type. Each argument is simply defined by its name. Multiple arguments are separated by commas. Arguments to a macro function can either be inputs (constants or variables) for the code in the body of the function to process or they can be outputs (variables only) for the code to modify and return. The following sample shows how to define a macro function to simplify the process of drawing text on the NXT LCD screen:

```
#define MyMacro(x, y, berase, msg) \
    mov dtArgs.Location.X, x \
    mov dtArgs.Location.Y, y \
    mov dtArgs.Options, berase \
    mov dtArgs.Text, msg \
    syscall DrawText, dtArgs

MyMacro(0, 0, TRUE, 'testing')
MyMacro(10, 20, FALSE, 'Please Work')
```

NBC macro functions are always expanded inline by the NBC preprocessor. This means that each call to a macro function results in another copy of the function's code being included in the program. Unless used judiciously, inline macro functions can lead to excessive code size.

### 3.2.4 Data Segments

#### Data Segments

Data segments contain all type definitions and variable declarations. Data segments are defined using the following syntax:

```
dseg segment
    // type definitions and variable declarations go here
dseg ends

thread main
    dseg segment
        // or here - still global, though
    dseg ends
endt
```

- [segment](#)

You can have multiple data segments in an NBC program. All variables are global regardless of where they are declared. Once declared, they may be used within all threads, subroutines, and macro functions. Their scope begins at the declaration and ends at the end of the program.

- [Type Definitions](#)
- [Structure Definitions](#)
- [Variable Declarations](#)

#### 3.2.4.1 segment

Start a data segment.

A data segment starts with the segment keyword.

```
dseg segment
    // data definitions go here
dseg ends
```

#### 3.2.4.2 Type Definitions

##### Type Definitions

Type definitions must be contained within a data segment. They are used to define new type aliases or new aggregate types (i.e., structures). A type alias is defined using the typedef keyword with the following syntax:

```
type_alias typedef existing_type
```

The new alias name may be any valid identifier. The existing type must be some type already known by the compiler. It can be a native type or a user-defined type. Once a type alias has been defined it can be used in subsequent variable declarations and aggregate type definitions. The following is an example of a simple type alias definition:

```
big typedef dword // big is now an alias for the dword type
```

### 3.2.4.3 Structure Definitions

#### Structure Definitions

Structure definitions must also be contained within a data segment. They are used to define a type which aggregates or contains other native or user-defined types. A structure definition is defined using the struct and ends keywords with the following syntax:

```
TypeName struct
  x byte
  y byte
TypeName ends
```

Structure definitions allow you to manage related data in a single combined type. They can be as simple or complex as the needs of your program dictate. The following is an example of a fairly complex structure:

```
MyPoint struct
  x byte
  y byte
MyPoint ends
ComplexStrut struct
  value1 big           // using a type alias
  value2 sdword
  buffer byte[]        // array of byte
  blen word
  extrastuff MyPoint[] // array of structs
  pt_1 MyPoint         // struct contains struct instances
  pt_2 MyPoint
ComplexStrut ends
```

### 3.2.4.4 Variable Declarations

#### Variable Declarations

All variable declarations must be contained within a data segment. They are used to declare variables for use in a code block such as a thread, subroutine, or macro function. A variable is declared using the following syntax:

```
var_name type_name optional_initialization
```

The variable name may be any valid identifier. The type name must be a type or type alias already known by the compiler. The optional initialization format depends on the variable type, but for non-aggregate (scalar) types the format is simply a constant integer or constant expression (which may not contain whitespace). See the examples later in this section.

The NXT firmware supports several different types of variables which are grouped into two categories: scalar types and aggregate types. Scalar types are a single integer value which may be signed or unsigned and occupy one, two, or four bytes of memory. The keywords for declaring variables of a scalar type are listed in the following table:

Type Name	Information
byte, ubyte, db	8 bit unsigned
sbyte	8 bit signed

<code>word, uword, dw</code>	16 bit unsigned
<code>sword</code>	16 bit signed
<code>dword, udword, dd</code>	32 bit unsigned
<code>sdword</code>	32 bit signed
<code>long, ulong</code>	32 bit unsigned (alias for dword, udword)
<code>slong</code>	32 bit signed (alias for sdword)
<code>float</code>	32 bit IEEE 754 floating point type
<code>mutex</code>	Special type used for exclusive subroutine access

Table 1. Scalar Types

Examples of scalar variable declarations are as follow:

```
dseg segment
  x byte          // initialized to zero by default
  y byte 12       // initialize to 12
  z sword -2048   // a signed value
  myVar big 0x12345 // use a type alias
  var1 dword 0xFF // value is 255
  myMutex mutex   // mutexes ignore initialization, if present
  bTrue byte 1     // byte variables can be used as booleans
dseg ends
```

Aggregate variables are either structures or arrays of some other type (either scalar or aggregate). Once a user-defined struct type has been defined it may be used to declare a variable of that type. Similarly, user-defined struct types can be used in array declarations. Arrays and structs may be nested (i.e., contained in other arrays or structures) as deeply as the needs of your program dictate, but nesting deeper than 2 or 3 levels may lead to slower program execution due to NXT firmware memory constraints.

Examples of aggregate variable declarations are as follow:

```
dseg segment
  buffer byte[] // starts off empty
  msg byte[] 'Testing'
  // msg is an array of byte =
  // (0x54, 0x65, 0x73, 0x74, 0x69, 0x6e, 0x67, 0x00)
  data long[] {0xabcd, 0xfade0} // two values in the array
  myStruct ComplexStruct // declare an instance of a struct
  Points MyPoint[] // declare an array of a structs
  msgs byte[][] // an array of an array of byte
dseg ends
```

Byte arrays may be initialized either by using braces containing a list of numeric values ({val1, val2, ..., valN}) or by using a string constant delimited with single-quote characters ('Testing'). Embedded single quote characters must be escaped using the '\` character. The '\` character can be part of the string by using two forward slashes: '\'. Arrays of any scalar type other than byte should be initialized using braces. Arrays of struct and nested arrays cannot be initialized.

#### 3.2.4.4.1 byte

The byte type

In NBC the byte type is an unsigned 8-bit value. This type can store values from zero to `UCHAR_MAX`. You can also define an unsigned 8-bit variable using the `ubyte` or `db` keywords.

```
dseg segment
  x byte 12
  b ubyte 0xE2
  test db 0xa0
dseg ends
```

#### 3.2.4.4.2 ubyte

The ubyte type

In NBC the ubyte type is an unsigned 8-bit value. This type can store values from zero to [UCHAR\\_MAX](#). You can also define an unsigned 8-bit variable using the [byte](#) or [db](#) keywords.

```
dseg segment
  x byte 12
  b ubyte 0xE2
  test db 0xa0
dseg ends
```

#### 3.2.4.4.3 db

The db type

In NBC the db type is an unsigned 8-bit value. This type can store values from zero to [UCHAR\\_MAX](#). You can also define an unsigned 8-bit variable using the [ubyte](#) or [byte](#) keywords.

```
dseg segment
  x byte 12
  b ubyte 0xE2
  test db 0xa0
dseg ends
```

#### 3.2.4.4.4 sbyte

The sbyte type

In NBC the sbyte type is a signed 8-bit value. This type can store values from [SCHAR\\_MIN](#) to [SCHAR\\_MAX](#). The sbyte type is often used to store the ASCII value of a single character.

```
dseg segment
  ch sbyte 12
dseg ends
```

#### 3.2.4.4.5 word

The word type

In NBC the word type is an unsigned 16-bit value. This type can store values from zero to [UINT\\_MAX](#). You can also define an unsigned 16-bit variable using the [uword](#) or [dw](#) keywords.

```
dseg segment
  x word 0xffff0
  y uword 62450
  z dw    48500
dseg ends
```

#### 3.2.4.4.6 uword

The uword type

In NBC the uword type is an unsigned 16-bit value. This type can store values from zero to [UINT\\_MAX](#). You can also define an unsigned 16-bit variable using the [word](#) or [dw](#) keywords.

```
dseg segment
  x word 0xffff0
  y uword 62450
  z dw    48500
dseg ends
```

#### 3.2.4.4.7 dw

The dw type

In NBC the dw type is an unsigned 16-bit value. This type can store values from zero to [UINT\\_MAX](#). You can also define an unsigned 16-bit variable using the [uword](#) or [word](#) keywords.

```
dseg segment
  x word 0xffff0
  y uword 62450
  z dw    48500
dseg ends
```

#### 3.2.4.4.8 sword

The sword type

In NBC the sword type is a signed 16-bit value. This type can store values from [INT\\_MIN](#) to [INT\\_MAX](#).

```
dseg segment
  x sword 0xffff
  y sword -23
dseg ends
```

#### 3.2.4.4.9 dword

The dword type

In NBC the dword type is an unsigned 32-bit value. The range of values that can be stored in a ulong variable is from zero to [ULONG\\_MAX](#).

```
dseg segment
  a ulong 0xdeadbeef
  b long 80000
  c dword 150000
  d udword 200000
  e dd 400000
dseg ends
```

#### 3.2.4.4.10 udword

The udword type

In NBC the udword type is an unsigned 32-bit value. The range of values that can be stored in a ulong variable is from zero to [ULONG\\_MAX](#).

```
dseg segment
  a ulong 0xdeadbeef
  b long 80000
  c dword 150000
  d udword 200000
  e dd 400000
dseg ends
```

#### 3.2.4.4.11 dd

The dd type

In NBC the dd type is an unsigned 32-bit value. The range of values that can be stored in a ulong variable is from zero to [ULONG\\_MAX](#).

```
dseg segment
  a ulong 0xdeadbeef
  b long 80000
  c dword 150000
  d udword 200000
  e dd 400000
dseg ends
```

#### 3.2.4.4.12 sdword

The sdword type

In NBC the sdword type is a signed 32-bit value. This type can store values from [LONG\\_MIN](#) to [LONG\\_MAX](#). You can also define a signed 32-bit variable using the [slong](#) keywords.

```
dseg segment
  x slong 2147000000
  y sdword -88235
dseg ends
```

#### 3.2.4.4.13 long

The long type

In NBC the long type is an unsigned 32-bit value. The range of values that can be stored in a ulong variable is from zero to [ULONG\\_MAX](#).

```
dseg segment
  a ulong 0xdeadbeef
  b long 80000
  c dword 150000
  d udword 200000
  e dd 400000
dseg ends
```

#### 3.2.4.4.14 ulong

The ulong type

In NBC the ulong type is an unsigned 32-bit value. The range of values that can be stored in a ulong variable is from zero to [ULONG\\_MAX](#). You can also define an unsigned 32-bit variable using the [long](#), [dword](#), [udword](#), or [dd](#) keywords.

```
dseg segment
  a ulong 0xdeadbeef
  b long 80000
  c dword 150000
  d udword 200000
  e dd 400000
dseg ends
```

#### 3.2.4.4.15 slong

The slong type

In NBC the slong type is a signed 32-bit value. This type can store values from [LONG\\_MIN](#) to [LONG\\_MAX](#). You can also define a signed 32-bit variable using the [sdword](#) keywords.

```
dseg segment
  x slong 2147000000
  y sdword -88235
dseg ends
```

#### 3.2.4.4.16 float

The float type

In NBC the float type is a 32-bit IEEE 754 single precision floating point representation. This is a binary format that occupies 32 bits (4 bytes) and its significand has a precision of 24 bits (about 7 decimal digits).

Floating point arithmetic will be slower than integer operations but if you need to easily store decimal values the float type is your best option. The standard NXT firmware provides the sqrt function which benefits from the ability to use the float type. In the enhanced NBC/NXC firmware there are many more native opcodes from the standard C math library which are designed to work with floats.

```
dseg segment
  pi float 3.14159
  e float 2.71828
  s2 float 1.4142
dseg ends
```

#### 3.2.4.4.17 mutex

The mutex type

In NBC the mutex type is a 32-bit value that is used to synchronize access to resources shared across multiple threads. For this reason there is never a reason to declare a mutex variable inside a task or a subroutine. It is designed for global variables that all tasks or functions can [acquire](#) or [release](#) in order to obtain exclusive access to a resource that other tasks or functions are also trying to use.

```
dseg segment
    motorMutex mutex
dseg ends

thread main
    acquire motorMutex
    // use the motor(s) protected by this mutex.
    release motorMutex
    wait MS_500
endt
```

## 3.3 The Preprocessor

The Preprocessor

NBC also includes a preprocessor that is modeled after the Standard C preprocessor. The C preprocessor processes a source code file before the compiler does. It handles such tasks as including code from other files, conditionally including or excluding blocks of code, stripping comments, defining simple and parameterized macros, and expanding macros wherever they are encountered in the source code.

The NBC preprocessor implements the following standard preprocessor directives: #include, #define, #ifdef, #ifndef, #endif, #if, #elif, #undef, ##, #line, #error, and #pragma. It also supports two non-standard directives: #download and #import. Its implementation is close to a standard C preprocessor's, so most preprocessor directives should work as C programmers expect in NBC. Any significant deviations are explained below.

- [#include](#)
- [#define](#)
- [## \(Concatenation\)](#)
- [Conditional Compilation](#)
- [#import](#)
- [#download](#)

### 3.3.1 #include

include

The #include command works as in Standard C, with the caveat that the filename must be enclosed in double quotes. There is no notion of a system include path, so enclosing a filename in angle brackets is forbidden.

```
#include "foo.h" // ok
#include <foo.h> // error!
```

NBC programs can begin with #include "NXTDefs.h" but they don't need to. This standard header file includes many important constants and macros, which form the core NBC API. NBC no longer requires that you manually include the NXTDefs.h header file. Unless you specifically tell the compiler to ignore the standard system files, this header file is included automatically.

### 3.3.2 #define

define

The #define command is used for macro substitution. Redefinition of a macro will result in a compiler warning.

```
#define TurnTime 3000 // 3 seconds
```

Macros are normally restricted to one line because the newline character at the end of the line acts as a terminator. However, you can write multiline macros by instructing the preprocessor to ignore the newline character. This is accomplished by escaping the newline character with a backslash ('\\'). The backslash character must be the very last character in the line or it will not extend the macro definition to the next line. The code sample below shows how to write a multi-line preprocessor macro.

```
#define square(x, result) \  
    mul result, x, x
```

The #undef directive may be used to remove a macro's definition.

### 3.3.3 ## (Concatenation)

Concatenation

The ## directive works similar to the C preprocessor. It is replaced by nothing, which causes tokens on either side to be concatenated together. Because it acts as a separator initially, it can be used within macro functions to produce identifiers via combination with parameter values.

```
#define ELEMENT_OUT(n) \  
    NumOut(0, LCD_LINE##n, b##n)

dseg segment
    b1 byte
    b2 byte 1
dseg ends

thread main
    ELEMENT_OUT(1)
    ELEMENT_OUT(2)
    wait SEC_2
endt
```

This is the same as writing

```
dseg segment
    b1 byte
    b2 byte 1
dseg ends

thread main
    NumOut(0, LCD_LINE1, b1)
    NumOut(0, LCD_LINE2, b2)
    wait SEC_2
endt
```

### 3.3.4 Conditional Compilation

Conditional Compilation

Conditional compilation works similar to the C preprocessor's conditional compilation. The following preprocessor directives may be used:

Directive	Meaning
#ifdef symbol	If symbol is defined then compile the following code
#ifndef symbol	If symbol is not defined then compile the following code
#else	Switch from compiling to not compiling and vice versa
#endif	Return to previous compiling state
#if condition	If the condition evaluates to true then compile the following code
#elif	Same as #else but used with #if

Table 7. Conditional compilation directives

See the NXTDefs.h header files for many examples of how to use conditional compilation.

### 3.3.5 #import

import

The #import directive lets you define a global byte array variable in your NBC program that contains the contents of the imported file. Like #include, this directive is followed by a filename enclosed in double quote characters. Following the filename you may optionally include a format string for constructing the name of the variable you want to define using this directive.

```
#import "myfile.txt" data
```

By default, the format string is s which means that the name of the file without any file extension will be the name of the variable. For instance, if the format string "data" were not specified in the example above, then the name of the byte array variable would be "myfile". In this case the name of the byte array variable will be "data".

The #import directive is often used in conjunction with the [GraphicArrayOut](#) and [GraphicArrayOutEx](#) API functions.

### 3.3.6 #download

download

The #download directive works in conjunction with the compiler's built-in download capability. It lets you tell the compiler to download a specified auxiliary file in addition to the .rxe file produced from your source code. If the file extension matches a type of source code that the compiler knows how to compile (such as .rs or .nbc) then the compiler will first compile the source before downloading the resulting binary. The name of the file to download (and optionally compile) is enclosed in double quote characters immediately following this directive. If the compiler is only told to compile the original source code then the #download directive is ignored.

```
#download "myfile.rs"
#download "mypicture.ric"
```

## 3.4 Compiler Tokens

### Compiler Tokens

NBC supports special tokens, which it replaces on compilation. The tokens are similar to preprocessor #define macros but they are actually handled directly by the compiler rather than the preprocessor. The supported tokens are as follows:

Token	Usage
<code>_FILE_</code>	This token is replaced with the currently active filename (no path)
<code>_LINE_</code>	This token is replaced with the current line number
<code>_VER_</code>	This token is replaced with the compiler version number
<code>_THREADNAME_</code>	This token is replaced with the current thread name
<code>_I_, J</code>	These tokens are replaced with the current value of I or J. They are both initialized to zero at the start of each thread or subroutine.
<code>_ResetI_, ResetJ</code>	These tokens are replaced with nothing. As a side effect the value of I or J is reset to zero.
<code>_Incl_, IncJ</code>	These tokens are replaced with nothing. As a side effect the value of I or J is incremented by one.
<code>_Decl_, DecJ</code>	These tokens are replaced with nothing. As a side effect the value of I or J is decremented by one.

Table 2. Compiler Tokens

The ## preprocessor directive can help make the use of compiler tokens more readable. `THREADNAME##_##_I_`: would become something like `main_1_`. Without the ## directive it would much harder to read the mixture of compiler tokens and underscores.

### 3.5 Expression Evaluator

#### Expression Evaluator

Constant expressions are supported by NBC for many statement arguments as well as variable initialization. Expressions are evaluated by the compiler when the program is compiled, not at run time. The compiler will return an error if it encounters an expression that contains whitespace. "4+4" is a valid constant expression but "4 + 4" is not.

The expression evaluator supports the following operators:

<code>+</code>	addition
<code>-</code>	subtraction
<code>*</code>	multiplication
<code>/</code>	division
<code>^</code>	exponent
<code>%</code>	modulo (remainder)
<code>&amp;</code>	bitwise and
<code> </code>	bitwise or
<code>~</code>	bitwise xor
<code>&lt;&lt;</code>	shift left
<code>&gt;&gt;</code>	shift right
<code>()</code>	grouping subexpressions
<code>PI</code>	constant value

Table 3. Constant Expression Operators

The expression evaluator also supports the following compile-time functions:

```

tan(x), sin(x), cos(x)
sinh(x), cosh(x)
arctan(x), cotan(x)
arg(x)
exp(x), ln(x), log10(x), log2(x), logn(x, n)

```

```
sqr(x), sqrt(x)
trunc(x), int(x), ceil(x), floor(x), heav(x)
abs(x), sign(x), zero(x), ph(x)
rnd(x), random(x)
max(x, y), min(x, y)
power(x, exp), intpower(x, exp)
```

Table 4. Constant Expression Functions

The following example demonstrates how to use a constant expression:

```
// expression value will be truncated to an integer
set val, 3+(PI*2)-sqrt(30)
```

## 3.6 Statements

### Statements

The body of a code block (thread, subroutine, or macro function) is composed of statements. All statements are terminated with the newline character.

- Assignment Statements
- Math Statements
- Logic Statements
- Bit Manipulation Statements
- Comparison Statements
- Control Flow Statements
- `syscall`
- Timing Statements
- Array Statements
- String Statements
- Scheduling Statements
- Input Statements
- Output Statements
- Compile-time Statements

### 3.6.1 Assignment Statements

#### Assignment Statements

Assignment statements enable you to copy values from one variable to another or to simply set the value of a variable. In NBC there are five ways to assign a new value to a variable.

- `mov`
- `set`
- `setclump`
- `setlabel`
- `addrf`

### 3.6.1.1 mov

#### The mov statement

The mov statement assigns the value of its second argument to its first argument. The first argument must be the name of a variable. It can be of any valid variable type except mutex. The second argument can be a variable or a numeric or string constant. If a constant is used, the compiler creates a variable behind the scenes and initializes it to the specified constant value.

Both arguments to the mov statement must be of compatible types. A scalar value can be assigned to another scalar variable, regardless of type, structs can be assigned to struct variables if the structure types are the same, and arrays can be assigned to an array variable provided that the type contained in the arrays are the same. The syntax of the mov statement is shown below.

```
mov x, y      // set x equal to y
```

### 3.6.1.2 set

#### The set statement

The set statement also assigns its first argument to have the value of its second argument. The first argument must be the name of a variable. It must be a scalar type. The second argument must be a numeric constant or constant expression. You should never use set with a variable of type float. For float types use [mov](#) instead. The syntax of the set statement is shown below.

```
set x, 10      // set x equal to 10
```

Because all arguments must fit into a 2-byte value in the NXT executable, the second argument of the set statement is limited to a 16 bit signed or unsigned value (-32768..65535).

### 3.6.1.3 setclump

#### The setclump statement

The setclump statement assigns its first argument to have the value of its second argument. The first argument must be the name of a variable. It must be a scalar type. The second argument must be a thread (aka clump) name. The syntax of the setclump statement is shown below.

```
setclump x, MyFunction // set x equal to the numeric value of MyFunction
```

### 3.6.1.4 setlabel

#### The setlabel statement

The setlabel statement assigns its first argument to have the value of its second argument. The first argument must be the name of a variable. It must be a scalar type. The second argument must be a label name. The syntax of the setlabel statement is shown below.

```
setlabel x, MyLabel // set x equal to the address of MyLabel
```

### 3.6.1.5 addrof

#### The addrof statement

The addrof statement gets the address of its input (second) argument and stores it in the output (first) argument. The third argument is a flag which indicates whether the address should be absolute or relative.

Relative addresses can be used like pointers along with the IOMapWrite [syscall](#) function. The relative address is an offset from the start of the VM memory pool ([CommandOffsetMemoryPool](#)). An absolute address is only useful for

cases where module IOMap structures expose a pointer address field. One example is the pFont address in the Display module IOMap structure ([DisplayOffsetPFont](#)). The syntax of the addrof statement is shown below.

```
// addrof dest, src, brelative?
addrof ptrFont, fontdataArray, FALSE
```

### 3.6.2 Math Statements

#### Math Statements

Math statements enable you to perform basic math operations on data in your NBC programs. Unlike high level programming languages where mathematical expressions use standard math operators (such as \*, -, +, /), in NBC, as with other assembly languages, math operations are expressed as statements with the math operation name coming first, followed by the arguments to the operation. Nearly all the statements in this family have one output argument and two input arguments. The exceptions include sqrt (square root), neg (negate), abs (absolute value), and sign statements, which are unary statements having a single input argument.

Math statements in NBC differ from traditional assembly math statements because many of the operations can handle arguments of scalar, array, and struct types rather than only scalar types. If, for example, you multiply an array by a scalar then each of the elements in the resulting array will be the corresponding element in the original array multiplied by the scalar value.

Only the abs (absolute value) and sign statements require that their arguments are scalar types. When using the standard NXT firmware these two statements are implemented by the compiler since the firmware does not have built-in support for them. If you install the enhanced NBC/NXC firmware and tell the compiler to target it using the -EF command line switch then these statements will be handled directly by the firmware itself rather than by the compiler.

This family of statements also includes several math operations that are supported only by the enhanced NBC/NXC firmware. These statements cannot be used at all if you tell the compiler to generate code for the standard NXT firmware. The enhanced NBC/NXC firmware math statements are mostly unary functions with a single input argument and a single output argument. They are sin, cos, tan, sind, cosd, tand, asin, acos, atan, asind, acosd, atand, sinh, cosh, tanh, sinh, cosh, tanh, ceil, floor, trunc, frac, exp, log, and log10. There are three enhanced firmware math statements which take two input arguments: pow, atan2, and atan2d. And the muldiv statement takes three input arguments.

- [add](#)
- [sub](#)
- [mul](#)
- [div](#)
- [mod](#)
- [neg](#)
- [abs](#)
- [sign](#)
- [sqrt](#)
- [sin](#)
- [cos](#)
- [tan](#)
- [sind](#)

- [cosd](#)
- [tand](#)
- [asin](#)
- [acos](#)
- [atan](#)
- [atan2](#)
- [asind](#)
- [acosd](#)
- [atand](#)
- [atan2d](#)
- [sinh](#)
- [cosh](#)
- [tanh](#)
- [sinhd](#)
- [coshd](#)
- [tanhd](#)
- [ceil](#)
- [floor](#)
- [trunc](#)
- [frac](#)
- [exp](#)
- [log](#)
- [log10](#)
- [log10](#)
- [pow](#)
- [muldiv](#)

### 3.6.2.1 add

#### The add Statement

The add statement lets you add two input values together and store the result in the first argument. The first argument must be a variable but the second and third arguments can be variables, numeric constants, or constant expressions. The syntax of the add statement is shown below.

```
add x, x, y // add x and y and store result in x
```

### 3.6.2.2 sub

#### The sub Statement

The sub statement lets you subtract two input values and store the result in the first argument. The first argument must be a variable but the second and third arguments can be variables, numeric constants, or constant expressions. The syntax of the sub statement is shown below.

```
sub x, x, y // subtract y from x and store result in x
```

### 3.6.2.3 mul

#### The mul Statement

The mul statement lets you multiply two input values and store the result in the first argument. The first argument must be a variable but the second and third arguments can be variables, numeric constants, or constant expressions. The syntax of the mul statement is shown below.

```
mul x, x, x // set x equal to x^2
```

### 3.6.2.4 div

#### The div Statement

The div statement lets you divide two input values and store the result in the first argument. The first argument must be a variable but the second and third arguments can be variables, numeric constants, or constant expressions. The syntax of the div statement is shown below.

```
div x, x, 2 // set x equal to x / 2 (integer division)
```

### 3.6.2.5 mod

#### The mod Statement

The mod statement lets you calculate the modulus value (or remainder) of two input values and store the result in the first argument. The first argument must be a variable but the second and third arguments can be variables, numeric constants, or constant expressions. The syntax of the mod statement is shown below.

```
mod x, x, 4 // set x equal to x % 4 (0..3)
```

### 3.6.2.6 neg

#### The neg Statement

The neg statement lets you negate an input value and store the result in the first argument. The first argument must be a variable but the second argument can be a variable, a numeric constant, or a constant expression. The syntax of the neg statement is shown below.

```
neg x, y // set x equal to -y
```

### 3.6.2.7 abs

#### The abs Statement

The abs statement lets you take the absolute value of an input value and store the result in the first argument. The first argument must be a variable but the second argument can be a variable, a numeric constant, or a constant expression. The syntax of the abs statement is shown below.

```
abs x, y // set x equal to the absolute value of y
```

### 3.6.2.8 sign

#### The sign Statement

The sign statement lets you take the sign value (-1, 0, or 1) of an input value and store the result in the first argument. The first argument must be a variable but the second argument can be a variable, a numeric constant, or a constant expression. The syntax of the sign statement is shown below.

```
sign x, y // set x equal to -1, 0, or 1
```

### 3.6.2.9 sqrt

#### The sqrt Statement

The sqrt statement lets you take the square root of an input value and store the result in the first argument. The first argument must be a variable but the second argument can be a variable, a numeric constant, or a constant expression. The syntax of the sqrt statement is shown below.

```
sqrt x, y // set x equal to the square root of y
```

### 3.6.2.10 sin

#### The sin Statement

The sin statement lets you calculate the sine value of its input (second) argument (radians) and store the result in its output (first) argument. The syntax of the sin statement is shown below.

```
sin x, y // store the sine of y in x
```

### 3.6.2.11 cos

#### The cos Statement

The cos statement lets you calculate the cosine value of its input (second) argument (radians) and store the result in its output (first) argument. The syntax of the cos statement is shown below.

```
cos x, y // store the cosine of y in x
```

### 3.6.2.12 tan

#### The tan Statement

The tan statement lets you calculate the tangent value of its input (second) argument (radians) and store the result in its output (first) argument. The syntax of the tan statement is shown below.

```
tan x, y // store the tangent of y in x
```

### 3.6.2.13 sind

#### The sind Statement

The sind statement lets you calculate the sine value of its input (second) argument (degrees) and store the result in its output (first) argument. The syntax of the sind statement is shown below.

```
sind x, y // store the sine of y in x
```

### 3.6.2.14 cosd

#### The cosd Statement

The cosd statement lets you calculate the cosine value of its input (second) argument (degrees) and store the result in its output (first) argument. The syntax of the cosd statement is shown below.

```
cosd x, y // store the cosine of y in x
```

### 3.6.2.15 tand

#### The tand Statement

The tand statement lets you calculate the tangent value of its input (second) argument (degrees) and store the result in its output (first) argument. The syntax of the tand statement is shown below.

```
tand x, y // store the tangent of y in x
```

### 3.6.2.16 asin

#### The asin Statement

The asin statement lets you calculate the arc sine value of its input (second) argument and store the result (radians) in its output (first) argument. The syntax of the asin statement is shown below.

```
asin x, y // store the arc sine of y in x
```

### 3.6.2.17 acos

#### The acos Statement

The acos statement lets you calculate the arc cosine value of its input (second) argument and store the result (radians) in its output (first) argument. The syntax of the acos statement is shown below.

```
acos x, y // store the arc cosine of y in x
```

### 3.6.2.18 atan

#### The atan Statement

The atan statement lets you calculate the arc tangent value of its input (second) argument and store the result (radians) in its output (first) argument. The syntax of the atan statement is shown below.

```
atan x, y // store the arc tangent of y in x
```

### 3.6.2.19 atan2

#### The atan2 Statement

The atan2 statement lets you calculate the arc tangent value of its two input (second and third) arguments and store the result (radians) in its output (first) argument. The syntax of the atan2 statement is shown below.

```
atan2 result, y, x // store the arc tangent of y/x in result
```

### 3.6.2.20 asind

#### The asind Statement

The asind statement lets you calculate the arc sine value of its input (second) argument and store the result (degrees) in its output (first) argument. The syntax of the asind statement is shown below.

```
asind x, y // store the arc sine of y in x
```

### 3.6.2.21acosd

#### The acosd Statement

The acosd statement lets you calculate the arc cosine value of its input (second) argument and store the result (degrees) in its output (first) argument. The syntax of the acosd statement is shown below.

```
acosd x, y // store the arc cosine of y in x
```

### 3.6.2.22atand

#### The atand Statement

The atand statement lets you calculate the arc tangent value of its input (second) argument and store the result (degrees) in its output (first) argument. The syntax of the atand statement is shown below.

```
atand x, y // store the arc tangent of y in x
```

### 3.6.2.23atan2d

#### The atan2d Statement

The atan2d statement lets you calculate the arc tangent value of its two input (second and third) arguments and store the result (degrees) in its output (first) argument. The syntax of the atan2d statement is shown below.

```
atan2d result, y, x // store the arc tangent of y/x in result
```

### 3.6.2.24sinh

#### The sinh Statement

The sinh statement lets you calculate the hyperbolic sine value of its input (second) argument and store the result in its output (first) argument. The syntax of the sinh statement is shown below.

```
sinh x, y // store the hyperbolic sine of y in x
```

### 3.6.2.25cosh

#### The cosh Statement

The cosh statement lets you calculate the hyperbolic cosine value of its input (second) argument and store the result in its output (first) argument. The syntax of the cosh statement is shown below.

```
cosh x, y // store the hyperbolic cosine of y in x
```

### 3.6.2.26tanh

#### The tanh Statement

The tanh statement lets you calculate the hyperbolic tangent value of its input (second) argument and store the result in its output (first) argument. The syntax of the tanh statement is shown below.

```
tanh x, y // store the hyperbolic tangent of y in x
```

### 3.6.2.27sinhd

#### The sinhd Statement

The sinhd statement lets you calculate the hyperbolic sine value of its input (second) argument and store the result in its output (first) argument. The syntax of the sinhd statement is shown below.

```
sinhd x, y // store the hyperbolic sine of y in x
```

### 3.6.2.28 coshd

#### The coshd Statement

The coshd statement lets you calculate the hyperbolic cosine value of its input (second) argument and store the result in its output (first) argument. The syntax of the coshd statement is shown below.

```
coshd x, y // store the hyperbolic cosine of y in x
```

### 3.6.2.29 tanhd

#### The tanhd Statement

The tanhd statement lets you calculate the hyperbolic tangent value of its input (second) argument and store the result in its output (first) argument. The syntax of the tanhd statement is shown below.

```
tanhd x, y // store the hyperbolic tangent of y in x
```

### 3.6.2.30 ceil

#### The ceil Statement

The ceil statement lets you calculate the smallest integral value that is not less than the input (second) argument and store the result in its output (first) argument. The syntax of the ceil statement is shown below.

```
ceil x, y // store the ceil of y in x
```

### 3.6.2.31 floor

#### The floor Statement

The floor statement lets you calculate the largest integral value that is not greater than the input (second) argument and store the result in its output (first) argument. The syntax of the floor statement is shown below.

```
floor x, y // store the floor of y in x
```

### 3.6.2.32 trunc

#### The trunc Statement

The trunc statement lets you calculate the integer part of its input (second) argument and store the result in its output (first) argument. The syntax of the trunc statement is shown below.

```
trunc x, y // store the trunc of y in x
```

### 3.6.2.33 frac

#### The frac Statement

The frac statement lets you calculate the fractional part of its input (second) argument and store the result in its output (first) argument. The syntax of the frac statement is shown below.

```
frac x, y // store the frac of y in x
```

### 3.6.2.34 exp

#### The exp Statement

The exp statement lets you calculate the base-e exponential function of x, which is the e number raised to the power x. The syntax of the exp statement is shown below.

```
exp result, x // store the value of e^x in result
```

### 3.6.2.35 log

#### The log Statement

The log statement lets you calculate the natural logarithm of x. The syntax of the log statement is shown below.

```
log result, x // store the natural logarithm of x
```

### 3.6.2.36 log10

#### The log10 Statement

The log10 statement lets you calculate the base-10 logarithm of x. The syntax of the log10 statement is shown below.

```
log10 result, x // store the base-10 logarithm of x
```

### 3.6.2.37 log10

#### The log10 Statement

The log10 statement lets you calculate the base-10 logarithm of x. The syntax of the log10 statement is shown below.

```
log10 result, x // store the base-10 logarithm of x
```

### 3.6.2.38 pow

#### The pow Statement

The pow statement lets you calculate the value of x raised to the y power and store the result in the output (first) argument. The syntax of the pow statement is shown below.

```
pow result, x, y// store the x^y in result
```

### 3.6.2.39 muldiv

#### The muldiv Statement

The muldiv statement lets you multiply two 32-bit values and then divide the 64-bit result by a third 32-bit value. The syntax of the muldiv statement is shown below.

```
muldiv result, a, b, c// store the a*b/c in result
```

## 3.6.3 Logic Statements

#### Logic Statements

Logic statements let you perform basic logical operations on data in your NBC program. As with the math statements, the logical operation name begins the statement and it is followed by the arguments to the logical operation. All the statements in this family have one output argument and two input arguments except the logical not statement. Each statement supports arguments of any type, scalar, array, or struct.

- [and](#)
- [or](#)
- [xor](#)
- [not](#)

### 3.6.3.1 and

#### The and Statement

The and statement lets you bitwise and together two input values and store the result in the first argument. The first argument must be a variable but the second and third arguments can be a variable, a numeric constant, or a constant expression. The syntax of the and statement is shown below.

```
and x, x, y // x = x & y
```

### 3.6.3.2 or

#### The or Statement

The or statement lets you bitwise or together two input values and store the result in the first argument. The first argument must be a variable but the second and third arguments can be a variable, a numeric constant, or a constant expression. The syntax of the or statement is shown below.

```
or x, x, y // x = x | y
```

### 3.6.3.3 xor

#### The xor Statement

The xor statement lets you bitwise exclusive or together two input values and store the result in the first argument. The first argument must be a variable but the second and third arguments can be a variable, a numeric constant, or a constant expression. The syntax of the xor statement is shown below.

```
xor x, x, y // x = x ^ y
```

### 3.6.3.4 not

#### The not Statement

The not statement lets you logically not its input value and store the result in the first argument. The first argument must be a variable but the second argument can be a variable, a numeric constant, or a constant expression. The syntax of the not statement is shown below.

```
not x, x // x = !x (logical not - not bitwise)
```

## 3.6.4 Bit Manipulation Statements

### Bit Manipulation Statements

Bit manipulation statements enable you to perform basic bitwise operations on data in your NBC programs. All statements in this family have one output argument and two input arguments except the complement statement.

Using the standard NXT firmware the basic shift right and shift left statements (shr and shl) are implemented by the compiler since the firmware does not support shift operations at this time. If you install the enhanced NBC/NBC firmware and tell the compiler to target it using the -EF command line switch, then these operations will be handled directly by the firmware itself rather than by the compiler. The other bit manipulation statements described in this section are only available when targeting the enhanced firmware.

- [shr](#)
- [shl](#)

- [asr](#)
- [asl](#)
- [lsl](#)
- [lsl](#)
- [rotl](#)
- [rotl](#)
- [cmnt](#)

#### 3.6.4.1 shr

##### The shr Statement

The shr statement lets you shift right an input value by the number of bits specified by the second input argument and store the resulting value in the output argument. The output (first) argument must be a variable but the second and third arguments can be a variable, a numeric constant, or a constant expression. The syntax of the shr statement is shown below.

```
shr x, x, y // x = x >> y
```

#### 3.6.4.2 shl

##### The shl Statement

The shl statement lets you shift left an input value by the number of bits specified by the second input argument and store the resulting value in the output argument. The output (first) argument must be a variable but the second and third arguments can be a variable, a numeric constant, or a constant expression. The syntax of the shl statement is shown below.

```
shl x, x, y // x = x << y
```

#### 3.6.4.3 asr

##### The asr Statement

The asr statement lets you perform an arithmetic right shift operation. The output (first) argument must be a variable but the second and third arguments can be a variable, a numeric constant, or a constant expression. The syntax of the asr statement is shown below.

```
asr x, x, y // x = x >> y
```

#### 3.6.4.4 asl

##### The asl Statement

The asl statement lets you perform an arithmetic left shift operation. The output (first) argument must be a variable but the second and third arguments can be a variable, a numeric constant, or a constant expression. The syntax of the asl statement is shown below.

```
asl x, x, y // x = x << y
```

### 3.6.4.5 lsr

#### The lsr Statement

The lsr statement lets you perform a logical right shift operation. The output (first) argument must be a variable but the second and third arguments can be a variable, a numeric constant, or a constant expression. The syntax of the lsr statement is shown below.

```
lsr x, x, y
```

### 3.6.4.6 lsl

#### The lsl Statement

The lsl statement lets you perform a logical left shift operation. The output (first) argument must be a variable but the second and third arguments can be a variable, a numeric constant, or a constant expression. The syntax of the lsl statement is shown below.

```
lsl x, x, y
```

### 3.6.4.7 rotr

#### The rotr Statement

The rotr statement lets you perform a rotate right operation. The output (first) argument must be a variable but the second and third arguments can be a variable, a numeric constant, or a constant expression. The syntax of the rotr statement is shown below.

```
rotr x, x, y
```

### 3.6.4.8 rotl

#### The rotl Statement

The rotl statement lets you perform a rotate left operation. The output (first) argument must be a variable but the second and third arguments can be a variable, a numeric constant, or a constant expression. The syntax of the rotl statement is shown below.

```
rotl x, x, y
```

### 3.6.4.9 cmnt

#### The cmnt Statement

The cmnt statement lets you perform a bitwise complement operation. The output (first) argument must be a variable but the second can be a variable, a numeric constant, or a constant expression. The syntax of the cmnt statement is shown below.

```
cmnt x, y // x = ~y
```

## 3.6.5 Comparison Statements

### Comparison Statements

Comparison statements enable you to compare data in your NBC programs. These statements take a comparison code constant as their first argument. Valid comparison constants are listed in the [Comparison Constants](#) section. You can use scalar, array, and aggregate types for the compare or test argument(s).

- [cmp](#)
- [tst](#)

#### 3.6.5.1 cmp

##### The cmp Statement

The cmp statement lets you compare two different input sources. The output (second) argument must be a variable but the remaining arguments can be a variable, a numeric constant, or a constant expression. The syntax of the cmp statement is shown below.

```
cmp EQ, bXEqualsY, x, y // bXEqualsY = (x == y);
```

#### 3.6.5.2 tst

##### The tst Statement

The tst statement lets you compare an input source to zero. The output (second) argument must be a variable but the remaining argument can be a variable, a numeric constant, or a constant expression. The syntax of the tst statement is shown below.

```
tst GT, bXGTZero, x // bXGTZero = (x > 0);
```

### 3.6.6 Control Flow Statements

#### Control Flow Statements

Control flow statements enable you to manipulate or control the execution flow of your NBC programs. Some of these statements take a comparison code constant as their first argument. Valid comparison constants are listed in [Comparison Constants](#) section. You can use scalar, array, and aggregate types for the compare or test argument(s).

- [jmp](#)
- [brcmp](#)
- [brtst](#)
- [jmpabsvar](#)
- [brcmpabsvar](#)
- [brstabvars](#)
- [stop](#)

#### 3.6.6.1 jmp

##### The jmp Statement

The jmp statement lets you unconditionally jump from the current execution point to a new location. Its only argument is a label that specifies where program execution should resume. The syntax of the jmp statement is shown below.

```
jmp LoopStart // jump to the LoopStart label
```

### 3.6.6.2 brcmp

#### The brcmp Statement

The brcmp statement lets you conditionally jump from the current execution point to a new location. It is like the cmp statement except that instead of an output argument it has a label argument that specifies where program execution should resume. The syntax of the brcmp statement is shown below.

```
brcmp EQ, LoopStart, x, y // jump to LoopStart if x == y
```

### 3.6.6.3 brtst

#### The brtst Statement

The brtst statement lets you conditionally jump from the current execution point to a new location. It is like the tst statement except that instead of an output argument it has a label argument that specifies where program execution should resume. The syntax of the brtst statement is shown below.

```
brtst GT, lblXGTZero, x // jump to lblXGTZero if x > 0
```

### 3.6.6.4 jmpabsvar

#### The jmpabsvar Statement

The jmpabsvar statement lets you unconditionally jump from the current execution point to a new location. Its only argument is a variable that specifies where program execution should resume. The syntax of the jmpabsvar statement is shown below.

```
jmpabsvar var // jump to the label address specified by var
```

### 3.6.6.5 brcmpabsvar

#### The brcmpabsvar Statement

The brcmpabsvar statement lets you conditionally jump from the current execution point to a new location. It is like the cmp statement except that instead of an output argument it has a variable argument that specifies where program execution should resume. The syntax of the brcmpabsvar statement is shown below.

```
brcmpabsvar EQ, var, x, y // jump to the loop address specified by var if x ==  
y
```

### 3.6.6.6 brtstabsvar

#### The brtstabsvar Statement

The brtstabsvar statement lets you conditionally jump from the current execution point to a new location. It is like the tst statement except that instead of an output argument it has a variable argument that specifies where program execution should resume. The syntax of the brtstabsvar statement is shown below.

```
brtstabsvar GT, var, x // jump to the loop address specified by var if x > 0
```

### 3.6.6.7 stop

#### The stop Statement

The stop statement lets you stop program execution completely, depending on the value of its boolean input argument. The syntax of the stop statement is shown below.

```
stop bProgShouldStop // stop program if flag <> 0
```

### 3.6.7 syscall

#### The syscall Statement

The syscall statement enables execution of various system functions via a constant function ID and an aggregate type variable for passing arguments to and from the system function. The syntax of the syscall statement is shown below.

```
// ptArgs is a struct with input and output args
syscall SoundPlayTone, ptArgs
```

Valid syscall statement function IDs are listed in the [System Call function constants](#) section.

- [System call structures](#)

#### 3.6.7.1 System call structures

##### System call structures

- [TLocation](#)
- [TSize](#)
- [TFileOpen](#)
- [TFileReadWrite](#)
- [TFileClose](#)
- [TFileResolveHandle](#)
- [TFileRename](#)
- [TFileDelete](#)
- [TSoundPlayFile](#)
- [TSoundPlayTone](#)
- [TSoundGetState](#)
- [TSoundSetState](#)
- [TDrawText](#)
- [TDrawPoint](#)
- [TDrawLine](#)
- [TDrawCircle](#)
- [TDrawRect](#)
- [TDrawGraphic](#)
- [TSetScreenMode](#)
- [TReadButton](#)
- [TCommLSWrite](#)
- [TCommLSRead](#)

- [TCommLSCheckStatus](#)
- [TRandomNumber](#)
- [TGetStartTick](#)
- [TMessageWrite](#)
- [TMessageRead](#)
- [TCommBTCheckStatus](#)
- [TCommBTWrite](#)
- [TCommBTRead](#)
- [TKeepAlive](#)
- [TIOMapRead](#)
- [TIOMapWrite](#)
- [TInputPinFunction](#)
- [TIOMapReadByID](#)
- [TIOMapWriteByID](#)
- [TDisplayExecuteFunction](#)
- [TCommExecuteFunction](#)
- [TLoaderExecuteFunction](#)
- [TFileFind](#)
- [TCommHSControl](#)
- [TCommHSCheckStatus](#)
- [TCommHSReadWrite](#)
- [TCommLSWriteEx](#)
- [TFileSeek](#)
- [TFileResize](#)
- [TDrawGraphicArray](#)
- [TDrawPolygon](#)
- [TDrawEllipse](#)
- [TDrawFont](#)
- [TColorSensorRead](#)
- [TDatalogWrite](#)
- [TDatalogGetTimes](#)
- [TSetSleepTimeout](#)
- [TCommBTOffOn](#)

- [TCommBTConnection](#)
- [TReadSemData](#)
- [TWriteSemData](#)
- [TUpdateCalibCacheInfo](#)
- [TComputeCalibValue](#)
- [TListFiles](#)
- [TMemoryManager](#)
- [TReadLastResponse](#)
- [TFileTell](#)
- [TRandomEx](#)

#### 3.6.7.1.1 TLocation

The TLocation structure.

```
TLocation      struct
X             sword // The X coordinate. Valid range is from 0 to 99
            inclusive.
Y             sword // The Y coordinate. Valid range is from 0 to 63
            inclusive.
TLocation     ends
```

For text drawing the Y value must be a multiple of 8.

#### 3.6.7.1.2 TSize

The TSize structure.

```
TSize   struct
Width  sword // The rectangle width.
Height sword // The rectangle height.
TSize   ends
```

#### 3.6.7.1.3 TFileOpen

The TFileOpen structure.

```
// FileOpenRead, FileOpenWrite, FileOpenAppend, FileOpenWriteLinear,
// FileOpenWriteNonLinear, FileOpenReadLinear
TFileOpen    struct
Result      word   // The function call result.
FileHandle  byte   // The returned file handle to use for subsequent file
                  operations.
Filename    byte[] // The name of the file to open or create.
Length      dword  // The desired maximum file capacity or the returned
                  available length in the file.
TFileOpen   ends
```

Possible Result values include [Loader module error codes](#).

### 3.6.7.1.4 TFileReadWrite

The TFileReadWrite structure.

```
// FileRead, FileWrite
TFileReadWrite struct
Result word // The function call result.
FileHandle byte // The file handle to access.
Buffer byte[] // The buffer to store read bytes or containing
bytes to write.
Length dword // The number of bytes to read or the returned
number of bytes written.
TFileReadWrite ends
```

Possible Result values include [Loader module error codes](#).

### 3.6.7.1.5 TFileClose

The TFileClose structure.

```
// FileClose
TFileClose struct
Result word // The function call result.
FileHandle byte // The file handle to close.
TFileClose ends
```

Possible Result values include [Loader module error codes](#).

### 3.6.7.1.6 TFileResolveHandle

The TFileResolveHandle structure.

```
// FileResolveHandle
TFileResolveHandle struct
Result word // The function call result.
FileHandle byte // The returned resolved file handle.
WriteHandle byte // True if the returned handle is a write handle.
Filename byte[] // The name of the file for which to resolve a
handle.
TFileResolveHandle ends
```

Possible Result values include [LDR\\_HANDLEALREADYCLOSED](#) and [LDR\\_SUCCESS](#).

### 3.6.7.1.7 TFileRename

The TFileRename structure.

```
// FileRename
TFileRename struct
Result word // The function call result.
OldFilename byte[] // The name of the file to be renamed.
NewFilename byte[] // The new name to give to the file.
TFileRename ends
```

Possible Result values include [Loader module error codes](#).

### 3.6.7.1.8 TFileDelete

The TFileDelete structure.

```
// FileDelete
TFileDelete struct
Result word // The function call result.
Filename byte[] // The name of the file to delete.
TFileDelete ends
```

Possible Result values include [Loader module error codes](#).

### 3.6.7.1.9 TSoundPlayFile

The TSoundPlayFile structure.

```
// SoundPlayFile
TSoundPlayFile struct
Result     sbyte    // The function call result, always NO_ERR.
Filename   byte[]   // The name of the file to play.
Loop       byte     // If true, loops at end of file.
Volume     byte     // The sound level. Valid values range from 0 to 4.
TSoundPlayFile ends
```

### 3.6.7.1.10 TSoundPlayTone

The TSoundPlayTone structure.

```
// SoundPlayTone
TSoundPlayTone struct
Result     sbyte    // The function call result, always NO_ERR.
Frequency word    // The tone frequency.
Duration   word    // The tone duration in milliseconds.
Loop       byte     // If true, loops forever.
Volume     byte     // The sound level. Valid values range from 0 to 4.
TSoundPlayTone ends
```

See the [Tone constants](#) group for Frequency values. See the [Time constants](#) group for Duration values.

### 3.6.7.1.11 TSoundGetState

The TSoundGetState structure.

```
// SoundGetState
TSoundGetState struct
State      byte    // The returned sound state.
Flags     byte    // The returned sound flags.
TSoundGetState ends
```

See the [SoundState constants](#) group for State values. See the [SoundFlags constants](#) group for Flags values.

### 3.6.7.1.12 TSoundSetState

The TSoundSetState structure.

```
// SoundSetState
TSoundSetState struct
Result     byte    // The function call result, same as State.
State      byte    // The new sound state.
Flags     byte    // The new sound flags.
TSoundSetState ends
```

See the [SoundState constants](#) group for State values. See the [SoundFlags constants](#) group for Flags values.

### 3.6.7.1.13 TDrawText

The TDrawText structure.

```
// DrawText
TDrawText struct
Result     sbyte    // The function call result. NO_ERR means it
               succeeded.
Location   TLocation // The location in X, LCD line number coordinates.
Text       byte[]   // The text to draw on the LCD.
Options    dword    // The options to use when writing to the LCD.
TDrawText ends
```

See [Drawing option constants](#) for valid Options values.

### 3.6.7.1.14 TDrawPoint

The TDrawPoint structure.

```
// DrawPoint
TDrawPoint    struct
Result        sbyte      // The function call result. NO_ERR means it
                succeeded.
Location     TLocation  // The point location on screen.
Options       dword      // The options to use when writing to the LCD.
TDrawPoint    ends
```

See [Drawing option constants](#) for valid Options values.

### 3.6.7.1.15 TDrawLine

The TDrawLine structure.

```
// DrawLine
TDrawLine     struct
Result        sbyte      // The function call result. NO_ERR means it
                succeeded.
StartLoc     TLocation  // The location of the starting point.
EndLoc       TLocation  // The location of the ending point.
Options       dword      // The options to use when writing to the LCD.
TDrawLine    ends
```

See [Drawing option constants](#) for valid Options values.

### 3.6.7.1.16 TDrawCircle

The TDrawCircle structure.

```
// DrawCircle
TDrawCircle   struct
Result        sbyte      // The function call result. NO_ERR means it
                succeeded.
Center       TLocation  // The location of the circle center.
Size         byte       // The circle radius.
Options       dword      // The options to use when writing to the LCD.
TDrawCircle  ends
```

See [Drawing option constants](#) for valid Options values.

### 3.6.7.1.17 TDrawRect

The TDrawRect structure.

```
// DrawRect
TDrawRect    struct
Result        sbyte      // The function call result. NO_ERR means it
                succeeded.
Location     TLocation  // The top left corner location.
Size         TSize      // The width and height of the rectangle.
Options       dword      // The options to use when writing to the LCD.
TDrawRect    ends
```

See [Drawing option constants](#) for valid Options values.

### 3.6.7.1.18 TDrawGraphic

The TDrawGraphic structure.

```
// DrawGraphic
```

```

TDrawGraphic struct
Result     sbyte    // The function call result.
Location   TLocation // The location on screen.
Filename   byte[]   // The RIC file name.
Variables  sdword[] // The variables passed as RIC arguments.
Options    dword    // The options to use when writing to the LCD.
TDrawGraphic ends

```

See [Drawing option constants](#) for valid Options values. Possible values for Result include [Loader module error codes](#), [ERR\\_FILE](#), and [NO\\_ERR](#).

### 3.6.7.1.19 TSetScreenMode

The TSetScreenMode structure.

```

// SetScreenMode
TSetScreenMode struct
Result     sbyte    // The function call result, always NO_ERR.
ScreenMode dword   // The requested screen mode.
TSetScreenMode ends

```

The standard NXT firmware only supports setting the ScreenMode to [SCREEN\\_MODE\\_RESTORE](#). If you install the NBC/NXC enhanced standard NXT firmware this system function also supports setting the ScreenMode to [SCREEN\\_MODE\\_CLEAR](#).

### 3.6.7.1.20 TReadButton

The TReadButton structure.

```

// ReadButton
TReadButton struct
Result     sbyte    // The function call result, ERR_INVALID_PORT or NO_ERR.
Index      byte     // The requested button index.
Pressed    byte     // The returned button state.
Count      byte     // The returned button pressed count.
Reset      byte     // If true, the count is reset after reading.
TReadButton ends

```

See the [Button name constants](#) group for Index values.

### 3.6.7.1.21 TCommLSWrite

The TCommLSWrite structure

```

// CommLSWrite
TCommLSWrite struct
Result     sbyte    // The function call result.
Port       byte     // The port to which the I2C device is connected.
Buffer    byte[]   // The buffer containing data to be written to the I2C
device.
ReturnLen  byte     // The number of bytes that you want to read from the
I2C device after writing the data. If no read is planned set this to zero.
TCommLSWrite ends

```

Possible Result values include [ERR\\_COMM\\_CHAN\\_INVALID](#), [ERR\\_COMM\\_CHAN\\_NOT\\_READY](#), [ERR\\_INVALID\\_SIZE](#), and [NO\\_ERR](#).

### 3.6.7.1.22 TCommLSRead

The TCommLSRead structure

```

// CommLSRead
TCommLSRead struct
Result     sbyte    // The function call result.

```

```

Port          byte    // The port to which the I2C device is connected.
Buffer        byte[]   // The buffer used to store the bytes read from the I2C
                     device.
BufferLen     byte    // The size of the output buffer on input. This field
                     is not updated during the function call.
TCommLSRead  ends

```

Possible Result values include [ERR\\_COMM\\_BUS\\_ERR](#), [ERR\\_COMM\\_CHAN\\_INVALID](#), [STAT\\_COMM\\_PENDING](#), [ERR\\_COMM\\_CHAN\\_NOT\\_READY](#), [ERR\\_INVALID\\_SIZE](#), and [NO\\_ERR](#).

#### 3.6.7.1.23 TCommLSCheckStatus

The TCommLSCheckStatus structure

```

// CommLSCheckStatus
TCommLSCheckStatus  struct
Result      sbyte // The function call result.
Port        byte  // The port to which the I2C device is connected.
BytesReady   byte  // The number of bytes ready to read from the specified
                     port.
TCommLSCheckStatus ends

```

Possible Result values include [ERR\\_COMM\\_BUS\\_ERR](#), [ERR\\_COMM\\_CHAN\\_INVALID](#), [STAT\\_COMM\\_PENDING](#), [ERR\\_COMM\\_CHAN\\_NOT\\_READY](#), and [NO\\_ERR](#).

#### 3.6.7.1.24 TRandomNumber

The TRandomNumber structure

```

// RandomNumber
TRandomNumber   struct
Result        sword // The random number.
TRandomNumber ends

```

#### 3.6.7.1.25 TGetStartTick

The TGetStartTick structure

```

// GetStartTick
TGetStartTick   struct
Result        dword // The returned tick value.
TGetStartTick ends

```

#### 3.6.7.1.26 TMessagewrite

The TMessagewrite structure

```

// MessageWrite
TMessagewrite   struct
Result        sbyte // The function call result. NO_ERR means it
                     succeeded.
QueueID       byte  // The queue identifier.
Message        byte[] // The message to write.
TMessagewrite ends

```

See the [Mailbox constants](#) group for QueueID values

#### 3.6.7.1.27 TMessageread

The TMessageread structure

```
// MessageRead
```

```

TMessageRead    struct
Result          sbyte // The function call result. NO_ERR means it succeeded.
QueueID         byte  // The queue identifier.
Remove          byte  // If true, remove the read message from the queue.
Message         byte[] // The contents of the mailbox/queue.
TMessageRead   ends

```

See the [Mailbox constants](#) group for QueueID values

#### 3.6.7.1.28 TCommBTCheckStatus

The TCommBTCheckStatus structure

```

// CommBTCheckStatus
TCommBTCheckStatus  struct
Result          sbyte
Connection      byte
TCommBTCheckStatus ends

```

#### 3.6.7.1.29 TCommBTWrite

The TCommBTWrite structure

```

// CommBTWrite
TCommBTWrite    struct
Result          sbyte
Connection      byte
Buffer          byte[]
TCommBTWrite   ends

```

#### 3.6.7.1.30 TCommBTRead

The TCommBTRead structure

```

// CommBTRead
TCommBTRead    struct
Result          sbyte
Count           byte
Buffer          byte[]
TCommBTRead   ends

```

#### 3.6.7.1.31 TKeepAlive

The TKeepAlive structure

```

// KeepAlive
TKeepAlive     struct
Result          dword
TKeepAlive    ends

```

#### 3.6.7.1.32 TIOMapRead

The TIOMapRead structure

```

// IOMapRead
TIOMapRead     struct
Result          sbyte
ModuleName     byte[]
Offset          word
Count           word
Buffer          byte[]
TIOMapRead    ends

```

### 3.6.7.1.33 TIOMapWrite

The TIOMapWrite structure

```
// IOMapWrite
TIOMapWrite    struct
Result        sbyte
ModuleName   byte[]
Offset        word
Buffer        byte[]
TIOMapWrite   ends
```

### 3.6.7.1.34 TInputPinFunction

The TInputPinFunction structure

```
// InputPinFunction
TInputPinFunction struct
Result word
Cmd byte
Port byte
Pin byte
Data byte
TInputPinFunction ends
```

### 3.6.7.1.35 TIOMapReadByID

The TIOMapReadByID structure

```
// IOMapReadByID
TIOMapReadByID struct
Result        sbyte
ModuleID     long
Offset        word
Count        word
Buffer        byte[]
TIOMapReadByID ends
```

### 3.6.7.1.36 TIOMapWriteByID

The TIOMapWriteByID structure

```
// IOMapWriteByID
TIOMapWriteByID struct
Result        sbyte
ModuleID     long
Offset        word
Buffer        byte[]
TIOMapWriteByID ends
```

### 3.6.7.1.37 TDisplayExecuteFunction

The TDisplayExecuteFunction structure

```
// DisplayExecuteFunction
TDisplayExecuteFunction struct
Status byte
Cmd byte
On byte
X1 byte
Y1 byte
X2 byte
Y2 byte
TDisplayExecuteFunction ends
```

### 3.6.7.1.38 TCommExecuteFunction

The TCommExecuteFunction structure

```
// CommExecuteFunction
TCommExecuteFunction struct
    Result word
    Cmd byte
    Param1 byte
    Param2 byte
    Param3 byte
    Name byte[]
    RetVal word
TCommExecuteFunction ends
```

### 3.6.7.1.39 TLoaderExecuteFunction

The TLoaderExecuteFunction structure

```
// LoaderExecuteFunction
TLoaderExecuteFunction struct
    Result word
    Cmd byte
    Filename byte[]
    Buffer byte[]
    Length long
TLoaderExecuteFunction ends
```

### 3.6.7.1.40 TFileFind

The TFileFind structure

```
// FileFindFirst, FileFindNext
TFileFind struct
    Result word
    FileHandle byte
    Filename byte[]
    Length dword
TFileFind ends
```

### 3.6.7.1.41 TCommHSControl

The TCommHSControl structure

```
// CommHSControl
TCommHSControl struct
    Result sbyte
    Command byte
    BaudRate byte
    Mode word
TCommHSControl ends
```

### 3.6.7.1.42 TCommHSCheckStatus

The TCommHSCheckStatus structure

```
// CommHSCheckStatus
TCommHSCheckStatus struct
    SendingData byte
    DataAvailable byte
TCommHSCheckStatus ends
```

### 3.6.7.1.43 TCommHSReadWrite

The TCommHSReadWrite structure

```
// CommHSRead, CommHSWrite
TCommHSReadWrite    struct
  Status sbyte
  Buffer byte[]
TCommHSReadWrite    ends
```

#### 3.6.7.1.44 TCommLSWriteEx

The TCommLSWriteEx structure

```
// CommLSWriteEx
TCommLSWriteEx    struct
  Result      sbyte
  Port       byte
  Buffer     byte[]
  ReturnLen   byte
  NoRestartOnRead byte
TCommLSWriteEx    ends
```

#### 3.6.7.1.45 TFileSeek

The TFileSeek structure

```
//FileSeek
TFileSeek    struct
  Result      word
  FileHandle  byte
  Origin      byte
  Length      sdword
TFileSeek    ends
```

#### 3.6.7.1.46 TFileResize

The TFileResize structure

```
//FileResize
TFileResize    struct
  Result      word
  FileHandle  byte
  NewSize     word
TFileResize    ends
```

#### 3.6.7.1.47 TDrawGraphicArray

The TDrawGraphicArray structure

```
// DrawGraphicArray
TDrawGraphicArray    struct
  Result      sbyte    // The function call result. NO_ERR means it
                     succeeded.
  Location    TLocation // The location on screen.
  Data        byte[]   // A byte array containing the RIC opcodes.
  Variables   sdword[] // The variables passed as RIC arguments.
  Options     dword    // The options to use when writing to the LCD.
TDrawGraphicArray    ends
```

See [Drawing option constants](#) for valid Options values.

#### 3.6.7.1.48 TDrawPolygon

The TDrawPolygon structure

```
// DrawPolygon
TDrawPolygon    struct
  Result      sbyte    // The function call result. NO_ERR means it
                     succeeded.
  Points      TLocation[] // The polygon vertices on the screen.
  Options     dword    // The options to use when writing to the LCD.
TDrawPolygon    ends
```

See [Drawing option constants](#) for valid Options values.

#### 3.6.7.1.49 TDrawEllipse

The TDrawEllipse structure

```
// DrawEllipse
TDrawEllipse    struct
Result          sbyte      // The function call result. NO_ERR means it
                   succeeded.
Center          TLocation // The location on screen.
SizeX           byte
SizeY           byte
Options         dword      // The options to use when writing to the LCD.
TDrawEllipse    ends
```

See [Drawing option constants](#) for valid Options values.

#### 3.6.7.1.50 TDrawFont

The TDrawFont structure

```
// DrawFont
TDrawFont       struct
Result          sbyte      // The function call result. NO_ERR means it
                   succeeded.
Location        TLocation // The location on screen.
Filename        byte[]     // The filename of the font.
Text            byte[]     // The text to draw on the LCD.
Options         dword      // The options to use when writing to the LCD.
TDrawFont       ends
```

See [Drawing option constants](#) for valid Options values.

#### 3.6.7.1.51 TColorSensorRead

Parameters for the ColorSensorRead system call.

This structure is used when calling the [ColorSensorRead](#) system call function. Choose the sensor port ([NBC Input port constants](#)) and after calling the function read the sensor values from the ColorValue field or the raw, normalized, or scaled value arrays.

```
// ColorSensorRead
TColorSensorRead   struct
Result          sbyte      // The function call result. NO_ERR means it
                   succeeded.
Port            byte       // The sensor port.
ColorValue      sword      // The color value returned by the sensor.
RawArray         word[]    // Raw color values returned by the sensor.
NormalizedArray word[]    // Normalized color values returned by the
                           sensor.
ScaledArray      sword[]   // Scaled color values returned by the sensor.
Invalid          byte      // Are the sensor values valid?
TColorSensorRead ends
```

See [NBC Input port constants](#) for valid Port values. See [Color values](#) for valid ColorValue values. See [Color sensor array indices](#) for array index constants used to read values from the RawArray, NormalizedArray, and ScaledArray fields.

#### 3.6.7.1.52 TDatalogWrite

The TDatalogWrite structure

```
// DatalogWrite
TDatalogWrite    struct
Result          sbyte
Message         byte[]
TDatalogWrite   ends
```

### 3.6.7.1.53 TDatalogGetTimes

The TDatalogGetTimes structure

```
// DatalogGetTimes
TDatalogGetTimes     struct
  SyncTime      dword
  SyncTick      dword
TDatalogGetTimes     ends
```

### 3.6.7.1.54 TSetSleepTimeout

The TSetSleepTimeout structure

```
// SetSleepTimeout
TSetSleepTimeout     struct
  Result        sbyte
  TheSleepTimeoutMS  dword
TSetSleepTimeout     ends
```

### 3.6.7.1.55 TCommBTOnOff

The TCommBTOnOff structure

```
// CommBTOnOff
TCommBTOnOff     struct
  Result        word
  PowerState    byte
TCommBTOnOff     ends
```

### 3.6.7.1.56 TCommBTConnection

The TCommBTConnection structure

```
// CommBTConnection
TCommBTConnection   struct
  Result        word
  Action        byte
  Name         byte []
  ConnectionSlot byte
TCommBTConnection   ends
```

### 3.6.7.1.57 TReadSemData

The TReadSemData structure

```
// ReadSemData
TReadSemData struct
  SemData      byte
  Request      byte
TReadSemData ends
```

### 3.6.7.1.58 TWriteSemData

The TWriteSemData structure

```
// WriteSemData
TWriteSemData struct
  SemData      byte
  Request      byte
  NewVal      byte
  ClearBits    byte
TWriteSemData ends
```

### 3.6.7.1.59 TUpdateCalibCacheInfo

The TUpdateCalibCacheInfo structure

```
// UpdateCalibCacheInfo
TUpdateCalibCacheInfo struct
Result byte
Name byte[]
MinVal word
MaxVal word
TUpdateCalibCacheInfo ends
```

### 3.6.7.1.60 TComputeCalibValue

The TComputeCalibValue structure

```
// ComputeCalibValue
TComputeCalibValue struct
Result byte
Name byte[]
RawVal word
TComputeCalibValue ends
```

### 3.6.7.1.61 TListFiles

The TListFiles structure

```
// ListFiles
TListFiles struct
Result sbyte
Pattern byte[]
FileList byte[][][]
TListFiles ends
```

### 3.6.7.1.62 TMemoryManager

The TMemoryManager structure

```
// MemoryManager
TMemoryManager struct
Result sbyte
Compact byte
PoolSize word
DataspaceSize word
TMemoryManager ends
```

### 3.6.7.1.63 TReadLastResponse

The TReadLastResponse structure

```
// ReadLastResponse
TReadLastResponse struct
Result sbyte
Clear byte
Length byte
Command byte
Buffer byte[]
TReadLastResponse ends
```

### 3.6.7.1.64 TFileTell

The TFileTell structure

```
// FileTell
TFileTell struct
Result sbyte
FileHandle byte
Position dword
TFileTell ends
```

### 3.6.7.1.65 TRandomEx

The TRandomEx structure

```
// RandomEx
TRandomEx      struct
Seed    long
ReSeed byte
TRandomEx      ends
```

## 3.6.8 Timing Statements

### Timing Statements

Timing statements enable you to pause the execution of a thread or obtain information about the system tick counter in your NBC programs. When using the standard NXT firmware NBC implements the wait and waitv statements as thread-specific subroutine calls due to them not being implemented. The enhanced NBC/NXC firmware implements these statements natively. If needed, you can implement simple wait loops using gettick.

```
add endTick, currTick, waitms
Loop:
    gettick currTick
brcmp LT, Loop, currTick, endTick
```

- [wait](#)
- [waitv](#)
- [wait2](#)
- [gettick](#)

### 3.6.8.1 wait

The wait Statement

The wait statement suspends the current thread for the number of milliseconds specified by its constant argument. The syntax of the wait statement is shown below.

```
wait 1000 // wait for 1 second
```

### 3.6.8.2 waitv

The waitv Statement

The waitv statement acts like wait but it takes a variable argument. If you use a constant argument with waitv the compiler will generate a temporary variable for you. The syntax of the waitv statement is shown below.

```
waitv iDelay // wait for the number of milliseconds in iDelay
```

### 3.6.8.3 wait2

The wait2 Statement

The wait2 statement suspends the current thread for the number of milliseconds specified by its second argument. If the second argument is NA then the wait time is zero milliseconds, which simply rotates the queue. The current timer value is returned in the first argument if it is not NA. The syntax of the wait2 statement is shown below.

```
set ms, 1000
wait2 NA, ms // wait for 1 second
```

### 3.6.8.4 gettick

#### The gettick Statement

The gettick statement retrieves the current system tick count. The syntax of the gettick statement is shown below.

```
gettick x // set x to the current system tick count
```

### 3.6.9 Array Statements

#### Array Statements

Array statements enable you to populate and manipulate arrays in your NBC programs.

- [NBC Programmer's Guide](#)
- [replace](#)
- [arrsize](#)
- [arrinit](#)
- [arrsubset](#)
- [arrbuilder](#)
- [arrop](#)

#### 3.6.9.1 index

##### The index Statement

The index statement extracts a single element from the source array and returns the value in the output (first) argument. The last argument is the index of the desired element. The syntax of the index statement is shown below.

```
// extract arrayValues[index] and store it in value
index value, arrayValues, index
```

#### 3.6.9.2 replace

##### The replace Statement

The replace statement replaces one or more items in a source array and stores the modified array contents in an output array. The array source argument (second) can be the same variable as the array destination (first) argument to replace without copying the array. The index of the element(s) to be replaced is specified via the third argument. The new value (last) argument can be an array, in which case multiple items are replaced. The syntax of the replace statement is shown below.

```
// replace arValues[idx] with x in arNew (arValues is unchanged)
replace arNew, arValues, idx, x
```

#### 3.6.9.3 arrsize

##### The arrsize Statement

The arrsize statement returns the number of elements in the input array (second) argument in the scalar output (first) argument. The syntax of the arrsize statement is shown below.

```
arrsize nSize, arValues // nSize == length of array
```

### 3.6.9.4 arrinit

#### The arrinit Statement

The arrinit statement initializes the output array (first) argument using the value (second) and size (third) arguments provided. The syntax of the arrinit statement is shown below.

```
// initialize arValues with nSize zeros
arrinit arValues, 0, nSize
```

### 3.6.9.5 arrsubset

#### The arrsubset Statement

The arrsubset statement copies a subset of the input array (second) argument to the output array (first) argument. The subset begins at the specified index (third) argument. The number of elements in the subset is specified using the length (fourth) argument. The syntax of the arrsubset statement is shown below.

```
// copy the first x elements to arSub
arrsubset arSub, arValues, NA, x
```

### 3.6.9.6 arrbuild

#### The arrbuild Statement

The arrbuild statement constructs an output array from a variable number of input arrays, scalars, or aggregates. The types of all the input arguments must be compatible with the type of the output array (first) argument. You must provide one or more comma-separated input arguments. The syntax of the arrbuild statement is shown below.

```
// build data array from 3 sources
arrbuild arData, arStart, arBody, arEnd
```

### 3.6.9.7 arrop

#### The arrop Statement

The arrop statement lets you perform several different operations on an array containing numeric values. The operations are [OPARR\\_SUM](#), [OPARR\\_MEAN](#), [OPARR\\_SUMSQR](#), [OPARR\\_STD](#), [OPARR\\_MIN](#), [OPARR\\_MAX](#), [OPARR\\_SORT](#), [OPARR\\_TOUPPER](#), and [OPARR\\_TOLOWER](#).

In the case of [OPARR\\_SORT](#) the output parameter should be an array of the same type as the input array. With [OPARR\\_TOUPPER](#) and [OPARR\\_TOLOWER](#) the input and output parameters should both be strings.

In all the other cases the output parameter can be any scalar type large enough to hold the resulting value. If the data in the array is of the float type then the output should also be of the float type.

The fourth and fifth arguments indicate the starting element and the number of elements to operate on. To use the entire input array you simply pass the [NA](#) constant in as the value for start and length. The syntax of the arrop statement is shown below.

```
// execute an array operation
// arrop op, dest, src, start, len
arrop OPARR_SUM, sum, data, NA, NA
arrop OPARR_SORT, data2, data, NA, NA
arrop OPARR_TOLOWER, str_out, str_in, NA, NA
```

## 3.6.10 String Statements

### String Statements

String statements enable you to populate and manipulate null-terminated byte arrays (aka strings) in your NBC programs.

- [flatten](#)
- [unflatten](#)
- [numtostr](#)
- [fmtnum](#)
- [strtonum](#)
- [strsubset](#)
- [strcat](#)
- [arrtostr](#)
- [strtoarr](#)
- [strindex](#)
- [strreplace](#)
- [strlen](#)

#### 3.6.10.1 flatten

##### The flatten Statement

The flatten statement converts its input (second) argument into its string output (first) argument. The syntax of the flatten statement is shown below.

```
flatten strData, args // copy args structure to strData
```

#### 3.6.10.2 unflatten

##### The unflatten Statement

The unflatten statement converts its input string (third) argument to the output (first) argument type. If the default value (fourth) argument type does not match the flattened data type exactly, including array sizes, then error output (second) argument will be set to TRUE and the output argument will contain a copy of the default argument. The syntax of the unflatten statement is shown below.

```
unflatten args, bErr, strSource, x // convert string to cluster
```

#### 3.6.10.3 numtostr

##### The numtostr Statement

The numtostr statement converts its scalar input (second) argument to a string output (first) argument. The syntax of the numtostr statement is shown below.

```
numtostr strValue, value // convert value to a string
```

#### 3.6.10.4 fmtnum

##### The fmtnum Statement

The fmtnum statement converts its scalar input (third) argument to a string output (first) argument. The format of the string output is specified via the format string (second) argument. The syntax of the fmtnum statement is shown below.

```
fmtnum strValue, fmtStr, value // convert value to a string
```

### 3.6.10.5 strtonum

#### The strtonum Statement

The strtonum statement parses its input string (third) argument into a numeric output (first) argument, advancing an offset output (second) argument past the numeric string. The initial input offset (fourth) argument determines where the string parsing begins. The default (fifth) argument is the value that is returned by the statement if an error occurs while parsing the string. The syntax of the strtonum statement is shown below.

```
// parse string into num
strtonum value, idx, strValue, idx, nZero
```

### 3.6.10.6 strsubset

#### The strsubset Statement

The strsubset statement copies a subset of the input string (second) argument to the output string (first) argument. The subset begins at the specified index (third) argument. The number of characters in the subset is specified using the length (fourth) argument. The syntax of the strsubset statement is shown below.

```
// copy the first x characters in strSource to strSub
strsubset strSub, strSource, NA, x
```

### 3.6.10.7 strcat

#### The strcat Statement

The strcat statement constructs an output string from a variable number of input strings. The input arguments must all be null-terminated byte arrays. You must provide one or more comma-separated input arguments. The syntax of the strcat statement is shown below.

```
// build data string from 3 sources
strcat strData, strStart, strBody, strEnd
```

### 3.6.10.8 arrtostr

#### The arrtostr Statement

The arrtostr statement copies the input byte array (second) argument into its output string (first) argument and adds a null-terminator byte at the end. The syntax of the arrtostr statement is shown below.

```
arrtostr strData, arrData // convert byte array to string
```

### 3.6.10.9 strtoarr

#### The strtoarr Statement

The strtoarr statement copies the input string (second) argument into its output byte array (first) argument excluding the last byte, which should be a null. The syntax of the strtoarr statement is shown below.

```
strtoarr arrData, strData // convert string to byte array
```

### 3.6.10.10 strindex

#### The strindex Statement

The strindex statement extracts a single element from the source string and returns the value in the output (first) argument. The last argument is the index of the desired element. The syntax of the strindex statement is shown below.

```
// extract strVal[idx] and store it in val
strindex val, strVal, idx
```

### 3.6.10.11 strreplace

#### The strreplace Statement

The strreplace statement replaces one or more characters in a source string and stores the modified string in an output string. The string source argument (second) can be the same variable as the string destination (first) argument to replace without copying the string. The index of the character(s) to be replaced is specified via the third argument. The new value (fourth) argument can be a string, in which case multiple characters are replaced. The syntax of the strreplace statement is shown below.

```
// replace strValues[idx] with newStr in strNew
strreplace strNew, strValues, idx, newStr
```

### 3.6.10.12 strlen

#### The strlen Statement

The strlen statement returns the length of the input string (second) argument in the scalar output (first) argument. The syntax of the strlen statement is shown below.

```
strlen nSize, strMsg // nSize == length of strMsg
```

## 3.6.11 Scheduling Statements

### Scheduling Statements

Scheduling statements enable you to control the execution of multiple threads and the calling of subroutines in your NBC programs.

- [exit](#)
- [exitto](#)
- [exittovar](#)
- [start](#)
- [startvar](#)
- [stopthread](#)
- [stopthreadvar](#)
- [priority](#)
- [precedes](#)
- [follows](#)
- [acquire](#)
- [release](#)
- [subcall](#)
- [subcallvar](#)
- [subret](#)
- [call](#)
- [callvar](#)
- [return](#)

### 3.6.11.1 exit

#### The exit Statement

The exit statement finalizes the current thread and schedules zero or more dependant threads by specifying start and end dependency list indices. The thread indices are zero-based and inclusive. The two arguments are optional, in which case the compiler automatically adds indices for all the dependencies. The syntax of the exit statement is shown below.

```
exit 0, 2 // schedule this thread's 3 dependants  
exit // schedule all this thread's dependants
```

### 3.6.11.2 exitto

#### The exitto Statement

The exitto statement exits the current thread and schedules the specified thread to begin executing. The syntax of the exitto statement is shown below.

```
exitto worker // exit now and schedule worker thread
```

### 3.6.11.3 exittovar

#### The exittovar Statement

The exittovar statement exits the current thread and schedules the specified thread to begin executing. The syntax of the exittovar statement is shown below.

```
exittovar var // exit now and schedule another thread specified by var
```

### 3.6.11.4 start

#### The start Statement

The start statement causes the thread specified in the statement to start running immediately. Using the standard NXT firmware this statement is implemented by the compiler using a set of compiler-generated subroutines. The enhanced NBC/NXC firmware implements this statement natively. The syntax of the start statement is shown below.

```
start worker // start the worker thread
```

### 3.6.11.5 startvar

#### The startvar Statement

The startvar statement causes the thread specified in the statement to start running immediately. Using the standard NX-T firmware this statement is implemented by the compiler using a set of compiler-generated subroutines. The enhanced NBC/NXC firmware implements this statement natively. The syntax of the startvar statement is shown below.

```
startvar var // start another thread specified by var
```

### 3.6.11.6 stopthread

#### The stopthread Statement

The stopthread statement causes the thread specified in the statement to stop running immediately. This statement cannot be used with the standard NXT firmware. It is supported by the enhanced NBC/NXC firmware. The syntax of the stopthread statement is shown below.

```
stopthread worker // stop the worker thread
```

### 3.6.11.7 stopthreadvar

The stopthreadvar Statement

The stopthreadvar statement causes the thread specified in the statement to stop running immediately. This statement cannot be used with the standard NXT firmware. It is supported by the enhanced NBC/NXC firmware. The syntax of the stopthreadvar statement is shown below.

```
stopthreadvar var // stop the thread specified by var
```

### 3.6.11.8 priority

The priority Statement

The priority statement modifies the priority of the thread specified in the statement. This statement cannot be used with the standard NXT firmware. It is supported by the enhanced NBC/NXC firmware. The syntax of the priority statement is shown below.

```
priority worker, 50 // change the priority of the worker thread
```

### 3.6.11.9 precedes

The precedes Statement

The precedes statement causes the compiler to mark the threads listed in the statement as dependants of the current thread. A subset of these threads will begin executing once the current thread exits, depending on the form of the exit statement used at the end of the current thread. The syntax of the precedes statement is shown below.

```
precedes worker, music, walking // configure dependant threads
```

### 3.6.11.10 follows

The follows Statement

The follows statement causes the compiler to mark the current thread as a dependant of the threads listed in the statement. The current thread will be scheduled to execute if all of the threads that precede it have exited and scheduled it for execution. The syntax of the follows statement is shown below.

```
follows main // configure thread dependencies
```

### 3.6.11.11 acquire

The acquire Statement

The acquire statement acquires the named mutex. If the mutex is already acquired the current thread waits until it becomes available. The syntax of the acquire statement is shown below.

```
acquire muFoo // acquire mutex for subroutine
```

### 3.6.11.12 release

The release Statement

The release statement releases the named mutex allowing other threads to acquire it. The syntax of the release statement is shown below.

```
release muFoo // release mutex for subroutine
```

### 3.6.11.13 subcall

#### The subcall Statement

The subcall statement calls into the named thread/subroutine and waits for a return (which might not come from the same thread). The second argument is a variable used to store the return address. The syntax of the subcall statement is shown below.

```
subcall drawText, retDrawText // call drawText subroutine
```

### 3.6.11.14 subcallvar

#### The subcallvar Statement

The subcallvar statement calls into the named thread/subroutine and waits for a return (which might not come from the same thread). The second argument is a variable used to store the return address. The syntax of the subcallvar statement is shown below.

```
subcallvar var, retDrawText // call the subroutine specified by var
```

### 3.6.11.15 subret

#### The subret Statement

The subret statement returns from a thread to the return address value contained in its input argument. The syntax of the subret statement is shown below.

```
subret retDrawText // return to calling routine
```

### 3.6.11.16 call

#### The call Statement

The call statement executes the named subroutine and waits for a return. The argument should specify a thread that was declared using the subroutine keyword. The syntax of the call statement is shown below.

```
call MyFavoriteSubroutine // call routine
```

### 3.6.11.17 callvar

#### The callvar Statement

The callvar statement executes the named subroutine and waits for a return. The argument should specify a thread that was declared using the subroutine keyword. The syntax of the callvar statement is shown below.

```
callvar var // call the routine specified by var
```

### 3.6.11.18 return

#### The return Statement

The return statement returns from a subroutine. The compiler automatically handles the return address for call and return when they are used with subroutines rather than threads. The syntax of the return statement is shown below.

```
return // return to calling routine
```

### 3.6.12 Input Statements

#### Input Statements

Input statements enable you to configure the four input ports and read analog sensor values in your NBC programs. Both statements in this category use input field identifiers to control which attribute of the input port you are manipulating.

- [setin](#)
- [getin](#)

##### 3.6.12.1 setin

###### The setin Statement

The setin statement sets an input field of a sensor on a port to the value specified in its first argument. The port is specified via the second argument. The input field identifier is the third argument. Valid input field identifiers are listed in the [Input field constants](#) section. Valid port constants are listed in the [NBC Input port constants](#) section. The syntax of the setin statement is shown below.

```
setin IN_TYPE_SWITCH, IN_1, TypeField // set sensor
      to switch type
setin IN_MODE_BOOLEAN, IN_1, InputModeField //
      set to boolean mode
```

##### 3.6.12.2 getin

###### The getin Statement

The getin statement reads a value from an input field of a sensor on a port and writes the value to its first argument. The port is specified via the second argument. The input field identifier is the third argument. Valid input field identifiers are listed in the [Input field constants](#) section. Valid port constants are listed in the [NBC Input port constants](#) section. The syntax of the getin statement is shown below.

```
getin rVal, thePort, RawValue // read raw sensor value
getin sVal, thePort, ScaledValue // read scaled sensor value
getin nVal, thePort, NormalizedValue // read normalized value
```

### 3.6.13 Output Statements

#### Output Statements

Output statements enable you to configure and control the three NXT outputs in your NBC programs. Both statements in this category use output field identifiers to control which attribute of the output you are manipulating.

- [setout](#)
- [getout](#)

##### 3.6.13.1 setout

###### The setout Statement

The setout statement sets one or more output fields of a motor on one or more ports to the value specified by the coupled input arguments. The first argument is either a scalar value specifying a single port or a byte array specifying multiple ports. After the port argument you then provide one or more pairs of output field identifiers and values. You can set multiple fields via a single statement. Valid output field identifiers are listed in the [Output field constants](#) section. Valid output port constants are listed in the [Output port constants](#) section. The syntax of the setout statement is shown below.

```

set theMode, OUT_MODE_MOTORON // set mode to motor on
set rsVal, OUT_RUNSTATE_RUNNING // motor running
set thePort, OUT_A // set port to #1
set pwr, -75 // negative power means reverse motor direction
// set output values
setout thePort, OutputModeField, theMode, RunStateField
, rsVal, PowerField, pwr

```

### 3.6.13.2 getout

#### The getout Statement

The getout statement reads a value from an output field of a sensor on a port and writes the value to its first output argument. The port is specified via the second argument. The output field identifier is the third argument. Valid output field identifiers are listed in the [Output field constants](#) section. Valid output port constants are listed in the [Output port constants](#) section. The syntax of the getout statement is shown below.

```

getout rmVal, thePort, RegModeField // read motor regulation mode
getout tlVal, thePort, TachoLimitField // read tachometer limit
    value
getout rcVal, thePort, RotationCountField // read the
    rotation count

```

### 3.6.14 Compile-time Statements

#### Compile-time Statements

Compile-time statements and functions enable you to perform simple compiler operations at the time you compile your NBC programs.

- [sizeof](#)
- [valueof](#)
- [isconst](#)
- [compchk](#)
- [compif](#)
- [compelse](#)
- [compend](#)
- [compchktype](#)

#### 3.6.14.1 sizeof

##### The sizeof function

The `sizeof(arg)` compiler function returns the size of the variable you pass into it. The syntax of the `sizeof` function is shown below.

```

dseg segment
    arg byte
    argsize byte
dseg ends
// ...
set argsize, sizeof(arg) // argsize == 1

```

### 3.6.14.2 valueof

The valueof function

The valueof(arg) compiler function returns the value of the constant expression you pass into it. The syntax of the valueof function is shown below.

```
set argval, valueof(4+3*2) // argval == 10
```

### 3.6.14.3 isconst

The isconst function

The isconst(arg) compiler function returns TRUE if the argument you pass into it is a constant and FALSE if it is not a constant. The syntax of the isconst function is shown below.

```
set argval, isconst(4+3*2) // argval == TRUE
```

### 3.6.14.4 compchk

The compchk Statement

The compchk compiler statement takes a comparison constant as its first argument. The second and third arguments must be constants or constant expressions that can be evaluated by the compiler during program compilation. It reports a compiler error if the comparison expression does not evaluate to TRUE. Valid comparison constants are listed in the [Comparison Constants](#) section. The syntax of the compchk statement is shown below.

```
compchk EQ, sizeof(arg3), 2
```

### 3.6.14.5 compif

The compif Statement

The compif statement works together with the compelse and comprehend compiler statements to create a compile-time if-else statement that enables you to control whether or not sections of code should be included in the compiler output. The compif statement takes a comparison constant as its first argument. The second and third arguments must be constants or constant expressions that can be evaluated by the compiler during program compilation. If the comparison expression is true then code immediate following the statement will be included in the executable. The compiler if statement ends when the compiler finds the next comprehend statement. To optionally provide an else clause use the compelse statement between the compif and comprehend statements. Valid comparison constants are listed in the [Comparison Constants](#) section. The syntax of the compif statement is demonstrated in the example below.

```
compif EQ, sizeof(arg3), 2
// compile this if sizeof(arg3) == 2
compelse
// compile this if sizeof(arg3) != 2
comprend
```

### 3.6.14.6 compelse

The compelse Statement

The compelse statement works together with the compif and comprehend compiler statements to create a compile-time if-else statement that enables you to control whether or not sections of code should be included in the compiler output. If the comparison expression in the compif statement is false then code immediately following the compelse statement will be included in the executable. The compelse block ends when the compiler finds the next comprehend statement. The syntax of the compelse statement is shown in the example below.

```
compif EQ, sizeof(arg3), 2
// compile this if sizeof(arg3) == 2
compelse
// compile this if sizeof(arg3) != 2
comprend
```

### 3.6.14.7 compend

The compend Statement

The compend statement works together with the compif and compelse compiler statements to create a compile-time if-else statement that enables you to control whether or not sections of code should be included in the compiler output. The compif and compelse blocks end when the compiler finds the next compend statement. The syntax of the compend statement is shown in the example below.

```
compif EQ, sizeof(arg3), 2
    // compile this if sizeof(arg3) == 2
compelse
    // compile this if sizeof(arg3) != 2
compend
```

### 3.6.14.8 compchktype

The compchktype Statement

The compchktype compiler statement takes a variable as its first argument. The second argument must be type name that can be evaluated by the compiler during program compilation. It reports a compiler error if the type of the variable does not match the second argument. The syntax of the compchktype statement is shown below.

```
compchktype _args, TDrawText
syscall DrawText, _args
```

## 4 TXGPacket

The TXGPacket structure

```
// XGPacket
TXGPacket struct
    AccAngle sword
    TurnRate sword
    XAxis sword
    YAxis sword
    ZAxis sword
TXGPacket ends
```

## 5 Module Documentation

### 5.1 Comparison Constants

Logical comparison operators for use in brtst, tst, tstset, brcmp, cmp, and cmpset.

#### Macros

- #define LT 0x00
- #define GT 0x01
- #define LTEQ 0x02
- #define GTEQ 0x03
- #define EQ 0x04
- #define NEQ 0x05

### 5.1.1 Detailed Description

Logical comparison operators for use in brtst, tst, tstset, brcmp, cmp, and cmpset.

### 5.1.2 Macro Definition Documentation

#### 5.1.2.1 #define EQ 0x04

The first value is equal to the second.

#### 5.1.2.2 #define GT 0x01

The first value is greater than the second.

#### 5.1.2.3 #define GTEQ 0x03

The first value is greater than or equal to the second.

#### 5.1.2.4 #define LT 0x00

The first value is less than the second.

#### 5.1.2.5 #define LTEQ 0x02

The first value is less than or equal to the second.

#### 5.1.2.6 #define NEQ 0x05

The first value is not equal to the second.

## 5.2 NXT Firmware Modules

Documentation common to all NXT firmware modules.

### Modules

- [Button module](#)  
*Constants and functions related to the Button module.*
- [Comm module](#)  
*Constants and functions related to the Comm module.*
- [Command module](#)  
*Constants and functions related to the Command module.*
- [Display module](#)  
*Constants and functions related to the Display module.*
- [IOCtrl module](#)  
*Constants and functions related to the IOCtrl module.*
- [Input module](#)  
*Constants and functions related to the Input module.*
- [Loader module](#)  
*Constants and functions related to the Loader module.*
- [Low Speed module](#)  
*Constants and functions related to the Low Speed module.*
- [NXT firmware module IDs](#)  
*Constant numeric IDs for all the NXT firmware modules.*
- [NXT firmware module names](#)  
*Constant string names for all the NXT firmware modules.*
- [Output module](#)  
*Constants and functions related to the Output module.*
- [Sound module](#)  
*Constants and functions related to the Sound module.*
- [Ui module](#)  
*Constants and functions related to the Ui module.*

### 5.2.1 Detailed Description

Documentation common to all NXT firmware modules.

## 5.3 Input module

Constants and functions related to the Input module.

### Modules

- [Input module constants](#)

*Constants that are part of the NXT firmware's Input module.*

- [Input module functions](#)

*Functions for accessing and modifying input module features.*

#### 5.3.1 Detailed Description

Constants and functions related to the Input module. The NXT input module encompasses all sensor inputs except for digital I2C (LowSpeed) sensors.

There are four sensors, which internally are numbered 0, 1, 2, and 3. This is potentially confusing since they are externally labeled on the NXT as sensors 1, 2, 3, and 4. To help mitigate this confusion, the NBC port name constants [IN\\_1](#), [IN\\_2](#), [IN\\_3](#), and [IN\\_4](#) may be used when a sensor port is required. See [NBC Input port constants](#).

## 5.4 Input module constants

Constants that are part of the NXT firmware's Input module.

### Modules

- [Color calibration constants](#)  
*Constants for use with the color calibration functions.*
- [Color calibration state constants](#)  
*Constants for use with the color calibration state function.*
- [Color sensor array indices](#)  
*Constants for use with color sensor value arrays to index RGB and blank return values.*
- [Color values](#)  
*Constants for use with the ColorValue returned by the color sensor in full color mode.*
- [Constants to use with the Input module's Pin function](#)  
*Constants for use with the Input module's Pin function.*
- [Input field constants](#)  
*Constants for use with SetInput() and GetInput().*
- [Input module IOMAP offsets](#)  
*Constant offsets into the Input module IOMAP structure.*
- [Input port digital pin constants](#)  
*Constants for use when directly controlling or reading a port's digital pin state.*
- [NBC Input port constants](#)  
*Input port constants are used when calling sensor control API functions.*
- [Sensor types and modes](#)  
*Constants that are used for defining sensor types and modes.*

### Macros

- #define INPUT\_CUSTOMINACTIVE 0x00
- #define INPUT\_CUSTOM9V 0x01
- #define INPUT\_CUSTOMACTIVE 0x02
- #define INPUT\_INVALID\_DATA 0x01

#### 5.4.1 Detailed Description

Constants that are part of the NXT firmware's Input module.

#### 5.4.2 Macro Definition Documentation

##### 5.4.2.1 #define INPUT\_CUSTOM9V 0x01

Custom sensor 9V

##### 5.4.2.2 #define INPUT\_CUSTOMACTIVE 0x02

Custom sensor active

5.4.2.3 #define INPUT\_CUSTOMINACTIVE 0x00

Custom sensor inactive

5.4.2.4 #define INPUT\_INVALID\_DATA 0x01

Invalid data flag

## 5.5 Sensor types and modes

Constants that are used for defining sensor types and modes.

### Modules

- [NBC sensor mode constants](#)

*Use sensor mode constants to configure an input port for the desired sensor mode.*

- [NBC sensor type constants](#)

*Use sensor type constants to configure an input port for a specific type of sensor.*

### 5.5.1 Detailed Description

Constants that are used for defining sensor types and modes. The sensor ports on the NXT are capable of interfacing to a variety of different sensors. It is up to the program to tell the NXT what kind of sensor is attached to each port. Calling [SetSensorType](#) configures a sensor's type. There are 16 sensor types, each corresponding to a specific type of LEGO RCX or NXT sensor. Two of these types are for NXT I2C digital sensors, either 9V powered or unpowered, and a third is used to configure port [IN\\_4](#) as a high-speed RS-485 serial port. A seventeenth type ([IN\\_TYPE\\_CUSTOM](#)) is for use with custom analog sensors. And an eighteenth type ([IN\\_TYPE\\_NO\\_SENSOR](#)) is used to indicate that no sensor has been configured, effectively turning off the specified port.

In general, a program should configure the type to match the actual sensor. If a sensor port is configured as the wrong type, the NXT may not be able to read it accurately. Use the [NBC sensor type constants](#).

The NXT allows a sensor to be configured in different modes. The sensor mode determines how a sensor's raw value is processed. Some modes only make sense for certain types of sensors, for example [IN\\_MODE\\_ANGLESTEP](#) is useful only with rotation sensors. Call [SetSensorMode](#) to set the sensor mode. The possible modes are shown below. Use the [NBC sensor mode constants](#).

The NXT provides a boolean conversion for all sensors - not just touch sensors. This boolean conversion is normally based on preset thresholds for the raw value. A "low" value (less than 460) is a boolean value of 1. A high value (greater than 562) is a boolean value of 0. This conversion can be modified: a slope value between 0 and 31 may be added to a sensor's mode when calling [SetSensorMode](#). If the sensor's value changes more than the slope value during a certain time (3ms), then the sensor's boolean state will change. This allows the boolean state to reflect rapid changes in the raw value. A rapid increase will result in a boolean value of 0, a rapid decrease is a boolean value of 1.

Even when a sensor is configured for some other mode (i.e. [IN\\_MODE\\_PCTFULLSCALE](#)), the boolean conversion will still be carried out.

## 5.6 Output module

Constants and functions related to the Output module.

### Modules

- [Output module constants](#)

*Constants that are part of the NXT firmware's Output module.*

- [Output module functions](#)

*Functions for accessing and modifying output module features.*

#### 5.6.1 Detailed Description

Constants and functions related to the Output module. The NXT output module encompasses all the motor outputs.

Nearly all of the NBC API functions dealing with outputs take either a single output or a set of outputs as their first argument. Depending on the function call, the output or set of outputs may be a constant or a variable containing an appropriate output port value. The constants [OUT\\_A](#), [OUT\\_B](#), and [OUT\\_C](#) are used to identify the three outputs. Unlike NQC, adding individual outputs together does not combine multiple outputs. Instead, the NBC API provides predefined combinations of outputs: [OUT\\_AB](#), [OUT\\_AC](#), [OUT\\_BC](#), and [OUT\\_ABC](#). Manually combining outputs involves creating an array and adding two or more of the three individual output constants to the array.

Output power levels can range 0 (lowest) to 100 (highest). Negative power levels reverse the direction of rotation (i.e., forward at a power level of -100 actually means reverse at a power level of 100).

The outputs each have several fields that define the current state of the output port. These fields are defined in the [Output field constants](#) section.

## 5.7 Output module constants

Constants that are part of the NXT firmware's Output module.

### Modules

- [Output field constants](#)  
*Constants for use with SetOutput() and GetOutput().*
- [Output module IOMAP offsets](#)  
*Constant offsets into the Output module IOMAP structure.*
- [Output port constants](#)  
*Output port constants are used when calling motor control API functions.*
- [Output port mode constants](#)  
*Use these constants to configure the desired mode for the specified motor(s): coast, motoron, brake, or regulated.*
- [Output port option constants](#)  
*Use these constants to configure the desired options for the specified motor(s): hold at limit and ramp down to limit.*
- [Output port regulation mode constants](#)  
*Use these constants to configure the desired regulation mode for the specified motor(s): none, speed regulation, multi-motor synchronization, or position regulation (requires the enhanced NBC/NXC firmware version 1.31+).*
- [Output port run state constants](#)  
*Use these constants to configure the desired run state for the specified motor(s): idle, rampup, running, rampdown, or hold.*
- [Output port update flag constants](#)  
*Use these constants to specify which motor values need to be updated.*
- [Output regulation option constants](#)  
*Use these constants to configure the desired options for position regulation.*
- [PID constants](#)  
*PID constants are for adjusting the Proportional, Integral, and Derivative motor controller parameters.*
- [Tachometer counter reset flags](#)  
*Use these constants to specify which of the three tachometer counters should be reset.*

### 5.7.1 Detailed Description

Constants that are part of the NXT firmware's Output module.

## 5.8 Command module

Constants and functions related to the Command module.

### Modules

- [Command module constants](#)

*Constants that are part of the NXT firmware's Command module.*

- [Command module functions](#)

*Functions for accessing and modifying Command module features.*

#### 5.8.1 Detailed Description

Constants and functions related to the Command module. The NXT command module encompasses support for the execution of user programs via the NXT virtual machine. It also implements the direct command protocol support that enables the NXT to respond to USB or Bluetooth requests from other devices such as a PC or another NXT brick.

## 5.9 Command module constants

Constants that are part of the NXT firmware's Command module.

### Modules

- [Array operation constants](#)

*Constants for use with the NXC ArrayOp function and the NBC arrop statement.*

- [Command module IOMAP offsets](#)

*Constant offsets into the Command module IOMAP structure.*

- [Communications specific errors](#)

*Constants defining communication error conditions.*

- [Fatal errors](#)

*Constants defining various fatal error conditions.*

- [General errors](#)

*Constants defining general error conditions.*

- [Program status constants](#)

*Constants defining various states of the command module virtual machine.*

- [Remote control \(direct commands\) errors](#)

*Constants defining errors that can occur during remote control (RC) direct command operations.*

- [System Call function constants](#)

*Constants for use in the SysCall() function or NBC syscall statement.*

- [Time constants](#)

*Constants for use with the Wait() function.*

- [VM state constants](#)

*Constants defining possible VM states.*

### Macros

- #define STAT\_MSG\_EMPTY\_MAILBOX 64
- #define STAT\_COMM\_PENDING 32
- #define POOL\_MAX\_SIZE 32768
- #define NO\_ERR 0

#### 5.9.1 Detailed Description

Constants that are part of the NXT firmware's Command module.

#### 5.9.2 Macro Definition Documentation

##### 5.9.2.1 #define NO\_ERR 0

Successful execution of the specified command

##### 5.9.2.2 #define POOL\_MAX\_SIZE 32768

Maximum size of memory pool, in bytes

**5.9.2.3 #define STAT\_COMM\_PENDING 32**

Pending setup operation in progress

**5.9.2.4 #define STAT\_MSG\_EMPTY\_MAILBOX 64**

Specified mailbox contains no new messages

## 5.10 Comm module

Constants and functions related to the Comm module.

### Modules

- [Comm module constants](#)

*Constants that are part of the NXT firmware's Comm module.*

- [Comm module functions](#)

*Functions for accessing and modifying Comm module features.*

### 5.10.1 Detailed Description

Constants and functions related to the Comm module. The NXT comm module encompasses support for all forms of Bluetooth, USB, and HiSpeed communication.

You can use the Bluetooth communication methods to send information to other devices connected to the NXT brick. The NXT firmware also implements a message queuing or mailbox system which you can access using these methods.

Communication via Bluetooth uses a master/slave connection system. One device must be designated as the master device before you run a program using Bluetooth. If the NXT is the master device then you can configure up to three slave devices using connection 1, 2, and 3 on the NXT brick. If your NXT is a slave device then connection 0 on the brick must be reserved for the master device.

Programs running on the master NXT brick can send packets of data to any connected slave devices using the Bluetooth-Write method. Slave devices write response packets to the message queuing system where they wait for the master device to poll for the response.

Using the direct command protocol, a master device can send messages to slave NXT bricks in the form of text strings addressed to a particular mailbox. Each mailbox on the slave NXT brick is a circular message queue holding up to five messages. Each message can be up to 58 bytes long.

To send messages from a master NXT brick to a slave brick, use BluetoothWrite on the master brick to send a Message-Write direct command packet to the slave. Then, you can use ReceiveMessage on the slave brick to read the message. The slave NXT brick must be running a program when an incoming message packet is received. Otherwise, the slave NXT brick ignores the message and the message is dropped.

## 5.11 Button module

Constants and functions related to the Button module.

### Modules

- [Button module constants](#)

*Constants that are part of the NXT firmware's Button module.*

- [Button module functions](#)

*Functions for accessing and modifying Button module features.*

### 5.11.1 Detailed Description

Constants and functions related to the Button module. The NXT button module encompasses support for the 4 buttons on the NXT brick.

## 5.12 IOCtrl module

Constants and functions related to the IOCtrl module.

### Modules

- [IOCtrl module constants](#)

*Constants that are part of the NXT firmware's IOCtrl module.*

- [IOCtrl module functions](#)

*Functions for accessing and modifying IOCtrl module features.*

### 5.12.1 Detailed Description

Constants and functions related to the IOCtrl module. The NXT ioctl module encompasses low-level communication between the two processors that control the NXT. The NBC API exposes two functions that are part of this module.

## 5.13 Loader module

Constants and functions related to the Loader module.

### Modules

- [Loader module constants](#)

*Constants that are part of the NXT firmware's Loader module.*

- [Loader module functions](#)

*Functions for accessing and modifying Loader module features.*

### 5.13.1 Detailed Description

Constants and functions related to the Loader module. The NXT loader module encompasses support for the NXT file system. The NXT supports creating files, opening existing files, reading, writing, renaming, and deleting files.

Files in the NXT file system must adhere to the 15.3 naming convention for a maximum filename length of 19 characters. While multiple files can be opened simultaneously, a maximum of 4 files can be open for writing at any given time.

When accessing files on the NXT, errors can occur. The NBC API defines several constants that define possible result codes. They are listed in the [Loader module error codes](#) section.

## 5.14 Sound module

Constants and functions related to the Sound module.

### Modules

- [Sound module constants](#)

*Constants that are part of the NXT firmware's Sound module.*

- [Sound module functions](#)

*Functions for accessing and modifying sound module features.*

### 5.14.1 Detailed Description

Constants and functions related to the Sound module. The NXT sound module encompasses all sound output features. The NXT provides support for playing basic tones as well as two different types of files.

Sound files (.rso) are like .wav files. They contain thousands of sound samples that digitally represent an analog waveform. With sounds files the NXT can speak or play music or make just about any sound imaginable.

Melody files are like MIDI files. They contain multiple tones with each tone being defined by a frequency and duration pair. When played on the NXT a melody file sounds like a pure sine-wave tone generator playing back a series of notes. While not as fancy as sound files, melody files are usually much smaller than sound files.

When a sound or a file is played on the NXT, execution of the program does not wait for the previous playback to complete. To play multiple tones or files sequentially it is necessary to wait for the previous tone or file playback to complete first. This can be done via the [wait](#) statement or by using the sound state value within a while loop.

The NBC API defines frequency and duration constants which may be used in calls to [PlayTone](#) or [PlayToneEx](#). Frequency constants start with [TONE\\_A3](#) (the 'A' pitch in octave 3) and go to [TONE\\_B7](#) (the 'B' pitch in octave 7). Duration constants start with [MS\\_1](#) (1 millisecond) and go up to [MIN\\_1](#) (60000 milliseconds) with several constants in between. See [NBCCommon.h](#) for the complete list.

## 5.15 Ui module

Constants and functions related to the Ui module.

### Modules

- [Ui module constants](#)

*Constants that are part of the NXT firmware's Ui module.*

- [Ui module functions](#)

*Functions for accessing and modifying Ui module features.*

### 5.15.1 Detailed Description

Constants and functions related to the Ui module. The NXT UI module encompasses support for various aspects of the user interface for the NXT brick.

## 5.16 Low Speed module

Constants and functions related to the Low Speed module.

### Modules

- [LowSpeed module constants](#)

*Constants that are part of the NXT firmware's LowSpeed module.*

- [LowSpeed module functions](#)

*Functions for accessing and modifying low speed module features.*

### 5.16.1 Detailed Description

Constants and functions related to the Low Speed module. The NXT low speed module encompasses support for digital I2C sensor communication.

Use the `lowspeed` (aka I2C) communication methods to access devices that use the I2C protocol on the NXT brick's four input ports.

You must set the input port's `TypeField` property to `IN_TYPE_LOWSPEED` or `IN_TYPE_LOWSPEED_9V` on a given port before using an I2C device on that port. Use `IN_TYPE_LOWSPEED_9V` if your device requires 9V power from the NXT brick. Remember that you also need to set the input port's `InvalidDataField` property to true after setting a new `TypeField`, and then wait in a loop for the NXT firmware to set `InvalidDataField` back to false. This process ensures that the firmware has time to properly initialize the port, including the 9V power lines, if applicable. Some digital devices might need additional time to initialize after power up.

The `SetSensorLowspeed` API function sets the specified port to `IN_TYPE_LOWSPEED_9V` and calls `ResetSensor` to perform the `InvalidDataField` reset loop described above.

When communicating with I2C devices, the NXT firmware uses a master/slave setup in which the NXT brick is always the master device. This means that the firmware is responsible for controlling the write and read operations. The NXT firmware maintains write and read buffers for each port, and the three main Lowspeed (I2C) methods described below enable you to access these buffers.

A call to `LowspeedWrite` starts an asynchronous transaction between the NXT brick and a digital I2C device. The program continues to run while the firmware manages sending bytes from the write buffer and reading the response bytes from the device. Because the NXT is the master device, you must also specify the number of bytes to expect from the device in response to each write operation. You can exchange up to 16 bytes in each direction per transaction.

After you start a write transaction with `LowspeedWrite`, use `LowspeedStatus` in a loop to check the status of the port. If `LowspeedStatus` returns a status code of 0 and a count of bytes available in the read buffer, the system is ready for you to use `LowspeedRead` to copy the data from the read buffer into the buffer you provide.

Note that any of these calls might return various status codes at any time. A status code of 0 means the port is idle and the last transaction (if any) did not result in any errors. Negative status codes and the positive status code 32 indicate errors. There are a few possible errors per call.

Valid low speed return values include `NO_ERR` as well as the error codes listed in the [Communications specific errors](#) section.

## 5.17 Display module

Constants and functions related to the Display module.

### Modules

- [Display module constants](#)

*Constants that are part of the NXT firmware's Display module.*

- [Display module functions](#)

*Functions for accessing and modifying display module features.*

### 5.17.1 Detailed Description

Constants and functions related to the Display module. The NXT display module encompasses support for drawing to the NXT LCD. The NXT supports drawing points, lines, rectangles, and circles on the LCD. It supports drawing graphic icon files on the screen as well as text and numbers. With the enhanced NBC/NXC firmware you can also draw ellipses and polygons as well as text and numbers using custom RIC-based font files. Also, all of the drawing operations have several drawing options for how the shapes are drawn to the LCD.

The LCD screen has its origin (0, 0) at the bottom left-hand corner of the screen with the positive Y-axis extending upward and the positive X-axis extending toward the right. The NBC API provides constants for use in the [NumOut](#) and [TextOut](#) functions which make it possible to specify LCD line numbers between 1 and 8 with line 1 being at the top of the screen and line 8 being at the bottom of the screen. These constants ([LCD\\_LINE1](#), [LCD\\_LINE2](#), [LCD\\_LINE3](#), [LCD\\_LINE4](#), [LCD\\_LINE5](#), [LCD\\_LINE6](#), [LCD\\_LINE7](#), [LCD\\_LINE8](#)) should be used as the Y coordinate in NumOut and TextOut calls. Values of Y other than these constants will be adjusted so that text and numbers are on one of 8 fixed line positions.

## 5.18 HiTechnic API Functions

Functions for accessing and modifying HiTechnic devices.

### Modules

- [HiTechnic device constants](#)

*Constants that are for use with HiTechnic devices.*

### Macros

- `#define SetSensorHTGyro(_port) __SetSensorHTGyro(_port)`  
*Set sensor as HiTechnic Gyro.*
- `#define ReadSensorHTGyro(_p, _offset, _val) __ReadSensorHTGyro(_p, _offset, _val)`  
*Read HiTechnic Gyro sensor.*
- `#define SetSensorHTMagnet(_port) __SetSensorHTGyro(_port)`  
*Set sensor as HiTechnic Magnet.*
- `#define ReadSensorHTMagnet(_p, _offset, _val) __ReadSensorHTGyro(_p, _offset, _val)`  
*Read HiTechnic Magnet sensor.*
- `#define SetSensorHTEOPD(_port, _bStd) __SetSensorHTEOPD(_port, _bStd)`  
*Set sensor as HiTechnic EOPD.*
- `#define ReadSensorHTEOPD(_port, _val) __ReadSensorHTEOPD(_port, _val)`  
*Read HiTechnic EOPD sensor.*
- `#define SetSensorHTForce(_port) __SetSensorHTForce(_port)`  
*Set sensor as HiTechnic Force.*
- `#define ReadSensorHTForce(_p, _val) __ReadSensorHTForce(_p, _val)`  
*Read HiTechnic Force sensor.*
- `#define ReadSensorHTTouchMultiplexer(_p, _t1, _t2, _t3, _t4) __ReadSensorHTTouchMultiplexer(_p, _t1, _t2, _t3, _t4)`  
*Read HiTechnic touch multiplexer.*
- `#define HTPowerFunctionCommand(_port, _channel, _outa, _outb, _result) __HTPFCComboDirect(_port, _channel, _outa, _outb, _result)`  
*HTPowerFunctionCommand function.*
- `#define HTIRTrain(_port, _channel, _func, _result) __HTIRTrain(_port, _channel, _func, FALSE, _result)`  
*HTIRTrain function.*
- `#define HTPFComboDirect(_port, _channel, _outa, _outb, _result) __HTPFCComboDirect(_port, _channel, _outa, _outb, _result)`  
*HTPFCComboDirect function.*
- `#define HTPFComboPWM(_port, _channel, _outa, _outb, _result) __HTPFCComboPWM(_port, _channel, _outa, _outb, _result)`  
*HTPFCComboPWM function.*
- `#define HTPFRawOutput(_port, _nibble0, _nibble1, _nibble2, _result) __HTPFRawOutput(_port, _nibble0, _nibble1, _nibble2, _result)`  
*HTPFRawOutput function.*
- `#define HTPFRepeat(_port, _count, _delay, _result) __HTPFRRepeatLastCommand(_port, _count, _delay, _result)`  
*HTPFRRepeat function.*

- #define `HTPFSingleOutputCST`(`_port`, `_channel`, `_out`, `_func`, `_result`) \_\_`HTPFSingleOutput`(`_port`, `_channel`, `_out`, `_func`, `TRUE`, `_result`)  
*HTPFSingleOutputCST function.*
- #define `HTPFSingleOutputPWM`(`_port`, `_channel`, `_out`, `_func`, `_result`) \_\_`HTPFSingleOutput`(`_port`, `_channel`, `_out`, `_func`, `FALSE`, `_result`)  
*HTPFSingleOutputPWM function.*
- #define `HTPFSinglePin`(`_port`, `_channel`, `_out`, `_pin`, `_func`, `_cont`, `_result`) \_\_`HTPFSinglePin`(`_port`, `_channel`, `_out`, `_pin`, `_func`, `_cont`, `_result`)  
*HTPFSinglePin function.*
- #define `HTPFTrain`(`_port`, `_channel`, `_func`, `_result`) \_\_`HTIRTrain`(`_port`, `_channel`, `_func`, `TRUE`, `_result`)  
*HTPFTrain function.*
- #define `HTRCXSetIRLinkPort`(`_port`) \_\_`HTRCXSetIRLinkPort`(`_port`)  
*HTRCXSetIRLinkPort function.*
- #define `HTRCXBatteryLevel`(`_result`) \_\_`HTRCXBatteryLevel`(`_result`)  
*HTRCXBatteryLevel function.*
- #define `HTRCXPoll`(`_src`, `_value`, `_result`) \_\_`HTRCXPoll`(`_src`, `_value`, `_result`)  
*HTRCXPoll function Send the Poll command to an RCX to read a signed 2-byte value at the specified source and value combination.*
- #define `HTRCXPollMemory`(`_memaddress`, `_result`) \_\_`HTRCXPollMemory`(`_memaddress`, `_result`)  
*HTRCXPollMemory function.*
- #define `HTRCXAddToDatalog`(`_src`, `_value`) \_\_`HTRCXAddToDatalog`(`_src`, `_value`)  
*HTRCXAddToDatalog function.*
- #define `HTRCXClearAllEvents`() \_\_`HTRCXOpNoArgs`(`RCX_ClearAllEventsOp`)  
*HTRCXClearAllEvents function.*
- #define `HTRCXClearCounter`(`_counter`) \_\_`HTRCXClearCounter`(`_counter`)  
*HTRCXClearCounter function.*
- #define `HTRCXClearMsg`() \_\_`HTRCXOpNoArgs`(`RCX_ClearMsgOp`)  
*HTRCXClearMsg function.*
- #define `HTRCXClearSensor`(`_port`) \_\_`HTRCXClearSensor`(`_port`)  
*HTRCXClearSensor function.*
- #define `HTRCXClearSound`() \_\_`HTRCXOpNoArgs`(`RCX_ClearSoundOp`)  
*HTRCXClearSound function.*
- #define `HTRCXClearTimer`(`_timer`) \_\_`HTRCXClearTimer`(`_timer`)  
*HTRCXClearTimer function.*
- #define `HTRCXCreateDatalog`(`_size`) \_\_`HTRCXCreateDatalog`(`_size`)  
*HTRCXCreateDatalog function.*
- #define `HTRCXDecCounter`(`_counter`) \_\_`HTRCXDecCounter`(`_counter`)  
*HTRCXDecCounter function.*
- #define `HTRCXDeleteSub`(`_s`) \_\_`HTRCXDeleteSub`(`_s`)  
*HTRCXDeleteSub function.*
- #define `HTRCXDeleteSubs`() \_\_`HTRCXOpNoArgs`(`RCX_DeleteSubsOp`)  
*HTRCXDeleteSubs function.*
- #define `HTRCXDeleteTask`(`_t`) \_\_`HTRCXDeleteTask`(`_t`)  
*HTRCXDeleteTask function.*
- #define `HTRCXDeleteTasks`() \_\_`HTRCXOpNoArgs`(`RCX_DeleteTasksOp`)  
*HTRCXDeleteTasks function.*
- #define `HTRCXDisableOutput`(`_outputs`) \_\_`HTRCXSetGlobalOutput`(`_outputs`, `RCX_OUT_OFF`)  
*HTRCXDisableOutput function.*

- #define HTRCXEnableOutput(\_outputs) \_\_HTRCXSetGlobalOutput(\_outputs, RCX\_OUT\_ON)  
*HTRCXEnableOutput function.*
- #define HTRCXEvent(\_src, \_value) \_\_HTRCXEvent(\_src, \_value)  
*HTRCXEvent function.*
- #define HTRCXFloat(\_outputs) \_\_HTRCXSetOutput(\_outputs, RCX\_OUT\_FLOAT)  
*HTRCXFloat function.*
- #define HTRCXFwd(\_outputs) \_\_HTRCXSetDirection(\_outputs, RCX\_OUT\_FWD)  
*HTRCXFwd function.*
- #define HTRCXIncCounter(\_counter) \_\_HTRCXIncCounter(\_counter)  
*HTRCXIncCounter function.*
- #define HTRCXIinvertOutput(\_outputs) \_\_HTRCXSetGlobalDirection(\_outputs, RCX\_OUT\_REV)  
*HTRCXIinvertOutput function.*
- #define HTRCXMuteSound() \_\_HTRCXOpNoArgs(RCX\_MuteSoundOp)  
*HTRCXMuteSound function.*
- #define HTRCXObvertOutput(\_outputs) \_\_HTRCXSetGlobalDirection(\_outputs, RCX\_OUT\_FWD)  
*HTRCXObvertOutput function.*
- #define HTRCXOff(\_outputs) \_\_HTRCXSetOutput(\_outputs, RCX\_OUT\_OFF)  
*HTRCXOff function.*
- #define HTRCXOn(\_outputs) \_\_HTRCXSetOutput(\_outputs, RCX\_OUT\_ON)  
*HTRCXOn function.*
- #define HTRCXOnFor(\_outputs, \_ms) \_\_HTRCXOnFor(\_outputs, \_ms)  
*HTRCXOnFor function.*
- #define HTRCXOnFwd(\_outputs) \_\_HTRCXOnFwd(\_outputs)  
*HTRCXOnFwd function.*
- #define HTRCXOnRev(\_outputs) \_\_HTRCXOnRev(\_outputs)  
*HTRCXOnRev function.*
- #define HTRCXPBTurnOff() \_\_HTRCXOpNoArgs(RCX\_PBTurnOffOp)  
*HTRCXPBTurnOff function.*
- #define HTRCXPing() \_\_HTRCXOpNoArgs(RCX\_PingOp)  
*HTRCXPing function.*
- #define HTRCXPlaySound(\_snd) \_\_HTRCXPlaySound(\_snd)  
*HTRCXPlaySound function.*
- #define HTRCXPlayTone(\_freq, \_duration) \_\_HTRCXPlayTone(\_freq, \_duration)  
*HTRCXPlayTone function.*
- #define HTRCXPlayToneVar(\_varnum, \_duration) \_\_HTRCXPlayToneVar(\_varnum, \_duration)  
*HTRCXPlayToneVar function.*
- #define HTRCXRemote(\_cmd) \_\_HTRCXRemote(\_cmd)  
*HTRCXRemote function.*
- #define HTRCXRev(\_outputs) \_\_HTRCXSetDirection(\_outputs, RCX\_OUT\_REV)  
*HTRCXRev function.*
- #define HTRCXSelectDisplay(\_src, \_value) \_\_HTRCXSelectDisplay(\_src, \_value)  
*HTRCXSelectDisplay function.*
- #define HTRCXSelectProgram(\_prog) \_\_HTRCXSelectProgram(\_prog)  
*HTRCXSelectProgram function.*
- #define HTRCXSendSerial(\_first, \_count) \_\_HTRCXSendSerial(\_first, \_count)  
*HTRCXSendSerial function.*
- #define HTRCXSetDirection(\_outputs, \_dir) \_\_HTRCXSetDirection(\_outputs, \_dir)

- `#define HTRCXSetEvent(_evt, _src, _type) __HTRCXSetEvent(_evt, _src, _type)`  
*HTRCXSetEvent function.*
- `#define HTRCXSetGlobalDirection(_outputs, _dir) __HTRCXSetGlobalDirection(_outputs, _dir)`  
*HTRCXSetGlobalDirection function.*
- `#define HTRCXSetGlobalOutput(_outputs, _mode) __HTRCXSetGlobalOutput(_outputs, _mode)`  
*HTRCXSetGlobalOutput function.*
- `#define HTRCXSetMaxPower(_outputs, _pwrsrc, _pwrval) __HTRCXSetMaxPower(_outputs, _pwrsrc, _pwrval)`  
*HTRCXSetMaxPower function.*
- `#define HTRCXSetMessage(_msg) __HTRCXSetMessage(_msg)`  
*HTRCXSetMessage function.*
- `#define HTRCXSetOutput(_outputs, _mode) __HTRCXSetOutput(_outputs, _mode)`  
*HTRCXSetOutput function.*
- `#define HTRCXSetPower(_outputs, _pwrsrc, _pwrval) __HTRCXSetPower(_outputs, _pwrsrc, _pwrval)`  
*HTRCXSetPower function.*
- `#define HTRCXSetPriority(_p) __HTRCXSetPriority(_p)`  
*HTRCXSetPriority function.*
- `#define HTRCXSetSensorMode(_port, _mode) __HTRCXSetSensorMode(_port, _mode)`  
*HTRCXSetSensorMode function.*
- `#define HTRCXSetSensorType(_port, _type) __HTRCXSetSensorType(_port, _type)`  
*HTRCXSetSensorType function.*
- `#define HTRCXSetSleepTime(_t) __HTRCXSetSleepTime(_t)`  
*HTRCXSetSleepTime function.*
- `#define HTRCXSetTxPower(_pwr) __HTRCXSetTxPower(_pwr)`  
*HTRCXSetTxPower function.*
- `#define HTRCXSetWatch(_hours, _minutes) __HTRCXSetWatch(_hours, _minutes)`  
*HTRCXSetWatch function.*
- `#define HTRCXStartTask(_t) __HTRCXStartTask(_t)`  
*HTRCXStartTask function.*
- `#define HTRCXStopAllTasks() __HTRCXOpNoArgs(RCX_StopAllTasksOp)`  
*HTRCXStopAllTasks function.*
- `#define HTRCXStopTask(_t) __HTRCXStopTask(_t)`  
*HTRCXStopTask function.*
- `#define HTRCXToggle(_outputs) __HTRCXSetDirection(_outputs, RCX_OUT_TOGGLE)`  
*HTRCXToggle function.*
- `#define HTRCXUnmuteSound() __HTRCXOpNoArgs(RCX_UnmuteSoundOp)`  
*HTRCXUnmuteSound function.*
- `#define HTScoutCalibrateSensor() __HTRCXOpNoArgs(RCX_LSCalibrateOp)`  
*HTScoutCalibrateSensor function.*
- `#define HTScoutMuteSound() __HTScoutMuteSound()`  
*HTScoutMuteSound function.*
- `#define HTScoutSelectSounds(_grp) __HTScoutSelectSounds(_grp)`  
*HTScoutSelectSounds function.*
- `#define HTScoutSendVLL(_src, _value) __HTScoutSendVLL(_src, _value)`  
*HTScoutSendVLL function.*
- `#define HTScoutSetEventFeedback(_src, _value) __HTScoutSetEventFeedback(_src, _value)`

- HTScoutSetEventFeedback function.*
- `#define HTScoutSetLight(_x) __HTScoutSetLight(_x)`  
*HTScoutSetLight function.*
  - `#define HTScoutSetScoutMode(_mode) __HTScoutSetScoutMode(_mode)`  
*HTScoutSetScoutMode function.*
  - `#define HTScoutSetSensorClickTime(_src, _value) __HTScoutSetSensorClickTime(_src, _value)`  
*HTScoutSetSensorClickTime function.*
  - `#define HTScoutSetSensorHysteresis(_src, _value) __HTScoutSetSensorHysteresis(_src, _value)`  
*HTScoutSetSensorHysteresis function.*
  - `#define HTScoutSetSensorLowerLimit(_src, _value) __HTScoutSetSensorLowerLimit(_src, _value)`  
*HTScoutSetSensorLowerLimit function.*
  - `#define HTScoutSetSensorUpperLimit(_src, _value) __HTScoutSetSensorUpperLimit(_src, _value)`  
*HTScoutSetSensorUpperLimit function.*
  - `#define HTScoutUnmuteSound() __HTScoutUnmuteSound()`  
*HTScoutUnmuteSound function.*
  - `#define ReadSensorHTCompass(_port, _value) __ReadSensorHTCompass(_port, _value)`  
*Read HiTechnic compass.*
  - `#define ReadSensorHTColorNum(_port, _value) __ReadSensorHTColorNum(_port, _value)`  
*Read HiTechnic color sensor color number.*
  - `#define ReadSensorHTIRSeekerDir(_port, _value) __ReadSensorHTIRSeekerDir(_port, _value)`  
*Read HiTechnic IRSeeker direction.*
  - `#define ReadSensorHTIRSeeker2Addr(_port, _reg, _value) __ReadSensorHTIRSeeker2Addr(_port, _reg, _value)`  
*Read HiTechnic IRSeeker2 register.*
  - `#define ReadSensorHTAccel(_port, _x, _y, _z, _result) __ReadSensorHTAccel(_port, _x, _y, _z, _result)`  
*Read HiTechnic acceleration values.*
  - `#define ReadSensorHTColor(_port, _ColorNum, _Red, _Green, _Blue, _result) __ReadSensorHTColor(_port, _ColorNum, _Red, _Green, _Blue, _result)`  
*Read HiTechnic Color values.*
  - `#define ReadSensorHTRawColor(_port, _Red, _Green, _Blue, _result) __ReadSensorHTRawColor(_port, _Red, _Green, _Blue, _result)`  
*Read HiTechnic Color raw values.*
  - `#define ReadSensorHTNormalizedColor(_port, _ColorIdx, _Red, _Green, _Blue, _result) __ReadSensorHTNormalizedColor(_port, _ColorIdx, _Red, _Green, _Blue, _result)`  
*Read HiTechnic Color normalized values.*
  - `#define ReadSensorHTIRSeeker(_port, _dir, _s1, _s3, _s5, _s7, _s9, _result) __ReadSensorHTIRSeeker(_port, _dir, _s1, _s3, _s5, _s7, _s9, _result)`  
*Read HiTechnic IRSeeker values.*
  - `#define ReadSensorHTIRSeeker2DC(_port, _dir, _s1, _s3, _s5, _s7, _s9, _avg, _result) __ReadSensorHTIRSeeker2DC(_port, _dir, _s1, _s3, _s5, _s7, _s9, _avg, _result)`  
*Read HiTechnic IRSeeker2 DC values.*
  - `#define ReadSensorHTIRSeeker2AC(_port, _dir, _s1, _s3, _s5, _s7, _s9, _result) __ReadSensorHTIRSeeker2AC(_port, _dir, _s1, _s3, _s5, _s7, _s9, _result)`  
*Read HiTechnic IRSeeker2 AC values.*
  - `#define SetHTIRSeeker2Mode(_port, _mode, _result) __SetHTIRSeeker2Mode(_port, _mode, _result)`  
*Set HiTechnic IRSeeker2 mode.*
  - `#define SetHTColor2Mode(_port, _mode, _result) __SetHTColor2Mode(_port, _mode, _result)`  
*Set HiTechnic Color2 mode.*

- #define `ReadSensorHTColor2Active(_port, _ColorNum, _Red, _Green, _Blue, _White, _result)` \_\_ReadSensorHTColor2Active(\_port, \_ColorNum, \_Red, \_Green, \_Blue, \_White, \_result)  
*Read HiTechnic Color2 active values.*
- #define `ReadSensorHTNormalizedColor2Active(_port, _ColorIdx, _Red, _Green, _Blue, _result)` \_\_ReadSensorHTNormalizedColor2Active(\_port, \_ColorIdx, \_Red, \_Green, \_Blue, \_result)  
*Read HiTechnic Color2 normalized active values.*
- #define `ReadSensorHTRawColor2(_port, _Red, _Green, _Blue, _White, _result)` \_\_ReadSensorHTRawColor2(-\_port, \_Red, \_Green, \_Blue, \_White, \_result)  
*Read HiTechnic Color2 raw values.*
- #define `ReadSensorHTIRReceiver(_port, _pfdata, _result)` \_\_ReadSensorHTIRReceiver(\_port, \_pfdata, \_result)  
*Read HiTechnic IRReceiver Power Function bytes.*
- #define `ReadSensorHTIRReceiverEx(_port, _reg, _pfchar, _result)` \_\_ReadSensorHTIRReceiverEx(\_port, \_reg, \_pfchar, \_result)  
*Read HiTechnic IRReceiver Power Function value.*
- #define `ResetSensorHTAngle(_port, _mode, _result)` \_\_ResetSensorHTAngle(\_port, \_mode, \_result)  
*Reset HiTechnic Angle sensor.*
- #define `ReadSensorHTAngle(_port, _Angle, _AccAngle, _RPM, _result)` \_\_ReadSensorHTAngle(\_port, \_Angle, \_AccAngle, \_RPM, \_result)  
*Read HiTechnic Angle sensor values.*
- #define `ResetHTBarometricCalibration(_port, _result)` \_\_ResetHTBarometricCalibration(\_port, \_result)  
*Reset HiTechnic Barometric sensor calibration.*
- #define `SetHTBarometricCalibration(_port, _cal, _result)` \_\_SetHTBarometricCalibration(\_port, \_cal, \_result)  
*Set HiTechnic Barometric sensor calibration.*
- #define `ReadSensorHTBarometric(_port, _temp, _press, _result)` \_\_ReadSensorHTBarometric(\_port, \_temp, \_press, \_result)  
*Read HiTechnic Barometric sensor values.*
- #define `ReadSensorHTProtoAnalog(_port, _input, _value, _result)` \_\_ReadSensorHTProtoAnalog(\_port, HT\_ADDR\_PROTOBOARD, \_input, \_value, \_result)  
*Read HiTechnic Prototype board analog input value.*
- #define `ReadSensorHTProtoAllAnalog(_port, _a0, _a1, _a2, _a3, _a4, _result)` \_\_ReadSensorHTProtoAll-Analog(\_port, \_a0, \_a1, \_a2, \_a3, \_a4, \_result)  
*Read all HiTechnic Prototype board analog input values.*
- #define `SetSensorHTProtoDigitalControl(_port, _value, _result)` \_\_SetSensorHTProtoDigitalControl(\_port, HT\_ADDR\_PROTOBOARD, \_value, \_result)  
*Set HiTechnic Prototype board digital pin direction.*
- #define `ReadSensorHTProtoDigitalControl(_port, _out, _result)` \_\_MSReadValue(\_port, HT\_ADDR\_PROTOBOARD, HTPROTO\_REG\_DCTRL, 1, \_out, \_result)  
*Read HiTechnic Prototype board digital pin control value.*
- #define `SetSensorHTProtoDigital(_port, _value, _result)` \_\_SetSensorHTProtoDigital(\_port, HT\_ADDR\_PROTOBOARD, \_value, \_result)  
*Set HiTechnic Prototype board digital output values.*
- #define `ReadSensorHTProtoDigital(_port, _value, _result)` \_\_ReadSensorHTProtoDigital(\_port, HT\_ADDR\_PROTOBOARD, \_value, \_result)  
*Read HiTechnic Prototype board digital input values.*
- #define `ReadSensorHTSuperProAnalog(_port, _input, _value, _result)` \_\_ReadSensorHTProtoAnalog(\_port, HT\_ADDR\_SUPERPRO, \_input, \_value, \_result)  
*Read HiTechnic SuperPro board analog input value.*

- #define `ReadSensorHTSuperProAllAnalog(_port, _a0, _a1, _a2, _a3, _result)` \_\_ReadSensorHTSuperProAllAnalog(\_port, \_a0, \_a1, \_a2, \_a3, \_result)  
*Read all HiTechnic SuperPro board analog input values.*
- #define `SetSensorHTSuperProDigitalControl(_port, _value, _result)` \_\_SetSensorHTProtoDigitalControl(\_port, HT\_ADDR\_SUPERPRO, \_value, \_result)  
*Control HiTechnic SuperPro board digital pin direction.*
- #define `ReadSensorHTSuperProDigitalControl(_port, _out, _result)` \_\_MSReadValue(\_port, HT\_ADDR\_SUPERPRO, HTSPRO\_REG\_DCTRL, 1, \_out, \_result)  
*Read HiTechnic SuperPro digital control value.*
- #define `SetSensorHTSuperProDigital(_port, _value, _result)` \_\_SetSensorHTProtoDigital(\_port, HT\_ADDR\_SUPERPRO, \_value, \_result)  
*Set HiTechnic SuperPro board digital output values.*
- #define `ReadSensorHTSuperProDigital(_port, _value, _result)` \_\_ReadSensorHTProtoDigital(\_port, HT\_ADDR\_SUPERPRO, \_value, \_result)  
*Read HiTechnic SuperPro board digital input values.*
- #define `SetSensorHTSuperProAnalogOut(_port, _dac, _mode, _freq, _volt, _result)` \_\_SetSensorHTSuperProAnalogOut(\_port, \_dac, \_mode, \_freq, \_volt, \_result)  
*Set HiTechnic SuperPro board analog output parameters.*
- #define `ReadSensorHTSuperProAnalogOut(_port, _dac, _mode, _freq, _volt, _result)` \_\_ReadSensorHTSuperProAnalogOut(\_port, \_dac, \_mode, \_freq, \_volt, \_result)  
*Read HiTechnic SuperPro board analog output parameters.*
- #define `SetSensorHTSuperProLED(_port, _value, _result)` \_\_SetSensorHTSuperProLED(\_port, \_value, \_result)  
*Set HiTechnic SuperPro LED value.*
- #define `ReadSensorHTSuperProLED(_port, _out, _result)` \_\_MSReadValue(\_port, HT\_ADDR\_SUPERPRO, HTSPRO\_REG\_LED, 1, \_out, \_result)  
*Read HiTechnic SuperPro LED value.*
- #define `SetSensorHTSuperProStrobe(_port, _value, _result)` \_\_SetSensorHTSuperProStrobe(\_port, \_value, \_result)  
*Set HiTechnic SuperPro strobe value.*
- #define `ReadSensorHTSuperProStrobe(_port, _out, _result)` \_\_MSReadValue(\_port, HT\_ADDR\_SUPERPRO, HTSPRO\_REG\_STROBE, 1, \_out, \_result)  
*Read HiTechnic SuperPro strobe value.*
- #define `SetSensorHTSuperProProgramControl(_port, _value, _result)` \_\_SetSensorHTSuperProProgramControl(\_port, \_value, \_result)  
*Set HiTechnic SuperPro program control value.*
- #define `ReadSensorHTSuperProProgramControl(_port, _out, _result)` \_\_MSReadValue(\_port, HT\_ADDR\_SUPERPRO, HTSPRO\_REG\_CTRL, 1, \_out, \_result)  
*Read HiTechnic SuperPro program control value.*
- #define `SetSensorHTPIRDeadband(_port, _value, _result)` \_\_SetSensorHTPIRDeadband(\_port, \_value, \_result)  
*Set HiTechnic PIR deadband value.*
- #define `ReadSensorHTPIR(_port, _out, _result)` \_\_MSReadValue(\_port, HT\_ADDR\_PIR, HTPIR\_REG\_READING, 1, \_out, \_result)  
*Read HiTechnic SuperPro strobe value.*

### 5.18.1 Detailed Description

Functions for accessing and modifying HiTechnic devices.

### 5.18.2 Macro Definition Documentation

5.18.2.1 #define HTIRTrain( \_port, \_channel, \_func, \_result ) \_\_HTIRTrain(\_port, \_channel, \_func, FALSE, \_result)

HTIRTrain function.

Control an IR Train receiver set to the specified channel using the HiTechnic iRLink device. Valid func values are [TRAIN\\_FUNC\\_STOP](#), [TRAIN\\_FUNC\\_INCR\\_SPEED](#), [TRAIN\\_FUNC\\_DECR\\_SPEED](#), and [TRAIN\\_FUNC\\_TOGGLE\\_LIGHT](#). Valid channel values are [TRAIN\\_CHANNEL\\_1](#) through [TRAIN\\_CHANNEL\\_3](#) and [TRAIN\\_CHANNEL\\_ALL](#). The port must be configured as a Lowspeed port before using this function.

#### Parameters

<a href="#">_port</a>	The sensor port. See <a href="#">NBC Input port constants</a> .
<a href="#">_channel</a>	The IR Train channel. See <a href="#">IR Train channel constants</a> .
<a href="#">_func</a>	The IR Train function. See <a href="#">PF/IR Train function constants</a>
<a href="#">_result</a>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

5.18.2.2 #define HTPFComboDirect( \_port, \_channel, \_outa, \_outb, \_result ) \_\_HTPFComboDirect(\_port, \_channel, \_outa, \_outb, \_result)

HTPFComboDirect function.

Execute a pair of Power Function motor commands on the specified channel using the HiTechnic iRLink device. Commands for outa and outb are [PF\\_CMD\\_STOP](#), [PF\\_CMD\\_REV](#), [PF\\_CMD\\_FWD](#), and [PF\\_CMD\\_BRAKE](#). Valid channels are [PF\\_CHANNEL\\_1](#) through [PF\\_CHANNEL\\_4](#). The port must be configured as a Lowspeed port before using this function.

#### Parameters

<a href="#">_port</a>	The sensor port. See <a href="#">NBC Input port constants</a> .
<a href="#">_channel</a>	The Power Function channel. See <a href="#">Power Function channel constants</a> .
<a href="#">_outa</a>	The Power Function command for output A. See <a href="#">Power Function command constants</a> .
<a href="#">_outb</a>	The Power Function command for output B. See <a href="#">Power Function command constants</a> .
<a href="#">_result</a>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

5.18.2.3 #define HTPFComboPWM( \_port, \_channel, \_outa, \_outb, \_result ) \_\_HTPFComboPWM(\_port, \_channel, \_outa, \_outb, \_result)

HTPFComboPWM function.

Control the speed of both outputs on a Power Function receiver set to the specified channel using the HiTechnic iRLink device. Valid output values are [PF\\_PWM\\_FLOAT](#), [PF\\_PWM\\_FWD1](#), [PF\\_PWM\\_FWD2](#), [PF\\_PWM\\_FWD3](#), [PF\\_PWM\\_FWD4](#), [PF\\_PWM\\_FWD5](#), [PF\\_PWM\\_FWD6](#), [PF\\_PWM\\_FWD7](#), [PF\\_PWM\\_BRAKE](#), [PF\\_PWM\\_REV7](#), [PF\\_PWM\\_REV6](#), [PF\\_PWM\\_REV5](#), [PF\\_PWM\\_REV4](#), [PF\\_PWM\\_REV3](#), [PF\\_PWM\\_REV2](#), and [PF\\_PWM\\_REV1](#). Valid channels are [PF\\_CHANNEL\\_1](#) through [PF\\_CHANNEL\\_4](#). The port must be configured as a Lowspeed port before using this function.

#### Parameters

<a href="#">_port</a>	The sensor port. See <a href="#">NBC Input port constants</a> .
<a href="#">_channel</a>	The Power Function channel. See <a href="#">Power Function channel constants</a> .
<a href="#">_outa</a>	The Power Function PWM command for output A. See <a href="#">Power Function PWM option constants</a> .
<a href="#">_outb</a>	The Power Function PWM command for output B. See <a href="#">Power Function PWM option constants</a> .
<a href="#">_result</a>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

```
5.18.2.4 #define HTPFRawOutput( _port, _nibble0, _nibble1, _nibble2, _result ) __HTPFRawOutput(_port, _nibble0, _nibble1, _nibble2, _result)
```

HTPFRawOutput function.

Control a Power Function receiver set to the specified channel using the HiTechnic iRLink device. Build the raw data stream using the 3 nibbles (4 bit values). The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_nibble0</code>	The first raw data nibble.
<code>_nibble1</code>	The second raw data nibble.
<code>_nibble2</code>	The third raw data nibble.
<code>_result</code>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

```
5.18.2.5 #define HTPFRepeat( _port, _count, _delay, _result ) __HTPFRepeatLastCommand(_port, _count, _delay, _result)
```

HTPFRepeat function.

Repeat sending the last Power Function command using the HiTechnic iRLink device. Specify the number of times to repeat the command and the number of milliseconds of delay between each repetition. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_count</code>	The number of times to repeat the command.
<code>_delay</code>	The number of milliseconds to delay between each repetition.
<code>_result</code>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

```
5.18.2.6 #define HTPFSingleOutputCST( _port, _channel, _out, _func, _result ) __HTPFSingleOutput(_port, _channel, _out, _func, TRUE, _result)
```

HTPFSingleOutputCST function.

Control a single output on a Power Function receiver set to the specified channel using the HiTechnic iRLink device. Select the desired output using [PF\\_OUT\\_A](#) or [PF\\_OUT\\_B](#). Valid functions are [PF\\_CST\\_CLEAR1\\_CLEAR2](#), [PF\\_CST\\_SET1\\_CLEAR2](#), [PF\\_CST\\_CLEAR1\\_SET2](#), [PF\\_CST\\_SET1\\_SET2](#), [PF\\_CST\\_INCREMENT\\_PWM](#), [PF\\_CST\\_DECREMENT\\_PWM](#), [PF\\_CST\\_FULL\\_FWD](#), [PF\\_CST\\_FULL\\_REV](#), and [PF\\_CST\\_TOGGLE\\_DIR](#). Valid channels are [PF\\_CHANNEL\\_1](#) through [PF\\_CHANNEL\\_4](#). The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_channel</code>	The Power Function channel. See <a href="#">Power Function channel constants</a> .
<code>_out</code>	The Power Function output. See <a href="#">Power Function output constants</a> .
<code>_func</code>	The Power Function CST function. See <a href="#">Power Function CST options constants</a> .
<code>_result</code>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

```
5.18.2.7 #define HTPFSingleOutputPWM( _port, _channel, _out, _func, _result ) __HTPFSingleOutput(_port, _channel, _out, _func, FALSE, _result)
```

HTPFSingleOutputPWM function.

Control the speed of a single output on a Power Function receiver set to the specified channel using the HiTechnic iRLink

device. Select the desired output using `PF_OUT_A` or `PF_OUT_B`. Valid functions are `PF_PWM_FLOAT`, `PF_PWM_FWD1`, `PF_PWM_FWD2`, `PF_PWM_FWD3`, `PF_PWM_FWD4`, `PF_PWM_FWD5`, `PF_PWM_FWD6`, `PF_PWM_FWD7`, `PF_PWM_BRAKE`, `PF_PWM_REV7`, `PF_PWM_REV6`, `PF_PWM_REV5`, `PF_PWM_REV4`, `PF_PWM_REV3`, `PF_PWM_REV2`, and `PF_PWM_REV1`. Valid channels are `PF_CHANNEL_1` through `PF_CHANNEL_4`. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_channel</code>	The Power Function channel. See <a href="#">Power Function channel constants</a> .
<code>_out</code>	The Power Function output. See <a href="#">Power Function output constants</a> .
<code>_func</code>	The Power Function PWM function. See <a href="#">Power Function PWM option constants</a> .
<code>_result</code>	The function call result. <code>NO_ERR</code> or <a href="#">Communications specific errors</a> .

```
5.18.2.8 #define HTPFSinglePin( _port, _channel, _out, _pin, _func, _cont, _result ) __HTPFSinglePin(_port, _channel, _out, _pin, _func, _cont, _result)
```

HTPFSinglePin function.

Control a single pin on a Power Function receiver set to the specified channel using the HiTechnic iRLLink device. Select the desired output using `PF_OUT_A` or `PF_OUT_B`. Select the desired pin using `PF_PIN_C1` or `PF_PIN_C2`. Valid functions are `PF_FUNC_NOCHANGE`, `PF_FUNC_CLEAR`, `PF_FUNC_SET`, and `PF_FUNC_TOGGLE`. Valid channels are `PF_CHANNEL_1` through `PF_CHANNEL_4`. Specify whether the mode by passing true (continuous) or false (timeout) as the final parameter. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_channel</code>	The Power Function channel. See <a href="#">Power Function channel constants</a> .
<code>_out</code>	The Power Function output. See <a href="#">Power Function output constants</a> .
<code>_pin</code>	The Power Function pin. See <a href="#">Power Function pin constants</a> .
<code>_func</code>	The Power Function single pin function. See <a href="#">Power Function single pin function constants</a> .
<code>_cont</code>	Control whether the mode is continuous or timeout.
<code>_result</code>	The function call result. <code>NO_ERR</code> or <a href="#">Communications specific errors</a> .

```
5.18.2.9 #define HTPFTrain( _port, _channel, _func, _result ) __HTIRTrain(_port, _channel, _func, TRUE, _result)
```

HTPFTrain function.

Control both outputs on a Power Function receiver set to the specified channel using the HiTechnic iRLLink device as if it were an IR Train receiver. Valid function values are `TRAIN_FUNC_STOP`, `TRAIN_FUNC_INCR_SPEED`, `TRAIN_FUNC_DECR_SPEED`, and `TRAIN_FUNC_TOGGLE_LIGHT`. Valid channels are `PF_CHANNEL_1` through `PF_CHANNEL_4`. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_channel</code>	The Power Function channel. See <a href="#">Power Function channel constants</a> .
<code>_func</code>	The Power Function train function. See <a href="#">PF/IR Train function constants</a> .
<code>_result</code>	The function call result. <code>NO_ERR</code> or <a href="#">Communications specific errors</a> .

```
5.18.2.10 #define HTPowerFunctionCommand( _port, _channel, _outa, _outb, _result ) __HTPFCComboDirect(_port, _channel, _outa, _outb, _result)
```

HTPowerFunctionCommand function.

Execute a pair of Power Function motor commands on the specified channel using the HiTechnic iRLink device. Commands for outa and outb are [PF\\_CMD\\_STOP](#), [PF\\_CMD\\_REV](#), [PF\\_CMD\\_FWD](#), and [PF\\_CMD\\_BRAKE](#). Valid channels are [PF\\_CHANNEL\\_1](#) through [PF\\_CHANNEL\\_4](#). The port must be configured as a Lowspeed port before using this function.

#### Parameters

<a href="#">_port</a>	The sensor port. See <a href="#">NBC Input port constants</a> .
<a href="#">_channel</a>	The Power Function channel. See <a href="#">Power Function channel constants</a> .
<a href="#">_outa</a>	The Power Function command for output A. See <a href="#">Power Function command constants</a> .
<a href="#">_outb</a>	The Power Function command for output B. See <a href="#">Power Function command constants</a> .
<a href="#">_result</a>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

#### 5.18.2.11 #define HTRCXAddToDatalog( [\\_src](#), [\\_value](#) ) \_\_HTRCXAddToDatalog([\\_src](#), [\\_value](#))

HTRCXAddToDatalog function.

Send the AddToDatalog command to an RCX.

#### Parameters

<a href="#">_src</a>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<a href="#">_value</a>	The RCX value.

#### 5.18.2.12 #define HTRCXBatteryLevel( [\\_result](#) ) \_\_HTRCXBatteryLevel([\\_result](#))

HTRCXBatteryLevel function.

Send the BatteryLevel command to an RCX to read the current battery level.

#### Parameters

<a href="#">_result</a>	The RCX battery level.
-------------------------	------------------------

#### 5.18.2.13 #define HTRCXClearAllEvents( ) \_\_HTRCXOpNoArgs(RCX\_ClearAllEventsOp)

HTRCXClearAllEvents function.

Send the ClearAllEvents command to an RCX.

#### 5.18.2.14 #define HTRCXClearCounter( [\\_counter](#) ) \_\_HTRCXClearCounter([\\_counter](#))

HTRCXClearCounter function.

Send the ClearCounter command to an RCX.

#### Parameters

<a href="#">_counter</a>	The counter to clear.
--------------------------	-----------------------

#### 5.18.2.15 #define HTRCXClearMsg( ) \_\_HTRCXOpNoArgs(RCX\_ClearMsgOp)

HTRCXClearMsg function.

Send the ClearMsg command to an RCX.

**5.18.2.16 #define HTRCXClearSensor( *\_port* ) \_\_HTRCXClearSensor(*\_port*)**

HTRCXClearSensor function.

Send the ClearSensor command to an RCX.

**Parameters**

<i>_port</i>	The RCX port number.
--------------	----------------------

**5.18.2.17 #define HTRCXClearSound( ) \_\_HTRCXOpNoArgs(RCX\_ClearSoundOp)**

HTRCXClearSound function.

Send the ClearSound command to an RCX.

**5.18.2.18 #define HTRCXClearTimer( *\_timer* ) \_\_HTRCXClearTimer(*\_timer*)**

HTRCXClearTimer function.

Send the ClearTimer command to an RCX.

**Parameters**

<i>_timer</i>	The timer to clear.
---------------	---------------------

**5.18.2.19 #define HTRCXCreateDatalog( *\_size* ) \_\_HTRCXCreateDatalog(*\_size*)**

HTRCXCreateDatalog function.

Send the CreateDatalog command to an RCX.

**Parameters**

<i>_size</i>	The new datalog size.
--------------	-----------------------

**5.18.2.20 #define HTRCXDecCounter( *\_counter* ) \_\_HTRCXDecCounter(*\_counter*)**

HTRCXDecCounter function.

Send the DecCounter command to an RCX.

**Parameters**

<i>_counter</i>	The counter to decrement.
-----------------	---------------------------

**5.18.2.21 #define HTRCXDeleteSub( *\_s* ) \_\_HTRCXDeleteSub(*\_s*)**

HTRCXDeleteSub function.

Send the DeleteSub command to an RCX.

**Parameters**

<i>_s</i>	The subroutine number to delete.
-----------	----------------------------------

**5.18.2.22 #define HTRCXDeleteSubs( ) \_\_HTRCXOpNoArgs(RCX\_DeleteSubsOp)**

HTRCXDeleteSubs function.

Send the DeleteSubs command to an RCX.

**5.18.2.23 #define HTRCXDeleteTask( \_t ) \_\_HTRCXDeleteTask(\_t)**

HTRCXDeleteTask function.

Send the DeleteTask command to an RCX.

**Parameters**

<i>_t</i>	The task number to delete.
-----------	----------------------------

**5.18.2.24 #define HTRCXDeleteTasks( ) \_\_HTRCXOpNoArgs(RCX\_DeleteTasksOp)**

HTRCXDeleteTasks function.

Send the DeleteTasks command to an RCX.

**5.18.2.25 #define HTRCXDisableOutput( \_outputs ) \_\_HTRCXSetGlobalOutput(\_outputs, RCX\_OUT\_OFF)**

HTRCXDisableOutput function.

Send the DisableOutput command to an RCX.

**Parameters**

<i>_outputs</i>	The RCX output(s) to disable. See <a href="#">RCX output constants</a> .
-----------------	--

**5.18.2.26 #define HTRCXEnableOutput( \_outputs ) \_\_HTRCXSetGlobalOutput(\_outputs, RCX\_OUT\_ON)**

HTRCXEnableOutput function.

Send the EnableOutput command to an RCX.

**Parameters**

<i>_outputs</i>	The RCX output(s) to enable. See <a href="#">RCX output constants</a> .
-----------------	---

**5.18.2.27 #define HTRCXEvent( \_src, \_value ) \_\_HTRCXEvent(\_src, \_value)**

HTRCXEvent function.

Send the Event command to an RCX.

**Parameters**

<i>_src</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_value</i>	The RCX value.

**5.18.2.28 #define HTRCXFloat( \_outputs ) \_\_HTRCXSetOutput(\_outputs, RCX\_OUT\_FLOAT)**

HTRCXFloat function.

Send commands to an RCX to float the specified outputs.

**Parameters**

<i>_outputs</i>	The RCX output(s) to float. See <a href="#">RCX output constants</a> .
-----------------	--

**5.18.2.29 #define HTRCXFwd( *\_outputs* ) \_\_HTRCXSetDirection(*\_outputs*, RCX\_OUT\_FWD)**

HTRCXFwd function.

Send commands to an RCX to set the specified outputs to the forward direction.

**Parameters**

<i>_outputs</i>	The RCX output(s) to set forward. See <a href="#">RCX output constants</a> .
-----------------	--

**5.18.2.30 #define HTRCXIncCounter( *\_counter* ) \_\_HTRCXIncCounter(*\_counter*)**

HTRCXIncCounter function.

Send the IncCounter command to an RCX.

**Parameters**

<i>_counter</i>	The counter to increment.
-----------------	---------------------------

**5.18.2.31 #define HTRCXInvertOutput( *\_outputs* ) \_\_HTRCXSetGlobalDirection(*\_outputs*, RCX\_OUT\_REV)**

HTRCXInvertOutput function.

Send the InvertOutput command to an RCX.

**Parameters**

<i>_outputs</i>	The RCX output(s) to invert. See <a href="#">RCX output constants</a> .
-----------------	---

**5.18.2.32 #define HTRCXMuteSound( ) \_\_HTRCXOpNoArgs(RCX\_MuteSoundOp)**

HTRCXMuteSound function.

Send the MuteSound command to an RCX.

**5.18.2.33 #define HTRCXObvertOutput( *\_outputs* ) \_\_HTRCXSetGlobalDirection(*\_outputs*, RCX\_OUT\_FWD)**

HTRCXObvertOutput function.

Send the ObvertOutput command to an RCX.

**Parameters**

<i>_outputs</i>	The RCX output(s) to obvert. See <a href="#">RCX output constants</a> .
-----------------	---

**5.18.2.34 #define HTRCXOff( *\_outputs* ) \_\_HTRCXSetOutput(*\_outputs*, RCX\_OUT\_OFF)**

HTRCXOff function.

Send commands to an RCX to turn off the specified outputs.

**Parameters**

<i>_outputs</i>	The RCX output(s) to turn off. See <a href="#">RCX output constants</a> .
-----------------	---

5.18.2.35 #define HTRECXOn( *\_outputs* ) \_\_HTRECXSetOutput(*\_outputs*, RCX\_OUT\_ON)

HTRECXOn function.

Send commands to an RCX to turn on the specified outputs.

**Parameters**

<i>_outputs</i>	The RCX output(s) to turn on. See <a href="#">RCX output constants</a> .
-----------------	--

5.18.2.36 #define HTRECXOnFor( *\_outputs*, *\_ms* ) \_\_HTRECXOnFor(*\_outputs*, *\_ms*)

HTRECXOnFor function.

Send commands to an RCX to turn on the specified outputs in the forward direction for the specified duration.

**Parameters**

<i>_outputs</i>	The RCX output(s) to turn on. See <a href="#">RCX output constants</a> .
<i>_ms</i>	The number of milliseconds to leave the outputs on

5.18.2.37 #define HTRECXOnFwd( *\_outputs* ) \_\_HTRECXOnFwd(*\_outputs*)

HTRECXOnFwd function.

Send commands to an RCX to turn on the specified outputs in the forward direction.

**Parameters**

<i>_outputs</i>	The RCX output(s) to turn on in the forward direction. See <a href="#">RCX output constants</a> .
-----------------	---

5.18.2.38 #define HTRECXOnRev( *\_outputs* ) \_\_HTRECXOnRev(*\_outputs*)

HTRECXOnRev function.

Send commands to an RCX to turn on the specified outputs in the reverse direction.

**Parameters**

<i>_outputs</i>	The RCX output(s) to turn on in the reverse direction. See <a href="#">RCX output constants</a> .
-----------------	---

5.18.2.39 #define HTRECXPTurnOff( ) \_\_HTRECXOpNoArgs(RCX\_PBTurnOffOp)

HTRECXPTurnOff function.

Send the PBTurnOff command to an RCX.

5.18.2.40 #define HTRECXPing( ) \_\_HTRECXOpNoArgs(RCX\_PingOp)

HTRECXPing function.

Send the Ping command to an RCX.

**5.18.2.41 #define HTRCXPlaySound(  *snd*  ) \_\_HTRCXPlaySound( *snd* )**

HTRCXPlaySound function.

Send the PlaySound command to an RCX.

**Parameters**

<i> snd </i>	The sound number to play.
--------------	---------------------------

**5.18.2.42 #define HTRCXPlayTone(  *freq, duration*  ) \_\_HTRCXPlayTone( *freq, duration* )**

HTRCXPlayTone function.

Send the PlayTone command to an RCX.

**Parameters**

<i> freq </i>	The frequency of the tone to play.
<i> duration </i>	The duration of the tone to play.

**5.18.2.43 #define HTRCXPlayToneVar(  *varnum, duration*  ) \_\_HTRCXPlayToneVar( *varnum, duration* )**

HTRCXPlayToneVar function.

Send the PlayToneVar command to an RCX.

**Parameters**

<i> varnum </i>	The variable containing the tone frequency to play.
<i> duration </i>	The duration of the tone to play.

**5.18.2.44 #define HTRCXPoll(  *src, value, result*  ) \_\_HTRCXPoll( *src, value, result* )**

HTRCXPoll function Send the Poll command to an RCX to read a signed 2-byte value at the specified source and value combination.

**Parameters**

<i> src </i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i> value </i>	The RCX value.
<i> result </i>	The value read from the specified port and value.

**5.18.2.45 #define HTRCXPollMemory(  *memaddress, result*  ) \_\_HTRCXPollMemory( *memaddress, result* )**

HTRCXPollMemory function.

Send the PollMemory command to an RCX.

**Parameters**

<i> memaddress </i>	The RCX memory address.
<i> result </i>	The value read from the specified address.

**5.18.2.46 #define HTRCXRemote( \_cmd ) \_\_HTRCXRemote(\_cmd)**

HTRCXRemote function.

Send the Remote command to an RCX.

**Parameters**

<i>_cmd</i>	The RCX IR remote command to send. See <a href="#">RCX IR remote constants</a> .
-------------	--

**5.18.2.47 #define HTRCXRev( \_outputs ) \_\_HTRCXSetDirection(\_outputs, RCX\_OUT\_REV)**

HTRCXRev function.

Send commands to an RCX to set the specified outputs to the reverse direction.

**Parameters**

<i>_outputs</i>	The RCX output(s) to reverse direction. See <a href="#">RCX output constants</a> .
-----------------	--

**5.18.2.48 #define HTRCXSelectDisplay( \_src, \_value ) \_\_HTRCXSelectDisplay(\_src, \_value)**

HTRCXSelectDisplay function.

Send the SelectDisplay command to an RCX.

**Parameters**

<i>_src</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_value</i>	The RCX value.

**5.18.2.49 #define HTRCXSelectProgram( \_prog ) \_\_HTRCXSelectProgram(\_prog)**

HTRCXSelectProgram function.

Send the SelectProgram command to an RCX.

**Parameters**

<i>_prog</i>	The program number to select.
--------------	-------------------------------

**5.18.2.50 #define HTRCXSendSerial( \_first, \_count ) \_\_HTRCXSendSerial(\_first, \_count)**

HTRCXSendSerial function.

Send the SendSerial command to an RCX.

**Parameters**

<i>_first</i>	The first byte address.
<i>_count</i>	The number of bytes to send.

**5.18.2.51 #define HTRCXSetDirection( \_outputs, \_dir ) \_\_HTRCXSetDirection(\_outputs, \_dir)**

HTRCXSetDirection function.

Send the SetDirection command to an RCX to configure the direction of the specified outputs.

## Parameters

<i>_outputs</i>	The RCX output(s) to set direction. See <a href="#">RCX output constants</a> .
<i>_dir</i>	The RCX output direction. See <a href="#">RCX output direction constants</a> .

5.18.2.52 #define HTRCXSetEvent( *\_evt*, *\_src*, *\_type* ) \_\_HTRCXSetEvent(*\_evt*, *\_src*, *\_type*)

HTRCXSetEvent function.

Send the SetEvent command to an RCX.

## Parameters

<i>_evt</i>	The event number to set.
<i>_src</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_type</i>	The event type.

5.18.2.53 #define HTRCXSetGlobalDirection( *\_outputs*, *\_dir* ) \_\_HTRCXSetGlobalDirection(*\_outputs*, *\_dir*)

HTRCXSetGlobalDirection function.

Send the SetGlobalDirection command to an RCX.

## Parameters

<i>_outputs</i>	The RCX output(s) to set global direction. See <a href="#">RCX output constants</a> .
<i>_dir</i>	The RCX output direction. See <a href="#">RCX output direction constants</a> .

5.18.2.54 #define HTRCXSetGlobalOutput( *\_outputs*, *\_mode* ) \_\_HTRCXSetGlobalOutput(*\_outputs*, *\_mode*)

HTRCXSetGlobalOutput function.

Send the SetGlobalOutput command to an RCX.

## Parameters

<i>_outputs</i>	The RCX output(s) to set global mode. See <a href="#">RCX output constants</a> .
<i>_mode</i>	The RCX output mode. See <a href="#">RCX output mode constants</a> .

5.18.2.55 #define HTRCXSetIRLinkPort( *\_port* ) \_\_HTRCXSetIRLinkPort(*\_port*)

HTRCXSetIRLinkPort function.

Set the global port in advance of using the HTRCX\* and HTScout\* API functions for sending RCX and Scout messages over the HiTechnic iRLink device. The port must be configured as a Lowspeed port before using any of the HiTechnic RCX and Scout iRLink functions.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
--------------	---

5.18.2.56 #define HTRCXSetMaxPower( *\_outputs*, *\_pwrsrc*, *\_pwrval* ) \_\_HTRCXSetMaxPower(*\_outputs*, *\_pwrsrc*, *\_pwrval*)

HTRCXSetMaxPower function.

Send the SetMaxPower command to an RCX.

**Parameters**

<i>_outputs</i>	The RCX output(s) to set max power. See <a href="#">RCX output constants</a> .
<i>_pwrsrc</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_pwrval</i>	The RCX value.

5.18.2.57 #define HTRCXSetMessage( *\_msg* ) \_\_HTRCXSetMessage(*\_msg*)

HTRCXSetMessage function.

Send the SetMessage command to an RCX.

**Parameters**

<i>_msg</i>	The numeric message to send.
-------------	------------------------------

5.18.2.58 #define HTRCXSetOutput( *\_outputs*, *\_mode* ) \_\_HTRCXSetOutput(*\_outputs*, *\_mode*)

HTRCXSetOutput function.

Send the SetOutput command to an RCX to configure the mode of the specified outputs

**Parameters**

<i>_outputs</i>	The RCX output(s) to set mode. See <a href="#">RCX output constants</a> .
<i>_mode</i>	The RCX output mode. See <a href="#">RCX output mode constants</a> .

5.18.2.59 #define HTRCXSetPower( *\_outputs*, *\_pwrsrc*, *\_pwrval* ) \_\_HTRCXSetPower(*\_outputs*, *\_pwrsrc*, *\_pwrval*)

HTRCXSetPower function.

Send the SetPower command to an RCX to configure the power level of the specified outputs.

**Parameters**

<i>_outputs</i>	The RCX output(s) to set power. See <a href="#">RCX output constants</a> .
<i>_pwrsrc</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_pwrval</i>	The RCX value.

5.18.2.60 #define HTRCXSetPriority( *\_p* ) \_\_HTRCXSetPriority(*\_p*)

HTRCXSetPriority function.

Send the SetPriority command to an RCX.

**Parameters**

<i>_p</i>	The new task priority.
-----------	------------------------

5.18.2.61 #define HTRCXSetSensorMode( *\_port*, *\_mode* ) \_\_HTRCXSetSensorMode(*\_port*, *\_mode*)

HTRCXSetSensorMode function.

Send the SetSensorMode command to an RCX.

**Parameters**

<i>_port</i>	The RCX sensor port.
<i>_mode</i>	The RCX sensor mode.

5.18.2.62 #define HTRCXSetSensorType( *\_port*, *\_type* ) \_\_HTRCXSetSensorType(*\_port*, *\_type*)

HTRCXSetSensorType function.

Send the SetSensorType command to an RCX.

**Parameters**

<i>_port</i>	The RCX sensor port.
<i>_type</i>	The RCX sensor type.

5.18.2.63 #define HTRCXSetSleepTime( *\_t* ) \_\_HTRCXSetSleepTime(*\_t*)

HTRCXSetSleepTime function.

Send the SetSleepTime command to an RCX.

**Parameters**

<i>_t</i>	The new sleep time value.
-----------	---------------------------

5.18.2.64 #define HTRCXSetTxPower( *\_pwr* ) \_\_HTRCXSetTxPower(*\_pwr*)

HTRCXSetTxPower function.

Send the SetTxPower command to an RCX.

**Parameters**

<i>_pwr</i>	The IR transmit power level.
-------------	------------------------------

5.18.2.65 #define HTRCXSetWatch( *\_hours*, *\_minutes* ) \_\_HTRCXSetWatch(*\_hours*, *\_minutes*)

HTRCXSetWatch function.

Send the SetWatch command to an RCX.

**Parameters**

<i>_hours</i>	The new watch time hours value.
<i>_minutes</i>	The new watch time minutes value.

5.18.2.66 #define HTRCXStartTask( *\_t* ) \_\_HTRCXStartTask(*\_t*)

HTRCXStartTask function.

Send the StartTask command to an RCX.

**Parameters**

<i>_t</i>	The task number to start.
-----------	---------------------------

5.18.2.67 #define HTRCXStopAllTasks( ) \_\_HTRCXOpNoArgs(RCX\_StopAllTasksOp)

HTRCXStopAllTasks function.

Send the StopAllTasks command to an RCX.

5.18.2.68 #define HTRCXStopTask( \_t ) \_\_HTRCXStopTask(\_t)

HTRCXStopTask function.

Send the StopTask command to an RCX.

Parameters

<i>_t</i>	The task number to stop.
-----------	--------------------------

5.18.2.69 #define HTRCXToggle( \_outputs ) \_\_HTRCXSetDirection(\_outputs, RCX\_OUT\_TOGGLE)

HTRCXToggle function.

Send commands to an RCX to toggle the direction of the specified outputs.

Parameters

<i>_outputs</i>	The RCX output(s) to toggle. See <a href="#">RCX output constants</a> .
-----------------	---

5.18.2.70 #define HTRCXUnmuteSound( ) \_\_HTRCXOpNoArgs(RCX\_UnmuteSoundOp)

HTRCXUnmuteSound function.

Send the UnmuteSound command to an RCX.

5.18.2.71 #define HTScoutCalibrateSensor( ) \_\_HTRCXOpNoArgs(RCX\_LSCalibrateOp)

HTScoutCalibrateSensor function.

Send the CalibrateSensor command to a Scout.

5.18.2.72 #define HTScoutMuteSound( ) \_\_HTScoutMuteSound()

HTScoutMuteSound function.

Send the MuteSound command to a Scout.

5.18.2.73 #define HTScoutSelectSounds( \_grp ) \_\_HTScoutSelectSounds(\_grp)

HTScoutSelectSounds function.

Send the SelectSounds command to a Scout.

Parameters

<i>_grp</i>	The Scout sound group to select.
-------------	----------------------------------

5.18.2.74 #define HTScoutSendVLL( \_src, \_value ) \_\_HTScoutSendVLL(\_src, \_value)

HTScoutSendVLL function.

Send the SendVLL command to a Scout.

## Parameters

<code>_src</code>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The Scout value.

**5.18.2.75 #define HTScoutSetEventFeedback( `_src`, `_value` ) \_\_HTScoutSetEventFeedback(`_src`, `_value`)**

HTScoutSetEventFeedback function.

Send the SetEventFeedback command to a Scout.

## Parameters

<code>_src</code>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The Scout value.

**5.18.2.76 #define HTScoutSetLight( `_x` ) \_\_HTScoutSetLight(`_x`)**

HTScoutSetLight function.

Send the SetLight command to a Scout.

## Parameters

<code>_x</code>	Set the light on or off using this value. See <a href="#">Scout light constants</a> .
-----------------	---

**5.18.2.77 #define HTScoutSetScoutMode( `_mode` ) \_\_HTScoutSetScoutMode(`_mode`)**

HTScoutSetScoutMode function.

Send the SetScoutMode command to a Scout.

## Parameters

<code>_mode</code>	Set the scout mode. See <a href="#">Scout mode constants</a> .
--------------------	--

**5.18.2.78 #define HTScoutSetSensorClickTime( `_src`, `_value` ) \_\_HTScoutSetSensorClickTime(`_src`, `_value`)**

HTScoutSetSensorClickTime function.

Send the SetSensorClickTime command to a Scout.

## Parameters

<code>_src</code>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The Scout value.

**5.18.2.79 #define HTScoutSetSensorHysteresis( `_src`, `_value` ) \_\_HTScoutSetSensorHysteresis(`_src`, `_value`)**

HTScoutSetSensorHysteresis function.

Send the SetSensorHysteresis command to a Scout.

## Parameters

<code>_src</code>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The Scout value.

**5.18.2.80 #define HTScoutSetSensorLowerLimit( \_src, \_value ) \_\_HTScoutSetSensorLowerLimit(\_src, \_value)**

HTScoutSetSensorLowerLimit function.

Send the SetSensorLowerLimit command to a Scout.

**Parameters**

<u>_src</u>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<u>_value</u>	The Scout value.

**5.18.2.81 #define HTScoutSetSensorUpperLimit( \_src, \_value ) \_\_HTScoutSetSensorUpperLimit(\_src, \_value)**

HTScoutSetSensorUpperLimit function.

Send the SetSensorUpperLimit command to a Scout.

**Parameters**

<u>_src</u>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<u>_value</u>	The Scout value.

**5.18.2.82 #define HTScoutUnmuteSound( ) \_\_HTScoutUnmuteSound()**

HTScoutUnmuteSound function.

Send the UnmuteSound command to a Scout.

**5.18.2.83 #define ReadSensorHTAccel( \_port, \_x, \_y, \_z, \_result ) \_\_ReadSensorHTAccel(\_port, \_x, \_y, \_z, \_result)**

Read HiTechnic acceleration values.

Read X, Y, and Z axis acceleration values from the HiTechnic Accelerometer sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

**Parameters**

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_x</u>	The output x-axis acceleration.
<u>_y</u>	The output y-axis acceleration.
<u>_z</u>	The output z-axis acceleration.
<u>_result</u>	The function call result.

**5.18.2.84 #define ReadSensorHTAngle( \_port, \_Angle, \_AccAngle, \_RPM, \_result ) \_\_ReadSensorHTAngle(\_port, \_Angle, \_AccAngle, \_RPM, \_result)**

Read HiTechnic Angle sensor values.

Read values from the HiTechnic Angle sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

**Parameters**

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_Angle</u>	Current angle in degrees (0-359).
<u>_AccAngle</u>	Accumulated angle in degrees (-2147483648 to 2147483647).
<u>_RPM</u>	rotations per minute (-1000 to 1000).

<i>_result</i>	The function call result.
----------------	---------------------------

5.18.2.85 #define ReadSensorHTBarometric( *\_port*, *\_temp*, *\_press*, *\_result* ) \_\_ReadSensorHTBarometric(*\_port*, *\_temp*, *\_press*, *\_result*)

Read HiTechnic Barometric sensor values.

Read values from the HiTechnic Barometric sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_temp</i>	Current temperature in 1/10ths of degrees Celcius.
<i>_press</i>	Current barometric pressure in 1/1000 inches of mercury.
<i>_result</i>	The function call result.

5.18.2.86 #define ReadSensorHTColor( *\_port*, *\_ColorNum*, *\_Red*, *\_Green*, *\_Blue*, *\_result* ) \_\_ReadSensorHTColor(*\_port*, *\_ColorNum*, *\_Red*, *\_Green*, *\_Blue*, *\_result*)

Read HiTechnic Color values.

Read color number, red, green, and blue values from the HiTechnic Color sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_ColorNum</i>	The output color number.
<i>_Red</i>	The red color value.
<i>_Green</i>	The green color value.
<i>_Blue</i>	The blue color value.
<i>_result</i>	The function call result.

5.18.2.87 #define ReadSensorHTColor2Active( *\_port*, *\_ColorNum*, *\_Red*, *\_Green*, *\_Blue*, *\_White*, *\_result* ) \_\_ReadSensorHTColor2Active(*\_port*, *\_ColorNum*, *\_Red*, *\_Green*, *\_Blue*, *\_White*, *\_result*)

Read HiTechnic Color2 active values.

Read color number, red, green, and blue values from the HiTechnic Color2 sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_ColorNum</i>	The output color number.
<i>_Red</i>	The red color value.
<i>_Green</i>	The green color value.
<i>_Blue</i>	The blue color value.
<i>_White</i>	The white color value.
<i>_result</i>	The function call result.

**5.18.2.88 #define ReadSensorHTColorNum( \_port, \_value ) \_\_ReadSensorHTColorNum(\_port, \_value)**

Read HiTechnic color sensor color number.

Read the color number from the HiTechnic Color sensor on the specified port. The port must be configured as a Lowspeed port before using this function.

**Parameters**

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_value</u>	The color number.

**5.18.2.89 #define ReadSensorHTCompass( \_port, \_value ) \_\_ReadSensorHTCompass(\_port, \_value)**

Read HiTechnic compass.

Read the compass heading value of the HiTechnic Compass sensor on the specified port. The port must be configured as a Lowspeed port before using this function.

**Parameters**

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_value</u>	The compass heading.

**5.18.2.90 #define ReadSensorHTEOPD( \_port, \_val ) \_\_ReadSensorHTEOPD(\_port, \_val)**

Read HiTechnic EOPD sensor.

Read the HiTechnic EOPD sensor on the specified port.

**Parameters**

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_val</u>	The EOPD sensor reading.

**5.18.2.91 #define ReadSensorHTForce( \_p, \_val ) \_\_ReadSensorHTForce(\_p, \_val)**

Read HiTechnic Force sensor.

Read the HiTechnic Force sensor on the specified port.

**Parameters**

<u>_p</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_val</u>	The Force sensor reading.

**5.18.2.92 #define ReadSensorHTGyro( \_p, \_offset, \_val ) \_\_ReadSensorHTGyro(\_p, \_offset, \_val)**

Read HiTechnic Gyro sensor.

Read the HiTechnic Gyro sensor on the specified port. The offset value should be calculated by averaging several readings with an offset of zero while the sensor is perfectly still.

**Parameters**

<u>_p</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_offset</u>	The zero offset.
<u>_val</u>	The Gyro sensor reading.

**5.18.2.93 #define ReadSensorHTIRReceiver( \_port, \_pfdata, \_result ) \_\_ReadSensorHTIRReceiver(\_port, \_pfdata, \_result)**

Read HiTechnic IRReceiver Power Function bytes.

Read Power Function bytes from the HiTechnic IRReceiver sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

**Parameters**

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_pfdata</i>	Eight bytes of power function remote IR data.
<i>_result</i>	The function call result.

**5.18.2.94 #define ReadSensorHTIRReceiverEx( \_port, \_reg, \_pfchar, \_result ) \_\_ReadSensorHTIRReceiverEx(\_port, \_reg, \_pfchar, \_result)**

Read HiTechnic IRReceiver Power Function value.

Read a Power Function byte from the HiTechnic IRReceiver sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

**Parameters**

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_reg</i>	The power function data offset. See <a href="#">HiTechnic IRReceiver constants</a> .
<i>_pfchar</i>	A single byte of power function remote IR data.
<i>_result</i>	The function call result.

**5.18.2.95 #define ReadSensorHTIRSeeker( \_port, \_dir, \_s1, \_s3, \_s5, \_s7, \_s9, \_result ) \_\_ReadSensorHTIRSeeker(\_port, \_dir, \_s1, \_s3, \_s5, \_s7, \_s9, \_result)**

Read HiTechnic IRSeeker values.

Read direction, and five signal strength values from the HiTechnic IRSeeker sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

**Parameters**

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_dir</i>	The direction.
<i>_s1</i>	The signal strength from sensor 1.
<i>_s3</i>	The signal strength from sensor 3.
<i>_s5</i>	The signal strength from sensor 5.
<i>_s7</i>	The signal strength from sensor 7.
<i>_s9</i>	The signal strength from sensor 9.
<i>_result</i>	The function call result.

**5.18.2.96 #define ReadSensorHTIRSeeker2AC( \_port, \_dir, \_s1, \_s3, \_s5, \_s7, \_s9, \_result ) \_\_ReadSensorHTIRSeeker2AC(\_port, \_dir, \_s1, \_s3, \_s5, \_s7, \_s9, \_result)**

Read HiTechnic IRSeeker2 AC values.

Read direction, and five signal strength values from the HiTechnic IRSeeker2 sensor in AC mode. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_dir</code>	The direction.
<code>_s1</code>	The signal strength from sensor 1.
<code>_s3</code>	The signal strength from sensor 3.
<code>_s5</code>	The signal strength from sensor 5.
<code>_s7</code>	The signal strength from sensor 7.
<code>_s9</code>	The signal strength from sensor 9.
<code>_result</code>	The function call result.

5.18.2.97 #define `ReadSensorHTIRSeeker2Addr( _port, _reg, _value ) __ReadSensorHTIRSeeker2Addr(_port, _reg, _value)`

Read HiTechnic IRSeeker2 register.

Read a register value from the HiTechnic IR Seeker2 on the specified port. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_reg</code>	The register address. See <a href="#">HiTechnic IRSeeker2 constants</a> .
<code>_value</code>	The IRSeeker2 register value.

5.18.2.98 #define `ReadSensorHTIRSeeker2DC( _port, _dir, _s1, _s3, _s5, _s7, _s9, _avg, _result ) __ReadSensorHTIRSeeker2DC(_port, _dir, _s1, _s3, _s5, _s7, _s9, _avg, _result)`

Read HiTechnic IRSeeker2 DC values.

Read direction, five signal strength, and average strength values from the HiTechnic IRSeeker2 sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_dir</code>	The direction.
<code>_s1</code>	The signal strength from sensor 1.
<code>_s3</code>	The signal strength from sensor 3.
<code>_s5</code>	The signal strength from sensor 5.
<code>_s7</code>	The signal strength from sensor 7.
<code>_s9</code>	The signal strength from sensor 9.
<code>_avg</code>	The average signal strength.
<code>_result</code>	The function call result.

5.18.2.99 #define `ReadSensorHTIRSeekerDir( _port, _value ) __ReadSensorHTIRSeekerDir(_port, _value)`

Read HiTechnic IRSeeker direction.

Read the direction value of the HiTechnic IR Seeker on the specified port. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_value</code>	The IRSeeker direction.

---

```
5.18.2.100 #define ReadSensorHTMagnet( _p, _offset, _val ) __ReadSensorHTGyro(_p, _offset, _val)
```

Read HiTechnic Magnet sensor.

Read the HiTechnic Magnet sensor on the specified port. The offset value should be calculated by averaging several readings with an offset of zero while the sensor is perfectly still.

#### Parameters

<code>_p</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_offset</code>	The zero offset.
<code>_val</code>	The Magnet sensor reading.

---

```
5.18.2.101 #define ReadSensorHTNormalizedColor( _port, _ColorIdx, _Red, _Green, _Blue, _result
) __ReadSensorHTNormalizedColor(_port, _ColorIdx, _Red, _Green, _Blue, _result)
```

Read HiTechnic Color normalized values.

Read the color index and the normalized red, green, and blue values from the HiTechnic Color sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_ColorIdx</code>	The output color index.
<code>_Red</code>	The normalized red color value.
<code>_Green</code>	The normalized green color value.
<code>_Blue</code>	The normalized blue color value.
<code>_result</code>	The function call result.

---

```
5.18.2.102 #define ReadSensorHTNormalizedColor2Active( _port, _ColorIdx, _Red, _Green, _Blue, _result
) __ReadSensorHTNormalizedColor2Active(_port, _ColorIdx, _Red, _Green, _Blue, _result)
```

Read HiTechnic Color2 normalized active values.

Read the color index and the normalized red, green, and blue values from the HiTechnic Color2 sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_ColorIdx</code>	The output color index.
<code>_Red</code>	The normalized red color value.
<code>_Green</code>	The normalized green color value.
<code>_Blue</code>	The normalized blue color value.
<code>_result</code>	The function call result.

---

```
5.18.2.103 #define ReadSensorHTPIR( _port, _out, _result ) __MSReadValue(_port, HT_ADDR_PIR, HTPIR_REG_READING,
1, _out, _result)
```

Read HiTechnic SuperPro strobe value.

Read the HiTechnic SuperPro strobe value. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_out</code>	The strobe value. See <a href="#">SuperPro Strobe control constants</a> .
<code>_result</code>	The function call result.

**5.18.2.104 #define ReadSensorHTProtoAllAnalog( `_port`, `_a0`, `_a1`, `_a2`, `_a3`, `_a4`, `_result` ) \_\_ReadSensorHTProtoAllAnalog(`_port`, `_a0`, `_a1`, `_a2`, `_a3`, `_a4`, `_result`)**

Read all HiTechnic Prototype board analog input values.

Read all 5 analog input values from the HiTechnic prototype board. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_a0</code>	The A0 analog input value.
<code>_a1</code>	The A1 analog input value.
<code>_a2</code>	The A2 analog input value.
<code>_a3</code>	The A3 analog input value.
<code>_a4</code>	The A4 analog input value.
<code>_result</code>	The function call result.

**5.18.2.105 #define ReadSensorHTProtoAnalog( `_port`, `_input`, `_value`, `_result` ) \_\_ReadSensorHTProtoAnalog(`_port`, `HT_ADDR_PROTOBOARD`, `_input`, `_value`, `_result`)**

Read HiTechnic Prototype board analog input value.

Read an analog input value from the HiTechnic prototype board. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_input</code>	The analog input. See <a href="#">HiTechnic Prototype board analog input constants</a> .
<code>_value</code>	The analog input value.
<code>_result</code>	The function call result.

**5.18.2.106 #define ReadSensorHTProtoDigital( `_port`, `_value`, `_result` ) \_\_ReadSensorHTProtoDigital(`_port`, `HT_ADDR_PROTOBOARD`, `_value`, `_result`)**

Read HiTechnic Prototype board digital input values.

Read digital input values from the HiTechnic prototype board. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_value</code>	The digital input values. See <a href="#">SuperPro digital pin constants</a> .
<code>_result</code>	The function call result.

```
5.18.2.107 #define ReadSensorHTProtoDigitalControl( _port, _out, _result ) __MSReadValue(_port, HT_ADDR_PROTOBOARD,  
HTPROTO_REG_DCTRL, 1, _out, _result)
```

Read HiTechnic Prototype board digital pin control value.

Read the HiTechnic prototype board digital control value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_out</i>	The digital control value. See <a href="#">SuperPro LED control constants</a> .
<i>_result</i>	The function call result.

```
5.18.2.108 #define ReadSensorHTRawColor( _port, _Red, _Green, _Blue, _result ) __ReadSensorHTRawColor(_port, _Red,  
_Green, _Blue, _result)
```

Read HiTechnic Color raw values.

Read the raw red, green, and blue values from the HiTechnic Color sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_Red</i>	The raw red color value.
<i>_Green</i>	The raw green color value.
<i>_Blue</i>	The raw blue color value.
<i>_result</i>	The function call result.

```
5.18.2.109 #define ReadSensorHTRawColor2( _port, _Red, _Green, _Blue, _White, _result ) __ReadSensorHTRawColor2(_port,  
_Red, _Green, _Blue, _White, _result)
```

Read HiTechnic Color2 raw values.

Read the raw red, green, and blue values from the HiTechnic Color2 sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_Red</i>	The raw red color value.
<i>_Green</i>	The raw green color value.
<i>_Blue</i>	The raw blue color value.
<i>_White</i>	The raw white color value.
<i>_result</i>	The function call result.

```
5.18.2.110 #define ReadSensorHTSuperProAllAnalog( _port, _a0, _a1, _a2, _a3, _result  
 ) __ReadSensorHTSuperProAllAnalog(_port, _a0, _a1, _a2, _a3, _result)
```

Read all HiTechnic SuperPro board analog input values.

Read all 4 analog input values from the HiTechnic SuperPro board. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_a0</code>	The A0 analog input value.
<code>_a1</code>	The A1 analog input value.
<code>_a2</code>	The A2 analog input value.
<code>_a3</code>	The A3 analog input value.
<code>_result</code>	The function call result.

```
5.18.2.111 #define ReadSensorHTSuperProAnalog( _port, _input, _value, _result ) __ReadSensorHTProtoAnalog(_port,  
HT_ADDR_SUPERPRO,_input,_value,_result)
```

Read HiTechnic SuperPro board analog input value.

Read an analog input value from the HiTechnic SuperPro board. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_input</code>	The analog input. See <a href="#">HiTechnic SuperPro analog input index constants</a> .
<code>_value</code>	The analog input value.
<code>_result</code>	The function call result.

```
5.18.2.112 #define ReadSensorHTSuperProAnalogOut( _port, _dac, _mode, _freq, _volt, _result  
 ) __ReadSensorHTSuperProAnalogOut(_port, _dac, _mode, _freq, _volt, _result)
```

Read HiTechnic SuperPro board analog output parameters.

Read the analog output parameters on the HiTechnic SuperPro board. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_dac</code>	The analog output index. See <a href="#">HiTechnic SuperPro analog output index constants</a> .
<code>_mode</code>	The analog output mode. See <a href="#">SuperPro analog output mode constants</a> .
<code>_freq</code>	The analog output frequency. Between 1 and 8191.
<code>_volt</code>	The analog output voltage level. A 10 bit value (0..1023).
<code>_result</code>	The function call result.

```
5.18.2.113 #define ReadSensorHTSuperProDigital( _port, _value, _result ) __ReadSensorHTProtoDigital(_port,  
HT_ADDR_SUPERPRO,_value,_result)
```

Read HiTechnic SuperPro board digital input values.

Read digital input values from the HiTechnic SuperPro board. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_value</code>	The digital input values. See <a href="#">SuperPro digital pin constants</a> .
<code>_result</code>	The function call result.

```
5.18.2.114 #define ReadSensorHTSuperProDigitalControl( _port, _out, _result ) __MSReadValue(_port, HT_ADDR_SUPERPRO,  
HTSPRO_REG_DCTRL, 1, _out, _result)
```

Read HiTechnic SuperPro digital control value.

Read the HiTechnic SuperPro digital control value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_out</i>	The digital control value. See <a href="#">SuperPro LED control constants</a> .
<i>_result</i>	The function call result.

```
5.18.2.115 #define ReadSensorHTSuperProLED( _port, _out, _result ) __MSReadValue(_port, HT_ADDR_SUPERPRO,  
HTSPRO_REG_LED, 1, _out, _result)
```

Read HiTechnic SuperPro LED value.

Read the HiTechnic SuperPro LED value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_out</i>	The LED value. See <a href="#">SuperPro LED control constants</a> .
<i>_result</i>	The function call result.

```
5.18.2.116 #define ReadSensorHTSuperProProgramControl( _port, _out, _result ) __MSReadValue(_port,  
HT_ADDR_SUPERPRO, HTSPRO_REG_CTRL, 1, _out, _result)
```

Read HiTechnic SuperPro program control value.

Read the HiTechnic SuperPro program control value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_out</i>	The program control value.
<i>_result</i>	The function call result.

```
5.18.2.117 #define ReadSensorHTSuperProStrobe( _port, _out, _result ) __MSReadValue(_port, HT_ADDR_SUPERPRO,  
HTSPRO_REG_STROBE, 1, _out, _result)
```

Read HiTechnic SuperPro strobe value.

Read the HiTechnic SuperPro strobe value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_out</i>	The strobe value. See <a href="#">SuperPro Strobe control constants</a> .
<i>_result</i>	The function call result.

5.18.2.118 #define ReadSensorHTTouchMultiplexer( *\_p*, *\_t1*, *\_t2*, *\_t3*, *\_t4* ) \_\_ReadSensorHTTouchMultiplexer(*\_p*, *\_t1*, *\_t2*, *\_t3*, *\_t4*)

Read HiTechnic touch multiplexer.

Read touch sensor values from the HiTechnic touch multiplexer device.

#### Parameters

<i>_p</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_t1</i>	The value of touch sensor 1.
<i>_t2</i>	The value of touch sensor 2.
<i>_t3</i>	The value of touch sensor 3.
<i>_t4</i>	The value of touch sensor 4.

5.18.2.119 #define ResetHTBarometricCalibration( *\_port*, *\_result* ) \_\_ResetHTBarometricCalibration(*\_port*, *\_result*)

Reset HiTechnic Barometric sensor calibration.

Reset the HiTechnic Barometric sensor to its factory calibration. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_result</i>	The function call result.

5.18.2.120 #define ResetSensorHTAngle( *\_port*, *\_mode*, *\_result* ) \_\_ResetSensorHTAngle(*\_port*, *\_mode*, *\_result*)

Reset HiTechnic Angle sensor.

Reset the HiTechnic Angle sensor on the specified port. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_mode</i>	The Angle reset mode. See <a href="#">HiTechnic Angle sensor constants</a> .
<i>_result</i>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

5.18.2.121 #define SetHTBarometricCalibration( *\_port*, *\_cal*, *\_result* ) \_\_SetHTBarometricCalibration(*\_port*, *\_cal*, *\_result*)

Set HiTechnic Barometric sensor calibration.

Set the HiTechnic Barometric sensor pressure calibration value. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_cal</i>	The new pressure calibration value.
<i>_result</i>	The function call result.

5.18.2.122 #define SetHTColor2Mode( *\_port*, *\_mode*, *\_result* ) \_\_SetHTColor2Mode(*\_port*, *\_mode*, *\_result*)

Set HiTechnic Color2 mode.

Set the mode of the HiTechnic Color2 sensor on the specified port. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_mode</code>	The Color2 mode. See <a href="#">HiTechnic Color2 constants</a> .
<code>_result</code>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

5.18.2.123 `#define SetHTIRSeeker2Mode( _port, _mode, _result ) __SetHTIRSeeker2Mode(_port, _mode, _result)`

Set HiTechnic IRSeeker2 mode.

Set the mode of the HiTechnic IRSeeker2 sensor on the specified port. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_mode</code>	The IRSeeker2 mode. See <a href="#">HiTechnic IRSeeker2 constants</a> .
<code>_result</code>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

5.18.2.124 `#define SetSensorHTEOPD( _port, _bStd ) __SetSensorHTEOPD(_port, _bStd)`

Set sensor as HiTechnic EOPD.

Configure the sensor on the specified port as a HiTechnic EOPD sensor.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_bStd</code>	Configure in standard or long-range mode.

5.18.2.125 `#define SetSensorHTForce( _port ) __SetSensorHTForce(_port)`

Set sensor as HiTechnic Force.

Configure the sensor on the specified port as a HiTechnic Force sensor.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
--------------------	---

5.18.2.126 `#define SetSensorHTGyro( _port ) __SetSensorHTGyro(_port)`

Set sensor as HiTechnic Gyro.

Configure the sensor on the specified port as a HiTechnic Gyro sensor.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
--------------------	---

5.18.2.127 `#define SetSensorHTMagnet( _port ) __SetSensorHTGyro(_port)`

Set sensor as HiTechnic Magnet.

Configure the sensor on the specified port as a HiTechnic Magnet sensor.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
--------------------	---

**5.18.2.128 #define SetSensorHTPIRDeadband( `_port`, `_value`, `_result` ) \_\_SetSensorHTPIRDeadband(`_port`, `_value`, `_result`)**

Set HiTechnic PIR deadband value.

Set the HiTechnic PIR deadband value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_value</code>	The deadband value. Valid values are 0..47.
<code>_result</code>	The function call result.

**5.18.2.129 #define SetSensorHTProtoDigital( `_port`, `_value`, `_result` ) \_\_SetSensorHTProtoDigital(`_port`,  
HT\_ADDR\_PROTOBOARD, `_value`, `_result`)**

Set HiTechnic Prototype board digital output values.

Set the digital pin output values on the HiTechnic prototype board. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_value</code>	The digital pin output values. See <a href="#">SuperPro digital pin constants</a> .
<code>_result</code>	The function call result.

**5.18.2.130 #define SetSensorHTProtoDigitalControl( `_port`, `_value`, `_result` ) \_\_SetSensorHTProtoDigitalControl(`_port`,  
HT\_ADDR\_PROTOBOARD, `_value`, `_result`)**

Set HiTechnic Prototype board digital pin direction.

Set which of the six digital pins on the HiTechnic prototype board should be outputs. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_value</code>	The digital pin control value. See <a href="#">SuperPro digital pin constants</a> . OR into this value the pins that you want to be output pins. The pins not included in the value will be input pins.
<code>_result</code>	The function call result.

**5.18.2.131 #define SetSensorHTSuperProAnalogOut( `_port`, `_dac`, `_mode`, `_freq`, `_volt`, `_result` ) \_\_SetSensorHTSuperProAnalogOut(`_port`, `_dac`, `_mode`, `_freq`, `_volt`, `_result`)**

Set HiTechnic SuperPro board analog output parameters.

Set the analog output parameters on the HiTechnic SuperPro board. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_dac</code>	The analog output index. See <a href="#">HiTechnic SuperPro analog output index constants</a> .
<code>_mode</code>	The analog output mode. See <a href="#">SuperPro analog output mode constants</a> .
<code>_freq</code>	The analog output frequency. Between 1 and 8191.
<code>_volt</code>	The analog output voltage level. A 10 bit value (0..1023).
<code>_result</code>	The function call result.

5.18.2.132 `#define SetSensorHTSuperProDigital( _port, _value, _result ) __SetSensorHTProtoDigital(_port, HT_ADDR_SUPERPRO, _value, _result)`

Set HiTechnic SuperPro board digital output values.

Set the digital pin output values on the HiTechnic SuperPro board. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_value</code>	The digital pin output values. See <a href="#">SuperPro digital pin constants</a> .
<code>_result</code>	The function call result.

5.18.2.133 `#define SetSensorHTSuperProDigitalControl( _port, _value, _result ) __SetSensorHTProtoDigitalControl(_port, HT_ADDR_SUPERPRO, _value, _result)`

Control HiTechnic SuperPro board digital pin direction.

Control the direction of the eight digital pins on the HiTechnic SuperPro board. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_value</code>	The digital pin control value. See <a href="#">SuperPro digital pin constants</a> . OR into this value the pins that you want to be output pins. The pins not included in the value will be input pins.
<code>_result</code>	The function call result.

5.18.2.134 `#define SetSensorHTSuperProLED( _port, _value, _result ) __SetSensorHTSuperProLED(_port, _value, _result)`

Set HiTechnic SuperPro LED value.

Set the HiTechnic SuperPro LED value. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_value</code>	The LED value. See <a href="#">SuperPro LED control constants</a> .
<code>_result</code>	The function call result.

5.18.2.135 `#define SetSensorHTSuperProProgramControl( _port, _value, _result ) __SetSensorHTSuperProProgramControl(_port, _value, _result)`

Set HiTechnic SuperPro program control value.

Set the HiTechnic SuperPro program control value. The port must be configured as a Lowspeed port before using this function.

**Parameters**

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_value</i>	The program control value.
<i>_result</i>	The function call result.

**5.18.2.136 #define SetSensorHTSuperProStrobe( *\_port*, *\_value*, *\_result* ) \_\_SetSensorHTSuperProStrobe(*\_port*, *\_value*, *\_result*)**

Set HiTechnic SuperPro strobe value.

Set the HiTechnic SuperPro strobe value. The port must be configured as a Lowspeed port before using this function.

**Parameters**

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_value</i>	The strobe value. See <a href="#">SuperPro Strobe control constants</a> .
<i>_result</i>	The function call result.

## 5.19 SuperPro analog output mode constants

Constants for controlling the 2 analog output modes.

### Macros

- #define DAC\_MODE\_DCOUT 0
- #define DAC\_MODE\_SINEWAVE 1
- #define DAC\_MODE\_SQUAREWAVE 2
- #define DAC\_MODE\_SAWPOSWAVE 3
- #define DAC\_MODE\_SAWNEGWAVE 4
- #define DAC\_MODE\_TRIANGLEWAVE 5
- #define DAC\_MODE\_PWMVOLTAGE 6
- #define DAC\_MODE\_RESTART\_MASK 0x80

### 5.19.1 Detailed Description

Constants for controlling the 2 analog output modes. Two analog outputs, which can span 0 to 3.3 volts, can be programmed to output a steady voltage or can be programmed to output a selection of waveforms over a range of frequencies.

In the DC output mode, the DAC0/DAC1 voltage fields control the voltage on the two analog outputs in increments of ~3.2mV from 0 - 1023 giving 0 - 3.3v.

In waveform modes, the channel outputs will center on 1.65 volts when generating waveforms. The DAC0/DAC1 voltage fields control the signal levels of the waveforms by adjusting the peak to peak signal levels from 0 - 3.3v.

In PWFM voltage mode, the channel outputs will create a variable mark:space ratio square wave at 3.3v signal level. The average output voltage is set by the O0/O1 voltage fields.

### 5.19.2 Macro Definition Documentation

#### 5.19.2.1 #define DAC\_MODE\_DCOUT 0

Steady (DC) voltage output.

#### 5.19.2.2 #define DAC\_MODE\_PWMVOLTAGE 6

PWM square wave output.

#### 5.19.2.3 #define DAC\_MODE\_RESTART\_MASK 0x80

Add mask to DAC mode constants to force waveform generation from the start of the wave table.

#### 5.19.2.4 #define DAC\_MODE\_SAWNEGWAVE 4

Negative going sawtooth output.

#### 5.19.2.5 #define DAC\_MODE\_SAWPOSWAVE 3

Positive going sawtooth output.

#### 5.19.2.6 #define DAC\_MODE\_SINEWAVE 1

Sine wave output.

5.19.2.7 #define DAC\_MODE\_SQUAREWAVE 2

Square wave output.

5.19.2.8 #define DAC\_MODE\_TRIANGLEWAVE 5

Triangle wave output.

## 5.20 SuperPro LED control constants

Constants for controlling the 2 onboard LEDs.

### Macros

- `#define LED_BLUE 0x02`
- `#define LED_RED 0x01`
- `#define LED_NONE 0x00`

#### 5.20.1 Detailed Description

Constants for controlling the 2 onboard LEDs.

#### 5.20.2 Macro Definition Documentation

##### 5.20.2.1 `#define LED_BLUE 0x02`

Turn on the blue onboard LED.

##### 5.20.2.2 `#define LED_NONE 0x00`

Turn off the onboard LEDs.

##### 5.20.2.3 `#define LED_RED 0x01`

Turn on the red onboard LED.

## 5.21 SuperPro digital pin constants

Constants for controlling the 8 digital pins.

### Macros

- #define DIGI\_PIN0 0x01
- #define DIGI\_PIN1 0x02
- #define DIGI\_PIN2 0x04
- #define DIGI\_PIN3 0x08
- #define DIGI\_PIN4 0x10
- #define DIGI\_PIN5 0x20
- #define DIGI\_PIN6 0x40
- #define DIGI\_PIN7 0x80

### 5.21.1 Detailed Description

Constants for controlling the 8 digital pins. The eight digital inputs are returned as a byte representing the state of the eight inputs. The eight digital outputs are controlled by two bytes, the first of which sets the state of any of the signals which have been defined as outputs and the second of which controls the input/output state of each signal.

### 5.21.2 Macro Definition Documentation

#### 5.21.2.1 #define DIGI\_PIN0 0x01

Access digital pin 0 (B0)

#### 5.21.2.2 #define DIGI\_PIN1 0x02

Access digital pin 1 (B1)

#### 5.21.2.3 #define DIGI\_PIN2 0x04

Access digital pin 2 (B2)

#### 5.21.2.4 #define DIGI\_PIN3 0x08

Access digital pin 3 (B3)

#### 5.21.2.5 #define DIGI\_PIN4 0x10

Access digital pin 4 (B4)

#### 5.21.2.6 #define DIGI\_PIN5 0x20

Access digital pin 5 (B5)

#### 5.21.2.7 #define DIGI\_PIN6 0x40

Access digital pin 6 (B6)

#### 5.21.2.8 #define DIGI\_PIN7 0x80

Access digital pin 7 (B7)

## 5.22 SuperPro Strobe control constants

Constants for manipulating the six digital strobe outputs.

### Macros

- #define STROBE\_S0 0x01
- #define STROBE\_S1 0x02
- #define STROBE\_S2 0x04
- #define STROBE\_S3 0x08
- #define STROBE\_READ 0x10
- #define STROBE\_WRITE 0x20

### 5.22.1 Detailed Description

Constants for manipulating the six digital strobe outputs. Six digital strobe outputs are available. One is pre-configured as a read strobe, another is pre-configured as a write strobe while the other four can be set to a high or low logic level. These strobe lines enable external devices to synchronize with the digital data port and multiplex the eight digital input/output bits to wider bit widths.

The RD and WR bits set the inactive state of the read and write strobe outputs. Thus, if these bits are set to 0, the strobe outputs will pulse high.

### 5.22.2 Macro Definition Documentation

#### 5.22.2.1 #define STROBE\_READ 0x10

Access read pin (RD)

#### 5.22.2.2 #define STROBE\_S0 0x01

Access strobe 0 pin (S0)

#### 5.22.2.3 #define STROBE\_S1 0x02

Access strobe 1 pin (S1)

#### 5.22.2.4 #define STROBE\_S2 0x04

Access strobe 2 pin (S2)

#### 5.22.2.5 #define STROBE\_S3 0x08

Access strobe 3 pin (S3)

#### 5.22.2.6 #define STROBE\_WRITE 0x20

Access write pin (WR)

## 5.23 MindSensors API Functions

Functions for accessing and modifying MindSensors devices.

### Modules

- [MindSensors device constants](#)

*Constants that are for use with MindSensors devices.*

### Macros

- `#define ReadSensorMSCompass(_port, _i2caddr, _value) __ReadSensorMSCompass(_port, _i2caddr, _value)`  
*Read mindsensors compass value.*
- `#define ReadSensorMSDROD(_port, _value) __ReadSensorMSDROD(_port, _value)`  
*Read mindsensors DROD value.*
- `#define SetSensorMSDRODActive(_port) __SetSensorMSDRODActive(_port)`  
*Configure a mindsensors DROD active sensor.*
- `#define SetSensorMSDRODInactive(_port) __SetSensorMSDRODInactive(_port)`  
*Configure a mindsensors DROD inactive sensor.*
- `#define ReadSensorNXTSumoEyes(_port, _value) __ReadSensorNXTSumoEyes(_port, _value)`  
*Read mindsensors NXTSumoEyes value.*
- `#define SetSensorNXTSumoEyesLong(_port) __SetSensorNXTSumoEyesLong(_port)`  
*Configure a mindsensors NXTSumoEyes long range sensor.*
- `#define SetSensorNXTSumoEyesShort(_port) __SetSensorNXTSumoEyesShort(_port)`  
*Configure a mindsensors NXTSumoEyes short range sensor.*
- `#define ReadSensorMSPressureRaw(_port, _value) __ReadSensorMSPressureRaw(_port, _value)`  
*Read mindsensors raw pressure value.*
- `#define ReadSensorMSPressure(_port, _value) __ReadSensorMSPressure(_port, _value)`  
*Read mindsensors processed pressure value.*
- `#define SetSensorMSPressure(_port) __SetSensorMSPressure(_port)`  
*Configure a mindsensors pressure sensor.*
- `#define SetSensorMSTouchMux(_port) __SetSensorMSTouchMux(_port)`  
*Configure a mindsensors touch sensor multiplexer.*
- `#define ReadSensorMSAccel(_port, _i2caddr, _x, _y, _z, _result) __ReadSensorMSAccel(_port, _i2caddr, _x, _y, _z, _result)`  
*Read mindsensors acceleration values.*
- `#define ReadSensorMSPlayStation(_port, _i2caddr, _b1, _b2, _xleft, _yleft, _xright, _yright, _result) __ReadSensorMSPlayStation(_port, _i2caddr, _b1, _b2, _xleft, _yleft, _xright, _yright, _result)`  
*Read mindsensors playstation controller values.*
- `#define ReadSensorMSRTClock(_port, _sec, _min, _hrs, _dow, _date, _month, _year, _result) __ReadSensorMSRTClock(_port, _sec, _min, _hrs, _dow, _date, _month, _year, _result)`  
*Read mindsensors RTClock values.*
- `#define ReadSensorMSTilt(_port, _i2caddr, _x, _y, _z, _result) __ReadSensorMSTilt(_port, _i2caddr, _x, _y, _z, _result)`  
*Read mindsensors tilt values.*
- `#define PFMateSend(_port, _i2caddr, _channel, _motors, _cmdA, _spdA, _cmdB, _spdB, _result) __PFMateSend(_port, _i2caddr, _channel, _motors, _cmdA, _spdA, _cmdB, _spdB, _result)`  
*Send PFmate command.*

- #define `PFMateSendRaw(_port, _i2caddr, _channel, _b1, _b2, _result)` `__PFMateSendRaw(_port, _i2caddr, _channel, _b1, _b2, _result)`  
*Send raw PFMate command.*
- #define `MSReadValue(_port, _i2caddr, _reg, _bytes, _out, _result)` `__MSReadValue(_port, _i2caddr, _reg, _bytes, _out, _result)`  
*Read a mindsensors device value.*
- #define `MSEnergize(_port, _i2caddr, _result)` `__I2CSendCmd(_port, _i2caddr, MS_CMD_ENERGIZED, _result)`  
*Turn on power to device.*
- #define `MSDeenergize(_port, _i2caddr, _result)` `__I2CSendCmd(_port, _i2caddr, MS_CMD_DEENERGIZED, _result)`  
*Turn off power to device.*
- #define `MSADPAOn(_port, _i2caddr, _result)` `__I2CSendCmd(_port, _i2caddr, MS_CMD_ADPA_ON, _result)`  
*Turn on mindsensors ADPA mode.*
- #define `MSADPAOff(_port, _i2caddr, _result)` `__I2CSendCmd(_port, _i2caddr, MS_CMD_ADPA_OFF, _result)`  
*Turn off mindsensors ADPA mode.*
- #define `DISTNxGP2D12(_port, _i2caddr, _result)` `__I2CSendCmd(_port, _i2caddr, DIST_CMD_GP2D12, _result)`  
*Configure DIST-Nx as GP2D12.*
- #define `DISTNxGP2D120(_port, _i2caddr, _result)` `__I2CSendCmd(_port, _i2caddr, DIST_CMD_GP2D120, _result)`  
*Configure DIST-Nx as GP2D120.*
- #define `DISTNxGP2YA02(_port, _i2caddr, _result)` `__I2CSendCmd(_port, _i2caddr, DIST_CMD_GP2YA02, _result)`  
*Configure DIST-Nx as GP2YA02.*
- #define `DISTNxGP2YA21(_port, _i2caddr, _result)` `__I2CSendCmd(_port, _i2caddr, DIST_CMD_GP2YA21, _result)`  
*Configure DIST-Nx as GP2YA21.*
- #define `ReadDISTNxDistance(_port, _i2caddr, _out, _result)` `__MSReadValue(_port, _i2caddr, DIST_REG_DISTANCE, 2, _out, _result)`  
*Read DIST-Nx distance value.*
- #define `ReadDISTNxMaxDistance(_port, _i2caddr, _out, _result)` `__MSReadValue(_port, _i2caddr, DIST_REG_DIST_MAX, 2, _out, _result)`  
*Read DIST-Nx maximum distance value.*
- #define `ReadDISTNxMinDistance(_port, _i2caddr, _out, _result)` `__MSReadValue(_port, _i2caddr, DIST_REG_DIST_MIN, 2, _out, _result)`  
*Read DIST-Nx minimum distance value.*
- #define `ReadDISTNxModuleType(_port, _i2caddr, _out, _result)` `__MSReadValue(_port, _i2caddr, DIST_REG_MODULE_TYPE, 1, _out, _result)`  
*Read DIST-Nx module type value.*
- #define `ReadDISTNxNumPoints(_port, _i2caddr, _out, _result)` `__MSReadValue(_port, _i2caddr, DIST_REG_NUM_POINTS, 1, _out, _result)`  
*Read DIST-Nx num points value.*
- #define `ReadDISTNxVoltage(_port, _i2caddr, _out, _result)` `__MSReadValue(_port, _i2caddr, DIST_REG_VOLT, 2, _out, _result)`  
*Read DIST-Nx voltage value.*
- #define `ACCLNxCalibrateX(_port, _i2caddr, _result)` `__I2CSendCmd(_port, _i2caddr, ACCL_CMD_X_CAL, _result)`  
*Calibrate ACCL-Nx X-axis.*

- #define ACCLNxCalibrateXEnd(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, ACCL\_CMD\_X\_CAL-END, \_result)
 

*Stop calibrating ACCL-Nx X-axis.*
- #define ACCLNxCalibrateY(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, ACCL\_CMD\_Y\_CAL, \_result)
 

*Calibrate ACCL-Nx Y-axis.*
- #define ACCLNxCalibrateYEnd(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, ACCL\_CMD\_Y\_CAL-END, \_result)
 

*Stop calibrating ACCL-Nx Y-axis.*
- #define ACCLNxCalibrateZ(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, ACCL\_CMD\_Z\_CAL, \_result)
 

*Calibrate ACCL-Nx Z-axis.*
- #define ACCLNxCalibrateZEnd(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, ACCL\_CMD\_Z\_CAL-END, \_result)
 

*Stop calibrating ACCL-Nx Z-axis.*
- #define ACCLNxResetCalibration(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, ACCL\_CMD\_RESET\_CAL, \_result)
 

*Reset ACCL-Nx calibration.*
- #define SetACCLNxSensitivity(\_port, \_i2caddr, \_slevel, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, \_slevel, \_result)
 

*Set ACCL-Nx sensitivity.*
- #define ReadACCLNxSensitivity(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, ACCL\_REG\_sENS\_LVL, 1, \_out, \_result)
 

*Read ACCL-Nx sensitivity value.*
- #define ReadACCLNxXOffset(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, ACCL\_REG\_X\_OFFSET, 2, \_out, \_result)
 

*Read ACCL-Nx X offset value.*
- #define ReadACCLNxXRange(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, ACCL\_REG\_X\_RANGE, 2, \_out, \_result)
 

*Read ACCL-Nx X range value.*
- #define ReadACCLNxYOffset(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, ACCL\_REG\_Y\_OFFSET, 2, \_out, \_result)
 

*Read ACCL-Nx Y offset value.*
- #define ReadACCLNxYRange(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, ACCL\_REG\_Y\_RANGE, 2, \_out, \_result)
 

*Read ACCL-Nx Y range value.*
- #define ReadACCLNxZOffset(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, ACCL\_REG\_Z\_OFFSET, 2, \_out, \_result)
 

*Read ACCL-Nx Z offset value.*
- #define ReadACCLNxZRange(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, ACCL\_REG\_Z\_RANGE, 2, \_out, \_result)
 

*Read ACCL-Nx Z range value.*
- #define PSPNxDigital(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, PSP\_CMD\_DIGITAL, \_result)
 

*Configure PSP-Nx in digital mode.*
- #define PSPNxAnalog(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, PSP\_CMD\_ANALOG, \_result)
 

*Configure PSP-Nx in analog mode.*
- #define ReadNXTServoPosition(\_port, \_i2caddr, \_servo, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, NXTSERVO\_REG\_S1\_POS+\_servo\*2, 2, \_out, \_result)
 

*Read NXTServo servo position value.*

- #define `ReadNXTServoSpeed(_port, _i2caddr, _servo, _out, _result)` \_\_MSReadValue(\_port, \_i2caddr, **NXTSERVO\_REG\_S1\_SPEED**+\_servo, 1, \_out, \_result)
 

*Read NXTServo servo speed value.*
- #define `ReadNXTServoBatteryVoltage(_port, _i2caddr, _out, _result)` \_\_MSReadValue(\_port, \_i2caddr, **NXTSERVO\_REG\_VOLTAGE**, 1, \_out, \_result)
 

*Read NXTServo battery voltage value.*
- #define `SetNXTServoSpeed(_port, _i2caddr, _servo, _speed, _result)` \_\_MSWriteToRegister(\_port, \_i2caddr, **NXTSERVO\_REG\_S1\_SPEED**+\_servo, \_speed, \_result)
 

*Set NXTServo servo motor speed.*
- #define `SetNXTServoQuickPosition(_port, _i2caddr, _servo, _qpos, _result)` \_\_MSWriteToRegister(\_port, \_i2caddr, **NXTSERVO\_REG\_S1\_QPOS**+\_servo, \_qpos, \_result)
 

*Set NXTServo servo motor quick position.*
- #define `SetNXTServoPosition(_port, _i2caddr, _servo, _pos, _result)` \_\_MSWriteLEIntToRegister(\_port, \_i2caddr, \_reg, \_pos, \_result)
 

*Set NXTServo servo motor position.*
- #define `NXTServoReset(_port, _i2caddr, _result)` \_\_I2CSendCmd(\_port, \_i2caddr, **NXTSERVO\_CMD\_RESET**, \_result)
 

*Reset NXTServo properties.*
- #define `NXTServoHaltMacro(_port, _i2caddr, _result)` \_\_I2CSendCmd(\_port, \_i2caddr, **NXTSERVO\_CMD\_HALT**, \_result)
 

*Halt NXTServo macro.*
- #define `NXTServoResumeMacro(_port, _i2caddr, _result)` \_\_I2CSendCmd(\_port, \_i2caddr, **NXTSERVO\_CMD\_RESUME**, \_result)
 

*Resume NXTServo macro.*
- #define `NXTServoPauseMacro(_port, _i2caddr, _result)` \_\_I2CSendCmd(\_port, \_i2caddr, **NXTSERVO\_CMD\_PAUSE**, \_result)
 

*Pause NXTServo macro.*
- #define `NXTServoInit(_port, _i2caddr, _servo, _result)` \_\_NXTServoInit(\_port, \_i2caddr, \_servo, \_result)
 

*Initialize NXTServo servo properties.*
- #define `NXTServoGotoMacroAddress(_port, _i2caddr, _macro, _result)` \_\_NXTServoGotoMacroAddress(\_port, \_i2caddr, \_macro, \_result)
 

*Goto NXTServo macro address.*
- #define `NXTServoEditMacro(_port, _i2caddr, _result)` \_\_NXTServoEditMacro(\_port, \_i2caddr, \_result)
 

*Edit NXTServo macro.*
- #define `NXTServoQuitEdit(_port, _result)` \_\_MSWriteToRegister(\_port, **MS\_ADDR\_NXTSERVO\_EM**, **NXTSERVO\_EM\_REG\_CMD**, **NXTSERVO\_EM\_CMD\_QUIT**, \_result)
 

*Quit NXTServo macro edit mode.*
- #define `NXTHIDAsciiMode(_port, _i2caddr, _result)` \_\_I2CSendCmd(\_port, \_i2caddr, **NXTHID\_CMD\_ASCII**, \_result)
 

*Set NXTHID into ASCII data mode.*
- #define `NXTHIDDirectMode(_port, _i2caddr, _result)` \_\_I2CSendCmd(\_port, \_i2caddr, **NXTHID\_CMD\_DIRECT**, \_result)
 

*Set NXTHID into direct data mode.*
- #define `NXTHIDTransmit(_port, _i2caddr, _result)` \_\_I2CSendCmd(\_port, \_i2caddr, **NXTHID\_CMD\_TRANSMIT**, \_result)
 

*Transmit NXTHID character.*
- #define `NXTHIDLoadCharacter(_port, _i2caddr, _modifier, _character, _result)` \_\_NXTHIDLoadCharacter(\_port, \_i2caddr, \_modifier, \_character, \_result)
 

*Load NXTHID character.*

- #define NXTPowerMeterResetCounters(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, **NXTPM\_CMD\_RESET**, \_result)
 

*Reset NXTPowerMeter counters.*
- #define ReadNXTPowerMeterPresentCurrent(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, **NXTPM\_REG\_CURRENT**, 2, \_out, \_result)
 

*Read NXTPowerMeter present current.*
- #define ReadNXTPowerMeterPresentVoltage(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, **NXTPM\_REG\_VOLTAGE**, 2, \_out, \_result)
 

*Read NXTPowerMeter present voltage.*
- #define ReadNXTPowerMeterCapacityUsed(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, **NXTPM\_REG\_CAPACITY**, 2, \_out, \_result)
 

*Read NXTPowerMeter capacity used.*
- #define ReadNXTPowerMeterPresentPower(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, **NXTPM\_REG\_POWER**, 2, \_out, \_result)
 

*Read NXTPowerMeter present power.*
- #define ReadNXTPowerMeterTotalPowerConsumed(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, **NXTPM\_REG\_POWER**, 4, \_out, \_result)
 

*Read NXTPowerMeter total power consumed.*
- #define ReadNXTPowerMeterMaxCurrent(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, **NXTPM\_REG\_MAXCURRENT**, 2, \_out, \_result)
 

*Read NXTPowerMeter maximum current.*
- #define ReadNXTPowerMeterMinCurrent(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, **NXTPM\_REG\_MINCURRENT**, 2, \_out, \_result)
 

*Read NXTPowerMeter minimum current.*
- #define ReadNXTPowerMeterMaxVoltage(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, **NXTPM\_REG\_MAXVOLTAGE**, 2, \_out, \_result)
 

*Read NXTPowerMeter maximum voltage.*
- #define ReadNXTPowerMeterMinVoltage(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, **NXTPM\_REG\_MINVOLTAGE**, 2, \_out, \_result)
 

*Read NXTPowerMeter minimum voltage.*
- #define ReadNXTPowerMeterElapsedTime(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, **NXTPM\_REG\_TIME**, 4, \_out, \_result)
 

*Read NXTPowerMeter elapsed time.*
- #define ReadNXTPowerMeterErrorCount(\_port, \_i2caddr, \_out, \_result) \_\_MSReadValue(\_port, \_i2caddr, **NXTPM\_REG\_ERRORCOUNT**, 2, \_out, \_result)
 

*Read NXTPowerMeter error count.*
- #define NXTLineLeaderPowerDown(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, **NXTLL\_CMD\_POWERDOWN**, \_result)
 

*Powderdown NXTLineLeader device.*
- #define NXTLineLeaderPowerUp(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, **NXTLL\_CMD\_POWERUP**, \_result)
 

*Powerup NXTLineLeader device.*
- #define NXTLineLeaderInvert(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, **NXTLL\_CMD\_INVERT**, \_result)
 

*Invert NXTLineLeader colors.*
- #define NXTLineLeaderReset(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, **NXTLL\_CMD\_RESET**, \_result)
 

*Reset NXTLineLeader color inversion.*
- #define NXTLineLeaderSnapshot(\_port, \_i2caddr, \_result) \_\_I2CSendCmd(\_port, \_i2caddr, **NXTLL\_CMD\_SNAPSHOT**, \_result)
 

*Reset NXTLineLeader color inversion.*

- `#define NXTLineLeaderCalibrateWhite(_port, _i2caddr, _result) __I2CSendCmd(_port, _i2caddr, NXTLL_CMD_WHITE, _result)`

*Calibrate NXTLineLeader white color.*
- `#define NXTLineLeaderCalibrateBlack(_port, _i2caddr, _result) __I2CSendCmd(_port, _i2caddr, NXTLL_CMD_BLACK, _result)`

*Calibrate NXTLineLeader black color.*
- `#define ReadNXTLineLeaderSteering(_port, _i2caddr, _out, _result) __MSReadValue(_port, _i2caddr, NXTLL_REG_STEERING, 1, _out, _result)`

*Read NXTLineLeader steering.*
- `#define ReadNXTLineLeaderAverage(_port, _i2caddr, _out, _result) __MSReadValue(_port, _i2caddr, NXTLL_REG_AVERAGE, 1, _out, _result)`

*Read NXTLineLeader average.*
- `#define ReadNXTLineLeaderResult(_port, _i2caddr, _out, _result) __MSReadValue(_port, _i2caddr, NXTLL_REG_RESULT, 1, _out, _result)`

*Read NXTLineLeader result.*
- `#define SetNXTLineLeaderSetpoint(_port, _i2caddr, _value, _result) __MSWriteToRegister(_port, _i2caddr, NXTLL_REG_SETPOINT, _value, _result)`

*Write NXTLineLeader setpoint.*
- `#define SetNXTLineLeaderKpValue(_port, _i2caddr, _value, _result) __MSWriteToRegister(_port, _i2caddr, NXTLL_REG_KP_VALUE, _value, _result)`

*Write NXTLineLeader Kp value.*
- `#define SetNXTLineLeaderKiValue(_port, _i2caddr, _value, _result) __MSWriteToRegister(_port, _i2caddr, NXTLL_REG_KI_VALUE, _value, _result)`

*Write NXTLineLeader Ki value.*
- `#define SetNXTLineLeaderKdValue(_port, _i2caddr, _value, _result) __MSWriteToRegister(_port, _i2caddr, NXTLL_REG_KD_VALUE, _value, _result)`

*Write NXTLineLeader Kd value.*
- `#define SetNXTLineLeaderKpFactor(_port, _i2caddr, _value, _result) __MSWriteToRegister(_port, _i2caddr, NXTLL_REG_KP_FACTOR, _value, _result)`

*Write NXTLineLeader Kp factor.*
- `#define SetNXTLineLeaderKiFactor(_port, _i2caddr, _value, _result) __MSWriteToRegister(_port, _i2caddr, NXTLL_REG_KI_FACTOR, _value, _result)`

*Write NXTLineLeader Ki factor.*
- `#define SetNXTLineLeaderKdFactor(_port, _i2caddr, _value, _result) __MSWriteToRegister(_port, _i2caddr, NXTLL_REG_KD_FACTOR, _value, _result)`

*Write NXTLineLeader Kd factor.*
- `#define NRLink2400(_port, _i2caddr, _result) __I2CSendCmd(_port, _i2caddr, NRLINK_CMD_2400, _result)`

*Configure NRLink in 2400 baud mode.*
- `#define NRLink4800(_port, _i2caddr, _result) __I2CSendCmd(_port, _i2caddr, NRLINK_CMD_4800, _result)`

*Configure NRLink in 4800 baud mode.*
- `#define NRLinkFlush(_port, _i2caddr, _result) __I2CSendCmd(_port, _i2caddr, NRLINK_CMD_FLUSH, _result)`

*Flush NRLink buffers.*
- `#define NRLinkIRLong(_port, _i2caddr, _result) __I2CSendCmd(_port, _i2caddr, NRLINK_CMD_IR_LONG, _result)`

*Configure NRLink in IR long mode.*
- `#define NRLinkIRShort(_port, _i2caddr, _result) __I2CSendCmd(_port, _i2caddr, NRLINK_CMD_IR_SHORT, _result)`

*Configure NRLink in IR short mode.*

- `#define NRLinkSetPF(_port, _i2caddr, _result) __I2CSendCmd(_port, _i2caddr, NRLINK_CMD_SET_PF, _result)`

*Configure NRLink in IR short mode.*
- `#define NRLinkSetRCX(_port, _i2caddr, _result) __I2CSendCmd(_port, _i2caddr, NRLINK_CMD_SET_RCX, _result)`

*Configure NRLink in power function mode.*
- `#define NRLinkSetTrain(_port, _i2caddr, _result) __I2CSendCmd(_port, _i2caddr, NRLINK_CMD_SET_TRAIN, _result)`

*Configure NRLink in RCX mode.*
- `#define NRLinkTxRaw(_port, _i2caddr, _result) __I2CSendCmd(_port, _i2caddr, NRLINK_CMD_TX_RAW, _result)`

*Configure NRLink in IR train mode.*
- `#define ReadNRLinkStatus(_port, _i2caddr, _value, _result) __ReadNRLinkStatus(_port, _i2caddr, _value, _result)`

*Read NRLink status.*
- `#define RunNRLinkMacro(_port, _i2caddr, _macro, _result) __RunNRLinkMacro(_port, _i2caddr, _macro, _result)`

*Run NRLink macro.*
- `#define WriteNRLinkBytes(_port, _i2caddr, _bytes, _result) __WriteNRLinkBytes(_port, _i2caddr, _bytes, _result)`

*Write data to NRLink.*
- `#define MSIRTrain(_port, _i2caddr, _channel, _func, _result) __MSIRTrain(_port, _i2caddr, _channel, _func, FA-LSE, _result)`

*MSIRTrain function.*
- `#define MSPFComboDirect(_port, _i2caddr, _channel, _outa, _outb, _result) __MSPFComboDirect(_port, _i2caddr, _channel, _outa, _outb, _result)`

*MSPFComboDirect function.*
- `#define MSPFComboPWM(_port, _i2caddr, _channel, _outa, _outb, _result) __MSPFComboPWM(_port, _i2caddr, _channel, _outa, _outb, _result)`

*MSPFComboPWM function.*
- `#define MSPFRawOutput(_port, _i2caddr, _nibble0, _nibble1, _nibble2, _result) __MSPFRawOutput(_port, _i2caddr, _nibble0, _nibble1, _nibble2, _result)`

*MSPFRawOutput function.*
- `#define MSPFRepeat(_port, _i2caddr, _count, _delay, _result) __MSPFRepeatLastCommand(_port, _i2caddr, _count, _delay, _result)`

*MSPFRepeat function.*
- `#define MSPFSingleOutputCST(_port, _i2caddr, _channel, _out, _func, _result) __MSPFSingleOutput(_port, _i2caddr, _channel, _out, _func, TRUE, _result)`

*MSPFSingleOutputCST function.*
- `#define MSPFSingleOutputPWM(_port, _i2caddr, _channel, _out, _func, _result) __MSPFSingleOutput(_port, _i2caddr, _channel, _out, _func, FALSE, _result)`

*MSPFSingleOutputPWM function.*
- `#define MSPFSinglePin(_port, _i2caddr, _channel, _out, _pin, _func, _cont, _result) __MSPFSinglePin(_port, _i2caddr, _channel, _out, _pin, _func, _cont, _result)`

*MSPFSinglePin function.*

- #define **MSPFTrain**(*\_port*, *\_i2caddr*, *\_channel*, *\_func*, *\_result*) \_\_MSIRTrain(*\_port*, *\_i2caddr*, *\_channel*, *\_func*, **TR-UE**, *\_result*)  
*MSPFTrain function.*
- #define **MSRCXSetNRLinkPort**(*\_port*, *\_i2caddr*) \_\_MSRCXSetNRLink(*\_port*, *\_i2caddr*)  
*MSRCXSetNRLinkPort function.*
- #define **MSRCXBatteryLevel**(*\_result*) \_\_MSRCXBatteryLevel(*\_result*)  
*MSRCXBatteryLevel function.*
- #define **MSRCXPoll**(*\_src*, *\_value*, *\_result*) \_\_MSRCXPoll(*\_src*, *\_value*, *\_result*)  
*MSRCXPoll function.*
- #define **MSRCXPollMemory**(*\_memaddress*, *\_result*) \_\_MSRCXPollMemory(*\_memaddress*, *\_result*)  
*MSRCXPollMemory function.*
- #define **MSRCXAbsVar**(*\_varnum*, *\_src*, *\_value*) \_\_MSRCXVarOp(**RCX\_AbsVarOp**, *\_varnum*, *\_src*, *\_value*)  
*MSRCXAbsVar function.*
- #define **MSRCXAddToDatalog**(*\_src*, *\_value*) \_\_MSRCXAddToDatalog(*\_src*, *\_value*)  
*MSRCXAddToDatalog function.*
- #define **MSRCXAndVar**(*\_varnum*, *\_src*, *\_value*) \_\_MSRCXVarOp(**RCX\_AndVarOp**, *\_varnum*, *\_src*, *\_value*)  
*MSRCXAndVar function.*
- #define **MSRCXBoot**() \_\_MSRCXBoot()  
*MSRCXBoot function.*
- #define **MSRCXCalibrateEvent**(*\_evt*, *\_low*, *\_hi*, *\_hyst*) \_\_MSRCXCalibrateEvent(*\_evt*, *\_low*, *\_hi*, *\_hyst*)  
*MSRCXCalibrateEvent function.*
- #define **MSRCXClearAllEvents**() \_\_MSRCXOpNoArgs(**RCX\_ClearAllEventsOp**)  
*MSRCXClearAllEvents function.*
- #define **MSRCXClearCounter**(*\_counter*) \_\_MSRCXClearCounter(*\_counter*)  
*MSRCXClearCounter function.*
- #define **MSRCXClearMsg**() \_\_MSRCXOpNoArgs(**RCX\_ClearMsgOp**)  
*MSRCXClearMsg function.*
- #define **MSRCXClearSensor**(*\_port*) \_\_MSRCXClearSensor(*\_port*)  
*MSRCXClearSensor function.*
- #define **MSRCXClearSound**() \_\_MSRCXOpNoArgs(**RCX\_ClearSoundOp**)  
*MSRCXClearSound function.*
- #define **MSRCXClearTimer**(*\_timer*) \_\_MSRCXClearTimer(*\_timer*)  
*MSRCXClearTimer function.*
- #define **MSRCXCreateDatalog**(*\_size*) \_\_MSRCXCreateDatalog(*\_size*)  
*MSRCXCreateDatalog function.*
- #define **MSRCXDecCounter**(*\_counter*) \_\_MSRCXDecCounter(*\_counter*)  
*MSRCXDecCounter function.*
- #define **MSRCXDeleteSub**(*\_s*) \_\_MSRCXDeleteSub(*\_s*)  
*MSRCXDeleteSub function.*
- #define **MSRCXDeleteSubs**() \_\_MSRCXOpNoArgs(**RCX\_DeleteSubsOp**)  
*MSRCXDeleteSubs function.*
- #define **MSRCXDeleteTask**(*\_t*) \_\_MSRCXDeleteTask(*\_t*)  
*MSRCXDeleteTask function.*
- #define **MSRCXDeleteTasks**() \_\_MSRCXOpNoArgs(**RCX\_DeleteTasksOp**)  
*MSRCXDeleteTasks function.*
- #define **MSRCXDisableOutput**(*\_outputs*) \_\_MSRCXSetGlobalOutput(*\_outputs*, **RCX\_OUT\_OFF**)  
*MSRCXDisableOutput function.*

- #define **MSRCXDivVar**(\_varnum, \_src, \_value) \_\_MSRCXVarOp(**RCX\_DivVarOp**, \_varnum, \_src, \_value)  
*MSRCXDivVar function.*
- #define **MSRCXEnableOutput**(\_outputs) \_\_MSRCXSetGlobalOutput(\_outputs, **RCX\_OUT\_ON**)  
*MSRCXEnableOutput function.*
- #define **MSRCXEvent**(\_src, \_value) \_\_MSRCXEvent(\_src, \_value)  
*MSRCXEvent function.*
- #define **MSRCXFloat**(\_outputs) \_\_MSRCXSetOutput(\_outputs, **RCX\_OUT\_FLOAT**)  
*MSRCXFloat function.*
- #define **MSRCXFwd**(\_outputs) \_\_MSRCXSetDirection(\_outputs, **RCX\_OUT\_FWD**)  
*MSRCXFwd function.*
- #define **MSRCXIIncCounter**(\_counter) \_\_MSRCXIIncCounter(\_counter)  
*MSRCXIIncCounter function.*
- #define **MSRCXIInvertOutput**(\_outputs) \_\_MSRCXSetGlobalDirection(\_outputs, **RCX\_OUT\_REV**)  
*MSRCXIInvertOutput function.*
- #define **MSRCXMulVar**(\_varnum, \_src, \_value) \_\_MSRCXVarOp(**RCX\_MulVarOp**, \_varnum, \_src, \_value)  
*MSRCXMulVar function.*
- #define **MSRCXMuteSound**() \_\_MSRCXOpNoArgs(**RCX\_MuteSoundOp**)  
*MSRCXMuteSound function.*
- #define **MSRCXObvertOutput**(\_outputs) \_\_MSRCXSetGlobalDirection(\_outputs, **RCX\_OUT\_FWD**)  
*MSRCXObvertOutput function.*
- #define **MSRCXOff**(\_outputs) \_\_MSRCXSetOutput(\_outputs, **RCX\_OUT\_OFF**)  
*MSRCXOff function.*
- #define **MSRCXOn**(\_outputs) \_\_MSRCXSetOutput(\_outputs, **RCX\_OUT\_ON**)  
*MSRCXOn function.*
- #define **MSRCXOnFor**(\_outputs, \_ms) \_\_MSRCXOnFor(\_outputs, \_ms)  
*MSRCXOnFor function.*
- #define **MSRCXOnFwd**(\_outputs) \_\_MSRCXOnFwd(\_outputs)  
*MSRCXOnFwd function.*
- #define **MSRCXOnRev**(\_outputs) \_\_MSRCXOnRev(\_outputs)  
*MSRCXOnRev function.*
- #define **MSRCXOrVar**(\_varnum, \_src, \_value) \_\_MSRCXVarOp(**RCX\_OrVarOp**, \_varnum, \_src, \_value)  
*MSRCXOrVar function.*
- #define **MSRCXPBTurnOff**() \_\_MSRCXOpNoArgs(**RCX\_PBTurnOffOp**)  
*MSRCXPBTurnOff function.*
- #define **MSRCXPing**() \_\_MSRCXOpNoArgs(**RCX\_PingOp**)  
*MSRCXPing function.*
- #define **MSRCXPlaySound**(\_snd) \_\_MSRCXPlaySound(\_snd)  
*MSRCXPlaySound function.*
- #define **MSRCXPlayTone**(\_freq, \_duration) \_\_MSRCXPlayTone(\_freq, \_duration)  
*MSRCXPlayTone function.*
- #define **MSRCXPlayToneVar**(\_varnum, \_duration) \_\_MSRCXPlayToneVar(\_varnum, \_duration)  
*MSRCXPlayToneVar function.*
- #define **MSRCXRemote**(\_cmd) \_\_MSRCXRemote(\_cmd)  
*MSRCXRemote function.*
- #define **MSRCXReset**() \_\_MSRCXReset()  
*MSRCXReset function.*
- #define **MSRCXRev**(\_outputs) \_\_MSRCXSetDirection(\_outputs, **RCX\_OUT\_REV**)

- `#define MSRCXRev(_src, _value) __MSRCXSelectDisplay(_src, _value)`  
*MSRCXSelectDisplay function.*
- `#define MSRCXSelectProgram(_prog) __MSRCXSelectProgram(_prog)`  
*MSRCXSelectProgram function.*
- `#define MSRCXSendSerial(_first, _count) __MSRCXSendSerial(_first, _count)`  
*MSRCXSendSerial function.*
- `#define MSRCXSet(_dstsrc, _dstval, _src, _value) __MSRCXSet(_dstsrc, _dstval, _src, _value)`  
*MSRCXSet function.*
- `#define MSRCXSetDirection(_outputs, _dir) __MSRCXSetDirection(_outputs, _dir)`  
*MSRCXSetDirection function.*
- `#define MSRCXSetEvent(_evt, _src, _type) __MSRCXSetEvent(_evt, _src, _type)`  
*MSRCXSetEvent function.*
- `#define MSRCXSetGlobalDirection(_outputs, _dir) __MSRCXSetGlobalDirection(_outputs, _dir)`  
*MSRCXSetGlobalDirection function.*
- `#define MSRCXSetGlobalOutput(_outputs, _mode) __MSRCXSetGlobalOutput(_outputs, _mode)`  
*MSRCXSetGlobalOutput function.*
- `#define MSRCXSetMaxPower(_outputs, _pwrsrc, _pwrval) __MSRCXSetMaxPower(_outputs, _pwrsrc, _pwrval)`  
*MSRCXSetMaxPower function.*
- `#define MSRCXSetMessage(_msg) __MSRCXSetMessage(_msg)`  
*MSRCXSetMessage function.*
- `#define MSRCXSetOutput(_outputs, _mode) __MSRCXSetOutput(_outputs, _mode)`  
*MSRCXSetOutput function.*
- `#define MSRCXSetPower(_outputs, _pwrsrc, _pwrval) __MSRCXSetPower(_outputs, _pwrsrc, _pwrval)`  
*MSRCXSetPower function.*
- `#define MSRCXSetPriority(_p) __MSRCXSetPriority(_p)`  
*MSRCXSetPriority function.*
- `#define MSRCXSetSensorMode(_port, _mode) __MSRCXSetSensorMode(_port, _mode)`  
*MSRCXSetSensorMode function.*
- `#define MSRCXSetSensorType(_port, _type) __MSRCXSetSensorType(_port, _type)`  
*MSRCXSetSensorType function.*
- `#define MSRCXSetSleepTime(_t) __MSRCXSetSleepTime(_t)`  
*MSRCXSetSleepTime function.*
- `#define MSRCXSetTxPower(_pwr) __MSRCXSetTxPower(_pwr)`  
*MSRCXSetTxPower function.*
- `#define MSRCXSetUserDisplay(_src, _value, _precision) __MSRCXSetUserDisplay(_src, _value, _precision)`  
*MSRCXSetUserDisplay function.*
- `#define MSRCXSetVar(_varnum, _src, _value) __MSRCXVarOp(RCX_SetVarOp, _varnum, _src, _value)`  
*MSRCXSetVar function.*
- `#define MSRCXSetWatch(_hours, _minutes) __MSRCXSetWatch(_hours, _minutes)`  
*MSRCXSetWatch function.*
- `#define MSRCXSgnVar(_varnum, _src, _value) __MSRCXVarOp(RCX_SgnVarOp, _varnum, _src, _value)`  
*MSRCXSgnVar function.*
- `#define MSRCXStartTask(_t) __MSRCXStartTask(_t)`  
*MSRCXStartTask function.*
- `#define MSRCXStopAllTasks() __MSRCXOpNoArgs(RCX_StopAllTasksOp)`

- `#define MSRCXStopTask(_t) __MSRCXStopTask(_t)`  
*MSRCXStopTask function.*
- `#define MSRCXSubVar(_varnum, _src, _value) __MSRCXVarOp(RCX_SubVarOp, _varnum, _src, _value)`  
*MSRCXSubVar function.*
- `#define MSRCXSumVar(_varnum, _src, _value) __MSRCXVarOp(RCX_SumVarOp, _varnum, _src, _value)`  
*MSRCXSumVar function.*
- `#define MSRCXToggle(_outputs) __MSRCXSetDirection(_outputs, RCX_OUT_TOGGLE)`  
*MSRCXToggle function.*
- `#define MSRCXUnlock() __MSRCXUnlock()`  
*MSRCXUnlock function.*
- `#define MSRCXUnmuteSound() __MSRCXOpNoArgs(RCX_UnmuteSoundOp)`  
*MSRCXUnmuteSound function.*
- `#define MSScoutCalibrateSensor() __MSRCXOpNoArgs(RCX_LSCalibrateOp)`  
*MSScoutCalibrateSensor function.*
- `#define MSScoutMuteSound() __MSScoutMuteSound()`  
*MSScoutMuteSound function.*
- `#define MSScoutSelectSounds(_grp) __MSScoutSelectSounds(_grp)`  
*MSScoutSelectSounds function.*
- `#define MSScoutSendVLL(_src, _value) __MSScoutSendVLL(_src, _value)`  
*MSScoutSendVLL function.*
- `#define MSScoutSetCounterLimit(_ctr, _src, _value) __MSScoutSetCounterLimit(_ctr, _src, _value)`  
*MSScoutSetCounterLimit function.*
- `#define MSScoutSetEventFeedback(_src, _value) __MSScoutSetEventFeedback(_src, _value)`  
*MSScoutSetEventFeedback function.*
- `#define MSScoutSetLight(_x) __MSScoutSetLight(_x)`  
*MSScoutSetLight function.*
- `#define MSScoutSetScoutMode(_mode) __MSScoutSetScoutMode(_mode)`  
*MSScoutSetScoutMode function.*
- `#define MSScoutSetScoutRules(_m, _t, _l, _tm, _fx) __MSScoutSetScoutRules(_m, _t, _l, _tm, _fx)`  
*MSScoutSetScoutRules function.*
- `#define MSScoutSetSensorClickTime(_src, _value) __MSScoutSetSensorClickTime(_src, _value)`  
*MSScoutSetSensorClickTime function.*
- `#define MSScoutSetSensorHysteresis(_src, _value) __MSScoutSetSensorHysteresis(_src, _value)`  
*MSScoutSetSensorHysteresis function.*
- `#define MSScoutSetSensorLowerLimit(_src, _value) __MSScoutSetSensorLowerLimit(_src, _value)`  
*MSScoutSetSensorLowerLimit function.*
- `#define MSScoutSetSensorUpperLimit(_src, _value) __MSScoutSetSensorUpperLimit(_src, _value)`  
*MSScoutSetSensorUpperLimit function.*
- `#define MSScoutSetTimerLimit(_tmr, _src, _value) __MSScoutSetTimerLimit(_tmr, _src, _value)`  
*MSScoutSetTimerLimit function.*
- `#define MSScoutUnmuteSound() __MSScoutUnmuteSound()`  
*MSScoutUnmuteSound function.*

### 5.23.1 Detailed Description

Functions for accessing and modifying MindSensors devices.

### 5.23.2 Macro Definition Documentation

5.23.2.1 #define ACCLNxCalibrateX( \_port, \_i2caddr, \_result ) \_\_I2CSendCmd(\_port, \_i2caddr, ACCL\_CMD\_X\_CAL, \_result)

Calibrate ACCL-Nx X-axis.

Calibrate the mindsensors ACCL-Nx sensor X-axis. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_i2caddr</u>	The sensor I2C address. See sensor documentation for this value.
<u>_result</u>	The function call result.

5.23.2.2 #define ACCLNxCalibrateXEnd( \_port, \_i2caddr, \_result ) \_\_I2CSendCmd(\_port, \_i2caddr, ACCL\_CMD\_X\_CAL\_END, \_result)

Stop calibrating ACCL-Nx X-axis.

Stop calibrating the mindsensors ACCL-Nx sensor X-axis. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_i2caddr</u>	The sensor I2C address. See sensor documentation for this value.
<u>_result</u>	The function call result.

5.23.2.3 #define ACCLNxCalibrateY( \_port, \_i2caddr, \_result ) \_\_I2CSendCmd(\_port, \_i2caddr, ACCL\_CMD\_Y\_CAL, \_result)

Calibrate ACCL-Nx Y-axis.

Calibrate the mindsensors ACCL-Nx sensor Y-axis. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_i2caddr</u>	The sensor I2C address. See sensor documentation for this value.
<u>_result</u>	The function call result.

5.23.2.4 #define ACCLNxCalibrateYEnd( \_port, \_i2caddr, \_result ) \_\_I2CSendCmd(\_port, \_i2caddr, ACCL\_CMD\_Y\_CAL\_END, \_result)

Stop calibrating ACCL-Nx Y-axis.

Stop calibrating the mindsensors ACCL-Nx sensor Y-axis. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_i2caddr</u>	The sensor I2C address. See sensor documentation for this value.
<u>_result</u>	The function call result.

5.23.2.5 #define ACCLNxCalibrateZ( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, **ACCL\_CMD\_Z\_CAL**, *\_result*)

Calibrate ACCL-Nx Z-axis.

Calibrate the mindsensors ACCL-Nx sensor Z-axis. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.6 #define ACCLNxCalibrateZEnd( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, **ACCL\_CMD\_Z\_CAL\_END**, *\_result*)

Stop calibrating ACCL-Nx Z-axis.

Stop calibrating the mindsensors ACCL-Nx sensor Z-axis. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.7 #define ACCLNxResetCalibration( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, **ACCL\_CMD\_RESET\_CAL**, *\_result*)

Reset ACCL-Nx calibration.

Reset the mindsensors ACCL-Nx sensor calibration to factory settings. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.8 #define DISTNxGP2D12( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, **DIST\_CMD\_GP2D12**, *\_result*)

Configure DIST-Nx as GP2D12.

Configure the mindsensors DIST-Nx sensor as GP2D12. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.9 #define DISTNxGP2D120( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, DIST\_CMD\_GP2D120, *\_result*)

Configure DIST-Nx as GP2D120.

Configure the mindsensors DIST-Nx sensor as GP2D120. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.10 #define DISTNxGP2YA02( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, DIST\_CMD\_GP2YA02, *\_result*)

Configure DIST-Nx as GP2YA02.

Configure the mindsensors DIST-Nx sensor as GP2YA02. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.11 #define DISTNxGP2YA21( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, DIST\_CMD\_GP2YA21, *\_result*)

Configure DIST-Nx as GP2YA21.

Configure the mindsensors DIST-Nx sensor as GP2YA21. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.12 #define MSADPAOff( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, MS\_CMD\_ADPA\_OFF, *\_result*)

Turn off mindsensors ADPA mode.

Turn ADPA mode off for the mindsensors device on the specified port. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.13 #define MSADPAOn( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, MS\_CMD\_ADPA\_ON, *\_result*)

Turn on mindsensors ADPA mode.

Turn ADPA mode on for the mindsensors device on the specified port. The port must be configured as a Lowspeed port

before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	The function call result.

**5.23.2.14 #define MSDeenergize( `_port`, `_i2caddr`, `_result` ) \_\_I2CSendCmd(`_port`, `_i2caddr`, `MS_CMD_DEENERGIZED`, `_result`)**

Turn off power to device.

Turn power off for the mindsensors device on the specified port. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	The function call result.

**5.23.2.15 #define MSEnergize( `_port`, `_i2caddr`, `_result` ) \_\_I2CSendCmd(`_port`, `_i2caddr`, `MS_CMD_ENERGIZED`, `_result`)**

Turn on power to device.

Turn the power on for the mindsensors device on the specified port. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	The function call result.

**5.23.2.16 #define MSIRTrain( `_port`, `_i2caddr`, `_channel`, `_func`, `_result` ) \_\_MSIRTrain(`_port`, `_i2caddr`, `_channel`, `_func`, `FALSE`, `_result`)**

MSIRTrain function.

Control an IR Train receiver set to the specified channel using the mindsensors NRLink device. Valid function values are `TRAIN_FUNC_STOP`, `TRAIN_FUNC_INCR_SPEED`, `TRAIN_FUNC_DECR_SPEED`, and `TRAIN_FUNC_TOGGLE_LIGHT`. Valid channels are `TRAIN_CHANNEL_1` through `TRAIN_CHANNEL_3` and `TRAIN_CHANNEL_ALL`. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_channel</code>	The IR Train channel. See <a href="#">IR Train channel constants</a> .
<code>_func</code>	The IR Train function. See <a href="#">PF/IR Train function constants</a>
<code>_result</code>	The function call result. <code>NO_ERR</code> or <a href="#">Communications specific errors</a> .

---

```
5.23.2.17 #define MSPFComboDirect( _port, _i2caddr, _channel, _outa, _outb, _result ) __MSPFComboDirect(_port, _i2caddr,
    _channel, _outa, _outb, _result)
```

MSPFComboDirect function.

Execute a pair of Power Function motor commands on the specified channel using the mindsensors NRLink device. Commands for outa and outb are PF\_CMD\_STOP, PF\_CMD\_REV, PF\_CMD\_FWD, and PF\_CMD\_BRAKE. Valid channels are PF\_CHANNEL\_1 through PF\_CHANNEL\_4. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_channel</i>	The Power Function channel. See <a href="#">Power Function channel constants</a> .
<i>_outa</i>	The Power Function command for output A. See <a href="#">Power Function command constants</a> .
<i>_outb</i>	The Power Function command for output B. See <a href="#">Power Function command constants</a> .
<i>_result</i>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

---

```
5.23.2.18 #define MSPFComboPWM( _port, _i2caddr, _channel, _outa, _outb, _result ) __MSPFComboPWM(_port, _i2caddr,
    _channel, _outa, _outb, _result)
```

MSPFComboPWM function.

Control the speed of both outputs on a Power Function receiver set to the specified channel using the mindsensors NRLink device. Valid output values are PF\_PWM\_FLOAT, PF\_PWM\_FWD1, PF\_PWM\_FWD2, PF\_PWM\_FWD3, PF\_PWM\_FWD4, PF\_PWM\_FWD5, PF\_PWM\_FWD6, PF\_PWM\_FWD7, PF\_PWM\_BRAKE, PF\_PWM\_REV7, PF\_PWM\_REV6, PF\_PWM\_REV5, PF\_PWM\_REV4, PF\_PWM\_REV3, PF\_PWM\_REV2, and PF\_PWM\_REV1. Valid channels are PF\_CHANNEL\_1 through PF\_CHANNEL\_4. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_channel</i>	The Power Function channel. See <a href="#">Power Function channel constants</a> .
<i>_outa</i>	The Power Function PWM command for output A. See <a href="#">Power Function PWM option constants</a> .
<i>_outb</i>	The Power Function PWM command for output B. See <a href="#">Power Function PWM option constants</a> .
<i>_result</i>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

---

```
5.23.2.19 #define MSPFRawOutput( _port, _i2caddr, _nibble0, _nibble1, _nibble2, _result ) __MSPFRawOutput(_port, _i2caddr,
    _nibble0, _nibble1, _nibble2, _result)
```

MSPFRawOutput function.

Control a Power Function receiver set to the specified channel using the mindsensors NRLink device. Build the raw data stream using the 3 nibbles (4 bit values). The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_nibble0</i>	The first raw data nibble.
<i>_nibble1</i>	The second raw data nibble.
<i>_nibble2</i>	The third raw data nibble.
<i>_result</i>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

---

```
5.23.2.20 #define MSPFRepeat( _port, _i2caddr, _count, _delay, _result ) __MSPFRepeatLastCommand(_port, _i2caddr, _count,
 _delay, _result)
```

MSPFRepeat function.

Repeat sending the last Power Function command using the mindsensors NRLink device. Specify the number of times to repeat the command and the number of milliseconds of delay between each repetition. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_count</i>	The number of times to repeat the command.
<i>_delay</i>	The number of milliseconds to delay between each repetition.
<i>_result</i>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

---

```
5.23.2.21 #define MSPFSingleOutputCST( _port, _i2caddr, _channel, _out, _func, _result ) __MSPFSingleOutput(_port, _i2caddr,
 _channel, _out, _func, TRUE, _result)
```

MSPFSingleOutputCST function.

Control a single output on a Power Function receiver set to the specified channel using the mindsensors NRLink device. Select the desired output using [PF\\_OUT\\_A](#) or [PF\\_OUT\\_B](#). Valid functions are [PF\\_CST\\_CLEAR1\\_CLEAR2](#), [PF\\_CST\\_SET1\\_CLEAR2](#), [PF\\_CST\\_CLEAR1\\_SET2](#), [PF\\_CST\\_SET1\\_SET2](#), [PF\\_CST\\_INCREMENT\\_PWM](#), [PF\\_CST\\_DECREMENT\\_PWM](#), [PF\\_CST\\_FULL\\_FWD](#), [PF\\_CST\\_FULL\\_REV](#), and [PF\\_CST\\_TOGGLE\\_DIR](#). Valid channels are [PF\\_CHANNEL\\_1](#) through [PF\\_CHANNEL\\_4](#). The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_channel</i>	The Power Function channel. See <a href="#">Power Function channel constants</a> .
<i>_out</i>	The Power Function output. See <a href="#">Power Function output constants</a> .
<i>_func</i>	The Power Function CST function. See <a href="#">Power Function CST options constants</a> .
<i>_result</i>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

---

```
5.23.2.22 #define MSPFSingleOutputPWM( _port, _i2caddr, _channel, _out, _func, _result ) __MSPFSingleOutput(_port, _i2caddr,
 _channel, _out, _func, FALSE, _result)
```

MSPFSingleOutputPWM function.

Control the speed of a single output on a Power Function receiver set to the specified channel using the mindsensors NRLink device. Select the desired output using [PF\\_OUT\\_A](#) or [PF\\_OUT\\_B](#). Valid functions are [PF\\_PWM\\_FLOAT](#), [PF\\_PWM\\_FWD1](#), [PF\\_PWM\\_FWD2](#), [PF\\_PWM\\_FWD3](#), [PF\\_PWM\\_FWD4](#), [PF\\_PWM\\_FWD5](#), [PF\\_PWM\\_FWD6](#), [PF\\_PWM\\_FWD7](#), [PF\\_PWM\\_BRAKE](#), [PF\\_PWM\\_REV7](#), [PF\\_PWM\\_REV6](#), [PF\\_PWM\\_REV5](#), [PF\\_PWM\\_REV4](#), [PF\\_PWM\\_REV3](#), [PF\\_PWM\\_REV2](#), and [PF\\_PWM\\_REV1](#). Valid channels are [PF\\_CHANNEL\\_1](#) through [PF\\_CHANNEL\\_4](#). The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_channel</i>	The Power Function channel. See <a href="#">Power Function channel constants</a> .
<i>_out</i>	The Power Function output. See <a href="#">Power Function output constants</a> .
<i>_func</i>	The Power Function PWM function. See <a href="#">Power Function PWM option constants</a> .
<i>_result</i>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

---

```
5.23.2.23 #define MSPFSinglePin( _port, _i2caddr, _channel, _out, _pin, _func, _cont, _result ) __MSPFSinglePin(_port,
    _i2caddr, _channel, _out, _pin, _func, _cont, _result)
```

MSPFSinglePin function.

Control a single pin on a Power Function receiver set to the specified channel using the mindsensors NRLink device. Select the desired output using [PF\\_OUT\\_A](#) or [PF\\_OUT\\_B](#). Select the desired pin using [PF\\_PIN\\_C1](#) or [PF\\_PIN\\_C2](#). Valid functions are [PF\\_FUNC\\_NOCHANGE](#), [PF\\_FUNC\\_CLEAR](#), [PF\\_FUNC\\_SET](#), and [PF\\_FUNC\\_TOGGLE](#). Valid channels are [PF\\_CHANNEL\\_1](#) through [PF\\_CHANNEL\\_4](#). Specify whether the mode by passing true (continuous) or false (timeout) as the final parameter. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<a href="#">_port</a>	The sensor port. See <a href="#">NBC Input port constants</a> .
<a href="#">_i2caddr</a>	The sensor I2C address. See sensor documentation for this value.
<a href="#">_channel</a>	The Power Function channel. See <a href="#">Power Function channel constants</a> .
<a href="#">_out</a>	The Power Function output. See <a href="#">Power Function output constants</a> .
<a href="#">_pin</a>	The Power Function pin. See <a href="#">Power Function pin constants</a> .
<a href="#">_func</a>	The Power Function single pin function. See <a href="#">Power Function single pin function constants</a> .
<a href="#">_cont</a>	Control whether the mode is continuous or timeout.
<a href="#">_result</a>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

---

```
5.23.2.24 #define MSPFTrain( _port, _i2caddr, _channel, _func, _result ) __MSIRTrain(_port, _i2caddr, _channel, _func, TRUE,
    _result)
```

MSPFTrain function.

Control both outputs on a Power Function receiver set to the specified channel using the mindsensors NRLink device as if it were an IR Train receiver. Valid function values are [TRAIN\\_FUNC\\_STOP](#), [TRAIN\\_FUNC\\_INCR\\_SPEED](#), [TRAIN\\_FUNC\\_DECR\\_SPEED](#), and [TRAIN\\_FUNC\\_TOGGLE\\_LIGHT](#). Valid channels are [PF\\_CHANNEL\\_1](#) through [PF\\_CHANNEL\\_4](#). The port must be configured as a Lowspeed port before using this function.

#### Parameters

<a href="#">_port</a>	The sensor port. See <a href="#">NBC Input port constants</a> .
<a href="#">_i2caddr</a>	The sensor I2C address. See sensor documentation for this value.
<a href="#">_channel</a>	The Power Function channel. See <a href="#">Power Function channel constants</a> .
<a href="#">_func</a>	The Power Function train function. See <a href="#">PF/IR Train function constants</a> .
<a href="#">_result</a>	The function call result. <a href="#">NO_ERR</a> or <a href="#">Communications specific errors</a> .

---

```
5.23.2.25 #define MSRCXAbsVar( _varnum, _src, _value ) __MSRCXVarOp(RCX_AbsVarOp, _varnum, _src, _value)
```

MSRCXAbsVar function.

Send the AbsVar command to an RCX.

#### Parameters

<a href="#">_varnum</a>	The variable number to change.
<a href="#">_src</a>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<a href="#">_value</a>	The RCX value.

---

```
5.23.2.26 #define MSRCXAddToDatalog( _src, _value ) __MSRCXAddToDatalog(_src, _value)
```

MSRCXAddToDatalog function.

Send the AddToDatalog command to an RCX.

#### Parameters

<code>_src</code>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The RCX value.

**5.23.2.27 #define MSRCXAndVar( `_varnum`, `_src`, `_value` ) \_\_MSRCXVarOp(RCX\_AndVarOp, `_varnum`, `_src`, `_value`)**

MSRCXAndVar function.

Send the AndVar command to an RCX.

#### Parameters

<code>_varnum</code>	The variable number to change.
<code>_src</code>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The RCX value.

**5.23.2.28 #define MSRCXBatteryLevel( `_result` ) \_\_MSRCXBatteryLevel(`_result`)**

MSRCXBatteryLevel function.

Send the BatteryLevel command to an RCX to read the current battery level.

#### Parameters

<code>_result</code>	The RCX battery level.
----------------------	------------------------

**5.23.2.29 #define MSRCXBoot( ) \_\_MSRCXBoot()**

MSRCXBoot function.

Send the Boot command to an RCX.

**5.23.2.30 #define MSRCXCalibrateEvent( `_evt`, `_low`, `_hi`, `_hyst` ) \_\_MSRCXCalibrateEvent(`_evt`, `_low`, `_hi`, `_hyst`)**

MSRCXCalibrateEvent function.

Send the CalibrateEvent command to an RCX.

#### Parameters

<code>_evt</code>	The event number.
<code>_low</code>	The low threshold.
<code>_hi</code>	The high threshold.
<code>_hyst</code>	The hysteresis value.

**5.23.2.31 #define MSRCXClearAllEvents( ) \_\_MSRCXOpNoArgs(RCX\_ClearAllEventsOp)**

MSRCXClearAllEvents function.

Send the ClearAllEvents command to an RCX.

**5.23.2.32 #define MSRCXClearCounter( `_counter` ) \_\_MSRCXClearCounter(`_counter`)**

MSRCXClearCounter function.

Send the ClearCounter command to an RCX.

Parameters

<i>_counter</i>	The counter to clear.
-----------------	-----------------------

5.23.2.33 #define MSRCXClearMsg( ) \_\_MSRCXOpNoArgs(RCX\_ClearMsgOp)

MSRCXClearMsg function.

Send the ClearMsg command to an RCX.

5.23.2.34 #define MSRCXClearSensor( *\_port* ) \_\_MSRCXClearSensor(*\_port*)

MSRCXClearSensor function.

Send the ClearSensor command to an RCX.

Parameters

<i>_port</i>	The RCX port number.
--------------	----------------------

5.23.2.35 #define MSRCXClearSound( ) \_\_MSRCXOpNoArgs(RCX\_ClearSoundOp)

MSRCXClearSound function.

Send the ClearSound command to an RCX.

5.23.2.36 #define MSRCXClearTimer( *\_timer* ) \_\_MSRCXClearTimer(*\_timer*)

MSRCXClearTimer function.

Send the ClearTimer command to an RCX.

Parameters

<i>_timer</i>	The timer to clear.
---------------	---------------------

5.23.2.37 #define MSRCXCreateDatalog( *\_size* ) \_\_MSRCXCreateDatalog(*\_size*)

MSRCXCreateDatalog function.

Send the CreateDatalog command to an RCX.

Parameters

<i>_size</i>	The new datalog size.
--------------	-----------------------

5.23.2.38 #define MSRCXDecCounter( *\_counter* ) \_\_MSRCXDecCounter(*\_counter*)

MSRCXDecCounter function.

Send the DecCounter command to an RCX.

Parameters

<i>_counter</i>	The counter to decrement.
-----------------	---------------------------

## 5.23.2.39 #define MSRCXDeleteSub( \_s ) \_\_MSRCXDeleteSub(\_s)

MSRCXDeleteSub function.

Send the DeleteSub command to an RCX.

## Parameters

<code>_s</code>	The subroutine number to delete.
-----------------	----------------------------------

## 5.23.2.40 #define MSRCXDeleteSubs( ) \_\_MSRCXOpNoArgs(RCX\_DeleteSubsOp)

MSRCXDeleteSubs function.

Send the DeleteSubs command to an RCX.

## 5.23.2.41 #define MSRCXDeleteTask( \_t ) \_\_MSRCXDeleteTask(\_t)

MSRCXDeleteTask function.

Send the DeleteTask command to an RCX.

## Parameters

<code>_t</code>	The task number to delete.
-----------------	----------------------------

## 5.23.2.42 #define MSRCXDeleteTasks( ) \_\_MSRCXOpNoArgs(RCX\_DeleteTasksOp)

MSRCXDeleteTasks function.

Send the DeleteTasks command to an RCX.

## 5.23.2.43 #define MSRCXDisableOutput( \_outputs ) \_\_MSRCXSetGlobalOutput(\_outputs, RCX\_OUT\_OFF)

MSRCXDisableOutput function.

Send the DisableOutput command to an RCX.

## Parameters

<code>_outputs</code>	The RCX output(s) to disable. See <a href="#">RCX output constants</a> .
-----------------------	--

## 5.23.2.44 #define MSRCXDivVar( \_varnum, \_src, \_value ) \_\_MSRCXVarOp(RCX\_DivVarOp, \_varnum, \_src, \_value)

MSRCXDivVar function.

Send the DivVar command to an RCX.

## Parameters

<code>_varnum</code>	The variable number to change.
<code>_src</code>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The RCX value.

## 5.23.2.45 #define MSRCXEnableOutput( \_outputs ) \_\_MSRCXSetGlobalOutput(\_outputs, RCX\_OUT\_ON)

MSRCXEnableOutput function.

Send the EnableOutput command to an RCX.

**Parameters**

<i>_outputs</i>	The RCX output(s) to enable. See <a href="#">RCX output constants</a> .
-----------------	---

5.23.2.46 `#define MSRCXEvent( _src, _value ) __MSRCXEvent(_src, _value)`

MSRCXEvent function.

Send the Event command to an RCX.

**Parameters**

<i>_src</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_value</i>	The RCX value.

5.23.2.47 `#define MSRCXFloat( _outputs ) __MSRCXSetOutput(_outputs, RCX_OUT_FLOAT)`

MSRCXFloat function.

Send commands to an RCX to float the specified outputs.

**Parameters**

<i>_outputs</i>	The RCX output(s) to float. See <a href="#">RCX output constants</a> .
-----------------	--

5.23.2.48 `#define MSRCXFwd( _outputs ) __MSRCXSetDirection(_outputs, RCX_OUT_FWD)`

MSRCXFwd function.

Send commands to an RCX to set the specified outputs to the forward direction.

**Parameters**

<i>_outputs</i>	The RCX output(s) to set forward. See <a href="#">RCX output constants</a> .
-----------------	--

5.23.2.49 `#define MSRCXIncCounter( _counter ) __MSRCXIncCounter(_counter)`

MSRCXIncCounter function.

Send the IncCounter command to an RCX.

**Parameters**

<i>_counter</i>	The counter to increment.
-----------------	---------------------------

5.23.2.50 `#define MSRCXInvertOutput( _outputs ) __MSRCXSetGlobalDirection(_outputs, RCX_OUT_REV)`

MSRCXInvertOutput function.

Send the InvertOutput command to an RCX.

**Parameters**

<i>_outputs</i>	The RCX output(s) to invert. See <a href="#">RCX output constants</a> .
-----------------	---

5.23.2.51 #define MSRCXMulVar( *\_varnum*, *\_src*, *\_value* ) \_\_MSRCXVarOp(RCX\_MulVarOp, *\_varnum*, *\_src*, *\_value*)

MSRCXMulVar function.

Send the MulVar command to an RCX.

#### Parameters

<i>_varnum</i>	The variable number to change.
<i>_src</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_value</i>	The RCX value.

5.23.2.52 #define MSRCXMuteSound( ) \_\_MSRCXOpNoArgs(RCX\_MuteSoundOp)

MSRCXMuteSound function.

Send the MuteSound command to an RCX.

5.23.2.53 #define MSRCXObvertOutput( *\_outputs* ) \_\_MSRCXSetGlobalDirection(*\_outputs*, RCX\_OUT\_FWD)

MSRCXObvertOutput function.

Send the ObvertOutput command to an RCX.

#### Parameters

<i>_outputs</i>	The RCX output(s) to obvert. See <a href="#">RCX output constants</a> .
-----------------	---

5.23.2.54 #define MSRCXOff( *\_outputs* ) \_\_MSRCXSetOutput(*\_outputs*, RCX\_OUT\_OFF)

MSRCXOff function.

Send commands to an RCX to turn off the specified outputs.

#### Parameters

<i>_outputs</i>	The RCX output(s) to turn off. See <a href="#">RCX output constants</a> .
-----------------	---

5.23.2.55 #define MSRCXOn( *\_outputs* ) \_\_MSRCXSetOutput(*\_outputs*, RCX\_OUT\_ON)

MSRCXOn function.

Send commands to an RCX to turn on the specified outputs.

#### Parameters

<i>_outputs</i>	The RCX output(s) to turn on. See <a href="#">RCX output constants</a> .
-----------------	--

5.23.2.56 #define MSRCXOnFor( *\_outputs*, *\_ms* ) \_\_MSRCXOnFor(*\_outputs*, *\_ms*)

MSRCXOnFor function.

Send commands to an RCX to turn on the specified outputs in the forward direction for the specified duration.

#### Parameters

<i>_outputs</i>	The RCX output(s) to turn on. See <a href="#">RCX output constants</a> .
<i>_ms</i>	The number of milliseconds to leave the outputs on

5.23.2.57 #define MSRCXOnFwd( *\_outputs* ) \_\_MSRCXOnFwd(*\_outputs*)

MSRCXOnFwd function.

Send commands to an RCX to turn on the specified outputs in the forward direction.

**Parameters**

<i>_outputs</i>	The RCX output(s) to turn on in the forward direction. See <a href="#">RCX output constants</a> .
-----------------	---

5.23.2.58 #define MSRCXOnRev( *\_outputs* ) \_\_MSRCXOnRev(*\_outputs*)

MSRCXOnRev function.

Send commands to an RCX to turn on the specified outputs in the reverse direction.

**Parameters**

<i>_outputs</i>	The RCX output(s) to turn on in the reverse direction. See <a href="#">RCX output constants</a> .
-----------------	---

5.23.2.59 #define MSRCXOrVar( *\_varnum*, *\_src*, *\_value* ) \_\_MSRCXVarOp(RCX\_OrVarOp, *\_varnum*, *\_src*, *\_value*)

MSRCXOrVar function.

Send the OrVar command to an RCX.

**Parameters**

<i>_varnum</i>	The variable number to change.
<i>_src</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_value</i>	The RCX value.

5.23.2.60 #define MSRCXPBTurnOff( ) \_\_MSRCXOpNoArgs(RCX\_PBTurnOffOp)

MSRCXPBTurnOff function.

Send the PBTurnOff command to an RCX.

5.23.2.61 #define MSRCXPing( ) \_\_MSRCXOpNoArgs(RCX\_PingOp)

MSRCXPing function.

Send the Ping command to an RCX.

5.23.2.62 #define MSRCXPlaySound( *\_snd* ) \_\_MSRCXPlaySound(*\_snd*)

MSRCXPlaySound function.

Send the PlaySound command to an RCX.

**Parameters**

<i>_snd</i>	The sound number to play.
-------------	---------------------------

5.23.2.63 #define MSRCXPlayTone( *\_freq*, *\_duration* ) \_\_MSRCXPlayTone(*\_freq*, *\_duration*)

MSRCXPlayTone function.

Send the PlayTone command to an RCX.

#### Parameters

<i>_freq</i>	The frequency of the tone to play.
<i>_duration</i>	The duration of the tone to play.

5.23.2.64 #define MSRCXPlayToneVar( *\_varnum*, *\_duration* ) \_\_MSRCXPlayToneVar(*\_varnum*, *\_duration*)

MSRCXPlayToneVar function.

Send the PlayToneVar command to an RCX.

#### Parameters

<i>_varnum</i>	The variable containing the tone frequency to play.
<i>_duration</i>	The duration of the tone to play.

5.23.2.65 #define MSRCXPoll( *\_src*, *\_value*, *\_result* ) \_\_MSRCXPoll(*\_src*, *\_value*, *\_result*)

MSRCXPoll function.

Send the Poll command to an RCX to read a signed 2-byte value at the specified source and value combination.

#### Parameters

<i>_src</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_value</i>	The RCX value.
<i>_result</i>	The value read from the specified port and value.

5.23.2.66 #define MSRCXPollMemory( *\_memaddress*, *\_result* ) \_\_MSRCXPollMemory(*\_memaddress*, *\_result*)

MSRCXPollMemory function.

Send the PollMemory command to an RCX.

#### Parameters

<i>_memaddress</i>	The RCX memory address.
<i>_result</i>	The value read from the specified address.

5.23.2.67 #define MSRCXRemote( *\_cmd* ) \_\_MSRCXRemote(*\_cmd*)

MSRCXRemote function.

Send the Remote command to an RCX.

#### Parameters

<i>_cmd</i>	The RCX IR remote command to send. See <a href="#">RCX IR remote constants</a> .
-------------	--

5.23.2.68 #define MSRCXReset( ) \_\_MSRCXReset()

MSRCXReset function.

Send the Reset command to an RCX.

5.23.2.69 #define MSRCXRev( *\_outputs* ) \_\_MSRCXSetDirection(*\_outputs*, RCX\_OUT\_REV)

MSRCXRev function.

Send commands to an RCX to set the specified outputs to the reverse direction.

**Parameters**

<i>_outputs</i>	The RCX output(s) to reverse direction. See <a href="#">RCX output constants</a> .
-----------------	--

5.23.2.70 #define MSRCXSelectDisplay( *\_src*, *\_value* ) \_\_MSRCXSelectDisplay(*\_src*, *\_value*)

MSRCXSelectDisplay function.

Send the SelectDisplay command to an RCX.

**Parameters**

<i>_src</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_value</i>	The RCX value.

5.23.2.71 #define MSRCXSelectProgram( *\_prog* ) \_\_MSRCXSelectProgram(*\_prog*)

MSRCXSelectProgram function.

Send the SelectProgram command to an RCX.

**Parameters**

<i>_prog</i>	The program number to select.
--------------	-------------------------------

5.23.2.72 #define MSRCXSendSerial( *\_first*, *\_count* ) \_\_MSRCXSendSerial(*\_first*, *\_count*)

MSRCXSendSerial function.

Send the SendSerial command to an RCX.

**Parameters**

<i>_first</i>	The first byte address.
<i>_count</i>	The number of bytes to send.

5.23.2.73 #define MSRCXSet( *\_dstsrc*, *\_dstval*, *\_src*, *\_value* ) \_\_MSRCXSet(*\_dstsrc*, *\_dstval*, *\_src*, *\_value*)

MSRCXSet function.

Send the Set command to an RCX.

**Parameters**

<i>_dstsrc</i>	The RCX destination source. See <a href="#">RCX and Scout source constants</a> .
<i>_dstval</i>	The RCX destination value.
<i>_src</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_value</i>	The RCX value.

5.23.2.74 #define MSRCXSetDirection( *\_outputs*, *\_dir* ) \_\_MSRCXSetDirection(*\_outputs*, *\_dir*)

MSRCXSetDirection function.

Send the SetDirection command to an RCX to configure the direction of the specified outputs.

#### Parameters

<i>_outputs</i>	The RCX output(s) to set direction. See <a href="#">RCX output constants</a> .
<i>_dir</i>	The RCX output direction. See <a href="#">RCX output direction constants</a> .

5.23.2.75 #define MSRCXSetEvent( *\_evt*, *\_src*, *\_type* ) \_\_MSRCXSetEvent(*\_evt*, *\_src*, *\_type*)

MSRCXSetEvent function.

Send the SetEvent command to an RCX.

#### Parameters

<i>_evt</i>	The event number to set.
<i>_src</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_type</i>	The event type.

5.23.2.76 #define MSRCXSetGlobalDirection( *\_outputs*, *\_dir* ) \_\_MSRCXSetGlobalDirection(*\_outputs*, *\_dir*)

MSRCXSetGlobalDirection function.

Send the SetGlobalDirection command to an RCX.

#### Parameters

<i>_outputs</i>	The RCX output(s) to set global direction. See <a href="#">RCX output constants</a> .
<i>_dir</i>	The RCX output direction. See <a href="#">RCX output direction constants</a> .

5.23.2.77 #define MSRCXSetGlobalOutput( *\_outputs*, *\_mode* ) \_\_MSRCXSetGlobalOutput(*\_outputs*, *\_mode*)

MSRCXSetGlobalOutput function.

Send the SetGlobalOutput command to an RCX.

#### Parameters

<i>_outputs</i>	The RCX output(s) to set global mode. See <a href="#">RCX output constants</a> .
<i>_mode</i>	The RCX output mode. See <a href="#">RCX output mode constants</a> .

5.23.2.78 #define MSRCXSetMaxPower( *\_outputs*, *\_pwrsrc*, *\_pwrval* ) \_\_MSRCXSetMaxPower(*\_outputs*, *\_pwrsrc*, *\_pwrval*)

MSRCXSetMaxPower function.

Send the SetMaxPower command to an RCX.

#### Parameters

<i>_outputs</i>	The RCX output(s) to set max power. See <a href="#">RCX output constants</a> .
<i>_pwrsrc</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_pwrval</i>	The RCX value.

5.23.2.79 #define MSRCXSetMessage( *\_msg* ) \_\_MSRCXSetMessage(*\_msg*)

MSRCXSetMessage function.

Send the SetMessage command to an RCX.

## Parameters

<i>_msg</i>	The numeric message to send.
-------------	------------------------------

5.23.2.80 #define MSRCXSetNRLinkPort( *\_port*, *\_i2caddr* ) \_\_MSRCXSetNRLink(*\_port*, *\_i2caddr*)

MSRCXSetIRLinkPort function.

Set the global port in advance of using the MSRCX\* and MSScout\* API functions for sending RCX and Scout messages over the mindsensors NRLink device. The port must be configured as a Lowspeed port before using any of the mindsensors RCX and Scout NRLink functions.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.

5.23.2.81 #define MSRCXSetOutput( *\_outputs*, *\_mode* ) \_\_MSRCXSetOutput(*\_outputs*, *\_mode*)

MSRCXSetOutput function.

Send the SetOutput command to an RCX to configure the mode of the specified outputs

## Parameters

<i>_outputs</i>	The RCX output(s) to set mode. See <a href="#">RCX output constants</a> .
<i>_mode</i>	The RCX output mode. See <a href="#">RCX output mode constants</a> .

5.23.2.82 #define MSRCXSetPower( *\_outputs*, *\_pwrsrc*, *\_pwrval* ) \_\_MSRCXSetPower(*\_outputs*, *\_pwrsrc*, *\_pwrval*)

MSRCXSetPower function.

Send the SetPower command to an RCX to configure the power level of the specified outputs.

## Parameters

<i>_outputs</i>	The RCX output(s) to set power. See <a href="#">RCX output constants</a> .
<i>_pwrsrc</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_pwrval</i>	The RCX value.

5.23.2.83 #define MSRCXSetPriority( *\_p* ) \_\_MSRCXSetPriority(*\_p*)

MSRCXSetPriority function.

Send the SetPriority command to an RCX.

## Parameters

<i>_p</i>	The new task priority.
-----------	------------------------

5.23.2.84 #define MSRCXSetSensorMode( *\_port*, *\_mode* ) \_\_MSRCXSetSensorMode(*\_port*, *\_mode*)

MSRCXSetSensorMode function.

Send the SetSensorMode command to an RCX.

#### Parameters

<i>_port</i>	The RCX sensor port.
<i>_mode</i>	The RCX sensor mode.

5.23.2.85 #define MSRCXSetSensorType( *\_port*, *\_type* ) \_\_MSRCXSetSensorType(*\_port*, *\_type*)

MSRCXSetSensorType function.

Send the SetSensorType command to an RCX.

#### Parameters

<i>_port</i>	The RCX sensor port.
<i>_type</i>	The RCX sensor type.

5.23.2.86 #define MSRCXSetSleepTime( *\_t* ) \_\_MSRCXSetSleepTime(*\_t*)

MSRCXSetSleepTime function.

Send the SetSleepTime command to an RCX.

#### Parameters

<i>_t</i>	The new sleep time value.
-----------	---------------------------

5.23.2.87 #define MSRCXSetTxPower( *\_pwr* ) \_\_MSRCXSetTxPower(*\_pwr*)

MSRCXSetTxPower function.

Send the SetTxPower command to an RCX.

#### Parameters

<i>_pwr</i>	The IR transmit power level.
-------------	------------------------------

5.23.2.88 #define MSRCXSetUserDisplay( *\_src*, *\_value*, *\_precision* ) \_\_MSRCXSetUserDisplay(*\_src*, *\_value*, *\_precision*)

MSRCXSetUserDisplay function.

Send the SetUserDisplay command to an RCX.

#### Parameters

<i>_src</i>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<i>_value</i>	The RCX value.
<i>_precision</i>	The number of digits of precision.

5.23.2.89 #define MSRCXSetVar( *\_varnum*, *\_src*, *\_value* ) \_\_MSRCXVarOp(RCX\_SetVarOp, *\_varnum*, *\_src*, *\_value*)

MSRCXSetVar function.

Send the SetVar command to an RCX.

**Parameters**

<code>_varnum</code>	The variable number to change.
<code>_src</code>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The RCX value.

**5.23.2.90 #define MSRCXSetWatch( `_hours`, `_minutes` ) \_\_MSRCXSetWatch(`_hours`, `_minutes`)**

MSRCXSetWatch function.

Send the SetWatch command to an RCX.

**Parameters**

<code>_hours</code>	The new watch time hours value.
<code>_minutes</code>	The new watch time minutes value.

**5.23.2.91 #define MSRCXSgnVar( `_varnum`, `_src`, `_value` ) \_\_MSRCXVarOp(RCX\_SgnVarOp, `_varnum`, `_src`, `_value`)**

MSRCXSgnVar function.

Send the SgnVar command to an RCX.

**Parameters**

<code>_varnum</code>	The variable number to change.
<code>_src</code>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The RCX value.

**5.23.2.92 #define MSRCXStartTask( `_t` ) \_\_MSRCXStartTask(`_t`)**

MSRCXStartTask function.

Send the StartTask command to an RCX.

**Parameters**

<code>_t</code>	The task number to start.
-----------------	---------------------------

**5.23.2.93 #define MSRCXStopAllTasks( ) \_\_MSRCXOpNoArgs(RCX\_StopAllTasksOp)**

MSRCXStopAllTasks function.

Send the StopAllTasks command to an RCX.

**5.23.2.94 #define MSRCXStopTask( `_t` ) \_\_MSRCXStopTask(`_t`)**

MSRCXStopTask function.

Send the StopTask command to an RCX.

**Parameters**

<code>_t</code>	The task number to stop.
-----------------	--------------------------

5.23.2.95 #define MSRCXSubVar( \_varnum, \_src, \_value ) \_\_MSRCXVarOp(RCX\_SubVarOp, \_varnum, \_src, \_value)

MSRCXSubVar function.

Send the SubVar command to an RCX.

#### Parameters

<u>_varnum</u>	The variable number to change.
<u>_src</u>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<u>_value</u>	The RCX value.

5.23.2.96 #define MSRCXSumVar( \_varnum, \_src, \_value ) \_\_MSRCXVarOp(RCX\_SumVarOp, \_varnum, \_src, \_value)

MSRCXSumVar function.

Send the SumVar command to an RCX.

#### Parameters

<u>_varnum</u>	The variable number to change.
<u>_src</u>	The RCX source. See <a href="#">RCX and Scout source constants</a> .
<u>_value</u>	The RCX value.

5.23.2.97 #define MSRCXToggle( \_outputs ) \_\_MSRCXSetDirection(\_outputs, RCX\_OUT\_TOGGLE)

MSRCXToggle function.

Send commands to an RCX to toggle the direction of the specified outputs.

#### Parameters

<u>_outputs</u>	The RCX output(s) to toggle. See <a href="#">RCX output constants</a> .
-----------------	---

5.23.2.98 #define MSRCXUnlock( ) \_\_MSRCXUnlock()

MSRCXUnlock function.

Send the Unlock command to an RCX.

5.23.2.99 #define MSRCXUnmuteSound( ) \_\_MSRCXOpNoArgs(RCX\_UnmuteSoundOp)

MSRCXUnmuteSound function.

Send the UnmuteSound command to an RCX.

5.23.2.100 #define MSReadValue( \_port, \_i2caddr, \_reg, \_bytes, \_out, \_result ) \_\_MSReadValue(\_port, \_i2caddr, \_reg, \_bytes, \_out, \_result)

Read a mindsensors device value.

Read a one, two, or four byte value from a mindsensors sensor. The value must be stored with the least significant byte (LSB) first (i.e., little endian). Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_i2caddr</u>	The sensor I2C address. See sensor documentation for this value.

<i>_reg</i>	The device register to read.
<i>_bytes</i>	The number of bytes to read. Only 1, 2, or 4 byte values are supported.
<i>_out</i>	The value read from the device.
<i>_result</i>	The function call result.

5.23.2.101 #define MSScoutCalibrateSensor( ) \_\_MSRCXOpNoArgs(RCX\_LSCalibrateOp)

MSScoutCalibrateSensor function.

Send the CalibrateSensor command to a Scout.

5.23.2.102 #define MSScoutMuteSound( ) \_\_MSScoutMuteSound()

MSScoutMuteSound function.

Send the MuteSound command to a Scout.

5.23.2.103 #define MSScoutSelectSounds( *\_grp* ) \_\_MSScoutSelectSounds(*\_grp*)

MSScoutSelectSounds function.

Send the SelectSounds command to a Scout.

#### Parameters

<i>_grp</i>	The Scout sound group to select.
-------------	----------------------------------

5.23.2.104 #define MSScoutSendVLL( *\_src*, *\_value* ) \_\_MSScoutSendVLL(*\_src*, *\_value*)

MSScoutSendVLL function.

Send the SendVLL command to a Scout.

#### Parameters

<i>_src</i>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<i>_value</i>	The Scout value.

5.23.2.105 #define MSScoutSetCounterLimit( *\_ctr*, *\_src*, *\_value* ) \_\_MSScoutSetCounterLimit(*\_ctr*, *\_src*, *\_value*)

MSScoutSetCounterLimit function.

Send the SetCounterLimit command to a Scout.

#### Parameters

<i>_ctr</i>	The counter for which to set the limit.
<i>_src</i>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<i>_value</i>	The Scout value.

5.23.2.106 #define MSScoutSetEventFeedback( *\_src*, *\_value* ) \_\_MSScoutSetEventFeedback(*\_src*, *\_value*)

MSScoutSetEventFeedback function.

Send the SetEventFeedback command to a Scout.

## Parameters

<code>_src</code>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The Scout value.

5.23.2.107 #define MSScoutSetLight( `_x` ) \_\_MSScoutSetLight(`_x`)

MSScoutSetLight function.

Send the SetLight command to a Scout.

## Parameters

<code>_x</code>	Set the light on or off using this value. See <a href="#">Scout light constants</a> .
-----------------	---

5.23.2.108 #define MSScoutSetScoutMode( `_mode` ) \_\_MSScoutSetScoutMode(`_mode`)

MSScoutSetScoutMode function.

Send the SetScoutMode command to a Scout.

## Parameters

<code>_mode</code>	Set the scout mode. See <a href="#">Scout mode constants</a> .
--------------------	--

5.23.2.109 #define MSScoutSetScoutRules( `_m`, `_t`, `_l`, `_tm`, `_fx` ) \_\_MSScoutSetScoutRules(`_m`, `_t`, `_l`, `_tm`, `_fx`)

MSScoutSetScoutRules function.

Send the SetScoutRules command to a Scout.

## Parameters

<code>_m</code>	Scout motion rule. See <a href="#">Scout motion rule constants</a> .
<code>_t</code>	Scout touch rule. See <a href="#">Scout touch rule constants</a> .
<code>_l</code>	Scout light rule. See <a href="#">Scout light rule constants</a> .
<code>_tm</code>	Scout transmit rule. See <a href="#">Scout transmit rule constants</a> .
<code>_fx</code>	Scout special effects rule. See <a href="#">Scout special effect constants</a> .

5.23.2.110 #define MSScoutSetSensorClickTime( `_src`, `_value` ) \_\_MSScoutSetSensorClickTime(`_src`, `_value`)

MSScoutSetSensorClickTime function.

Send the SetSensorClickTime command to a Scout.

## Parameters

<code>_src</code>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The Scout value.

5.23.2.111 #define MSScoutSetSensorHysteresis( `_src`, `_value` ) \_\_MSScoutSetSensorHysteresis(`_src`, `_value`)

MSScoutSetSensorHysteresis function.

Send the SetSensorHysteresis command to a Scout.

## Parameters

<code>_src</code>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The Scout value.

5.23.2.112 `#define MSScoutSetSensorLowerLimit( _src, _value ) __MSScoutSetSensorLowerLimit(_src, _value)`

MSScoutSetSensorLowerLimit function.

Send the SetSensorLowerLimit command to a Scout.

## Parameters

<code>_src</code>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The Scout value.

5.23.2.113 `#define MSScoutSetSensorUpperLimit( _src, _value ) __MSScoutSetSensorUpperLimit(_src, _value)`

MSScoutSetSensorUpperLimit function.

Send the SetSensorUpperLimit command to a Scout.

## Parameters

<code>_src</code>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The Scout value.

5.23.2.114 `#define MSScoutSetTimerLimit( _tmr, _src, _value ) __MSScoutSetTimerLimit(_tmr, _src, _value)`

MSScoutSetTimerLimit function.

Send the SetTimerLimit command to a Scout.

## Parameters

<code>_tmr</code>	The timer for which to set a limit.
<code>_src</code>	The Scout source. See <a href="#">RCX and Scout source constants</a> .
<code>_value</code>	The Scout value.

5.23.2.115 `#define MSScoutUnmuteSound( ) __MSScoutUnmuteSound()`

MSScoutUnmuteSound function.

Send the UnmuteSound command to a Scout.

5.23.2.116 `#define NRLink2400( _port, _i2caddr, _result ) __I2CSendCmd(_port, _i2caddr, NRLINK_CMD_2400, _result)`

Configure NRLink in 2400 baud mode.

Configure the mindsensors NRLink device in 2400 baud mode. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	The function call result.

5.23.2.117 #define NRLink4800( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, **NRLINK\_CMD\_4800**, *\_result*)

Configure NRLink in 4800 baud mode.

Configure the mindsensors NRLink device in 4800 baud mode. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.118 #define NRLinkFlush( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, **NRLINK\_CMD\_FLUSH**, *\_result*)

Flush NRLink buffers.

Flush the mindsensors NRLink device buffers. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.119 #define NRLinkIRLong( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, **NRLINK\_CMD\_IR\_LONG**, *\_result*)

Configure NRLink in IR long mode.

Configure the mindsensors NRLink device in IR long mode. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.120 #define NRLinkIRShort( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, **NRLINK\_CMD\_IR\_SHORT**, *\_result*)

Configure NRLink in IR short mode.

Configure the mindsensors NRLink device in IR short mode. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.121 #define NRLinkSetPF( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, **NRLINK\_CMD\_SET\_PF**, *\_result*)

Configure NRLink in power function mode.

Configure the mindsensors NRLink device in power function mode. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.122 #define NRLinkSetRCX( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, **NRLINK\_CMD\_SET\_RCX**, *\_result*)

Configure NRLink in RCX mode.

Configure the mindsensors NRLink device in RCX mode. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.123 #define NRLinkSetTrain( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, **NRLINK\_CMD\_SET\_TRAIN**, *\_result*)

Configure NRLink in IR train mode.

Configure the mindsensors NRLink device in IR train mode. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.124 #define NRLinkTxRaw( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, **NRLINK\_CMD\_TX\_RAW**, *\_result*)

Configure NRLink in raw IR transmit mode.

Configure the mindsensors NRLink device in raw IR transmit mode. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.125 #define NXTHIDAsciiMode( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, **NXTHID\_CMD\_ASCII**, *\_result*)

Set NXTHID into ASCII data mode.

Set the NXTHID device into ASCII data mode. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.126 `#define NXTHIDDirectMode( _port, _i2caddr, _result ) __I2CSendCmd(_port, _i2caddr, NXTHID_CMD_DIRECT, _result)`

Set NXTHID into direct data mode.

Set the NXTHID device into direct data mode. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.127 `#define NXTHIDLoadCharacter( _port, _i2caddr, _modifier, _character, _result ) __NXTHIDLoadCharacter(_port, _i2caddr, _modifier, _character, _result)`

Load NXTHID character.

Load a character into the NXTHID device. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_modifier</code>	The key modifier.
<code>_character</code>	The character.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.128 `#define NXTHIDTransmit( _port, _i2caddr, _result ) __I2CSendCmd(_port, _i2caddr, NXTHID_CMD_TRANSMIT, _result)`

Transmit NXTHID character.

Transmit a single character to a computer using the NXTHID device. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.129 #define NXTLineLeaderCalibrateBlack( _port, _i2caddr, _result ) __I2CSendCmd(_port, _i2caddr,
NXTLL_CMD_BLACK, _result)
```

Calibrate NXTLineLeader black color.

Store calibration data for the black color. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.130 #define NXTLineLeaderCalibrateWhite( _port, _i2caddr, _result ) __I2CSendCmd(_port, _i2caddr,
NXTLL_CMD_WHITE, _result)
```

Calibrate NXTLineLeader white color.

Store calibration data for the white color. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.131 #define NXTLineLeaderInvert( _port, _i2caddr, _result ) __I2CSendCmd(_port, _i2caddr, NXTLL_CMD_INVERT,
_result)
```

Invert NXTLineLeader colors.

Invert color sensing so that the device can detect a white line on a black background. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.132 #define NXTLineLeaderPowerDown( _port, _i2caddr, _result ) __I2CSendCmd(_port, _i2caddr,
NXTLL_CMD_POWERDOWN, _result)
```

Powerdown NXTLineLeader device.

Put the NXTLineLeader to sleep so that it does not consume power when it is not required. The device wakes up on its own when any I2C communication happens or you can specifically wake it up by using the [NXTLineLeaderPowerUp](#) command. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.

<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.
----------------------	---

5.23.2.133 #define NXTLineLeaderPowerUp( `_port`, `_i2caddr`, `_result` ) \_\_I2CSendCmd(`_port`, `_i2caddr`, `NXTLL_CMD_POWERUP`, `_result`)

Powerup NXTLineLeader device.

Wake up the NXTLineLeader device so that it can be used. The device can be put to sleep using the [NXTLineLeader-PowerDown](#) command. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.134 #define NXTLineLeaderReset( `_port`, `_i2caddr`, `_result` ) \_\_I2CSendCmd(`_port`, `_i2caddr`, `NXTLL_CMD_RESET`, `_result`)

Reset NXTLineLeader color inversion.

Reset the NXTLineLeader color detection back to its default state (black line on a white background). The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.135 #define NXTLineLeaderSnapshot( `_port`, `_i2caddr`, `_result` ) \_\_I2CSendCmd(`_port`, `_i2caddr`, `NXTLL_CMD_SNAPSHOT`, `_result`)

Take NXTLineLeader line snapshot.

Takes a snapshot of the line under the sensor and tracks that position in subsequent tracking operations. This function also will set color inversion if it sees a white line on a black background. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.136 #define NXTPowerMeterResetCounters( `_port`, `_i2caddr`, `_result` ) \_\_I2CSendCmd(`_port`, `_i2caddr`, `NXTPM_CMD_RESET`, `_result`)

Reset NXTPowerMeter counters.

Reset the NXTPowerMeter counters back to zero. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

**5.23.2.137 #define NXTServoEditMacro( `_port`, `_i2caddr`, `_result` ) \_\_NXTServoEditMacro(`_port`, `_i2caddr`, `_result`)**

Edit NXTServo macro.

Put the NXTServo device into macro edit mode. This operation changes the I2C address of the device to 0x40. Macros are written to EEPROM addresses between 0x21 and 0xFF. Use [NXTServoQuitEdit](#) to return the device to its normal operation mode. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

**5.23.2.138 #define NXTServoGotoMacroAddress( `_port`, `_i2caddr`, `_macro`, `_result` ) \_\_NXTServoGotoMacroAddress(`_port`, `_i2caddr`, `_macro`, `_result`)**

Goto NXTServo macro address.

Run the macro found at the specified EEPROM macro address. This command re-initializes the macro environment. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_macro</code>	The EEPROM macro address.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

**5.23.2.139 #define NXTServoHaltMacro( `_port`, `_i2caddr`, `_result` ) \_\_I2CSendCmd(`_port`, `_i2caddr`, `NXTSERVO_CMD_HALT`, `_result`)**

Halt NXTServo macro.

Halt a macro executing on the NXTServo device. This command re-initializes the macro environment. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.140 #define NXTServoInit( \_port, \_i2caddr, \_servo, \_result ) \_\_NXTServoInit(\_port, \_i2caddr, \_servo, \_result)

Initialize NXTServo servo properties.

Store the initial speed and position properties of the servo motor 'n'. Current speed and position values of the nth servo is read from the servo speed register and servo position register and written to permanent memory. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_i2caddr</u>	The sensor I2C address. See sensor documentation for this value.
<u>_servo</u>	The servo number. See <a href="#">MindSensors NXTServo servo numbers</a> group.
<u>_result</u>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.141 #define NXTServoPauseMacro( \_port, \_i2caddr, \_result ) \_\_I2CSendCmd(\_port, \_i2caddr, NXTSERVO\_CMD\_PAUSE, \_result)

Pause NXTServo macro.

Pause a macro executing on the NXTServo device. This command will pause the currently executing macro, and save the environment for subsequent resumption. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_i2caddr</u>	The sensor I2C address. See sensor documentation for this value.
<u>_result</u>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.142 #define NXTServoQuitEdit( \_port, \_result ) \_\_MSWriteToRegister(\_port, MS\_ADDR\_NXTSERVO\_EM, NXTSERVO\_EM\_REG\_CMD, NXTSERVO\_EM\_CMD\_QUIT, \_result)

Quit NXTServo macro edit mode.

Stop editing NXTServo device macro EEPROM memory. Use [NXTServoEditMacro](#) to start editing a macro. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_result</u>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.143 #define NXTServoReset( \_port, \_i2caddr, \_result ) \_\_I2CSendCmd(\_port, \_i2caddr, NXTSERVO\_CMD\_RESET, \_result)

Reset NXTServo properties.

Reset NXTServo device properties to factory defaults. Initial position = 1500. Initial speed = 0. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_i2caddr</u>	The sensor I2C address. See sensor documentation for this value.

<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.
----------------------	---

5.23.2.144 #define NXTServoResumeMacro( `_port`, `_i2caddr`, `_result` ) \_\_I2CSendCmd(`_port`, `_i2caddr`, `NXTSERVO_CMD_RESUME`, `_result`)

Resume NXTServo macro.

Resume a macro executing on the NXTServo device. This command resumes executing a macro where it was paused last, using the same environment. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.145 #define PFMateSend( `_port`, `_i2caddr`, `_channel`, `_motors`, `_cmdA`, `_spdA`, `_cmdB`, `_spdB`, `_result` ) \_\_PFMateSend(`_port`, `_i2caddr`, `_channel`, `_motors`, `_cmdA`, `_spdA`, `_cmdB`, `_spdB`, `_result`)

Send PFMate command.

Send a PFMate command to the power function IR receiver. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_channel</code>	The power function IR receiver channel. See the <a href="#">PFMate channel constants</a> group.
<code>_motors</code>	The motor(s) to control. See the <a href="#">PFMate motor constants</a> group.
<code>_cmdA</code>	The power function command for motor A. See the <a href="#">Power Function command constants</a> group.
<code>_spdA</code>	The power function speed for motor A.
<code>_cmdB</code>	The power function command for motor B. See the <a href="#">Power Function command constants</a> group.
<code>_spdB</code>	The power function speed for motor B.
<code>_result</code>	The function call result.

5.23.2.146 #define PFMateSendRaw( `_port`, `_i2caddr`, `_channel`, `_b1`, `_b2`, `_result` ) \_\_PFMateSendRaw(`_port`, `_i2caddr`, `_channel`, `_b1`, `_b2`, `_result`)

Send raw PFMate command.

Send a raw PFMate command to the power function IR receiver. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_channel</code>	The power function IR receiver channel. See the <a href="#">PFMate channel constants</a> group.
<code>_b1</code>	Raw byte 1.
<code>_b2</code>	Raw byte 2.
<code>_result</code>	The function call result.

5.23.2.147 #define PSPNxAnalog( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, PSP\_CMD\_ANALOG, *\_result*)

Configure PSP-Nx in analog mode.

Configure the mindsensors PSP-Nx device in analog mode. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.148 #define PSPNxDigital( *\_port*, *\_i2caddr*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, PSP\_CMD\_DIGITAL, *\_result*)

Configure PSP-Nx in digital mode.

Configure the mindsensors PSP-Nx device in digital mode. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_result</i>	The function call result.

5.23.2.149 #define ReadACCLNxSensitivity( *\_port*, *\_i2caddr*, *\_out*, *\_result* ) \_\_MSReadValue(*\_port*, *\_i2caddr*, ACCL\_REG\_sENS\_LVL, 1, *\_out*, *\_result*)

Read ACCL-Nx sensitivity value.

Read the mindsensors ACCL-Nx sensitivity value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The sensitivity value.
<i>_result</i>	The function call result.

5.23.2.150 #define ReadACCLNxXOffset( *\_port*, *\_i2caddr*, *\_out*, *\_result* ) \_\_MSReadValue(*\_port*, *\_i2caddr*, ACCL\_REG\_X\_OFFSET, 2, *\_out*, *\_result*)

Read ACCL-Nx X offset value.

Read the mindsensors ACCL-Nx sensor's X offset value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The X offset value.
<i>_result</i>	The function call result.

---

```
5.23.2.151 #define ReadACCLNxXRange( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,
ACCL_REG_X_RANGE, 2, _out, _result)
```

Read ACCL-Nx X range value.

Read the mindsensors ACCL-Nx sensor's X range value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The X range value.
<i>_result</i>	The function call result.

---

```
5.23.2.152 #define ReadACCLNxYOffset( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,
ACCL_REG_Y_OFFSET, 2, _out, _result)
```

Read ACCL-Nx Y offset value.

Read the mindsensors ACCL-Nx sensor's Y offset value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The Y offset value.
<i>_result</i>	The function call result.

---

```
5.23.2.153 #define ReadACCLNxYRange( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,
ACCL_REG_Y_RANGE, 2, _out, _result)
```

Read ACCL-Nx Y range value.

Read the mindsensors ACCL-Nx sensor's Y range value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The Y range value.
<i>_result</i>	The function call result.

---

```
5.23.2.154 #define ReadACCLNxZOffset( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,
ACCL_REG_Z_OFFSET, 2, _out, _result)
```

Read ACCL-Nx Z offset value.

Read the mindsensors ACCL-Nx sensor's Z offset value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.

<i>_out</i>	The Z offset value.
<i>_result</i>	The function call result.

5.23.2.155 #define ReadACCLNxZRange( *\_port*, *\_i2caddr*, *\_out*, *\_result* ) \_\_MSReadValue(*\_port*, *\_i2caddr*, ACCL\_REG\_Z\_RANGE, 2, *\_out*, *\_result*)

Read ACCL-Nx Z range value.

Read the mindsensors ACCL-Nx sensor's Z range value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The Z range value.
<i>_result</i>	The function call result.

5.23.2.156 #define ReadDISTNxDistance( *\_port*, *\_i2caddr*, *\_out*, *\_result* ) \_\_MSReadValue(*\_port*, *\_i2caddr*, DIST\_REG\_DIST, 2, *\_out*, *\_result*)

Read DIST-Nx distance value.

Read the mindsensors DIST-Nx sensor's distance value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The distance value.
<i>_result</i>	The function call result.

5.23.2.157 #define ReadDISTNxMaxDistance( *\_port*, *\_i2caddr*, *\_out*, *\_result* ) \_\_MSReadValue(*\_port*, *\_i2caddr*, DIST\_REG\_DIST\_MAX, 2, *\_out*, *\_result*)

Read DIST-Nx maximum distance value.

Read the mindsensors DIST-Nx sensor's maximum distance value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The maximum distance value.
<i>_result</i>	The function call result.

5.23.2.158 #define ReadDISTNxMinDistance( *\_port*, *\_i2caddr*, *\_out*, *\_result* ) \_\_MSReadValue(*\_port*, *\_i2caddr*, DIST\_REG\_DIST\_MIN, 2, *\_out*, *\_result*)

Read DIST-Nx minimum distance value.

Read the mindsensors DIST-Nx sensor's minimum distance value. The port must be configured as a Lowspeed port before using this function.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The minimum distance value.
<i>_result</i>	The function call result.

```
5.23.2.159 #define ReadDISTNxModuleType( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,  
DIST_REG_MODULE_TYPE, 1, _out, _result)
```

Read DIST-Nx module type value.

Read the mindsensors DIST-Nx sensor's module type value. The port must be configured as a Lowspeed port before using this function.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The module type value.
<i>_result</i>	The function call result.

```
5.23.2.160 #define ReadDISTNxNumPoints( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,  
DIST_REG_NUM_POINTS, 1, _out, _result)
```

Read DIST-Nx num points value.

Read the mindsensors DIST-Nx sensor's num points value. The port must be configured as a Lowspeed port before using this function.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The num points value.
<i>_result</i>	The function call result.

```
5.23.2.161 #define ReadDISTNxVoltage( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr, DIST_REG_VOLT, 2,  
_out, _result)
```

Read DIST-Nx voltage value.

Read the mindsensors DIST-Nx sensor's voltage value. The port must be configured as a Lowspeed port before using this function.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The voltage value.
<i>_result</i>	The function call result.

```
5.23.2.162 #define ReadNRLinkBytes( _port, _i2caddr, _bytes, _result ) __ReadNRLinkBytes(_port, _i2caddr, _bytes, _result)
```

Read data from NRLink.

Read data from the mindsensors NRLink device on the specified port. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_bytes</code>	A byte array that will contain the data read from the device on output.
<code>_result</code>	The function call result.

**5.23.2.163 #define ReadNRLinkStatus( `_port`, `_i2caddr`, `_value`, `_result` ) \_\_ReadNRLinkStatus(`_port`, `_i2caddr`, `_value`, `_result`)**

Read NRLink status.

Read the status of the mindsensors NRLink device. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_value</code>	The mindsensors NRLink status.
<code>_result</code>	The function call result.

**5.23.2.164 #define ReadNXTLineLeaderAverage( `_port`, `_i2caddr`, `_out`, `_result` ) \_\_MSReadValue(`_port`, `_i2caddr`, `NXTLL_REG_AVERAGE`, 1, `_out`, `_result`)**

Read NXTLineLeader average.

Read the mindsensors NXTLineLeader device's average value. The average is a weighted average of the bits set to 1 based on the position. The left most bit has a weight of 10, second bit has a weight of 20, and so forth. When all 8 sensors are over a black surface the average will be 45. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_out</code>	The NXTLineLeader average value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

**5.23.2.165 #define ReadNXTLineLeaderResult( `_port`, `_i2caddr`, `_out`, `_result` ) \_\_MSReadValue(`_port`, `_i2caddr`, `NXTLL_REG_RESULT`, 1, `_out`, `_result`)**

Read NXTLineLeader result.

Read the mindsensors NXTLineLeader device's result value. This is a single byte showing the 8 sensor's readings. Each bit corresponding to the sensor where the line is seen is set to 1, otherwise it is set to 0. When all 8 sensors are over a black surface the result will be 255 (b11111111). The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.

<i>_out</i>	The NXTLineLeader result value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.166 #define ReadNXTLineLeaderSteering( *\_port*, *\_i2caddr*, *\_out*, *\_result* ) \_\_MSReadValue(*\_port*, *\_i2caddr*, **NXTLL\_REG\_STEERING**, 1, *\_out*, *\_result*)

Read NXTLineLeader steering.

Read the mindsensors NXTLineLeader device's steering value. This is the power returned by the sensor to correct your course. Add this value to your left motor and subtract it from your right motor. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The NXTLineLeader steering value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.167 #define ReadNXTPowerMeterCapacityUsed( *\_port*, *\_i2caddr*, *\_out*, *\_result* ) \_\_MSReadValue(*\_port*, *\_i2caddr*, **NXTPM\_REG\_CAPACITY**, 2, *\_out*, *\_result*)

Read NXTPowerMeter capacity used.

Read the mindsensors NXTPowerMeter device's capacity used since the last reset command. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The NXTPowerMeter capacity used value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.168 #define ReadNXTPowerMeterElapsedTime( *\_port*, *\_i2caddr*, *\_out*, *\_result* ) \_\_MSReadValue(*\_port*, *\_i2caddr*, **NXTPM\_REG\_TIME**, 4, *\_out*, *\_result*)

Read NXTPowerMeter elapsed time.

Read the mindsensors NXTPowerMeter device's elapsed time since the last reset command. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The NXTPowerMeter elapsed time value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.169 #define ReadNXTPowerMeterErrorCount( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,  
NXTPM_REG_ERRORCOUNT, 2, _out, _result)
```

Read NXTPowerMeter error count.

Read the mindsensors NXTPowerMeter device's error count value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The NXTPowerMeter error count value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.170 #define ReadNXTPowerMeterMaxCurrent( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,  
NXTPM_REG_MAXCURRENT, 2, _out, _result)
```

Read NXTPowerMeter maximum current.

Read the mindsensors NXTPowerMeter device's maximum current value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The NXTPowerMeter maximum current value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.171 #define ReadNXTPowerMeterMaxVoltage( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,  
NXTPM_REG_MAXVOLTAGE, 2, _out, _result)
```

Read NXTPowerMeter maximum voltage.

Read the mindsensors NXTPowerMeter device's maximum voltage value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The NXTPowerMeter maximum voltage value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.172 #define ReadNXTPowerMeterMinCurrent( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,  
NXTPM_REG_MINCURRENT, 2, _out, _result)
```

Read NXTPowerMeter minimum current.

Read the mindsensors NXTPowerMeter device's minimum current value. The port must be configured as a Lowspeed port before using this function.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The NXTPowerMeter minimum current value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.173 #define ReadNXTPowerMeterMinVoltage( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,  
NXTPM_REG_MINVOLTAGE, 2, _out, _result)
```

Read NXTPowerMeter minimum voltage.

Read the mindsensors NXTPowerMeter device's minimum voltage value. The port must be configured as a Lowspeed port before using this function.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The NXTPowerMeter minimum voltage value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.174 #define ReadNXTPowerMeterPresentCurrent( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,  
NXTPM_REG_CURRENT, 2, _out, _result)
```

Read NXTPowerMeter present current.

Read the mindsensors NXTPowerMeter device's present current value. The port must be configured as a Lowspeed port before using this function.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The NXTPowerMeter present current.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.175 #define ReadNXTPowerMeterPresentPower( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,  
NXTPM_REG_POWER, 2, _out, _result)
```

Read NXTPowerMeter present power.

Read the mindsensors NXTPowerMeter device's present power value. The port must be configured as a Lowspeed port before using this function.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The NXTPowerMeter present power value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.176 #define ReadNXTPowerMeterPresentVoltage( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,
NXTPM_REG_VOLTAGE, 2, _out, _result)
```

Read NXTPowerMeter present voltage.

Read the mindsensors NXTPowerMeter device's present voltage value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The NXTPowerMeter present voltage.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.177 #define ReadNXTPowerMeterTotalPowerConsumed( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,
NXTPM_REG_POWER, 4, _out, _result)
```

Read NXTPowerMeter total power consumed.

Read the mindsensors NXTPowerMeter device's total power consumed since the last reset command. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The NXTPowerMeter total power consumed value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.178 #define ReadNXTServoBatteryVoltage( _port, _i2caddr, _out, _result ) __MSReadValue(_port, _i2caddr,
NXTSERVO_REG_VOLTAGE, 1, _out, _result)
```

Read NXTServo battery voltage value.

Read the mindsensors NXTServo device's battery voltage value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_out</i>	The NXTServo battery voltage.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.179 #define ReadNXTServoPosition( _port, _i2caddr, _servo, _out, _result ) __MSReadValue(_port, _i2caddr,
NXTSERVO_REG_S1_POS+(_servo*2), 2, _out, _result)
```

Read NXTServo servo position value.

Read the mindsensors NXTServo device's servo position value. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_servo</code>	The servo number. See <a href="#">MindSensors NXTServo servo numbers</a> group.
<code>_out</code>	The specified servo's position value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.180 #define ReadNXTServoSpeed( _port, _i2caddr, _servo, _out, _result ) __MSReadValue(_port, _i2caddr,  
NXTSERVO_REG_S1_SPEED+_servo, 1, _out, _result)
```

Read NXTServo servo speed value.

Read the mindsensors NXTServo device's servo speed value. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_servo</code>	The servo number. See <a href="#">MindSensors NXTServo servo numbers</a> group.
<code>_out</code>	The specified servo's speed value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.181 #define ReadSensorMSAccel( _port, _i2caddr, _x, _y, _z, _result ) __ReadSensorMSAccel(_port, _i2caddr, _x, _y, _z,  
_result)
```

Read mindsensors acceleration values.

Read X, Y, and Z axis acceleration values from the mindsensors Accelerometer sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_x</code>	The output x-axis acceleration.
<code>_y</code>	The output y-axis acceleration.
<code>_z</code>	The output z-axis acceleration.
<code>_result</code>	The function call result.

```
5.23.2.182 #define ReadSensorMSCompass( _port, _i2caddr, _value ) __ReadSensorMSCompass(_port, _i2caddr, _value)
```

Read mindsensors compass value.

Return the Mindsensors Compass sensor value.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_value</code>	The mindsensors compass value

5.23.2.183 #define ReadSensorMSDROD( \_port, \_value ) \_\_ReadSensorMSDROD(\_port, \_value)

Read mindsensors DROD value.

Return the Mindsensors DROD sensor value.

#### Parameters

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_value</u>	The mindsensors DROD value

5.23.2.184 #define ReadSensorMSPlayStation( \_port, \_i2caddr, \_b1, \_b2, \_xleft, \_yleft, \_xright, \_yright, \_result ) \_\_ReadSensorMSPlayStation(\_port, \_i2caddr, \_b1, \_b2, \_xleft, \_yleft, \_xright, \_yright, \_result)

Read mindsensors playstation controller values.

Read playstation controller values from the mindsensors playstation sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_i2caddr</u>	The sensor I2C address. See sensor documentation for this value.
<u>_b1</u>	The button set 1 values. See <a href="#">MindSensors PSP-Nx button set 1 constants</a> .
<u>_b2</u>	The button set 2 values. See <a href="#">MindSensors PSP-Nx button set 2 constants</a> .
<u>_xleft</u>	The left joystick x value.
<u>_yleft</u>	The left joystick y value.
<u>_xright</u>	The right joystick x value.
<u>_yright</u>	The right joystick y value.
<u>_result</u>	The function call result.

5.23.2.185 #define ReadSensorMSPressure( \_port, \_value ) \_\_ReadSensorMSPressure(\_port, \_value)

Read mindsensors processed pressure value.

Return the Mindsensors pressure sensor processed value.

#### Parameters

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_value</u>	The mindsensors processed pressure value

5.23.2.186 #define ReadSensorMSPressureRaw( \_port, \_value ) \_\_ReadSensorMSPressureRaw(\_port, \_value)

Read mindsensors raw pressure value.

Return the Mindsensors pressure sensor raw value.

#### Parameters

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_value</u>	The mindsensors raw pressure value

---

```
5.23.2.187 #define ReadSensorMSRTClock( _port, _sec, _min, _hrs, _dow, _date, _month, _year, _result
) __ReadSensorMSRTClock(_port, _sec, _min, _hrs, _dow, _date, _month, _year, _result)
```

Read mindsensors RTClock values.

Read real-time clock values from the Mindsensors RTClock sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_sec</i>	The seconds.
<i>_min</i>	The minutes.
<i>_hrs</i>	The hours.
<i>_dow</i>	The day of week number.
<i>_date</i>	The day.
<i>_month</i>	The month.
<i>_year</i>	The year.
<i>_result</i>	The function call result.

---

```
5.23.2.188 #define ReadSensorMSTilt( _port, _i2caddr, _x, _y, _z, _result ) __ReadSensorMSTilt(_port, _i2caddr, _x, _y, _z, _result)
```

Read mindsensors tilt values.

Read X, Y, and Z axis tilt values from the mindsensors tilt sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_x</i>	The output x-axis tilt.
<i>_y</i>	The output y-axis tilt.
<i>_z</i>	The output z-axis tilt.
<i>_result</i>	The function call result.

---

```
5.23.2.189 #define ReadSensorNXTSumoEyes( _port, _value ) __ReadSensorNXTSumoEyes(_port, _value)
```

Read mindsensors NXTSumoEyes value.

Return the Mindsensors NXTSumoEyes sensor value.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_value</i>	The mindsensors NXTSumoEyes value

---

```
5.23.2.190 #define RunNRLinkMacro( _port, _i2caddr, _macro, _result ) __RunNRLinkMacro(_port, _i2caddr, _macro, _result)
```

Run NRLink macro.

Run the specified mindsensors NRLink device macro. The port must be configured as a Lowspeed port before using this function.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_macro</i>	The address of the macro to execute.
<i>_result</i>	The function call result.

5.23.2.191 #define SetACCLNxSensitivity( *\_port*, *\_i2caddr*, *\_slevel*, *\_result* ) \_\_I2CSendCmd(*\_port*, *\_i2caddr*, *\_slevel*, *\_result*)

Set ACCL-Nx sensitivity.

Reset the mindsensors ACCL-Nx sensor calibration to factory settings. The port must be configured as a Lowspeed port before using this function.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_slevel</i>	The sensitivity level. See <a href="#">MindSensors ACCL-Nx sensitivity level constants</a> .
<i>_result</i>	The function call result.

5.23.2.192 #define SetNXTLineLeaderKdFactor( *\_port*, *\_i2caddr*, *\_value*, *\_result* ) \_\_MSWriteToRegister(*\_port*, *\_i2caddr*, **NXTLL\_REG\_KD\_FACTOR**, *\_value*, *\_result*)

Write NXTLineLeader Kd factor.

Write a Kd divisor factor to the NXTLineLeader device. Value ranges between 1 and 255. Change this value if you need more granularities in Kd value. The port must be configured as a Lowspeed port before using this function.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_value</i>	The new Kd factor.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

5.23.2.193 #define SetNXTLineLeaderKdValue( *\_port*, *\_i2caddr*, *\_value*, *\_result* ) \_\_MSWriteToRegister(*\_port*, *\_i2caddr*, **NXTLL\_REG\_KD\_VALUE**, *\_value*, *\_result*)

Write NXTLineLeader Kd value.

Write a Kd value to the NXTLineLeader device. This value divided by PID Factor for Kd is the Derivative value for the PID control. Suggested value is 8 with a divisor factor of 32 (which is also a factory default), start with this value, and tune it to meet your needs. Value ranges between 0 and 255. The port must be configured as a Lowspeed port before using this function.

## Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_value</i>	The new Kd value.
<i>_result</i>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.194 #define SetNXTLineLeaderKiFactor( _port, _i2caddr, _value, _result ) __MSWriteToRegister(_port, _i2caddr,
NXTLL_REG_KI_FACTOR, _value, _result)
```

Write NXTLineLeader Ki factor.

Write a Ki divisor factor to the NXTLineLeader device. Value ranges between 1 and 255. Change this value if you need more granularities in Ki value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_value</code>	The new Ki factor.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.195 #define SetNXTLineLeaderKiValue( _port, _i2caddr, _value, _result ) __MSWriteToRegister(_port, _i2caddr,
NXTLL_REG_KI_VALUE, _value, _result)
```

Write NXTLineLeader Ki value.

Write a Ki value to the NXTLineLeader device. This value divided by PID Factor for Ki is the Integral value for the PID control. Suggested value is 0 with a divisor factor of 32 (which is also a factory default), start with this value, and tune it to meet your needs. Value ranges between 0 and 255. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_value</code>	The new Ki value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.196 #define SetNXTLineLeaderKpFactor( _port, _i2caddr, _value, _result ) __MSWriteToRegister(_port, _i2caddr,
NXTLL_REG_KP_FACTOR, _value, _result)
```

Write NXTLineLeader Kp factor.

Write a Kp divisor factor to the NXTLineLeader device. Value ranges between 1 and 255. Change this value if you need more granularities in Kp value. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_value</code>	The new Kp factor.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.197 #define SetNXTLineLeaderKpValue( _port, _i2caddr, _value, _result ) __MSWriteToRegister(_port, _i2caddr,
NXTLL_REG_KP_VALUE, _value, _result)
```

Write NXTLineLeader Kp value.

Write a Kp value to the NXTLineLeader device. This value divided by PID Factor for Kp is the Proportional value for the

PID control. Suggested value is 25 with a divisor factor of 32 (which is also a factory default), start with this value, and tune it to meet your needs. Value ranges between 0 and 255. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_value</code>	The new Kp value.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.198 #define SetNXTLineLeaderSetpoint( _port, _i2caddr, _value, _result ) __MSWriteToRegister(_port, _i2caddr,  
NXTLL_REG_SETPOINT, _value, _result)
```

Write NXTLineLeader setpoint.

Write a new setpoint value to the NXTLineLeader device. The Set Point is a value you can ask sensor to maintain the average to. The default value is 45, whereby the line is maintained in center of the sensor. If you need to maintain line towards left of the sensor, set the Set Point to a lower value (minimum: 10). If you need it to be towards on the right of the sensor, set it to higher value (maximum: 80). Set point is also useful while tracking an edge of dark and light areas. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_value</code>	The new setpoint value (10..80).
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.199 #define SetNXTServoPosition( _port, _i2caddr, _servo, _pos, _result ) __MSWriteLEIntToRegister(_port, _i2caddr, _reg,  
_pos, _result)
```

Set NXTServo servo motor position.

Set the position of a servo motor controlled by the NXTServo device. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_servo</code>	The servo number. See <a href="#">MindSensors NXTServo servo numbers</a> group.
<code>_pos</code>	The servo position. See <a href="#">MindSensors NXTServo position constants</a> group.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.200 #define SetNXTServoQuickPosition( _port, _i2caddr, _servo, _qpos, _result ) __MSWriteToRegister(_port, _i2caddr,  
NXTSERVO_REG_S1_QPOS+_servo, _qpos, _result)
```

Set NXTServo servo motor quick position.

Set the quick position of a servo motor controlled by the NXTServo device. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_servo</code>	The servo number. See <a href="#">MindSensors NXTServo servo numbers</a> group.
<code>_qpos</code>	The servo quick position. See <a href="#">MindSensors NXTServo quick position constants</a> group.
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.201 #define SetNXTServoSpeed( _port, _i2caddr, _servo, _speed, _result ) __MSWriteToRegister(_port, _i2caddr, NXTSERVO_REG_S1_SPEED+_servo, _speed, _result)
```

Set NXTServo servo motor speed.

Set the speed of a servo motor controlled by the NXTServo device. The port must be configured as a Lowspeed port before using this function.

## Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_i2caddr</code>	The sensor I2C address. See sensor documentation for this value.
<code>_servo</code>	The servo number. See <a href="#">MindSensors NXTServo servo numbers</a> group.
<code>_speed</code>	The servo speed. (0..255)
<code>_result</code>	A status code indicating whether the operation completed successfully or not. See <a href="#">TCommLS-CheckStatus</a> for possible Result values.

```
5.23.2.202 #define SetSensorMSDRODActive( _port ) __SetSensorMSDRODActive(_port)
```

Configure a mindsensors DROD active sensor.

Configure the specified port for an active mindsensors DROD sensor.

## Parameters

<code>_port</code>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------------	---

```
5.23.2.203 #define SetSensorMSDRODInactive( _port ) __SetSensorMSDRODInactive(_port)
```

Configure a mindsensors DROD inactive sensor.

Configure the specified port for an inactive mindsensors DROD sensor.

## Parameters

<code>_port</code>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------------	---

```
5.23.2.204 #define SetSensorMSPressure( _port ) __SetSensorMSPressure(_port)
```

Configure a mindsensors pressure sensor.

Configure the specified port for a mindsensors pressure sensor.

## Parameters

<code>_port</code>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------------	---

**5.23.2.205 #define SetSensorMSTouchMux( \_port ) \_\_SetSensorMSTouchMux(\_port)**

Configure a mindsensors touch sensor multiplexer.

Configure the specified port for a mindsensors touch sensor multiplexer.

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**5.23.2.206 #define SetSensorNXTSumoEyesLong( \_port ) \_\_SetSensorNXTSumoEyesLong(\_port)**

Configure a mindsensors NXTSumoEyes long range sensor.

Configure the specified port for a long range mindsensors NXTSumoEyes sensor.

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**5.23.2.207 #define SetSensorNXTSumoEyesShort( \_port ) \_\_SetSensorNXTSumoEyesShort(\_port)**

Configure a mindsensors NXTSumoEyes short range sensor.

Configure the specified port for a short range mindsensors NXTSumoEyes sensor.

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**5.23.2.208 #define WriteNRLinkBytes( \_port, \_i2caddr, \_bytes, \_result ) \_\_WriteNRLinkBytes(\_port, \_i2caddr, \_bytes, \_result)**

Write data to NRLink.

Write data to the mindsensors NRLink device on the specified port. The port must be configured as a Lowspeed port before using this function.

**Parameters**

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_i2caddr</i>	The sensor I2C address. See sensor documentation for this value.
<i>_bytes</i>	A byte array containing the data to write.
<i>_result</i>	The function call result.

## 5.24 Codatex API Functions

Functions for accessing and modifying Codatex devices.

### Modules

- [Codatex device constants](#)

*Constants that are for use with Codatex devices.*

### Macros

- `#define RFIDInit(_port, _result) __RFIDInit(_port, _result)`  
*RFIDInit function.*
- `#define RFIDMode(_port, _mode, _result) __RFIDMode(_port, _mode, _result)`  
*RFIDMode function.*
- `#define RFIDStatus(_port, _result) __RFIDStatus(_port, _result)`  
*RFIDStatus function.*
- `#define RFIDRead(_port, _output, _result) __RFIDRead(_port, _output, _result)`  
*RFIDRead function.*
- `#define RFIDStop(_port, _result) __RFIDStop(_port, _result)`  
*RFIDStop function.*
- `#define RFIDReadSingle(_port, _output, _result) __RFIDReadSingle(_port, _output, _result)`  
*RFIDReadSingle function.*
- `#define RFIDReadContinuous(_port, _output, _result) __RFIDReadContinuous(_port, _output, _result)`  
*RFIDReadContinuous function.*

### 5.24.1 Detailed Description

Functions for accessing and modifying Codatex devices.

### 5.24.2 Macro Definition Documentation

#### 5.24.2.1 #define RFIDInit( \_port, \_result ) \_\_RFIDInit(\_port, \_result)

RFIDInit function.

Initialize the Codatex RFID sensor.

##### Parameters

<code>_port</code>	The port to which the Codatex RFID sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_result</code>	The boolean function call result.

#### 5.24.2.2 #define RFIDMode( \_port, \_mode, \_result ) \_\_RFIDMode(\_port, \_mode, \_result)

RFIDMode function.

Configure the Codatex RFID sensor mode.

**Parameters**

<i>_port</i>	The port to which the Codatex RFID sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_mode</i>	The RFID sensor mode. See the <a href="#">Codatex RFID sensor modes</a> group.
<i>_result</i>	The boolean function call result.

5.24.2.3 #define RFIDRead( *\_port*, *\_output*, *\_result* ) \_\_RFIDRead(*\_port*, *\_output*, *\_result*)

RFIDRead function.

Read the Codatex RFID sensor value.

**Parameters**

<i>_port</i>	The port to which the Codatex RFID sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_output</i>	The five bytes of RFID data.
<i>_result</i>	The boolean function call result.

5.24.2.4 #define RFIDReadContinuous( *\_port*, *\_output*, *\_result* ) \_\_RFIDReadContinuous(*\_port*, *\_output*, *\_result*)

RFIDReadContinuous function.

Set the Codatex RFID sensor into continuous mode, if necessary, and read the RFID data.

**Parameters**

<i>_port</i>	The port to which the Codatex RFID sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_output</i>	The five bytes of RFID data.
<i>_result</i>	The boolean function call result.

5.24.2.5 #define RFIDReadSingle( *\_port*, *\_output*, *\_result* ) \_\_RFIDReadSingle(*\_port*, *\_output*, *\_result*)

RFIDReadSingle function.

Set the Codatex RFID sensor into single mode and read the RFID data.

**Parameters**

<i>_port</i>	The port to which the Codatex RFID sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_output</i>	The five bytes of RFID data.
<i>_result</i>	The boolean function call result.

5.24.2.6 #define RFIDStatus( *\_port*, *\_result* ) \_\_RFIDStatus(*\_port*, *\_result*)

RFIDStatus function.

Read the Codatex RFID sensor status.

**Parameters**

<i>_port</i>	The port to which the Codatex RFID sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_result</i>	The RFID sensor status.

**5.24.2.7 #define RFIDStop( \_port, \_result ) \_\_RFIDStop(\_port, \_result)**

RFIDStop function.

Stop the Codatex RFID sensor.

**Parameters**

<i>_port</i>	The port to which the Codatex RFID sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_result</i>	The boolean function call result.

## 5.25 Dexter Industries API Functions

Functions for accessing and modifying Dexter Industries devices.

### Modules

- [Dexter Industries device constants](#)

*Constants that are for use with Dexter Industries devices.*

### Macros

- `#define ReadSensorDIGPSStatus(_port, _status) __ReadSensorDIGPSStatus(_port, _status)`  
*ReadSensorDIGPSStatus function.*
- `#define ReadSensorDIGPSTime(_port, _result) __ReadSensorDIGPSTime(_port, _result)`  
*ReadSensorDIGPSTime function.*
- `#define ReadSensorDIGPSLatitude(_port, _result) __ReadSensorDIGPSLatitude(_port, _result)`  
*ReadSensorDIGPSLatitude function.*
- `#define ReadSensorDIGPSLongitude(_port, _result) __ReadSensorDIGPSLongitude(_port, _result)`  
*ReadSensorDIGPSLongitude function.*
- `#define ReadSensorDIGPSVelocity(_port, _result) __ReadSensorDIGPSVelocity(_port, _result)`  
*ReadSensorDIGPSVelocity function.*
- `#define ReadSensorDIGPSHeading(_port, _result) __ReadSensorDIGPSHeading(_port, _result)`  
*ReadSensorDIGPSHeading function.*
- `#define ReadSensorDIGPSDistanceToWaypoint(_port, _result) __ReadSensorDIGPSDistanceToWaypoint(_port, _result)`  
*ReadSensorDIGPSDistanceToWaypoint function.*
- `#define ReadSensorDIGPSHeadingToWaypoint(_port, _result) __ReadSensorDIGPSHeadingToWaypoint(_port, _result)`  
*ReadSensorDIGPSHeadingToWaypoint function.*
- `#define ReadSensorDIGPSRelativeHeading(_port, _result) __ReadSensorDIGPSRelativeHeading(_port, _result)`  
*ReadSensorDIGPSRelativeHeading function.*
- `#define SetSensorDIGPSWaypoint(_port, _lat, _long, _result) __SetSensorDIGPSWaypoint(_port, _lat, _long, _result)`  
*SetSensorDIGPSWaypoint function.*
- `#define SetSensorDIGyroEx(_port, _scale, _odr, _bw, _result) __SetSensorDIGyro(_port, _scale, _odr, _bw, _result)`  
*SetSensorDIGyroEx function.*
- `#define SetSensorDIGyro(_port, _result) __SetSensorDIGyro(_port, DIGYRO_CTRL4_SCALE_500, DIGYRO_CTRL1_DATARATE_100, DIGYRO_CTRL1_BANDWIDTH_1, _result)`  
*SetSensorDIGyro function.*
- `#define ReadSensorDIGyroRaw(_port, _vector, _result) __ReadSensorDIGyroRaw(_port, _vector, _result)`  
*ReadSensorDIGyroRaw function.*
- `#define ReadSensorDIGyro(_port, _vector, _result) __ReadSensorDIGyro(_port, _vector, _result)`  
*ReadSensorDIGyro function.*
- `#define ReadSensorDIGyroTemperature(_port, _out, _result) __ReadSensorDIGyroTemperature(_port, _out, _result)`  
*ReadSensorDIGyroTemperature function.*

- #define `ReadSensorDIGyroStatus(_port, _out, _result)` \_\_`ReadSensorDIGyroStatus(_port, _out, _result)`  
*ReadSensorDIGyroStatus function.*
- #define `SetSensorDIAcclEx(_port, _mode, _result)` \_\_`SetSensorDIAccl(_port, _mode, _result)`  
*SetSensorDIAcclEx function.*
- #define `SetSensorDIAccl(_port, _result)` \_\_`SetSensorDIAccl(_port, DIACCL_MODE_GLVL2, _result)`  
*SetSensorDIAccl function.*
- #define `ReadSensorDIAcclRaw(_port, _vector, _result)` \_\_`ReadSensorDIAcclRaw(_port, DIACCL_REG_XLOW, _vector, _result)`  
*ReadSensorDIAcclRaw function.*
- #define `ReadSensorDIAccl(_port, _vector, _result)` \_\_`ReadSensorDIAccl(_port, _vector, _result)`  
*ReadSensorDIAccl function.*
- #define `ReadSensorDIAccl8Raw(_port, _vector, _result)` \_\_`ReadSensorDIAccl8Raw(_port, _vector, _result)`  
*ReadSensorDIAccl8Raw function.*
- #define `ReadSensorDIAccl8(_port, _vector, _result)` \_\_`ReadSensorDIAccl8(_port, _vector, _result)`  
*ReadSensorDIAccl8 function.*
- #define `ReadSensorDIAcclStatus(_port, _out, _result)` \_\_`ReadSensorDIAcclStatus(_port, _out, _result)`  
*ReadSensorDIAcclStatus function.*
- #define `ReadSensorDIAcclDrift(_port, _x, _y, _z, _result)` \_\_`ReadSensorDIAcclDrift(_port, _x, _y, _z, _result)`  
*ReadSensorDIAcclDrift function.*
- #define `SetSensorDIAcclDrift(_port, _x, _y, _z, _result)` \_\_`SetSensorDIAcclDrift(_port, _x, _y, _z, _result)`  
*SetSensorDIAcclDrift function.*

### 5.25.1 Detailed Description

Functions for accessing and modifying Dexter Industries devices.

### 5.25.2 Macro Definition Documentation

#### 5.25.2.1 #define `ReadSensorDIAccl( _port, _vector, _result )` \_\_`ReadSensorDIAccl(_port, _vector, _result)`

ReadSensorDIAccl function.

Read the scaled Dexter Industries IMU Accl X, Y, and Z axis 10-bit values.

##### Parameters

<code>_port</code>	The port to which the Dexter Industries IMU Accl sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_vector</code>	A variable of type TVector which will contain the scaled X, Y, anx Z values.
<code>_result</code>	The boolean function call result.

#### 5.25.2.2 #define `ReadSensorDIAccl8( _port, _vector, _result )` \_\_`ReadSensorDIAccl8(_port, _vector, _result)`

ReadSensorDIAccl8 function.

Read the scaled Dexter Industries IMU Accl X, Y, and Z axis 8-bit values.

##### Parameters

<code>_port</code>	The port to which the Dexter Industries IMU Accl sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
--------------------	--

<code>_vector</code>	A variable of type TVector which will contain the scaled X, Y, and Z values.
<code>_result</code>	The boolean function call result.

5.25.2.3 #define ReadSensorDIAccl8Raw( `_port`, `_vector`, `_result` ) \_\_ReadSensorDIAccl8Raw(`_port`, `_vector`, `_result`)

ReadSensorDIAccl8Raw function.

Read the raw Dexter Industries IMU Accl X, Y, and Z axis 8-bit values.

#### Parameters

<code>_port</code>	The port to which the Dexter Industries IMU Accl sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_vector</code>	A variable of type TVector which will contain the raw X, Y, and Z values.
<code>_result</code>	The boolean function call result.

5.25.2.4 #define ReadSensorDIAcclDrift( `_port`, `_x`, `_y`, `_z`, `_result` ) \_\_ReadSensorDIAcclDrift(`_port`, `_x`, `_y`, `_z`, `_result`)

ReadSensorDIAcclDrift function.

Read the Dexter Industries IMU Accl X, Y, and Z axis 10-bit drift values.

#### Parameters

<code>_port</code>	The port to which the Dexter Industries IMU Accl sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_x</code>	The X axis 10-bit drift value.
<code>_y</code>	The Y axis 10-bit drift value.
<code>_z</code>	The Z axis 10-bit drift value.
<code>_result</code>	The boolean function call result.

5.25.2.5 #define ReadSensorDIAcclRaw( `_port`, `_vector`, `_result` ) \_\_ReadSensorDIAcclRaw(`_port`, DIACCL\_REG\_XLOW, `_vector`, `_result`)

ReadSensorDIAcclRaw function.

Read the raw Dexter Industries IMU Accl X, Y, and Z axis 10-bit values.

#### Parameters

<code>_port</code>	The port to which the Dexter Industries IMU Accl sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_vector</code>	A variable of type TVector which will contain the raw X, Y, and Z values.
<code>_result</code>	The boolean function call result.

5.25.2.6 #define ReadSensorDIAcclStatus( `_port`, `_out`, `_result` ) \_\_ReadSensorDIAcclStatus(`_port`, `_out`, `_result`)

ReadSensorDIAcclStatus function.

Read the Dexter Industries IMU Accl status value.

#### Parameters

<code>_port</code>	The port to which the Dexter Industries IMU Accl sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
--------------------	--

<i>_out</i>	The output status value.
<i>_result</i>	The boolean function call result.

5.25.2.7 #define ReadSensorDIGPSDistanceToWaypoint( *\_port*, *\_result* ) \_\_ReadSensorDIGPSDistanceToWaypoint(*\_port*, *\_result*)

ReadSensorDIGPSDistanceToWaypoint function.

Read the distance remaining to reach the current waypoint in meters.

#### Parameters

<i>_port</i>	The port to which the Dexter Industries GPS sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_result</i>	The distance to the waypoint in meters

5.25.2.8 #define ReadSensorDIGPSHeading( *\_port*, *\_result* ) \_\_ReadSensorDIGPSHeading(*\_port*, *\_result*)

ReadSensorDIGPSHeading function.

Read the current heading in degrees.

#### Parameters

<i>_port</i>	The port to which the Dexter Industries GPS sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_result</i>	The current heading in degrees

5.25.2.9 #define ReadSensorDIGPSHeadingToWaypoint( *\_port*, *\_result* ) \_\_ReadSensorDIGPSHeadingToWaypoint(*\_port*, *\_result*)

ReadSensorDIGPSHeadingToWaypoint function.

Read the heading required to reach the current waypoint.

#### Parameters

<i>_port</i>	The port to which the Dexter Industries GPS sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_result</i>	The heading to the waypoint in degrees

5.25.2.10 #define ReadSensorDIGPSLatitude( *\_port*, *\_result* ) \_\_ReadSensorDIGPSLatitude(*\_port*, *\_result*)

ReadSensorDIGPSLatitude function.

Read the integer latitude reported by the GPS (ddddddddd; Positive = North; Negative = South).

#### Parameters

<i>_port</i>	The port to which the Dexter Industries GPS sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_result</i>	The integer latitude

5.25.2.11 #define ReadSensorDIGPSLongitude( *\_port*, *\_result* ) \_\_ReadSensorDIGPSLongitude(*\_port*, *\_result*)

ReadSensorDIGPSLongitude function.

Read the integer longitude reported by the GPS (ddddddddd; Positive = East; Negative = West).

#### Parameters

<code>_port</code>	The port to which the Dexter Industries GPS sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_result</code>	The integer longitude

5.25.2.12 #define `ReadSensorDIGPSRelativeHeading( _port, _result )` ...`ReadSensorDIGPSRelativeHeading(_port, _result)`

ReadSensorDIGPSRelativeHeading function.

Read the angle travelled since last request. Resets the request coordinates on the GPS sensor. Sends the angle of travel since the last call.

#### Parameters

<code>_port</code>	The port to which the Dexter Industries GPS sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_result</code>	The relative heading in degrees

5.25.2.13 #define `ReadSensorDIGPSStatus( _port, _status )` ...`ReadSensorDIGPSStatus(_port, _status)`

ReadSensorDIGPSStatus function.

Read the status of the GPS satellite link.

#### Parameters

<code>_port</code>	The port to which the Dexter Industries GPS sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_status</code>	The GPS status

5.25.2.14 #define `ReadSensorDIGPSTime( _port, _result )` ...`ReadSensorDIGPSTime(_port, _result)`

ReadSensorDIGPSTime function.

Read the current time reported by the GPS in UTC.

#### Parameters

<code>_port</code>	The port to which the Dexter Industries GPS sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_result</code>	The current time in UTC

5.25.2.15 #define `ReadSensorDIGPSVelocity( _port, _result )` ...`ReadSensorDIGPSVelocity(_port, _result)`

ReadSensorDIGPSVelocity function.

Read the current velocity in cm/s.

#### Parameters

<code>_port</code>	The port to which the Dexter Industries GPS sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_result</code>	The current velocity in cm/s

**5.25.2.16 #define ReadSensorDIGyro( \_port, \_vector, \_result ) \_\_ReadSensorDIGyro(\_port, \_vector, \_result)**

ReadSensorDIGyro function.

Read the scaled Dexter Industries IMU Gyro X, Y, and Z axis values.

**Parameters**

<i>_port</i>	The port to which the Dexter Industries IMU Gyro sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_vector</i>	A variable of type TVector which will contain the scaled X, Y, and Z values.
<i>_result</i>	The boolean function call result.

**5.25.2.17 #define ReadSensorDIGyroRaw( \_port, \_vector, \_result ) \_\_ReadSensorDIGyroRaw(\_port, \_vector, \_result)**

ReadSensorDIGyroRaw function.

Read the raw Dexter Industries IMU Gyro X, Y, and Z axis values.

**Parameters**

<i>_port</i>	The port to which the Dexter Industries IMU Gyro sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_vector</i>	A variable of type TVector which will contain the raw X, Y, and Z values.
<i>_result</i>	The boolean function call result.

**5.25.2.18 #define ReadSensorDIGyroStatus( \_port, \_out, \_result ) \_\_ReadSensorDIGyroStatus(\_port, \_out, \_result)**

ReadSensorDIGyroStatus function.

Read the Dexter Industries IMU Gyro status value.

**Parameters**

<i>_port</i>	The port to which the Dexter Industries IMU Gyro sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_out</i>	The output status value.
<i>_result</i>	The boolean function call result.

**5.25.2.19 #define ReadSensorDIGyroTemperature( \_port, \_out, \_result ) \_\_ReadSensorDIGyroTemperature(\_port, \_out, \_result)**

ReadSensorDIGyroTemperature function.

Read the Dexter Industries IMU Gyro temperature value.

**Parameters**

<i>_port</i>	The port to which the Dexter Industries IMU Gyro sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_out</i>	The output temperature value.
<i>_result</i>	The boolean function call result.

**5.25.2.20 #define SetSensorDIACcl( \_port, \_result ) \_\_SetSensorDIACcl(\_port, DIACCL\_MODE\_GLVL2, \_result)**

SetSensorDIACcl function.

Configure DIAccl device on the specified port with default mode of 2G.

#### Parameters

<code>_port</code>	The port to which the Dexter Industries IMU Accl sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_result</code>	The boolean function call result.

**5.25.2.21 #define SetSensorDIAcclDrift( `_port`, `_x`, `_y`, `_z`, `_result` ) \_\_SetSensorDIAcclDrift(`_port`, `_x`, `_y`, `_z`, `_result`)**

SetSensorDIAcclDrift function.

Set the Dexter Industries IMU Accl X, Y, and Z axis 10-bit drift values.

#### Parameters

<code>_port</code>	The port to which the Dexter Industries IMU Accl sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_x</code>	The X axis 10-bit drift value.
<code>_y</code>	The Y axis 10-bit drift value.
<code>_z</code>	The Z axis 10-bit drift value.
<code>_result</code>	The boolean function call result.

**5.25.2.22 #define SetSensorDIAcclEx( `_port`, `_mode`, `_result` ) \_\_SetSensorDIAccl(`_port`, `_mode`, `_result`)**

SetSensorDIAcclEx function.

Configure DIAccl device on the specified port with the specified mode.

#### Parameters

<code>_port</code>	The port to which the Dexter Industries IMU Accl sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_mode</code>	The mode of the device (2G, 4G, or 8G). See the <a href="#">Dexter Industries IMU Accelerometer mode control register constants</a> group. You may use a constant or a variable.
<code>_result</code>	The boolean function call result.

**5.25.2.23 #define SetSensorDIGPSWaypoint( `_port`, `_lat`, `_long`, `_result` ) \_\_SetSensorDIGPSWaypoint(`_port`, `_lat`, `_long`, `_result`)**

SetSensorDIGPSWaypoint function.

Set the coordinates of the waypoint destination. The GPS sensor uses this to calculate the heading and distance required to reach the waypoint.

#### Parameters

<code>_port</code>	The port to which the Dexter Industries GPS sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_lat</code>	The latitude of the waypoint.
<code>_long</code>	The longitude of the waypoint.
<code>_result</code>	The boolean function call result.

```
5.25.2.24 #define SetSensorDIGyro( _port, _result ) __SetSensorDIGyro(_port, DIGYRO_CTRL4_SCALE_500,  
                           DIGYRO_CTRL1_DATARATE_100, DIGYRO_CTRL1_BANDWIDTH_1, _result)
```

SetSensorDIGyro function.

Configure DIGyro device on the specified port with default scale of 500dps, output data rate of 100hz, and bandwidth level 1.

Parameters

<i>_port</i>	The port to which the Dexter Industries IMU Gyro sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_result</i>	The boolean function call result.

```
5.25.2.25 #define SetSensorDIGyroEx( _port, _scale, _odr, _bw, _result ) __SetSensorDIGyro(_port, _scale, _odr, _bw, _result)
```

SetSensorDIGyroEx function.

Configure DIGyro device on the specified port with the specified scale, output data rate, and bandwidth.

Parameters

<i>_port</i>	The port to which the Dexter Industries IMU Gyro sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_scale</i>	The full scale of the device (250dps, 500dps, or 2000dps). See the <a href="#">Dexter Industries IMU Gyro control register 4 constants</a> group. You may use a constant or a variable.
<i>_odr</i>	The output data rate of the device (100hz, 200hz, 400hz, or 800hz). See the <a href="#">Dexter Industries IMU Gyro control register 1 constants</a> group. You may use a constant or a variable.
<i>_bw</i>	The bandwidth of the device. See the <a href="#">Dexter Industries IMU Gyro control register 1 constants</a> group. You may use a constant or a variable.
<i>_result</i>	The boolean function call result.

## 5.26 Microinfinity API Functions

Functions for accessing and modifying Microinfinity devices.

### Modules

- [Microinfinity device constants](#)

*Constants that are for use with Microinfinity devices.*

### Macros

- `#define ResetMIXG1300L(_port, _result) __ResetMIXG1300L(_port, _result)`  
*ResetMIXG1300L function.*
- `#define ReadSensorMIXG1300LScale(_port, _result) __ReadSensorMIXG1300LScale(_port, _result)`  
*ReadSensorMIXG1300LScale function.*
- `#define SetSensorMIXG1300LScale(_port, _scale, _result) __SetSensorMIXG1300LScale(_port, _scale, _result)`  
*SetSensorMIXG1300LScale function.*
- `#define ReadSensorMIXG1300L(_port, _packet, _result)`  
*ReadSensorMIXG1300L function.*

### 5.26.1 Detailed Description

Functions for accessing and modifying Microinfinity devices.

### 5.26.2 Macro Definition Documentation

#### 5.26.2.1 #define ReadSensorMIXG1300L( \_port, \_packet, \_result )

##### Value:

```
compchktype _packet, TXGPacket \
__ReadSensorMIXG1300L(_port, _packet, _result)
```

ReadSensorMIXG1300L function.

Read Microinfinity CruizCore XG1300L values. Read accumulated angle, turn rate, and X, Y, and Z axis acceleration values from the Microinfinity CruizCore XG1300L sensor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

##### Parameters

<code>_port</code>	The sensor port. See <a href="#">NBC Input port constants</a> .
<code>_packet</code>	The output XK1300L data structure. See <a href="#">TXGPacket</a> .
<code>_result</code>	The function call result.

#### 5.26.2.2 #define ReadSensorMIXG1300LScale( \_port, \_result ) \_\_ReadSensorMIXG1300LScale(\_port, \_result)

ReadSensorMIXG1300LScale function.

Read the Microinfinity CruizCore XG1300L accelerometer scale. The accelerometer in the CruizCore XG1300L can

be set to operate with a scale ranging from +/-2G, +/-4G, or +/-8G. Returns the scale value that the device is currently configured to use. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_result</i>	The current scale value.

#### 5.26.2.3 #define ResetMIXG1300L( *\_port*, *\_result* ) \_\_ResetMIXG1300L(*\_port*, *\_result*)

ResetMIXG1300L function.

Reset the Microinfinity CruizCore XG1300L device.

During reset, the XG1300L will recomputed the bias drift value, therefore it must remain stationary. The bias drift value will change randomly over time due to temperature variations, however the internal algorithm in the XG1300L will compensate for these changes. We strongly recommend issuing a reset command to the XG1300L at the beginning of the program.

The reset function also resets the accumulate angle value to a zero. Since the accelerometers measurements are taken with respect to the sensor reference frame the reset function will have no effect in the accelerometer measurements.

Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_result</i>	The function call result.

#### 5.26.2.4 #define SetSensorMIXG1300LScale( *\_port*, *\_scale*, *\_result* ) \_\_SetSensorMIXG1300LScale(*\_port*, *\_scale*, *\_result*)

SetSensorMIXG1300LScale function.

Set the Microinfinity CruizCore XG1300L accelerometer scale factor. Returns a boolean value indicating whether or not the operation completed successfully. The port must be configured as a Lowspeed port before using this function.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_scale</i>	This value must be a constant. See <a href="#">Microinfinity CruizCore XG1300L</a> .
<i>_result</i>	The function call result.

## 5.27 RIC Macro Wrappers

Macro wrappers for use in defining RIC byte arrays.

### Macros

- `#define RICImgPoint(_X, _Y) (_X)&0xFF, (_X)>>8, (_Y)&0xFF, (_Y)>>8`  
*Output an RIC ImgPoint structure.*
- `#define RICImgRect(_Pt, _W, _H) _Pt, (_W)&0xFF, (_W)>>8, (_H)&0xFF, (_H)>>8`  
*Output an RIC ImgRect structure.*
- `#define RICOpDescription(_Options, _Width, _Height) 8, 0, 0, 0, (_Options)&0xFF, (_Options)>>8, (_Width)&0xFF, (_Width)>>8, (_Height)&0xFF, (_Height)>>8`  
*Output an RIC Description opcode.*
- `#define RICOpCopyBits(_CopyOptions, _DataAddr, _SrcRect, _DstPoint) 18, 0, 3, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, (_DataAddr)&0xFF, (_DataAddr)>>8, _SrcRect, _DstPoint`  
*Output an RIC CopyBits opcode.*
- `#define RICOpPixel(_CopyOptions, _Point, _Value) 10, 0, 4, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point, (_Value)&0xFF, (_Value)>>8`  
*Output an RIC Pixel opcode.*
- `#define RICOpLine(_CopyOptions, _Point1, _Point2) 12, 0, 5, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point1, _Point2`  
*Output an RIC Line opcode.*
- `#define RICOpRect(_CopyOptions, _Point, _Width, _Height) 12, 0, 6, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point, (_Width)&0xFF, (_Width)>>8, (_Height)&0xFF, (_Height)>>8`  
*Output an RIC Rect opcode.*
- `#define RICOpCircle(_CopyOptions, _Point, _Radius) 10, 0, 7, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point, (_Radius)&0xFF, (_Radius)>>8`  
*Output an RIC Circle opcode.*
- `#define RICOpNumBox(_CopyOptions, _Point, _Value) 10, 0, 8, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point, (_Value)&0xFF, (_Value)>>8`  
*Output an RIC NumBox opcode.*
- `#define RICOpSprite(_DataAddr, _Rows, _BytesPerRow, _SpriteData) ((_Rows*_BytesPerRow)+(((_Rows*_BytesPerRow)%2)+8)&0xFF, ((_Rows*_BytesPerRow)+(((_Rows*_BytesPerRow)%2)+8)>>8, 1, 0, (_DataAddr)&0xFF, (_DataAddr)>>8, (_Rows)&0xFF, (_Rows)>>8, (_BytesPerRow)&0xFF, (_BytesPerRow)>>8, _SpriteData)`  
*Output an RIC Sprite opcode.*
- `#define RICSpriteData(...)` \_\_VA\_ARGS\_\_  
*Output RIC sprite data.*
- `#define RICOpVarMap(_DataAddr, _MapCount, _MapFunction) ((_MapCount*4)+6)&0xFF, ((_MapCount*4)+6)>>8, 2, 0, (_DataAddr)&0xFF, (_DataAddr)>>8, (_MapCount)&0xFF, (_MapCount)>>8, _MapFunction`  
*Output an RIC VarMap opcode.*
- `#define RICMapElement(_Domain, _Range) (_Domain)&0xFF, (_Domain)>>8, (_Range)&0xFF, (_Range)>>8`  
*Output an RIC map element.*
- `#define RICMapFunction(_MapElement,...)` \_MapElement, \_\_VA\_ARGS\_\_  
*Output an RIC VarMap function.*
- `#define RICArg(_arg) ((_arg)|0x1000)`  
*Output an RIC parameterized argument.*
- `#define RICMapArg(_mapidx, _arg) ((_arg)|0x1000|((((_mapidx)&0xF)<<8))`

*Output an RIC parameterized and mapped argument.*

- `#define RICOpPolygon(_CopyOptions, _Count, _ThePoints) ((_Count*4)+6)&0xFF, ((_Count*4)+6)>>8, 10, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, (_Count)&0xFF, (_Count)>>8, _ThePoints`

*Output an RIC Polygon opcode.*

- `#define RICPolygonPoints(_pPoint1, _pPoint2,...) _pPoint1, _pPoint2, __VA_ARGS__`

*Output RIC polygon points.*

- `#define RICOpEllipse(_CopyOptions, _Point, _RadiusX, _RadiusY) 12, 0, 9, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point, (_RadiusX)&0xFF, (_RadiusX)>>8, (_RadiusY)&0xFF, (_RadiusY)>>8`

*Output an RIC Ellipse opcode.*

### 5.27.1 Detailed Description

Macro wrappers for use in defining RIC byte arrays.

### 5.27.2 Macro Definition Documentation

#### 5.27.2.1 #define RICArg( *\_arg* ) ((*\_arg*)|0x1000)

Output an RIC parameterized argument.

##### Parameters

<i>_arg</i>	The argument that you want to parameterize.
-------------	---

#### 5.27.2.2 #define RICImgPoint( *\_X*, *\_Y* ) (*\_X*)&0xFF, (*\_X*)>>8, (*\_Y*)&0xFF, (*\_Y*)>>8

Output an RIC ImgPoint structure.

##### Parameters

<i>_X</i>	The X coordinate.
<i>_Y</i>	The Y coordinate.

#### 5.27.2.3 #define RICImgRect( *\_Pt*, *\_W*, *\_H* ) *\_Pt*, (*\_W*)&0xFF, (*\_W*)>>8, (*\_H*)&0xFF, (*\_H*)>>8

Output an RIC ImgRect structure.

##### Parameters

<i>_Pt</i>	An ImgPoint. See <a href="#">RICImgPoint</a> .
<i>_W</i>	The rectangle width.
<i>_H</i>	The rectangle height.

#### 5.27.2.4 #define RICMapArg( *\_mapidx*, *\_arg* ) ((*\_arg*)|0x1000|((*\_mapidx*)&0xF)<<8))

Output an RIC parameterized and mapped argument.

##### Parameters

<i>_mapidx</i>	The varmap data address.
<i>_arg</i>	The parameterized argument you want to pass through a varmap.

5.27.2.5 #define RICMapElement( *\_Domain*, *\_Range* ) (*\_Domain\_Domain*)>>8, (*\_Range*)&0xFF, (*\_Range*)>>8

Output an RIC map element.

#### Parameters

<i>_Domain</i>	The map element domain.
<i>_Range</i>	The map element range.

5.27.2.6 #define RICMapFunction( *\_MapElement*, ... ) *\_MapElement*, \_\_VA\_ARGS\_\_

Output an RIC VarMap function.

#### Parameters

<i>_MapElement</i>	An entry in the varmap function. At least 2 elements are required. See <a href="#">RICMapElement</a> .
--------------------	--

5.27.2.7 #define RICOpCircle( *\_CopyOptions*, *\_Point*, *\_Radius* ) 10, 0, 7, 0, (*\_CopyOptions*)&0xFF, (*\_CopyOptions*)>>8, *\_Point*, (*\_Radius*)&0xFF, (*\_Radius*)>>8

Output an RIC Circle opcode.

#### Parameters

<i>_CopyOptions</i>	Circle copy options. See <a href="#">Drawing option constants</a> .
<i>_Point</i>	The circle's center point. See <a href="#">RICImgPoint</a> .
<i>_Radius</i>	The circle's radius.

5.27.2.8 #define RICOpCopyBits( *\_CopyOptions*, *\_DataAddr*, *\_SrcRect*, *\_DstPoint* ) 18, 0, 3, 0, (*\_CopyOptions*)&0xFF, (*\_CopyOptions*)>>8, (*\_DataAddr*)&0xFF, (*\_DataAddr*)>>8, *\_SrcRect*, *\_DstPoint*

Output an RIC CopyBits opcode.

#### Parameters

<i>_CopyOptions</i>	CopyBits copy options. See <a href="#">Drawing option constants</a> .
<i>_DataAddr</i>	The address of the sprite from which to copy data.
<i>_SrcRect</i>	The rectangular portion of the sprite to copy. See <a href="#">RICImgRect</a> .
<i>_DstPoint</i>	The LCD coordinate to which to copy the data. See <a href="#">RICImgPoint</a> .

5.27.2.9 #define RICOpDescription( *\_Options*, *\_Width*, *\_Height* ) 8, 0, 0, 0, (*\_Options*)&0xFF, (*\_Options*)>>8, (*\_Width*)&0xFF, (*\_Width*)>>8, (*\_Height*)&0xFF, (*\_Height*)>>8

Output an RIC Description opcode.

#### Parameters

<i>_Options</i>	RIC options.
<i>_Width</i>	The total RIC width.
<i>_Height</i>	The total RIC height.

```
5.27.2.10 #define RICOpEllipse( _CopyOptions, _Point, _RadiusX, _RadiusY ) 12, 0, 9, 0, (_CopyOptions)&0xFF,
            (_CopyOptions)>>8, _Point, (_RadiusX)&0xFF, (_RadiusX)>>8, (_RadiusY)&0xFF, (_RadiusY)>>8
```

Output an RIC Ellipse opcode.

#### Parameters

<i>_CopyOptions</i>	Ellipse copy options. See <a href="#">Drawing option constants</a> .
<i>_Point</i>	The center of the ellipse. See <a href="#">RICImgPoint</a> .
<i>_RadiusX</i>	The x-axis radius of the ellipse.
<i>_RadiusY</i>	The y-axis radius of the ellipse.

```
5.27.2.11 #define RICOpLine( _CopyOptions, _Point1, _Point2 ) 12, 0, 5, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point1,
            _Point2
```

Output an RIC Line opcode.

#### Parameters

<i>_CopyOptions</i>	Line copy options. See <a href="#">Drawing option constants</a> .
<i>_Point1</i>	The starting point of the line. See <a href="#">RICImgPoint</a> .
<i>_Point2</i>	The ending point of the line. See <a href="#">RICImgPoint</a> .

```
5.27.2.12 #define RICOpNumBox( _CopyOptions, _Point, _Value ) 10, 0, 8, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point,
            (_Value)&0xFF, (_Value)>>8
```

Output an RIC NumBox opcode.

#### Parameters

<i>_CopyOptions</i>	NumBox copy options. See <a href="#">Drawing option constants</a> .
<i>_Point</i>	The numbox bottom left corner. See <a href="#">RICImgPoint</a> .
<i>_Value</i>	The number to draw.

```
5.27.2.13 #define RICOpPixel( _CopyOptions, _Point, _Value ) 10, 0, 4, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point,
            (_Value)&0xFF, (_Value)>>8
```

Output an RIC Pixel opcode.

#### Parameters

<i>_CopyOptions</i>	Pixel copy options. See <a href="#">Drawing option constants</a> .
<i>_Point</i>	The pixel coordinate. See <a href="#">RICImgPoint</a> .
<i>_Value</i>	The pixel value (unused).

```
5.27.2.14 #define RICOpPolygon( _CopyOptions, _Count, _ThePoints ) ((_Count*4)+6)&0xFF, (((_Count*4)+6)>>8, 10, 0,
            (_CopyOptions)&0xFF, (_CopyOptions)>>8, (_Count)&0xFF, (_Count)>>8, _ThePoints
```

Output an RIC Polygon opcode.

#### Parameters

<i>_CopyOptions</i>	Polygon copy options. See <a href="#">Drawing option constants</a> .
<i>_Count</i>	The number of points in the polygon.
<i>_ThePoints</i>	The list of polygon points. See <a href="#">RICPolygonPoints</a> .

```
5.27.2.15 #define RICOpRect( _CopyOptions, _Point, _Width, _Height ) 12, 0, 6, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8,  
_Point, (_Width)&0xFF, (_Width)>>8, (_Height)&0xFF, (_Height)>>8
```

Output an RIC Rect opcode.

#### Parameters

<i>_CopyOptions</i>	Rect copy options. See <a href="#">Drawing option constants</a> .
<i>_Point</i>	The rectangle's top left corner. See <a href="#">RICImgPoint</a> .
<i>_Width</i>	The rectangle's width.
<i>_Height</i>	The rectangle's height.

```
5.27.2.16 #define RICOpSprite( _DataAddr, _Rows, _BytesPerPage, _SpriteData ) ((_Rows*_BytesPerPage)+(((_Rows*  
_BytesPerPage)%2)+8)&0xFF, ((_Rows*_BytesPerPage)+(((_Rows*_BytesPerPage)%2)+8))>>8, 1, 0, (_DataAddr)&0xFF,  
(_DataAddr)>>8, (_Rows)&0xFF, (_Rows)>>8, (_BytesPerPage)&0xFF, (_BytesPerPage)>>8, _SpriteData
```

Output an RIC Sprite opcode.

#### Parameters

<i>_DataAddr</i>	The address of the sprite.
<i>_Rows</i>	The number of rows of data.
<i>_BytesPerPage</i>	The number of bytes per row.
<i>_SpriteData</i>	The actual sprite data. See <a href="#">RICSpriteData</a> .

```
5.27.2.17 #define RICOpVarMap( _DataAddr, _MapCount, _MapFunction ) ((_MapCount*4)+6)&0xFF, ((_MapCount*4)+6)>>8, 2, 0,  
(_DataAddr)&0xFF, (_DataAddr)>>8, (_MapCount)&0xFF, (_MapCount)>>8, _MapFunction
```

Output an RIC VarMap opcode.

#### Parameters

<i>_DataAddr</i>	The address of the varmap.
<i>_MapCount</i>	The number of points in the function.
<i>_MapFunction</i>	The definition of the varmap function. See <a href="#">RICMapFunction</a> .

```
5.27.2.18 #define RICPolygonPoints( _pPoint1, _pPoint2, ... ) _pPoint1, _pPoint2, __VA_ARGS__
```

Output RIC polygon points.

#### Parameters

<i>_pPoint1</i>	The first polygon point. See <a href="#">RICImgPoint</a> .
<i>_pPoint2</i>	The second polygon point (at least 3 points are required). See <a href="#">RICImgPoint</a> .

```
5.27.2.19 #define RICSpriteData( ... ) __VA_ARGS__
```

Output RIC sprite data.

## 5.28 NXT firmware module names

Constant string names for all the NXT firmware modules.

### Macros

- `#define CommandModuleName "Command.mod"`
- `#define IOCtrlModuleName "IOCtrl.mod"`
- `#define LoaderModuleName "Loader.mod"`
- `#define SoundModuleName "Sound.mod"`
- `#define ButtonModuleName "Button.mod"`
- `#define UIModuleName "Ui.mod"`
- `#define InputModuleName "Input.mod"`
- `#define OutputModuleName "Output.mod"`
- `#define LowSpeedModuleName "Low Speed.mod"`
- `#define DisplayModuleName "Display.mod"`
- `#define CommModuleName "Comm.mod"`

### 5.28.1 Detailed Description

Constant string names for all the NXT firmware modules.

### 5.28.2 Macro Definition Documentation

#### 5.28.2.1 `#define ButtonModuleName "Button.mod"`

The button module name

#### 5.28.2.2 `#define CommandModuleName "Command.mod"`

The command module name

#### 5.28.2.3 `#define CommModuleName "Comm.mod"`

The Comm module name

#### 5.28.2.4 `#define DisplayModuleName "Display.mod"`

The display module name

#### 5.28.2.5 `#define InputModuleName "Input.mod"`

The input module name.

#### 5.28.2.6 `#define IOCtrlModuleName "IOCtrl.mod"`

The IOCtrl module name

#### 5.28.2.7 `#define LoaderModuleName "Loader.mod"`

The Loader module name

5.28.2.8 #define LowSpeedModuleName "Low Speed.mod"

The low speed module name

5.28.2.9 #define OutputModuleName "Output.mod"

The output module name

5.28.2.10 #define SoundModuleName "Sound.mod"

The sound module name

5.28.2.11 #define UIModuleName "Ui.mod"

The Ui module name

## 5.29 NXT firmware module IDs

Constant numeric IDs for all the NXT firmware modules.

### Macros

- #define CommandModuleID 0x00010001
- #define IOCtrlModuleID 0x00060001
- #define LoaderModuleID 0x00090001
- #define SoundModuleID 0x00080001
- #define ButtonModuleID 0x00040001
- #define UIModuleID 0x000C0001
- #defineInputModuleID 0x00030001
- #define OutputModuleID 0x00020001
- #define LowSpeedModuleID 0x000B0001
- #define DisplayModuleID 0x000A0001
- #define CommModuleID 0x00050001

### 5.29.1 Detailed Description

Constant numeric IDs for all the NXT firmware modules.

### 5.29.2 Macro Definition Documentation

#### 5.29.2.1 #define ButtonModuleID 0x00040001

The button module ID

#### 5.29.2.2 #define CommandModuleID 0x00010001

The command module ID

#### 5.29.2.3 #define CommModuleID 0x00050001

The Comm module ID

#### 5.29.2.4 #define DisplayModuleID 0x000A0001

The display module ID

#### 5.29.2.5 #defineInputModuleID 0x00030001

The input module ID

#### 5.29.2.6 #define IOCtrlModuleID 0x00060001

The IOCtrl module ID

#### 5.29.2.7 #define LoaderModuleID 0x00090001

The Loader module ID

5.29.2.8 #define LowSpeedModuleID 0x000B0001

The low speed module ID

5.29.2.9 #define OutputModuleID 0x00020001

The output module ID

5.29.2.10 #define SoundModuleID 0x00080001

The sound module ID

5.29.2.11 #define UIModuleID 0x000C0001

The UI module ID

## 5.30 Miscellaneous NBC/NXC constants

Miscellaneous constants for use in NBC and NXC.

### Modules

- [Data type limits](#)

*Constants that define various data type limits.*

- [Property constants](#)

*Use these constants for specifying the property for the GetProperty and SetProperty direct commands.*

- [Variable type constants](#)

*Use these constants for testing a variable type.*

### Macros

- `#define TRUE 1`
- `#define FALSE 0`
- `#define NA 0xFFFF`
- `#define PI 3.141593`
- `#define RADIANS_PER_DEGREE PI/180`
- `#define DEGREES_PER_RADIAN 180/PI`

#### 5.30.1 Detailed Description

Miscellaneous constants for use in NBC and NXC.

#### 5.30.2 Macro Definition Documentation

##### 5.30.2.1 `#define DEGREES_PER_RADIAN 180/PI`

Used for converting from radians to degrees

##### 5.30.2.2 `#define FALSE 0`

A false value

##### 5.30.2.3 `#define NA 0xFFFF`

The specified argument does not apply (aka unwired)

##### 5.30.2.4 `#define PI 3.141593`

A constant for PI

##### 5.30.2.5 `#define RADIANS_PER_DEGREE PI/180`

Used for converting from degrees to radians

##### 5.30.2.6 `#define TRUE 1`

A true value

## 5.31 Third-party NXT devices

Documentation for NXT devices made by companies other than LEGO such as HiTechnic, mindsensors.com, and Coda-Tex.

### Modules

- [Codatex API Functions](#)

*Functions for accessing and modifying Codatex devices.*

- [Dexter Industries API Functions](#)

*Functions for accessing and modifying Dexter Industries devices.*

- [HiTechnic API Functions](#)

*Functions for accessing and modifying HiTechnic devices.*

- [HiTechnic/mindsensors Power Function/IR Train constants](#)

*Constants that are for use with the HiTechnic IRLink or mindsensors nRLink in Power Function or IR Train mode.*

- [Microinfinity API Functions](#)

*Functions for accessing and modifying Microinfinity devices.*

- [MindSensors API Functions](#)

*Functions for accessing and modifying MindSensors devices.*

- [RCX constants](#)

*Constants that are for use with devices that communicate with the RCX or Scout programmable bricks via IR such as the HiTechnic IRLink or the MindSensors nRLink.*

### 5.31.1 Detailed Description

Documentation for NXT devices made by companies other than LEGO such as HiTechnic, mindsensors.com, and Coda-Tex.

## 5.32 Standard-C API functions

Documentation for various Standard-C library routines.

### Modules

- [cmath API](#)  
*Standard C cmath API functions.*
- [cstdlib API](#)  
*Standard C cstdlib API functions.*

#### 5.32.1 Detailed Description

Documentation for various Standard-C library routines.

## 5.33 A simple 3D graphics library

Documentation for a simple 3D graphics library.

### Modules

- [Graphics library actions](#)  
*Constants that are used to specify a graphics library action.*
- [Graphics library begin modes](#)  
*Constants that are used to specify the polygon surface begin mode.*
- [Graphics library cull mode](#)  
*Constants to use when setting the graphics library cull mode.*
- [Graphics library settings](#)  
*Constants that are used to configure the graphics library settings.*

### Macros

- `#define glInit() __glInit()`  
*Initialize graphics library.*
- `#define glSet(_glType, _glValue) __glSet(_glType, _glValue)`  
*Set graphics library options.*
- `#define glBeginObject(_glObjId) __glBeginObject(_glObjId)`  
*Begin defining an object.*
- `#define glEndObject() __glEndObject()`  
*Stop defining an object.*
- `#define glObjectAction(_objectId, _glAction, _glValue) __glObjectAction(_objectId, _glAction, _glValue)`  
*Perform an object action.*
- `#define glAddVertex(_glX, _glY, _glZ) __glAddVertex(_glX, _glY, _glZ)`  
*Add a vertex to an object.*
- `#define glBegin(_glBeginMode) __glBegin(_glBeginMode)`  
*Begin a new polygon for the current object.*
- `#define glEnd() __glEnd()`  
*Finish a polygon for the current object.*
- `#define glBeginRender() __glBeginRender()`  
*Begin a new render.*
- `#define glCallObject(_objectId) __glCallObject(_objectId)`  
*Call a graphic object.*
- `#define glFinishRender() __glFinishRender()`  
*Finish the current render.*
- `#define glSetAngleX(_glValue) __glSetAngleX(_glValue)`  
*Set the X axis angle.*
- `#define glAddToAngleX(_glValue) __glAddToAngleX(_glValue)`  
*Add to the X axis angle.*
- `#define glSetAngleY(_glValue) __glSetAngleY(_glValue)`  
*Set the Y axis angle.*
- `#define glAddToAngleY(_glValue) __glAddToAngleY(_glValue)`  
*Add to the Y axis angle.*

- `#define glSetAngleZ(_glValue) __glSetAngleZ(_glValue)`  
*Set the Z axis angle.*
- `#define glAddToAngleZ(_glValue) __glAddToAngleZ(_glValue)`  
*Add to the Z axis angle.*
- `#define glSin32768(_glAngle, _glResult) __glSin32768(_glAngle, _glResult)`  
*Table-based sine scaled by 32768.*
- `#define glCos32768(_glAngle, _glResult) __glCos32768(_glAngle, _glResult)`  
*Table-based cosine scaled by 32768.*
- `#define glBox(_glMode, _glSizeX, _glSizeY, _glSizeZ, _glObjId) __glBox(_glMode, _glSizeX, _glSizeY, _glSizeZ, _glObjId)`  
*Create a 3D box.*
- `#define glCube(_glMode, _glSize, _glObjId) __glBox(_glMode, _glSize, _glSize, _glSize, _glObjId)`  
*Create a 3D cube.*
- `#define glPyramid(_glMode, _glSizeX, _glSizeY, _glSizeZ, _glObjId) __glPyramid(_glMode, _glSizeX, _glSizeY, _glSizeZ, _glObjId)`  
*Create a 3D pyramid.*

### 5.33.1 Detailed Description

Documentation for a simple 3D graphics library. The library code was written by Arno van der Vegt.

### 5.33.2 Macro Definition Documentation

#### 5.33.2.1 `#define glAddToAngleX( _glValue ) __glAddToAngleX(_glValue)`

Add to the X axis angle.

Add the specified value to the existing X axis angle.

##### Parameters

<code>_glValue</code>	The value to add to the X axis angle.
-----------------------	---------------------------------------

#### 5.33.2.2 `#define glAddToAngleY( _glValue ) __glAddToAngleY(_glValue)`

Add to the Y axis angle.

Add the specified value to the existing Y axis angle.

##### Parameters

<code>_glValue</code>	The value to add to the Y axis angle.
-----------------------	---------------------------------------

#### 5.33.2.3 `#define glAddToAngleZ( _glValue ) __glAddToAngleZ(_glValue)`

Add to the Z axis angle.

Add the specified value to the existing Z axis angle.

##### Parameters

<code>_glValue</code>	The value to add to the Z axis angle.
-----------------------	---------------------------------------

**5.33.2.4 #define glAddVertex( \_gIX, \_gIY, \_gIZ ) \_\_glAddVertex(\_gIX, \_gIY, \_gIZ)**

Add a vertex to an object.

Add a vertex to an object currently being defined. This function should only be used between `glBegin` and `glEnd` which are themselves nested within a `glBeginObject` and `glEndObject` pair.

**Parameters**

<code>_gIX</code>	The X axis coordinate.
<code>_gIY</code>	The Y axis coordinate.
<code>_gIZ</code>	The Z axis coordinate.

**5.33.2.5 #define glBegin( \_glBeginMode ) \_\_glBegin(\_glBeginMode)**

Begin a new polygon for the current object.

Start defining a polygon surface for the current graphics object using the specified begin mode.

**Parameters**

<code>_glBeginMode</code>	The desired mode. See <a href="#">Graphics library begin modes</a> .
---------------------------	--

**5.33.2.6 #define glBeginObject( \_gObjId ) \_\_glBeginObject(\_gObjId)**

Begin defining an object.

Start the process of defining a graphics library object using low level functions such as `glBegin`, `glAddVertex`, and `glEnd`.

**Parameters**

<code>_gObjId</code>	The object index of the new object being created.
----------------------	---

**5.33.2.7 #define glBeginRender( ) \_\_glBeginRender()**

Begin a new render.

Start the process of rendering the existing graphic objects.

**5.33.2.8 #define glBox( \_glMode, \_glSizeX, \_glSizeY, \_glSizeZ, \_gObjId ) \_\_glBox(\_glMode, \_glSizeX, \_glSizeY, \_glSizeZ, \_gObjId)**

Create a 3D box.

Define a 3D box using the specified begin mode for all faces. The center of the box is at the origin of the XYZ axis with width, height, and depth specified via the `glSizeX`, `glSizeY`, and `glSizeZ` parameters.

**Parameters**

<code>_glMode</code>	The begin mode for each surface. See <a href="#">Graphics library begin modes</a> .
<code>_glSizeX</code>	The X axis size (width).
<code>_glSizeY</code>	The Y axis size (height).
<code>_glSizeZ</code>	The Z axis size (depth).
<code>_gObjId</code>	The object ID of the new object.

**5.33.2.9 #define glCallObject( \_gObjectid ) \_\_glCallObject(\_gObjectid)**

Call a graphic object.

Tell the graphics library that you want it to include the specified object in the render.

**Parameters**

<code>_glObjectId</code>	The desired object id.
--------------------------	------------------------

**5.33.2.10 #define glCos32768( `_glAngle`, `_glResult` ) \_\_glCos32768(`_glAngle`, `_glResult`)**

Table-based cosine scaled by 32768.

Return the cosine of the specified angle in degrees. The result is scaled by 32768.

**Parameters**

<code>_glAngle</code>	The angle in degrees.
<code>_glResult</code>	The cosine value scaled by 32768.

**5.33.2.11 #define glCube( `_glMode`, `_glSize`, `_glObjId` ) \_\_glBox(`_glMode`, `_glSize`, `_glSize`, `_glSize`, `_glObjId`)**

Create a 3D cube.

Define a 3D cube using the specified begin mode for all faces. The center of the box is at the origin of the XYZ axis with equal width, height, and depth specified via the `glSize` parameter.

**Parameters**

<code>_glMode</code>	The begin mode for each surface. See <a href="#">Graphics library begin modes</a> .
<code>_glSize</code>	The cube's width, height, and depth.
<code>_glObjId</code>	The object ID of the new object.

**5.33.2.12 #define glEnd( ) \_\_glEnd()**

Finish a polygon for the current object.

Stop defining a polgyon surface for the current graphics object.

**5.33.2.13 #define glEndObject( ) \_\_glEndObject()**

Stop defining an object.

Finish the process of defining a graphics library object. Call this function after you have completed the object definition.

**5.33.2.14 #define glFinishRender( ) \_\_glFinishRender()**

Finish the current render.

Rotate the vertex list, clear the screen, and draw the rendered objects to the LCD.

**5.33.2.15 #define glInit( ) \_\_glInit()**

Initialize graphics library.

Setup all the necessary data for the graphics library to function. Call this function before any other graphics library routine.

**5.33.2.16 #define glObjectAction( `_glObjectId`, `_glAction`, `_glValue` ) \_\_glObjectAction(`_glObjectId`, `_glAction`, `_glValue`)**

Perform an object action.

Execute the specified action on the specified object.

#### Parameters

<code>_glObjectId</code>	The object id.
<code>_glAction</code>	The action to perform on the object. See <a href="#">Graphics library actions</a> .
<code>_glValue</code>	The setting value.

5.33.2.17 `#define glPyramid( _glMode, _glSizeX, _glSizeY, _glSizeZ, _glObjId ) __glPyramid(_glMode, _glSizeX, _glSizeY, _glSizeZ, _glObjId)`

Create a 3D pyramid.

Define a 3D pyramid using the specified begin mode for all faces. The center of the pyramid is at the origin of the XYZ axis with width, height, and depth specified via the glSizeX, glSizeY, and glSizeZ parameters.

#### Parameters

<code>_glMode</code>	The begin mode for each surface. See <a href="#">Graphics library begin modes</a> .
<code>_glSizeX</code>	The X axis size (width).
<code>_glSizeY</code>	The Y axis size (height).
<code>_glSizeZ</code>	The Z axis size (depth).
<code>_glObjId</code>	The object ID of the new object.

5.33.2.18 `#define glSet( _glType, _glValue ) __glSet(_glType, _glValue)`

Set graphics library options.

Adjust graphic library settings for circle size and cull mode.

#### Parameters

<code>_glType</code>	The setting type. See <a href="#">Graphics library settings</a> .
<code>_glValue</code>	The setting value. For culling modes see <a href="#">Graphics library cull mode</a> .

5.33.2.19 `#define glSetAngleX( _glValue ) __glSetAngleX(_glValue)`

Set the X axis angle.

Set the X axis angle to the specified value.

#### Parameters

<code>_glValue</code>	The new X axis angle.
-----------------------	-----------------------

5.33.2.20 `#define glSetAngleY( _glValue ) __glSetAngleY(_glValue)`

Set the Y axis angle.

Set the Y axis angle to the specified value.

#### Parameters

<code>_glValue</code>	The new Y axis angle.
-----------------------	-----------------------

**5.33.2.21 #define glSetAngleZ( \_glValue ) \_\_glSetAngleZ(\_glValue)**

Set the Z axis angle.

Set the Z axis angle to the specified value.

**Parameters**

<i>_glValue</i>	The new Z axis angle.
-----------------	-----------------------

**5.33.2.22 #define glSin32768( \_glAngle, \_glResult ) \_\_glSin32768(\_glAngle, \_glResult)**

Table-based sine scaled by 32768.

Return the sine of the specified angle in degrees. The result is scaled by 32768.

**Parameters**

<i>_glAngle</i>	The angle in degrees.
<i>_glResult</i>	The sine value scaled by 32768.

## 5.34 Output module functions

Functions for accessing and modifying output module features.

### Macros

- `#define ResetTachoCount(_p) __resetTachoCount(_p)`  
*Reset tachometer counter.*
- `#define ResetBlockTachoCount(_p) __resetBlockTachoCount(_p)`  
*Reset block-relative counter.*
- `#define ResetRotationCount(_p) __resetRotationCount(_p)`  
*Reset program-relative counter.*
- `#define ResetAllTachoCounts(_p) __resetAllTachoCounts(_p)`  
*Reset all tachometer counters.*
- `#define OnFwdEx(_ports, _pwr, _reset) __OnFwdEx(_ports, _pwr, _reset)`  
*Run motors forward and reset counters.*
- `#define OnRevEx(_ports, _pwr, _reset) __OnRevEx(_ports, _pwr, _reset)`  
*Run motors backward and reset counters.*
- `#define OnFwdExPID(_ports, _pwr, _reset, _p, _i, _d) __OnFwdExPID(_ports, _pwr, _reset, _p, _i, _d)`  
*Run motors forward and reset counters.*
- `#define OnRevExPID(_ports, _pwr, _reset, _p, _i, _d) __OnRevExPID(_ports, _pwr, _reset, _p, _i, _d)`  
*Run motors backward and reset counters.*
- `#define OnFwd(_ports, _pwr) OnFwdEx(_ports, _pwr, RESET_BLOCKANDTACHO)`  
*Run motors forward.*
- `#define OnRev(_ports, _pwr) OnRevEx(_ports, _pwr, RESET_BLOCKANDTACHO)`  
*Run motors backward.*
- `#define CoastEx(_ports, _reset) __CoastEx(_ports, _reset)`  
*Coast motors and reset counters.*
- `#define OffEx(_ports, _reset) __OffEx(_ports, _reset)`  
*Turn motors off and reset counters.*
- `#define Coast(_ports) CoastEx(_ports, RESET_BLOCKANDTACHO)`  
*Coast motors.*
- `#define Off(_ports) OffEx(_ports, RESET_BLOCKANDTACHO)`  
*Turn motors off.*
- `#define Float(_ports) Coast(_ports)`  
*Float motors.*
- `#define OnFwdRegEx(_ports, _pwr, _regmode, _reset) __OnFwdRegEx(_ports, _pwr, _regmode, _reset)`  
*Run motors forward regulated and reset counters.*
- `#define OnRevRegEx(_ports, _pwr, _regmode, _reset) __OnRevRegEx(_ports, _pwr, _regmode, _reset)`  
*Run motors backward regulated and reset counters.*
- `#define OnFwdRegExPID(_ports, _pwr, _regmode, _reset, _p, _i, _d) __OnFwdRegExPID(_ports, _pwr, _regmode, _reset, _p, _i, _d)`  
*Run motors forward regulated and reset counters with PID factors.*
- `#define OnRevRegExPID(_ports, _pwr, _regmode, _reset, _p, _i, _d) __OnRevRegExPID(_ports, _pwr, _regmode, _reset, _p, _i, _d)`  
*Run motors backward regulated and reset counters with PID factors.*
- `#define OnFwdReg(_ports, _pwr, _regmode) OnFwdRegEx(_ports, _pwr, _regmode, RESET_BLOCKANDTACHO)`

- `#define OnRevReg(_ports, _pwr, _regmode) OnRevRegEx(_ports, _pwr, _regmode, RESET_BLOCKANDTAC-HO)`
  - Run motors forward regulated.*
- `#define OnFwdRegPID(_ports, _pwr, _regmode, _p, _i, _d) OnFwdRegExPID(_ports, _pwr, _regmode, RESET_BLOCKANDTACHO, _p, _i, _d)`
  - Run motors forward regulated.*
- `#define OnRevRegPID(_ports, _pwr, _regmode, _p, _i, _d) OnRevRegExPID(_ports, _pwr, _regmode, RESET_BLOCKANDTACHO, _p, _i, _d)`
  - Run motors forward regulated with PID factors.*
- `#define OnFwdSyncEx(_ports, _pwr, _turnpct, _reset) __OnFwdSyncEx(_ports, _pwr, _turnpct, _reset)`
  - Run motors forward synchronised and reset counters.*
- `#define OnRevSyncEx(_ports, _pwr, _turnpct, _reset) __OnRevSyncEx(_ports, _pwr, _turnpct, _reset)`
  - Run motors backward synchronised and reset counters.*
- `#define OnFwdSyncExPID(_ports, _pwr, _turnpct, _reset, _p, _i, _d) __OnFwdSyncExPID(_ports, _pwr, _turnpct, _reset, _p, _i, _d)`
  - Run motors forward synchronised and reset counters with PID factors.*
- `#define OnRevSyncExPID(_ports, _pwr, _turnpct, _reset, _p, _i, _d) __OnRevSyncExPID(_ports, _pwr, _turnpct, _reset, _p, _i, _d)`
  - Run motors backward synchronised and reset counters with PID factors.*
- `#define OnFwdSync(_ports, _pwr, _turnpct) OnFwdSyncEx(_ports, _pwr, _turnpct, RESET_BLOCKANDTAC-HO)`
  - Run motors forward synchronised.*
- `#define OnRevSync(_ports, _pwr, _turnpct) OnRevSyncEx(_ports, _pwr, _turnpct, RESET_BLOCKANDTAC-HO)`
  - Run motors backward synchronised.*
- `#define OnFwdSyncPID(_ports, _pwr, _turnpct, _p, _i, _d) OnFwdSyncExPID(_ports, _pwr, _turnpct, RESET_B-LOCKANDTACHO, _p, _i, _d)`
  - Run motors forward synchronised with PID factors.*
- `#define OnRevSyncPID(_ports, _pwr, _turnpct, _p, _i, _d) OnRevSyncExPID(_ports, _pwr, _turnpct, RESET_B-LOCKANDTACHO, _p, _i, _d)`
  - Run motors backward synchronised with PID factors.*
- `#define RotateMotorExPID(_ports, _pwr, _angle, _turnpct, _bSync, _bStop, _p, _i, _d) __RotateMotorExPID(_ports, _pwr, _angle, _turnpct, _bSync, _bStop, _p, _i, _d)`
  - Rotate motor.*
- `#define RotateMotorPID(_ports, _pwr, _angle, _p, _i, _d) __RotateMotorExPID(_ports, _pwr, _angle, 0, FALSE, TRUE, _p, _i, _d)`
  - Rotate motor with PID factors.*
- `#define RotateMotorEx(_ports, _pwr, _angle, _turnpct, _bSync, _bStop) __RotateMotorExPID(_ports, _pwr, _angle, _turnpct, _bSync, _bStop, PID_3, PID_1, PID_1)`
  - Rotate motor.*
- `#define RotateMotor(_ports, _pwr, _angle) __RotateMotorExPID(_ports, _pwr, _angle, 0, FALSE, TRUE, PID_3, PID_1, PID_1)`
  - Rotate motor.*
- `#define SetOutPwnFreq(_n) __setOutPwnFreq(_n)`
  - Set motor regulation frequency.*
- `#define GetOutPwnFreq(_n) __GetOutPwnFreq(_n)`
  - Get motor regulation frequency.*
- `#define SetOutRegulationTime(_n) __setOutRegulationTime(_n)`
  -

- `#define GetOutRegulationTime(_n) __GetOutRegulationTime(_n)`  
*Set motor regulation time.*
- `#define SetOutRegulationOptions(_n) __setOutRegulationOptions(_n)`  
*Get motor regulation time.*
- `#define GetOutRegulationOptions(_n) __GetOutRegulationOptions(_n)`  
*Set motor regulation options.*
- `#define GetOutRegulationOptions(_n) __GetOutRegulationOptions(_n)`  
*Get motor regulation options.*

### 5.34.1 Detailed Description

Functions for accessing and modifying output module features.

### 5.34.2 Macro Definition Documentation

#### 5.34.2.1 `#define Coast( _ports ) CoastEx(_ports, RESET_BLOCKANDTACHO)`

Coast motors.

Turn off the specified outputs, making them coast to a stop.

##### Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
---------------------	---

#### 5.34.2.2 `#define CoastEx( _ports, _reset ) __CoastEx(_ports, _reset)`

Coast motors and reset counters.

Turn off the specified outputs, making them coast to a stop.

##### Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_reset</code>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .

#### 5.34.2.3 `#define Float( _ports ) Coast(_ports)`

Float motors.

Make outputs float. Float is an alias for Coast.

##### Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
---------------------	---

## 5.34.2.4 #define GetOutPwnFreq( \_n ) \_\_GetOutPwnFreq(\_n)

Get motor regulation frequency.

Get the current motor regulation frequency.

## Parameters

<code>_n</code>	The motor regulation frequency in milliseconds
-----------------	--

## 5.34.2.5 #define GetOutRegulationOptions( \_n ) \_\_GetOutRegulationOptions(\_n)

Get motor regulation options.

Get the current motor regulation options.

## Parameters

<code>_n</code>	The motor regulation options
-----------------	------------------------------

## 5.34.2.6 #define GetOutRegulationTime( \_n ) \_\_GetOutRegulationTime(\_n)

Get motor regulation time.

Get the current motor regulation time.

## Parameters

<code>_n</code>	The motor regulation time in milliseconds
-----------------	---

## 5.34.2.7 #define Off( \_ports ) OffEx(\_ports, RESET\_BLOCKANDTACHO)

Turn motors off.

Turn the specified outputs off (with braking).

## Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
---------------------	---

## 5.34.2.8 #define OffEx( \_ports, \_reset ) \_\_OffEx(\_ports, \_reset)

Turn motors off and reset counters.

Turn the specified outputs off (with braking).

## Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_reset</code>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .

**5.34.2.9 #define OnFwd( \_ports, \_pwr ) OnFwdEx(\_ports, \_pwr, RESET\_BLOCKANDTACHO)**

Run motors forward.

Set outputs to forward direction and turn them on.

**Parameters**

<i>_ports</i>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<i>_pwr</i>	Output power, 0 to 100. Can be negative to reverse direction.

**5.34.2.10 #define OnFwdEx( \_ports, \_pwr, \_reset ) ...OnFwdEx(\_ports, \_pwr, \_reset)**

Run motors forward and reset counters.

Set outputs to forward direction and turn them on.

**Parameters**

<i>_ports</i>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<i>_pwr</i>	Output power, 0 to 100. Can be negative to reverse direction.
<i>_reset</i>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .

**5.34.2.11 #define OnFwdExPID( \_ports, \_pwr, \_reset, \_p, \_i, \_d ) ...OnFwdExPID(\_ports, \_pwr, \_reset, \_p, \_i, \_d)**

Run motors forward and reset counters.

Set outputs to forward direction and turn them on. Specify proportional, integral, and derivative factors.

**Parameters**

<i>_ports</i>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<i>_pwr</i>	Output power, 0 to 100. Can be negative to reverse direction.
<i>_reset</i>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .
<i>_p</i>	Proportional factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<i>_i</i>	Integral factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<i>_d</i>	Derivative factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .

**5.34.2.12 #define OnFwdReg( \_ports, \_pwr, \_regmode ) OnFwdRegEx(\_ports, \_pwr, \_regmode, RESET\_BLOCKANDTACHO)**

Run motors forward regulated.

Run the specified outputs forward using the specified regulation mode.

**Parameters**

<i>_ports</i>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<i>_pwr</i>	Output power, 0 to 100. Can be negative to reverse direction.

<code>_regmode</code>	Regulation mode, see <a href="#">Output port regulation mode constants</a> .
-----------------------	--

5.34.2.13 #define OnFwdRegEx( *\_ports*, *\_pwr*, *\_regmode*, *\_reset* ) ...OnFwdRegEx(*\_ports*, *\_pwr*, *\_regmode*, *\_reset*)

Run motors forward regulated and reset counters.

Run the specified outputs forward using the specified regulation mode.

#### Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_regmode</code>	Regulation mode, see <a href="#">Output port regulation mode constants</a> .
<code>_reset</code>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .

5.34.2.14 #define OnFwdRegExPID( *\_ports*, *\_pwr*, *\_regmode*, *\_reset*, *\_p*, *\_i*, *\_d* ) ...OnFwdRegExPID(*\_ports*, *\_pwr*, *\_regmode*, *\_reset*, *\_p*, *\_i*, *\_d*)

Run motors forward regulated and reset counters with PID factors.

Run the specified outputs forward using the specified regulation mode. Specify proportional, integral, and derivative factors.

#### Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_regmode</code>	Regulation mode, see <a href="#">Output port regulation mode constants</a> .
<code>_reset</code>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .
<code>_p</code>	Proportional factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_i</code>	Integral factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_d</code>	Derivative factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .

5.34.2.15 #define OnFwdRegPID( *\_ports*, *\_pwr*, *\_regmode*, *\_p*, *\_i*, *\_d* ) OnFwdRegExPID(*\_ports*, *\_pwr*, *\_regmode*,  
RESET\_BLOCKANDTACHO, *\_p*, *\_i*, *\_d*)

Run motors forward regulated with PID factors.

Run the specified outputs forward using the specified regulation mode. Specify proportional, integral, and derivative factors.

#### Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_regmode</code>	Regulation mode, see <a href="#">Output port regulation mode constants</a> .
<code>_p</code>	Proportional factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_i</code>	Integral factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_d</code>	Derivative factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .

**5.34.2.16 #define OnFwdSync( \_ports, \_pwr, \_turnpct ) OnFwdSyncEx(\_ports, \_pwr, \_turnpct, RESET\_BLOCKANDTACHO)**

Run motors forward synchronised.

Run the specified outputs forward with regulated synchronization using the specified turn ratio.

**Parameters**

<u>_ports</u>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<u>_pwr</u>	Output power, 0 to 100. Can be negative to reverse direction.
<u>_turnpct</u>	Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.

**5.34.2.17 #define OnFwdSyncEx( \_ports, \_pwr, \_turnpct, \_reset ) \_\_OnFwdSyncEx(\_ports, \_pwr, \_turnpct, \_reset)**

Run motors forward synchronised and reset counters.

Run the specified outputs forward with regulated synchronization using the specified turn ratio.

**Parameters**

<u>_ports</u>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<u>_pwr</u>	Output power, 0 to 100. Can be negative to reverse direction.
<u>_turnpct</u>	Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.
<u>_reset</u>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .

**5.34.2.18 #define OnFwdSyncExPID( \_ports, \_pwr, \_turnpct, \_reset, \_p, \_i, \_d ) \_\_OnFwdSyncExPID(\_ports, \_pwr, \_turnpct, \_reset, \_p, \_i, \_d)**

Run motors forward synchronised and reset counters with PID factors.

Run the specified outputs forward with regulated synchronization using the specified turn ratio. Specify proportional, integral, and derivative factors.

**Parameters**

<u>_ports</u>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<u>_pwr</u>	Output power, 0 to 100. Can be negative to reverse direction.
<u>_turnpct</u>	Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.
<u>_reset</u>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .
<u>_p</u>	Proportional factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<u>_i</u>	Integral factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<u>_d</u>	Derivative factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .

**5.34.2.19 #define OnFwdSyncPID( \_ports, \_pwr, \_turnpct, \_p, \_i, \_d ) OnFwdSyncExPID(\_ports, \_pwr, \_turnpct, RESET\_BLOCKANDTACHO, \_p, \_i, \_d)**

Run motors forward synchronised with PID factors.

Run the specified outputs forward with regulated synchronization using the specified turn ratio. Specify proportional, integral, and derivative factors.

## Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_turnpct</code>	Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.
<code>_p</code>	Proportional factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_i</code>	Integral factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_d</code>	Derivative factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .

5.34.2.20 `#define OnRev( _ports, _pwr ) OnRevEx(_ports, _pwr, RESET_BLOCKANDTACHO)`

Run motors backward.

Set outputs to reverse direction and turn them on.

## Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.

5.34.2.21 `#define OnRevEx( _ports, _pwr, _reset ) __OnRevEx(_ports, _pwr, _reset)`

Run motors backward and reset counters.

Set outputs to reverse direction and turn them on.

## Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_reset</code>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .

5.34.2.22 `#define OnRevExPID( _ports, _pwr, _reset, _p, _i, _d ) __OnRevExPID(_ports, _pwr, _reset, _p, _i, _d)`

Run motors backward and reset counters.

Set outputs to reverse direction and turn them on. Specify proportional, integral, and derivative factors.

## Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_reset</code>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .
<code>_p</code>	Proportional factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_i</code>	Integral factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_d</code>	Derivative factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .

**5.34.2.23 #define OnRevReg( \_ports, \_pwr, \_regmode ) OnRevRegEx(\_ports, \_pwr, \_regmode, RESET\_BLOCKANDTACHO)**

Run motors forward regulated.

Run the specified outputs in reverse using the specified regulation mode.

**Parameters**

<i>_ports</i>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<i>_pwr</i>	Output power, 0 to 100. Can be negative to reverse direction.
<i>_regmode</i>	Regulation mode, see <a href="#">Output port regulation mode constants</a> .

**5.34.2.24 #define OnRevRegEx( \_ports, \_pwr, \_regmode, \_reset ) \_\_OnRevRegEx(\_ports, \_pwr, \_regmode, \_reset)**

Run motors backward regulated and reset counters.

Run the specified outputs in reverse using the specified regulation mode.

**Parameters**

<i>_ports</i>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<i>_pwr</i>	Output power, 0 to 100. Can be negative to reverse direction.
<i>_regmode</i>	Regulation mode, see <a href="#">Output port regulation mode constants</a> .
<i>_reset</i>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .

**5.34.2.25 #define OnRevRegExPID( \_ports, \_pwr, \_regmode, \_reset, \_p, \_i, \_d ) \_\_OnRevRegExPID(\_ports, \_pwr, \_regmode, \_reset, \_p, \_i, \_d)**

Run motors backward regulated and reset counters with PID factors.

Run the specified outputs in reverse using the specified regulation mode. Specify proportional, integral, and derivative factors.

**Parameters**

<i>_ports</i>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<i>_pwr</i>	Output power, 0 to 100. Can be negative to reverse direction.
<i>_regmode</i>	Regulation mode, see <a href="#">Output port regulation mode constants</a> .
<i>_reset</i>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .
<i>_p</i>	Proportional factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<i>_i</i>	Integral factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<i>_d</i>	Derivative factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .

**5.34.2.26 #define OnRevRegPID( \_ports, \_pwr, \_regmode, \_p, \_i, \_d ) OnRevRegExPID(\_ports, \_pwr, \_regmode, RESET\_BLOCKANDTACHO, \_p, \_i, \_d)**

Run motors reverse regulated with PID factors.

Run the specified outputs in reverse using the specified regulation mode. Specify proportional, integral, and derivative factors.

## Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_regmode</code>	Regulation mode, see <a href="#">Output port regulation mode constants</a> .
<code>_p</code>	Proportional factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_i</code>	Integral factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_d</code>	Derivative factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .

5.34.2.27 `#define OnRevSync( _ports, _pwr, _turnpct ) OnRevSyncEx(_ports, _pwr, _turnpct, RESET_BLOCKANDTACHO)`

Run motors backward synchronised.

Run the specified outputs in reverse with regulated synchronization using the specified turn ratio.

## Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_turnpct</code>	Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.

5.34.2.28 `#define OnRevSyncEx( _ports, _pwr, _turnpct, _reset ) __OnRevSyncEx(_ports, _pwr, _turnpct, _reset)`

Run motors backward synchronised and reset counters.

Run the specified outputs in reverse with regulated synchronization using the specified turn ratio.

## Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_turnpct</code>	Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.
<code>_reset</code>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .

5.34.2.29 `#define OnRevSyncExPID( _ports, _pwr, _turnpct, _reset, _p, _i, _d ) __OnRevSyncExPID(_ports, _pwr, _turnpct, _reset, _p, _i, _d)`

Run motors backward synchronised and reset counters with PID factors.

Run the specified outputs in reverse with regulated synchronization using the specified turn ratio. Specify proportional, integral, and derivative factors.

## Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_turnpct</code>	Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.
<code>_reset</code>	Position counters reset control. It must be a constant, see <a href="#">Tachometer counter reset flags</a> .

<code>_p</code>	Proportional factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_i</code>	Integral factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_d</code>	Derivative factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .

5.34.2.30 `#define OnRevSyncPID( _ports, _pwr, _turnpct, _p, _i, _d ) OnRevSyncExPID(_ports, _pwr, _turnpct, RESET_BLOCKANDTACHO, _p, _i, _d)`

Run motors backward synchronised with PID factors.

Run the specified outputs in reverse with regulated synchronization using the specified turn ratio. Specify proportional, integral, and derivative factors.

#### Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_turnpct</code>	Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.
<code>_p</code>	Proportional factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_i</code>	Integral factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_d</code>	Derivative factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .

5.34.2.31 `#define ResetAllTachoCounts( _p ) __resetAllTachoCounts(_p)`

Reset all tachometer counters.

Reset all three position counters and reset the current tachometer limit goal for the specified outputs.

#### Parameters

<code>_p</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . For multiple outputs at the same time you need to add single output port values into a byte array and pass the array instead of a single numeric value.
-----------------	--

5.34.2.32 `#define ResetBlockTachoCount( _p ) __resetBlockTachoCount(_p)`

Reset block-relative counter.

Reset the block-relative position counter for the specified outputs.

#### Parameters

<code>_p</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . For multiple outputs at the same time you need to add single output port values into a byte array and pass the array instead of a single numeric value.
-----------------	--

5.34.2.33 `#define ResetRotationCount( _p ) __resetRotationCount(_p)`

Reset program-relative counter.

Reset the program-relative position counter for the specified outputs.

## Parameters

<code>_p</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . For multiple outputs at the same time you need to add single output port values into a byte array and pass the array instead of a single numeric value.
-----------------	--

5.34.2.34 #define ResetTachoCount( `_p` ) \_\_resetTachoCount(`_p`)

Reset tachometer counter.

Reset the tachometer count and tachometer limit goal for the specified outputs.

## Parameters

<code>_p</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . For multiple outputs at the same time you need to add single output port values into a byte array and pass the array instead of a single numeric value.
-----------------	--

5.34.2.35 #define RotateMotor( `_ports`, `_pwr`, `_angle` ) \_\_RotateMotorExPID(`_ports`, `_pwr`, `_angle`, 0, FALSE, TRUE, PID\_3, PID\_1, PID\_1)

Rotate motor.

Run the specified outputs forward for the specified number of degrees.

## Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_angle</code>	Angle limit, in degree. Can be negative to reverse direction.

5.34.2.36 #define RotateMotorEx( `_ports`, `_pwr`, `_angle`, `_turnpct`, `_bSync`, `_bStop` ) \_\_RotateMotorExPID(`_ports`, `_pwr`, `_angle`, `_turnpct`, `_bSync`, `_bStop`, PID\_3, PID\_1, PID\_1)

Rotate motor.

Run the specified outputs forward for the specified number of degrees.

## Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_angle</code>	Angle limit, in degree. Can be negative to reverse direction.
<code>_turnpct</code>	Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.
<code>_bSync</code>	Synchronise two motors. Should be set to true if a non-zero turn percent is specified or no turning will occur.
<code>_bStop</code>	Specify whether the motor(s) should brake at the end of the rotation.

5.34.2.37 #define RotateMotorExPID( `_ports`, `_pwr`, `_angle`, `_turnpct`, `_bSync`, `_bStop`, `_p`, `_i`, `_d` ) \_\_RotateMotorExPID(`_ports`, `_pwr`, `_angle`, `_turnpct`, `_bSync`, `_bStop`, `_p`, `_i`, `_d`)

Rotate motor.

Run the specified outputs forward for the specified number of degrees. Specify proportional, integral, and derivative factors.

#### Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_angle</code>	Angle limit, in degree. Can be negative to reverse direction.
<code>_turnpct</code>	Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.
<code>_bSync</code>	Synchronise two motors. Should be set to true if a non-zero turn percent is specified or no turning will occur.
<code>_bStop</code>	Specify whether the motor(s) should brake at the end of the rotation.
<code>_p</code>	Proportional factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_i</code>	Integral factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_d</code>	Derivative factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .

**5.34.2.38 #define RotateMotorPID( `_ports`, `_pwr`, `_angle`, `_p`, `_i`, `_d` ) \_\_RotateMotorExPID(`_ports`, `_pwr`, `_angle`, 0, FALSE, `TRUE`, `_p`, `_i`, `_d`)**

Rotate motor with PID factors.

Run the specified outputs forward for the specified number of degrees. Specify proportional, integral, and derivative factors.

#### Parameters

<code>_ports</code>	Desired output ports. Can be a constant or a variable, see <a href="#">Output port constants</a> . If you use a variable and want to control multiple outputs in a single call you need to use a byte array rather than a byte and store the output port values in the byte array before passing it into this function.
<code>_pwr</code>	Output power, 0 to 100. Can be negative to reverse direction.
<code>_angle</code>	Angle limit, in degree. Can be negative to reverse direction.
<code>_p</code>	Proportional factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_i</code>	Integral factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .
<code>_d</code>	Derivative factor used by the firmware's PID motor control algorithm. See <a href="#">PID constants</a> .

**5.34.2.39 #define SetOutPwnFreq( `_n` ) \_\_setOutPwnFreq(`_n`)**

Set motor regulation frequency.

Set the motor regulation frequency to the specified number of milliseconds.

#### Parameters

<code>_n</code>	The motor regulation frequency in milliseconds
-----------------	--

**5.34.2.40 #define SetOutRegulationOptions( `_n` ) \_\_setOutRegulationOptions(`_n`)**

Set motor regulation options.

Set the motor regulation options.

**Parameters**

<code>_n</code>	The motor regulation options
-----------------	------------------------------

**5.34.2.41 #define SetOutRegulationTime( *\_n* ) \_setOutRegulationTime(*\_n*)**

Set motor regulation time.

Set the motor regulation time to the specified number of milliseconds.

**Parameters**

<code>_n</code>	The motor regulation time in milliseconds
-----------------	---

## 5.35 Input module functions

Functions for accessing and modifying input module features.

### Macros

- `#define SetSensorType(_port, _t) setin _t, _port, TypeField`  
*Set sensor type.*
- `#define SetSensorMode(_port, _m) setin _m, _port, InputModeField`  
*Set sensor mode.*
- `#define ReadSensor(_port, _value) getin _value, _port, ScaledValueField`  
*Read sensor scaled value.*
- `#define ClearSensor(_port) setin 0, _port, ScaledValueField`  
*Clear a sensor value.*
- `#define SetSensorTouch(_port) __SetSensorTouch(_port)`  
*Configure a touch sensor.*
- `#define SetSensorLight(_port) __SetSensorLight(_port)`  
*Configure a light sensor.*
- `#define SetSensorSound(_port) __SetSensorSound(_port)`  
*Configure a sound sensor.*
- `#define SetSensorLowspeed(_port) __SetSensorLowspeed(_port)`  
*Configure an I2C sensor.*
- `#define SetSensorUltrasonic(_port) __SetSensorLowspeed(_port)`  
*Configure an ultrasonic sensor.*
- `#define SetSensorEMeter(_port) __SetSensorLowspeed(_port)`  
*Configure an EMeter sensor.*
- `#define SetSensorTemperature(_port)`  
*Configure a temperature sensor.*
- `#define SetSensorColorFull(_port) __SetSensorColorFull(_port)`  
*Configure an NXT 2.0 full color sensor.*
- `#define SetSensorColorRed(_port) __SetSensorColorRed(_port)`  
*Configure an NXT 2.0 red light sensor.*
- `#define SetSensorColorGreen(_port) __SetSensorColorGreen(_port)`  
*Configure an NXT 2.0 green light sensor.*
- `#define SetSensorColorBlue(_port) __SetSensorColorBlue(_port)`  
*Configure an NXT 2.0 blue light sensor.*
- `#define SetSensorColorNone(_port) __SetSensorColorNone(_port)`  
*Configure an NXT 2.0 no light sensor.*
- `#define ResetSensor(_port) __ResetSensor(_port)`  
*Reset the sensor port.*
- `#define ReadSensorColorRaw(_port, _rawVals, _result) __ReadSensorColorRaw(_port, _rawVals, _result)`  
*Read LEGO color sensor raw values.*
- `#define ReadSensorColorEx(_port, _colorval, _rawVals, _normVals, _scaledVals, _result) __ReadSensorColorEx(_port, _colorval, _rawVals, _normVals, _scaledVals, _result)`  
*Read LEGO color sensor extra.*
- `#define GetInCustomZeroOffset(_p, _n) __GetInCustomZeroOffset(_p, _n)`  
*Get the custom sensor zero offset.*

- #define `GetInSensorBoolean(_p, _n)` \_\_`GetInSensorBoolean(_p, _n)`  
*Read sensor boolean value.*
- #define `GetInDigiPinsDirection(_p, _n)` \_\_`GetInDigiPinsDirection(_p, _n)`  
*Read sensor digital pins direction.*
- #define `GetInDigiPinsStatus(_p, _n)` \_\_`GetInDigiPinsStatus(_p, _n)`  
*Read sensor digital pins status.*
- #define `GetInDigiPinsOutputLevel(_p, _n)` \_\_`GetInDigiPinsOutputLevel(_p, _n)`  
*Read sensor digital pins output level.*
- #define `GetInCustomPercentFullScale(_p, _n)` \_\_`GetInCustomPercentFullScale(_p, _n)`  
*Get the custom sensor percent full scale.*
- #define `GetInCustomActiveStatus(_p, _n)` \_\_`GetInCustomActiveStatus(_p, _n)`  
*Get the custom sensor active status.*
- #define `GetInColorCalibration(_p, _np, _nc, _n)` \_\_`GetInColorCalibration(_p, _np, _nc, _n)`  
*Read a LEGO color sensor calibration point value.*
- #define `GetInColorCalLimits(_p, _np, _n)` \_\_`GetInColorCalLimits(_p, _np, _n)`  
*Read a LEGO color sensor calibration limit value.*
- #define `GetInColorADRaw(_p, _nc, _n)` \_\_`GetInColorADRaw(_p, _nc, _n)`  
*Read a LEGO color sensor AD raw value.*
- #define `GetInColorSensorRaw(_p, _nc, _n)` \_\_`GetInColorSensorRaw(_p, _nc, _n)`  
*Read a LEGO color sensor raw value.*
- #define `GetInColorSensorValue(_p, _nc, _n)` \_\_`GetInColorSensorValue(_p, _nc, _n)`  
*Read a LEGO color sensor scaled value.*
- #define `GetInColorBoolean(_p, _nc, _n)` \_\_`GetInColorBoolean(_p, _nc, _n)`  
*Read a LEGO color sensor boolean value.*
- #define `GetInColorCalibrationState(_p, _n)` \_\_`GetInColorCalibrationState(_p, _n)`  
*Read LEGO color sensor calibration state.*
- #define `SetInCustomZeroOffset(_p, _n)` \_\_`setInCustomZeroOffset(_p, _n)`  
*Set custom zero offset.*
- #define `SetInSensorBoolean(_p, _n)` \_\_`setInSensorBoolean(_p, _n)`  
*Set sensor boolean value.*
- #define `SetInDigiPinsDirection(_p, _n)` \_\_`setInDigiPinsDirection(_p, _n)`  
*Set digital pins direction.*
- #define `SetInDigiPinsStatus(_p, _n)` \_\_`setInDigiPinsStatus(_p, _n)`  
*Set digital pins status.*
- #define `SetInDigiPinsOutputLevel(_p, _n)` \_\_`setInDigiPinsOutputLevel(_p, _n)`  
*Set digital pins output level.*
- #define `SetInCustomPercentFullScale(_p, _n)` \_\_`setInCustomPercentFullScale(_p, _n)`  
*Set percent full scale.*
- #define `SetInCustomActiveStatus(_p, _n)` \_\_`setInCustomActiveStatus(_p, _n)`  
*Set active status.*

### 5.35.1 Detailed Description

Functions for accessing and modifying input module features.

### 5.35.2 Macro Definition Documentation

#### 5.35.2.1 #define ClearSensor( *\_port* ) setin 0, *\_port*, ScaledValueField

Clear a sensor value.

Clear the value of a sensor - only affects sensors that are configured to measure a cumulative quantity such as rotation or a pulse count.

##### Parameters

<i>_port</i>	The port to clear. See <a href="#">NBC Input port constants</a> .
--------------	---

#### 5.35.2.2 #define GetInColorADRaw( *\_p*, *\_nc*, *\_n* ) \_\_GetInColorADRaw(*\_p*, *\_nc*, *\_n*)

Read a LEGO color sensor AD raw value.

This function lets you directly access a specific LEGO color sensor AD raw value. Both the port and the color index must be constants.

##### Parameters

<i>_p</i>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<i>_nc</i>	The color index. See <a href="#">Color sensor array indices</a> . Must be a constant.
<i>_n</i>	The AD raw value.

##### Warning

This function requires an NXT 2.0 compatible firmware.

#### 5.35.2.3 #define GetInColorBoolean( *\_p*, *\_nc*, *\_n* ) \_\_GetInColorBoolean(*\_p*, *\_nc*, *\_n*)

Read a LEGO color sensor boolean value.

This function lets you directly access a specific LEGO color sensor boolean value. Both the port and the color index must be constants.

##### Parameters

<i>_p</i>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<i>_nc</i>	The color index. See <a href="#">Color sensor array indices</a> . Must be a constant.
<i>_n</i>	The boolean value.

##### Warning

This function requires an NXT 2.0 compatible firmware.

#### 5.35.2.4 #define GetInColorCalibration( *\_p*, *\_np*, *\_nc*, *\_n* ) \_\_GetInColorCalibration(*\_p*, *\_np*, *\_nc*, *\_n*)

Read a LEGO color sensor calibration point value.

This function lets you directly access a specific LEGO color calibration point value. The port, point, and color index must be constants.

**Parameters**

<code>_p</code>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<code>_np</code>	The calibration point. See <a href="#">Color calibration constants</a> . Must be a constant.
<code>_nc</code>	The color index. See <a href="#">Color sensor array indices</a> . Must be a constant.
<code>_n</code>	The calibration point value.

**Warning**

This function requires an NXT 2.0 compatible firmware.

**5.35.2.5 #define GetInColorCalibrationState( `_p, _n` ) `_GetInColorCalibrationState(_p, _n)`**

Read LEGO color sensor calibration state.

This function lets you directly access the LEGO color calibration state. The port must be a constant.

**Parameters**

<code>_p</code>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<code>_n</code>	The calibration state.

**Warning**

This function requires an NXT 2.0 compatible firmware.

**5.35.2.6 #define GetInColorCalLimits( `_p, _np, _n` ) `_GetInColorCalLimits(_p, _np, _n)`**

Read a LEGO color sensor calibration limit value.

This function lets you directly access a specific LEGO color calibration limit value. The port and the point must be constants.

**Parameters**

<code>_p</code>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<code>_np</code>	The calibration point. See <a href="#">Color calibration constants</a> . Must be a constant.
<code>_n</code>	The calibration limit value.

**Warning**

This function requires an NXT 2.0 compatible firmware.

**5.35.2.7 #define GetInColorSensorRaw( `_p, _nc, _n` ) `_GetInColorSensorRaw(_p, _nc, _n)`**

Read a LEGO color sensor raw value.

This function lets you directly access a specific LEGO color sensor raw value. Both the port and the color index must be constants.

**Parameters**

<code>_p</code>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<code>_nc</code>	The color index. See <a href="#">Color sensor array indices</a> . Must be a constant.
<code>_n</code>	The raw value.

**Warning**

This function requires an NXT 2.0 compatible firmware.

**5.35.2.8 #define GetInColorSensorValue( \_p, \_nc, \_n ) \_\_GetInColorSensorValue(\_p, \_nc, \_n)**

Read a LEGO color sensor scaled value.

This function lets you directly access a specific LEGO color sensor scaled value. Both the port and the color index must be constants.

**Parameters**

<u>_p</u>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<u>_nc</u>	The color index. See <a href="#">Color sensor array indices</a> . Must be a constant.
<u>_n</u>	The scaled value.

**Warning**

This function requires an NXT 2.0 compatible firmware.

**5.35.2.9 #define GetInCustomActiveStatus( \_p, \_n ) \_\_GetInCustomActiveStatus(\_p, \_n)**

Get the custom sensor active status.

Return the custom sensor active status value of a sensor.

**Parameters**

<u>_p</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_n</u>	The custom sensor active status.

**5.35.2.10 #define GetInCustomPercentFullScale( \_p, \_n ) \_\_GetInCustomPercentFullScale(\_p, \_n)**

Get the custom sensor percent full scale.

Return the custom sensor percent full scale value of a sensor.

**Parameters**

<u>_p</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_n</u>	The custom sensor percent full scale.

**5.35.2.11 #define GetInCustomZeroOffset( \_p, \_n ) \_\_GetInCustomZeroOffset(\_p, \_n)**

Get the custom sensor zero offset.

Return the custom sensor zero offset value of a sensor.

**Parameters**

<u>_p</u>	The sensor port. See <a href="#">NBC Input port constants</a> .
<u>_n</u>	The custom sensor zero offset.

**5.35.2.12 #define GetInDigiPinsDirection( \_p, \_n ) \_\_GetInDigiPinsDirection(\_p, \_n)**

Read sensor digital pins direction.

Return the digital pins direction value of a sensor on the specified port.

**Parameters**

<u>_p</u>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<u>_n</u>	The sensor's digital pins direction.

**5.35.2.13 #define GetInDigiPinsOutputLevel( \_p, \_n ) \_\_GetInDigiPinsOutputLevel(\_p, \_n)**

Read sensor digital pins output level.

Return the digital pins output level value of a sensor on the specified port.

**Parameters**

<u>_p</u>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<u>_n</u>	The sensor's digital pins output level.

**5.35.2.14 #define GetInDigiPinsStatus( \_p, \_n ) \_\_GetInDigiPinsStatus(\_p, \_n)**

Read sensor digital pins status.

Return the digital pins status value of a sensor on the specified port.

**Parameters**

<u>_p</u>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<u>_n</u>	The sensor's digital pins status.

**5.35.2.15 #define GetInSensorBoolean( \_p, \_n ) \_\_GetInSensorBoolean(\_p, \_n)**

Read sensor boolean value.

Return the boolean value of a sensor on the specified port. Boolean conversion is either done based on preset cutoffs, or a slope parameter specified by calling SetSensorMode.

**Parameters**

<u>_p</u>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<u>_n</u>	The sensor's boolean value.

**5.35.2.16 #define ReadSensor( \_port, \_value ) getin \_value, \_port, ScaledValueField**

Read sensor scaled value.

Return the processed sensor reading for a sensor on the specified port.

**Parameters**

<u>_port</u>	The sensor port. See <a href="#">NBC Input port constants</a> . A variable whose value is the desired sensor port may also be used.
<u>_value</u>	The sensor's scaled value.

---

5.35.2.17 #define ReadSensorColorEx( *\_port*, *\_colorval*, *\_rawVals*, *\_normVals*, *\_scaledVals*, *\_result* ) \_\_ReadSensorColorEx(*\_port*,  
*\_colorval*, *\_rawVals*, *\_normVals*, *\_scaledVals*, *\_result*)

Read LEGO color sensor extra.

This function lets you read the LEGO color sensor. It returns the color value, and three arrays containing raw, normalized, and scaled color values for red, green, blue, and none indices.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_colorval</i>	The color value. See <a href="#">Color values</a> .
<i>_rawVals</i>	An array containing four raw color values. See <a href="#">Color sensor array indices</a> .
<i>_normVals</i>	An array containing four normalized color values. See <a href="#">Color sensor array indices</a> .
<i>_scaledVals</i>	An array containing four scaled color values. See <a href="#">Color sensor array indices</a> .
<i>_result</i>	The function call result.

#### Warning

This function requires an NXT 2.0 compatible firmware.

5.35.2.18 #define ReadSensorColorRaw( *\_port*, *\_rawVals*, *\_result* ) \_\_ReadSensorColorRaw(*\_port*, *\_rawVals*, *\_result*)

Read LEGO color sensor raw values.

This function lets you read the LEGO color sensor. It returns an array containing raw color values for red, green, blue, and none indices.

#### Parameters

<i>_port</i>	The sensor port. See <a href="#">NBC Input port constants</a> .
<i>_rawVals</i>	An array containing four raw color values. See <a href="#">Color sensor array indices</a> .
<i>_result</i>	The function call result.

#### Warning

This function requires an NXT 2.0 compatible firmware.

5.35.2.19 #define ResetSensor( *\_port* ) \_\_ResetSensor(*\_port*)

Reset the sensor port.

Sets the invalid data flag on the specified port and waits for it to become valid again. After changing the type or the mode of a sensor port you must call this function to give the firmware time to reconfigure the sensor port.

#### Parameters

<i>_port</i>	The port to reset. See <a href="#">NBC Input port constants</a> .
--------------	---

5.35.2.20 #define SetInCustomActiveStatus( *\_p*, *\_n* ) \_\_setInCustomActiveStatus(*\_p*, *\_n*)

Set active status.

Sets the active status value of a custom sensor.

## Parameters

<code>_p</code>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<code>_n</code>	The new active status value.

**5.35.2.21 #define SetInCustomPercentFullScale( `_p, _n` ) `_setInCustomPercentFullScale(_p, _n)`**

Set percent full scale.

Sets the percent full scale value of a custom sensor.

## Parameters

<code>_p</code>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<code>_n</code>	The new percent full scale value.

**5.35.2.22 #define SetInCustomZeroOffset( `_p, _n` ) `_setInCustomZeroOffset(_p, _n)`**

Set custom zero offset.

Sets the zero offset value of a custom sensor.

## Parameters

<code>_p</code>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<code>_n</code>	The new zero offset value.

**5.35.2.23 #define SetInDigiPinsDirection( `_p, _n` ) `_setInDigiPinsDirection(_p, _n)`**

Set digital pins direction.

Sets the digital pins direction value of a sensor.

## Parameters

<code>_p</code>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<code>_n</code>	The new digital pins direction value.

**5.35.2.24 #define SetInDigiPinsOutputLevel( `_p, _n` ) `_setInDigiPinsOutputLevel(_p, _n)`**

Set digital pins output level.

Sets the digital pins output level value of a sensor.

## Parameters

<code>_p</code>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<code>_n</code>	The new digital pins output level value.

**5.35.2.25 #define SetInDigiPinsStatus( `_p, _n` ) `_setInDigiPinsStatus(_p, _n)`**

Set digital pins status.

Sets the digital pins status value of a sensor.

**Parameters**

<u>_p</u>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<u>_n</u>	The new digital pins status value.

5.35.2.26 #define SetInSensorBoolean( \_p, \_n ) \_\_setInSensorBoolean(\_p, \_n)

Set sensor boolean value.

Sets the boolean value of a sensor.

**Parameters**

<u>_p</u>	The sensor port. See <a href="#">NBC Input port constants</a> . Must be a constant.
<u>_n</u>	The new boolean value.

5.35.2.27 #define SetSensorColorBlue( \_port ) \_\_SetSensorColorBlue(\_port)

Configure an NXT 2.0 blue light sensor.

Configure the sensor on the specified port as an NXT 2.0 color sensor in blue light mode. Requires an NXT 2.0 compatible firmware.

**Parameters**

<u>_port</u>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**Warning**

This function requires an NXT 2.0 compatible firmware.

5.35.2.28 #define SetSensorColorFull( \_port ) \_\_SetSensorColorFull(\_port)

Configure an NXT 2.0 full color sensor.

Configure the sensor on the specified port as an NXT 2.0 color sensor in full color mode. Requires an NXT 2.0 compatible firmware.

**Parameters**

<u>_port</u>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**Warning**

This function requires an NXT 2.0 compatible firmware.

5.35.2.29 #define SetSensorColorGreen( \_port ) \_\_SetSensorColorGreen(\_port)

Configure an NXT 2.0 green light sensor.

Configure the sensor on the specified port as an NXT 2.0 color sensor in green light mode. Requires an NXT 2.0 compatible firmware.

**Parameters**

<u>_port</u>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**Warning**

This function requires an NXT 2.0 compatible firmware.

**5.35.2.30 #define SetSensorColorNone( *\_port* ) \_\_SetSensorColorNone(*\_port*)**

Configure an NXT 2.0 no light sensor.

Configure the sensor on the specified port as an NXT 2.0 color sensor in no light mode. Requires an NXT 2.0 compatible firmware.

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**Warning**

This function requires an NXT 2.0 compatible firmware.

**5.35.2.31 #define SetSensorColorRed( *\_port* ) \_\_SetSensorColorRed(*\_port*)**

Configure an NXT 2.0 red light sensor.

Configure the sensor on the specified port as an NXT 2.0 color sensor in red light mode. Requires an NXT 2.0 compatible firmware.

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**Warning**

This function requires an NXT 2.0 compatible firmware.

**5.35.2.32 #define SetSensorEMeter( *\_port* ) \_\_SetSensorLowspeed(*\_port*)**

Configure an EMeter sensor.

Configure the sensor on the specified port as an EMeter sensor.

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**5.35.2.33 #define SetSensorLight( *\_port* ) \_\_SetSensorLight(*\_port*)**

Configure a light sensor.

Configure the sensor on the specified port as an NXT light sensor.

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**5.35.2.34 #define SetSensorLowspeed( \_port ) \_\_SetSensorLowspeed(\_port)**

Configure an I2C sensor.

Configure the sensor on the specified port as a 9V powered I2C digital sensor.

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**5.35.2.35 #define SetSensorMode( \_port, \_m ) setin \_m, \_port, InputModeField**

Set sensor mode.

Set a sensor's mode, which should be one of the predefined sensor mode constants. A slope parameter for boolean conversion, if desired, may be added to the mode. After changing the type or the mode of a sensor port you must call [ResetSensor](#) to give the firmware time to reconfigure the sensor port.

**See Also**

[SetSensorType\(\)](#)

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
<i>_m</i>	The desired sensor mode. See <a href="#">NBC sensor mode constants</a> .

**5.35.2.36 #define SetSensorSound( \_port ) \_\_SetSensorSound(\_port)**

Configure a sound sensor.

Configure the sensor on the specified port as a sound sensor.

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**5.35.2.37 #define SetSensorTemperature( \_port )****Value:**

```
__SetSensorLowspeed(_port) \
    __MSWriteToRegister(_port, LEGO_ADDR_TEMP, TEMP_REG_CONFIG
        , TEMP_RES_12BIT, __WDSC_LSStatus)
```

Configure a temperature sensor.

Configure the sensor on the specified port as a temperature sensor. Use this to setup the temperature sensor rather than [SetSensorLowspeed](#) so that the sensor is properly configured in 12-bit conversion mode.

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**5.35.2.38 #define SetSensorTouch( \_port ) \_\_SetSensorTouch(\_port)**

Configure a touch sensor.

Configure the sensor on the specified port as a touch sensor.

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

**5.35.2.39 #define SetSensorType( \_port, \_t ) setin \_t, \_port, TypeField**

Set sensor type.

Set a sensor's type, which must be one of the predefined sensor type constants. After changing the type or the mode of a sensor port you must call [ResetSensor](#) to give the firmware time to reconfigure the sensor port.

**See Also**

[SetSensorMode\(\)](#)

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
<i>_t</i>	The desired sensor type. See <a href="#">NBC sensor type constants</a> .

**5.35.2.40 #define SetSensorUltrasonic( \_port ) \_\_SetSensorLowspeed(\_port)**

Configure an ultrasonic sensor.

Configure the sensor on the specified port as an ultrasonic sensor.

**Parameters**

<i>_port</i>	The port to configure. See <a href="#">NBC Input port constants</a> .
--------------	---

## 5.36 LowSpeed module functions

Functions for accessing and modifying low speed module features.

### Modules

- [Low level LowSpeed module functions](#)

*Low level functions for accessing low speed module features.*

### Macros

- `#define ReadSensorUS(_port, _value) __ReadSensorUS(_port, _value, 15)`  
*Read ultrasonic sensor value.*
- `#define ReadSensorUSEx(_port, _values, _result) __ReadSensorUSEx(_port, _values, _result, 15)`  
*Read multiple ultrasonic sensor values.*
- `#define ReadSensorEMeter(_port, _vIn, _aIn, _vOut, _aOut, _joules, _wIn, _wOut, _result) __ReadSensorEMeter(_port, _vIn, _aIn, _vOut, _aOut, _joules, _wIn, _wOut, _result)`  
*Read the LEGO EMeter values.*
- `#define ConfigureTemperatureSensor(_port, _config, _result) __TempSendCmd(_port, _config, _result)`  
*Configure LEGO Temperature sensor options.*
- `#define ReadSensorTemperature(_port, _temp) __ReadSensorTemperature(_port, _temp)`  
*Read the LEGO Temperature sensor value.*
- `#define LowspeedStatus(_port, _bready, _result) __lowspeedStatus(_port, _bready, _result)`  
*Get lowspeed status.*
- `#define LowspeedCheckStatus(_port, _result) __lowspeedCheckStatus(_port, _result)`  
*Check lowspeed status.*
- `#define LowspeedBytesReady(_port, _bready) __lowspeedBytesReady(_port, _bready)`  
*Get lowspeed bytes ready.*
- `#define LowspeedWrite(_port, _retlen, _buffer, _result) __lowspeedWrite(_port, _retlen, _buffer, _result)`  
*Write lowspeed data.*
- `#define LowspeedRead(_port, _buflen, _buffer, _result) __lowspeedRead(_port, _buflen, _buffer, _result)`  
*Read lowspeed data.*
- `#define ReadI2CBytes(_port, _inbuf, _count, _outbuf, _result) __ReadI2CBytes(_port, _inbuf, _count, _outbuf, _result)`  
*Perform an I2C write/read transaction.*
- `#define ReadI2CDeviceInfo(_port, _i2caddr, _info, _strVal) __ReadI2CDeviceInfo(_port, _i2caddr, _info, _strVal)`  
*Read I2C device information.*
- `#define ReadI2CVersion(_port, _i2caddr, _strVal) __ReadI2CDeviceInfo(_port, _i2caddr, I2C_REG_VERSION, _strVal)`  
*Read I2C device version.*
- `#define ReadI2CVendorId(_port, _i2caddr, _strVal) __ReadI2CDeviceInfo(_port, _i2caddr, I2C_REG_VENDOR_ID, _strVal)`  
*Read I2C device vendor.*
- `#define ReadI2CDeviceId(_port, _i2caddr, _strVal) __ReadI2CDeviceInfo(_port, _i2caddr, I2C_REG_DEVICE_ID, _strVal)`  
*Read I2C device identifier.*

- #define [ReadI2CRegister](#)(*\_port*, *\_i2caddr*, *\_reg*, *\_out*, *\_result*) [\\_\\_MSReadValue](#)(*\_port*, *\_i2caddr*, *\_reg*, 1, *\_out*, *\_result*)  
*Read I2C register.*
- #define [Writel2CRegister](#)(*\_port*, *\_i2caddr*, *\_reg*, *\_val*, *\_result*) [\\_\\_MSWriteToRegister](#)(*\_port*, *\_i2caddr*, *\_reg*, *\_val*, *\_result*)  
*Write I2C register.*
- #define [I2CSendCommand](#)(*\_port*, *\_i2caddr*, *\_cmd*, *\_result*) [\\_\\_I2CSendCmd](#)(*\_port*, *\_i2caddr*, *\_cmd*, *\_result*)  
*Send an I2C command.*
- #define [SetI2COOptions](#)(*\_port*, *\_options*) [\\_\\_setI2COOptions](#)(*\_port*, *\_options*)  
*Set I2C options.*

### 5.36.1 Detailed Description

Functions for accessing and modifying low speed module features.

### 5.36.2 Macro Definition Documentation

#### 5.36.2.1 #define ConfigureTemperatureSensor( *\_port*, *\_config*, *\_result* ) [\\_\\_TempSendCmd](#)(*\_port*, *\_config*, *\_result*)

Configure LEGO Temperature sensor options.

Set various LEGO Temperature sensor options.

##### Parameters

<i>_port</i>	The port to which the LEGO EMeter sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_config</i>	The temperature sensor configuration settings. See <a href="#">LEGO temperature sensor constants</a> for configuration constants that can be ORed or added together.
<i>_result</i>	A status code indicating whether the read completed successfully or not. See <a href="#">TCommLSRead</a> for possible Result values.

#### 5.36.2.2 #define I2CSendCommand( *\_port*, *\_i2caddr*, *\_cmd*, *\_result* ) [\\_\\_I2CSendCmd](#)(*\_port*, *\_i2caddr*, *\_cmd*, *\_result*)

Send an I2C command.

Send a command to an I2C device at the standard command register: [I2C\\_REG\\_CMD](#). The I2C device uses the specified address.

##### Parameters

<i>_port</i>	The port to which the I2C device is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.
<i>_i2caddr</i>	The I2C device address.
<i>_cmd</i>	The command to send to the I2C device.
<i>_result</i>	A status code indicating whether the write completed successfully or not. See <a href="#">TCommLSCheckStatus</a> for possible Result values.

#### 5.36.2.3 #define LowspeedBytesReady( *\_port*, *\_bready* ) [\\_\\_lowspeedBytesReady](#)(*\_port*, *\_bready*)

Get lowspeed bytes ready.

This method checks the number of bytes that are ready to be read on the specified port. If the last operation on this port was a successful LowspeedWrite call that requested response data from the device then the return value will be the number of bytes in the internal read buffer.

#### Parameters

<code>_port</code>	The port to which the I2C device is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.
<code>_bready</code>	The number of bytes available to be read from the internal I2C buffer. The maximum number of bytes that can be read is 16.

#### See Also

[LowspeedRead](#), [LowspeedWrite](#), and [LowspeedStatus](#)

#### 5.36.2.4 #define LowspeedCheckStatus( `_port`, `_result` ) `__lowspeedCheckStatus(_port, _result)`

Check lowspeed status.

This method checks the status of the I2C communication on the specified port.

#### Parameters

<code>_port</code>	The port to which the I2C device is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.
<code>_result</code>	A status code indicating whether the write completed successfully or not. See <a href="#">TCommLSCheckStatus</a> for possible Result values. If the return value is <a href="#">NO_ERR</a> then the last operation did not cause any errors. Avoid calls to <a href="#">LowspeedRead</a> or <a href="#">LowspeedWrite</a> while <a href="#">LowspeedCheckStatus</a> returns <a href="#">STAT_COMM_PENDING</a> .

#### See Also

[LowspeedRead](#), [LowspeedWrite](#), and [LowspeedStatus](#)

#### 5.36.2.5 #define LowspeedRead( `_port`, `_buflen`, `_buffer`, `_result` ) `__lowspeedRead(_port, _buflen, _buffer, _result)`

Read lowspeed data.

Read the specified number of bytes from the I2C device on the specified port and store the bytes read in the byte array buffer provided. The maximum number of bytes that can be written or read is 16.

#### Parameters

<code>_port</code>	The port to which the I2C device is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.
<code>_buflen</code>	The initial size of the output buffer.
<code>_buffer</code>	A byte array that contains the data read from the internal I2C buffer. If the return value is negative then the output buffer will be empty.
<code>_result</code>	A status code indicating whether the write completed successfully or not. See <a href="#">TCommLSRead</a> for possible Result values. If the return value is <a href="#">NO_ERR</a> then the last operation did not cause any errors.

## See Also

[LowspeedWrite](#), [LowspeedCheckStatus](#), [LowspeedBytesReady](#), and [LowspeedStatus](#)

5.36.2.6 #define LowspeedStatus( *\_port*, *\_bready*, *\_result* ) \_\_lowspeedStatus(*\_port*, *\_bready*, *\_result*)

Get lowspeed status.

This method checks the status of the I2C communication on the specified port. If the last operation on this port was a successful LowspeedWrite call that requested response data from the device then bytesready will be set to the number of bytes in the internal read buffer.

## Parameters

<i>_port</i>	The port to which the I2C device is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.
<i>_bready</i>	The number of bytes available to be read from the internal I2C buffer. The maximum number of bytes that can be read is 16.
<i>_result</i>	A status code indicating whether the write completed successfully or not. See <a href="#">TCommLSCheckStatus</a> for possible Result values. If the return value is <a href="#">NO_ERR</a> then the last operation did not cause any errors. Avoid calls to <a href="#">LowspeedRead</a> or <a href="#">LowspeedWrite</a> while LowspeedStatus returns <a href="#">STAT_COMM_PENDING</a> .

## See Also

[LowspeedRead](#), [LowspeedWrite](#), and [LowspeedCheckStatus](#)

5.36.2.7 #define LowspeedWrite( *\_port*, *\_retlen*, *\_buffer*, *\_result* ) \_\_lowspeedWrite(*\_port*, *\_retlen*, *\_buffer*, *\_result*)

Write lowspeed data.

This method starts a transaction to write the bytes contained in the array buffer to the I2C device on the specified port. It also tells the I2C device the number of bytes that should be included in the response. The maximum number of bytes that can be written or read is 16.

## Parameters

<i>_port</i>	The port to which the I2C device is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.
<i>_retlen</i>	The number of bytes that should be returned by the I2C device.
<i>_buffer</i>	A byte array containing the address of the I2C device, the I2C device register at which to write data, and up to 14 bytes of data to be written at the specified register.
<i>_result</i>	A status code indicating whether the write completed successfully or not. See <a href="#">TCommLSWrite</a> for possible Result values. If the return value is <a href="#">NO_ERR</a> then the last operation did not cause any errors.

## See Also

[LowspeedRead](#), [LowspeedCheckStatus](#), [LowspeedBytesReady](#), and [LowspeedStatus](#)

5.36.2.8 #define ReadI2CBytes( *\_port*, *\_inbuf*, *\_count*, *\_outbuf*, *\_result* ) \_\_ReadI2CBytes(*\_port*, *\_inbuf*, *\_count*, *\_outbuf*, *\_result*)

Perform an I2C write/read transaction.

This method writes the bytes contained in the input buffer (inbuf) to the I2C device on the specified port, checks for the specified number of bytes to be ready for reading, and then tries to read the specified number (count) of bytes from the I2C device into the output buffer (outbuf).

This is a higher-level wrapper around the three main I2C functions. It also maintains a "last good read" buffer and returns values from that buffer if the I2C communication transaction fails.

#### Parameters

<code>_port</code>	The port to which the I2C device is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.
<code>_inbuf</code>	A byte array containing the address of the I2C device, the I2C device register at which to write data, and up to 14 bytes of data to be written at the specified register.
<code>_count</code>	The number of bytes that should be returned by the I2C device. On output count is set to the number of bytes in outbuf.
<code>_outbuf</code>	A byte array that contains the data read from the internal I2C buffer.
<code>_result</code>	Returns true or false indicating whether the I2C transaction succeeded or failed.

#### See Also

[LowspeedRead](#), [LowspeedWrite](#), [LowspeedCheckStatus](#), [LowspeedBytesReady](#), and [LowspeedStatus](#)

5.36.2.9 #define ReadI2CDeviceId( `_port`, `_i2caddr`, `_strVal` ) ReadI2CDeviceInfo(`_port`, `_i2caddr`, `I2C_REG_DEVICE_ID`, `_strVal`)

Read I2C device identifier.

Read standard I2C device identifier. The I2C device uses the specified address.

#### Parameters

<code>_port</code>	The port to which the I2C device is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.
<code>_i2caddr</code>	The I2C device address.
<code>_strVal</code>	A string containing the device identifier.

5.36.2.10 #define ReadI2CDeviceInfo( `_port`, `_i2caddr`, `_info`, `_strVal` ) \_\_ReadI2CDeviceInfo(`_port`, `_i2caddr`, `_info`, `_strVal`)

Read I2C device information.

Read standard I2C device information: version, vendor, and device ID. The I2C device uses the specified address.

#### Parameters

<code>_port</code>	The port to which the I2C device is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.
<code>_i2caddr</code>	The I2C device address.
<code>_info</code>	A value indicating the type of device information you are requesting. See <a href="#">Standard I2C constants</a> .
<code>_strVal</code>	A string containing the requested device information.

5.36.2.11 #define ReadI2CRegister( *\_port*, *\_i2caddr*, *\_reg*, *\_out*, *\_result* ) \_\_MSReadValue(*\_port*, *\_i2caddr*, *\_reg*, 1, *\_out*, *\_result*)

Read I2C register.

Read a single byte from an I2C device register.

#### Parameters

<i>_port</i>	The port to which the I2C device is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_i2caddr</i>	The I2C device address.
<i>_reg</i>	The I2C device register from which to read a single byte.
<i>_out</i>	The single byte read from the I2C device.
<i>_result</i>	A status code indicating whether the read completed successfully or not. See <a href="#">TCommLSRead</a> for possible Result values.

5.36.2.12 #define ReadI2CVendorId( *\_port*, *\_i2caddr*, *\_strVal* ) ReadI2CDeviceInfo(*\_port*, *\_i2caddr*, I2C\_REG\_VENDOR\_ID, *\_strVal*)

Read I2C device vendor.

Read standard I2C device vendor. The I2C device uses the specified address.

#### Parameters

<i>_port</i>	The port to which the I2C device is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.
<i>_i2caddr</i>	The I2C device address.
<i>_strVal</i>	A string containing the device vendor.

5.36.2.13 #define ReadI2CVersion( *\_port*, *\_i2caddr*, *\_strVal* ) ReadI2CDeviceInfo(*\_port*, *\_i2caddr*, I2C\_REG\_VERSION, *\_strVal*)

Read I2C device version.

Read standard I2C device version. The I2C device uses the specified address.

#### Parameters

<i>_port</i>	The port to which the I2C device is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.
<i>_i2caddr</i>	The I2C device address.
<i>_strVal</i>	A string containing the device version.

5.36.2.14 #define ReadSensorEMeter( *\_port*, *\_vIn*, *\_aIn*, *\_vOut*, *\_aOut*, *\_joules*, *\_wIn*, *\_wOut*, *\_result* ) \_\_ReadSensorEMeter(*\_port*, *\_vIn*, *\_aIn*, *\_vOut*, *\_aOut*, *\_joules*, *\_wIn*, *\_wOut*, *\_result*)

Read the LEGO EMeter values.

Read all the LEGO EMeter register values. They must all be read at once to ensure data coherency.

#### Parameters

<i>_port</i>	The port to which the LEGO EMeter sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
--------------	---

<code>_vIn</code>	Input voltage
<code>_aln</code>	Input current
<code>_vOut</code>	Output voltage
<code>_aOut</code>	Output current
<code>_joules</code>	The number of joules stored in E-Meter
<code>_wIn</code>	The number of watts generated
<code>_wOut</code>	The number of watts consumed
<code>_result</code>	A status code indicating whether the read completed successfully or not. See <a href="#">TCommLSRead</a> for possible Result values.

#### 5.36.2.15 #define ReadSensorTemperature( `_port`, `_temp` ) `_ReadSensorTemperature(_port, _temp)`

Read the LEGO Temperature sensor value.

Return the temperature sensor value in degrees celcius. Since a temperature sensor is an I2C digital sensor its value cannot be read using the standard Sensor(n) value. The port must be configured as a temperature sensor port before using this function. Use [SetSensorTemperature](#) to configure the port.

##### Parameters

<code>_port</code>	The port to which the temperature sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_temp</code>	The temperature sensor value in degrees celcius.

#### 5.36.2.16 #define ReadSensorUS( `_port`, `_value` ) `_ReadSensorUS(_port, _value, 15)`

Read ultrasonic sensor value.

Return the ultrasonic sensor distance value. Since an ultrasonic sensor is an I2C digital sensor its value cannot be read using the standard Sensor(n) value. The port must be configured as a Lowspeed port before using this function.

##### Parameters

<code>_port</code>	The port to which the ultrasonic sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_value</code>	The ultrasonic sensor distance value (0..255)

#### 5.36.2.17 #define ReadSensorUSEx( `_port`, `_values`, `_result` ) `_ReadSensorUSEx(_port, _values, _result, 15)`

Read multiple ultrasonic sensor values.

Return eight ultrasonic sensor distance values.

##### Parameters

<code>_port</code>	The port to which the ultrasonic sensor is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<code>_values</code>	An array of bytes that will contain the 8 distance values read from the ultrasonic sensor.
<code>_result</code>	A status code indicating whether the read completed successfully or not. See <a href="#">TCommLSRead</a> for possible Result values.

#### 5.36.2.18 #define SetI2COptions( `_port`, `_options` ) `_setI2COOptions(_port, _options)`

Set I2C options.

This method lets you modify I2C options. Use this function to turn on or off the fast I2C mode and also control whether the standard I2C mode performs a restart prior to the read operation.

**Parameters**

<i>_port</i>	The port whose I2C options you wish to change. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_options</i>	The new option value. See <a href="#">I2C option constants</a> .

5.36.2.19 #define Writel2CRegister( *\_port*, *\_i2caddr*, *\_reg*, *\_val*, *\_result* ) [\\_MSWriteToRegister](#)(*\_port*, *\_i2caddr*, *\_reg*, *\_val*, *\_result*)

Write I2C register.

Write a single byte to an I2C device register.

**Parameters**

<i>_port</i>	The port to which the I2C device is attached. See the <a href="#">NBC Input port constants</a> group. You may use a constant or a variable.
<i>_i2caddr</i>	The I2C device address.
<i>_reg</i>	The I2C device register to which to write a single byte.
<i>_val</i>	The byte to write to the I2C device.
<i>_result</i>	A status code indicating whether the write completed successfully or not. See <a href="#">TCommLSCheck-Status</a> for possible Result values.

## 5.37 Low level LowSpeed module functions

Low level functions for accessing low speed module features.

### Macros

- `#define GetLSInputBuffer(_p, _offset, _cnt, _data) __getLSInputBuffer(_p, _offset, _cnt, _data)`  
*Get I2C input buffer data.*
- `#define GetLSInputBufferInPtr(_p, _n) __GetLSInputBufferInPtr(_p, _n)`  
*Get I2C input buffer in-pointer.*
- `#define GetLSInputBufferOutPtr(_p, _n) __GetLSInputBufferOutPtr(_p, _n)`  
*Get I2C input buffer out-pointer.*
- `#define GetLSInputBufferBytesToRx(_p, _n) __GetLSInputBufferBytesToRx(_p, _n)`  
*Get I2C input buffer bytes to rx.*
- `#define GetLSSOutputBuffer(_p, _offset, _cnt, _data) __getLSSOutputBuffer(_p, _offset, _cnt, _data)`  
*Get I2C output buffer data.*
- `#define GetLSSOutputBufferInPtr(_p, _n) __GetLSSOutputBufferInPtr(_p, _n)`  
*Get I2C output buffer in-pointer.*
- `#define GetLSSOutputBufferOutPtr(_p, _n) __GetLSSOutputBufferOutPtr(_p, _n)`  
*Get I2C output buffer out-pointer.*
- `#define GetLSSOutputBufferBytesToRx(_p, _n) __GetLSSOutputBufferBytesToRx(_p, _n)`  
*Get I2C output buffer bytes to rx.*
- `#define GetLSMode(_p, _n) __GetLSMode(_p, _n)`  
*Get I2C mode.*
- `#define GetLSChannelState(_p, _n) __GetLSChannelState(_p, _n)`  
*Get I2C channel state.*
- `#define GetLSErrorType(_p, _n) __GetLSErrorType(_p, _n)`  
*Get I2C error type.*
- `#define GetLSState(_n) __GetLSState(_n)`  
*Get I2C state.*
- `#define GetLSSpeed(_n) __GetLSSpeed(_n)`  
*Get I2C speed.*
- `#define GetLSNoRestartOnRead(_n) __GetLSNoRestartOnRead(_n)`  
*Get I2C no restart on read setting.*

### 5.37.1 Detailed Description

Low level functions for accessing low speed module features.

### 5.37.2 Macro Definition Documentation

#### 5.37.2.1 `#define GetLSChannelState( _p, _n ) __GetLSChannelState(_p, _n)`

Get I2C channel state.

This method returns the value of the I2C channel state for the specified port.

### Parameters

<u>_p</u>	A constant port number (S1..S4). See <a href="#">NBC Input port constants</a> .
<u>_n</u>	The I2C port channel state. See <a href="#">LSCChannelState constants</a> .

#### 5.37.2.2 #define GetLSErrorType( \_p, \_n ) \_\_GetLSErrorType(\_p, \_n)

Get I2C error type.

This method returns the value of the I2C error type for the specified port.

##### Parameters

<u>_p</u>	A constant port number (S1..S4). See <a href="#">NBC Input port constants</a> .
<u>_n</u>	The I2C port error type. See <a href="#">LSErrorType constants</a> .

#### 5.37.2.3 #define GetLSInputBuffer( \_p, \_offset, \_cnt, \_data ) \_\_getLSInputBuffer(\_p, \_offset, \_cnt, \_data)

Get I2C input buffer data.

This method reads count bytes of data from the I2C input buffer for the specified port and writes it to the buffer provided.

##### Parameters

<u>_p</u>	A constant port number (S1..S4). See <a href="#">NBC Input port constants</a> .
<u>_offset</u>	A constant offset into the I2C input buffer.
<u>_cnt</u>	The number of bytes to read.
<u>_data</u>	The byte array reference which will contain the data read from the I2C input buffer.

#### 5.37.2.4 #define GetLSInputBufferBytesToRx( \_p, \_n ) \_\_GetLSInputBufferBytesToRx(\_p, \_n)

Get I2C input buffer bytes to rx.

This method returns the value of the bytes to rx field of the I2C input buffer for the specified port.

##### Parameters

<u>_p</u>	A constant port number (S1..S4). See <a href="#">NBC Input port constants</a> .
<u>_n</u>	The I2C input buffer's bytes to rx value.

#### 5.37.2.5 #define GetLSInputBufferInPtr( \_p, \_n ) \_\_GetLSInputBufferInPtr(\_p, \_n)

Get I2C input buffer in-pointer.

This method returns the value of the input pointer of the I2C input buffer for the specified port.

##### Parameters

<u>_p</u>	A constant port number (S1..S4). See <a href="#">NBC Input port constants</a> .
<u>_n</u>	The I2C input buffer's in-pointer value.

#### 5.37.2.6 #define GetLSInputBufferOutPtr( \_p, \_n ) \_\_GetLSInputBufferOutPtr(\_p, \_n)

Get I2C input buffer out-pointer.

This method returns the value of the output pointer of the I2C input buffer for the specified port.

**Parameters**

<code>_p</code>	A constant port number (S1..S4). See <a href="#">NBC Input port constants</a> .
<code>_n</code>	The I2C input buffer's out-pointer value.

**5.37.2.7 #define GetLSMode( `_p`, `_n` ) \_\_GetLSMode(`_p`, `_n`)**

Get I2C mode.

This method returns the value of the I2C mode for the specified port.

**Parameters**

<code>_p</code>	A constant port number (S1..S4). See <a href="#">NBC Input port constants</a> .
<code>_n</code>	The I2C port mode. See <a href="#">LSMode constants</a> .

**5.37.2.8 #define GetLSNoRestartOnRead( `_n` ) \_\_GetLSNoRestartOnRead(`_n`)**

Get I2C no restart on read setting.

This method returns the value of the I2C no restart on read field.

**Parameters**

<code>_n</code>	The I2C no restart on read field. See <a href="#">LSNoRestartOnRead constants</a> .
-----------------	---

**5.37.2.9 #define GetLSSOutputBuffer( `_p`, `_offset`, `_cnt`, `_data` ) \_\_getLSSOutputBuffer(`_p`, `_offset`, `_cnt`, `_data`)**

Get I2C output buffer data.

This method reads cnt bytes of data from the I2C output buffer for the specified port and writes it to the buffer provided.

**Parameters**

<code>_p</code>	A constant port number (S1..S4). See <a href="#">NBC Input port constants</a> .
<code>_offset</code>	A constant offset into the I2C output buffer.
<code>_cnt</code>	The number of bytes to read.
<code>_data</code>	The byte array reference which will contain the data read from the I2C output buffer.

**5.37.2.10 #define GetLSSOutputBufferBytesToRx( `_p`, `_n` ) \_\_GetLSSOutputBufferBytesToRx(`_p`, `_n`)**

Get I2C output buffer bytes to rx.

This method returns the value of the bytes to rx field of the I2C output buffer for the specified port.

**Parameters**

<code>_p</code>	A constant port number (S1..S4). See <a href="#">NBC Input port constants</a> .
<code>_n</code>	The I2C output buffer's bytes to rx value.

**5.37.2.11 #define GetLSSOutputBufferInPtr( `_p`, `_n` ) \_\_GetLSSOutputBufferInPtr(`_p`, `_n`)**

Get I2C output buffer in-pointer.

This method returns the value of the input pointer of the I2C output buffer for the specified port.

**Parameters**

<code>_p</code>	A constant port number (S1..S4). See <a href="#">NBC Input port constants</a> .
<code>_n</code>	The I2C output buffer's in-pointer value.

5.37.2.12 #define GetLSSOutputBufferOutPtr( `_p`, `_n` ) \_\_GetLSSOutputBufferOutPtr(`_p`, `_n`)

Get I2C output buffer out-pointer.

This method returns the value of the output pointer of the I2C output buffer for the specified port.

**Parameters**

<code>_p</code>	A constant port number (S1..S4). See <a href="#">NBC Input port constants</a> .
<code>_n</code>	The I2C output buffer's out-pointer value.

5.37.2.13 #define GetLSSpeed( `_n` ) \_\_GetLSSpeed(`_n`)

Get I2C speed.

This method returns the value of the I2C speed.

**Parameters**

<code>_n</code>	The I2C speed.
-----------------	----------------

**Warning**

This function is unimplemented within the firmware.

5.37.2.14 #define GetLSState( `_n` ) \_\_GetLSState(`_n`)

Get I2C state.

This method returns the value of the I2C state.

**Parameters**

<code>_n</code>	The I2C state. See <a href="#">LSSState constants</a> .
-----------------	---

## 5.38 Display module functions

Functions for accessing and modifying display module features.

### Macros

- `#define ClearLine(_line) __TextOutEx(0, _line, __BlankLine, 0)`  
*Clear a line on the LCD screen.*
- `#define PointOutEx(_x, _y, _options) __PointOutEx(_x, _y, _options)`  
*Draw a point with drawing options.*
- `#define PointOut(_x, _y) __PointOutEx(_x, _y, 0)`  
*Draw a point.*
- `#define ClearScreen() __PointOutEx(200, 200, 1)`  
*Clear LCD screen.*
- `#define LineOutEx(_x1, _y1, _x2, _y2, _options) __LineOutEx(_x1, _y1, _x2, _y2, _options)`  
*Draw a line with drawing options.*
- `#define LineOut(_x1, _y1, _x2, _y2) __LineOutEx(_x1, _y1, _x2, _y2, 0)`  
*Draw a line.*
- `#define RectOutEx(_x, _y, _w, _h, _options) __RectOutEx(_x, _y, _w, _h, _options)`  
*Draw a rectangle with drawing options.*
- `#define RectOut(_x, _y, _w, _h) __RectOutEx(_x, _y, _w, _h, 0)`  
*Draw a rectangle.*
- `#define CircleOutEx(_x, _y, _r, _options) __CircleOutEx(_x, _y, _r, _options)`  
*Draw a circle with drawing options.*
- `#define CircleOut(_x, _y, _r) __CircleOutEx(_x, _y, _r, 0)`  
*Draw a circle.*
- `#define NumOutEx(_x, _y, _num, _options) __NumOutEx(_x, _y, _num, _options)`  
*Draw a number with drawing options.*
- `#define NumOut(_x, _y, _num) __NumOutEx(_x, _y, _num, 0)`  
*Draw a number.*
- `#define TextOutEx(_x, _y, _txt, _options) __TextOutEx(_x, _y, _txt, _options)`  
*Draw text.*
- `#define TextOut(_x, _y, _txt) __TextOutEx(_x, _y, _txt, 0)`  
*Draw text.*
- `#define GraphicOutEx(_x, _y, _file, _vars, _options) __GraphicOutEx(_x, _y, _file, _vars, _options)`  
*Draw a graphic image with parameters and drawing options.*
- `#define GraphicOut(_x, _y, _file) __GraphicOutEx(_x, _y, _file, __GraphicOutEmptyVars, 0)`  
*Draw a graphic image.*
- `#define GraphicArrayOutEx(_x, _y, _data, _vars, _options) __GraphicArrayOutEx(_x, _y, _data, _vars, _options)`  
*Draw a graphic image from byte array with parameters and drawing options.*
- `#define GraphicArrayOut(_x, _y, _data) __GraphicArrayOutEx(_x, _y, _data, __GraphicOutEmptyVars, 0)`  
*Draw a graphic image from byte array.*
- `#define EllipseOutEx(_x, _y, _rX, _rY, _options) __EllipseOutEx(_x, _y, _rX, _rY, _options)`  
*Draw an ellipse with drawing options.*
- `#define EllipseOut(_x, _y, _rX, _rY) __EllipseOutEx(_x, _y, _rX, _rY, 0)`  
*Draw an ellipse.*
- `#define PolyOutEx(_points, _options) __PolyOutEx(_points, _options)`

- `#define PolyOut(_points) __PolyOutEx(_points,0)`  
*Draw a polygon.*
- `#define FontTextOutEx(_x, _y, _fnt, _txt, _options) __FontTextOutEx(_x,_y,_fnt,_txt,_options)`  
*Draw text with font and drawing options.*
- `#define FontTextOut(_x, _y, _fnt, _txt) __FontTextOutEx(_x,_y,_fnt,_txt,0)`  
*Draw text with font.*
- `#define FontNumOutEx(_x, _y, _fnt, _num, _options) __FontNumOutEx(_x,_y,_fnt,_num,_options)`  
*Draw a number with font and drawing options.*
- `#define FontNumOut(_x, _y, _fnt, _num) __FontNumOutEx(_x,_y,_fnt,_num,0)`  
*Draw a number with font.*
- `#define GetDisplayEraseMask(_n) __GetDisplayEraseMask(_n)`  
*Read the display erase mask value.*
- `#define GetDisplayUpdateMask(_n) __GetDisplayUpdateMask(_n)`  
*Read the display update mask value.*
- `#define GetDisplayFont(_n) __GetDisplayFont(_n)`  
*Read the display font memory address.*
- `#define GetDisplayDisplay(_n) __GetDisplayDisplay(_n)`  
*Read the display memory address.*
- `#define GetDisplayFlags(_n) __GetDisplayFlags(_n)`  
*Read the display flags.*
- `#define GetDisplayTextLinesCenterFlags(_n) __GetDisplayTextLinesCenterFlags(_n)`  
*Read the display text lines center flags.*
- `#define GetDisplayContrast(_n) __GetDisplayContrast(_n)`  
*Read the display contrast setting.*
- `#define GetDisplayNormal(_x, _line, _cnt, _data) __getDisplayNormal(_x, _line, _cnt, _data)`  
*Read pixel data from the normal display buffer.*
- `#define GetDisplayPopup(_x, _line, _cnt, _data) __getDisplayPopup(_x, _line, _cnt, _data)`  
*Read pixel data from the popup display buffer.*
- `#define SetDisplayFont(_n) __setDisplayFont(_n)`  
*Set the display font memory address.*
- `#define SetDisplayDisplay(_n) __setDisplayDisplay(_n)`  
*Set the display memory address.*
- `#define SetDisplayEraseMask(_n) __setDisplayEraseMask(_n)`  
*Set the display erase mask.*
- `#define SetDisplayFlags(_n) __setDisplayFlags(_n)`  
*Set the display flags.*
- `#define SetDisplayTextLinesCenterFlags(_n) __setDisplayTextLinesCenterFlags(_n)`  
*Set the display text lines center flags.*
- `#define SetDisplayUpdateMask(_n) __setDisplayUpdateMask(_n)`  
*Set the display update mask.*
- `#define SetDisplayContrast(_n) __setDisplayContrast(_n)`  
*Set the display contrast.*
- `#define SetDisplayNormal(_x, _line, _cnt, _data) __setDisplayNormal(_x, _line, _cnt, _data)`  
*Write pixel data to the normal display buffer.*
- `#define SetDisplayPopup(_x, _line, _cnt, _data) __setDisplayPopup(_x, _line, _cnt, _data)`  
*Write pixel data to the popup display buffer.*

### 5.38.1 Detailed Description

Functions for accessing and modifying display module features.

### 5.38.2 Macro Definition Documentation

#### 5.38.2.1 #define CircleOut( *\_x*, *\_y*, *\_r* ) \_\_CircleOutEx(*\_x*,*\_y*,*\_r*,0)

Draw a circle.

This function lets you draw a circle on the screen with its center at the specified x and y location, using the specified radius.

#### See Also

[TDrawCircle](#)

#### Parameters

<i>_x</i>	The x value for the center of the circle.
<i>_y</i>	The y value for the center of the circle.
<i>_r</i>	The radius of the circle.

#### 5.38.2.2 #define CircleOutEx( *\_x*, *\_y*, *\_r*, *\_options* ) \_\_CircleOutEx(*\_x*,*\_y*,*\_r*,*\_options*)

Draw a circle with drawing options.

This function lets you draw a circle on the screen with its center at the specified x and y location, using the specified radius. Also specify drawing options. Valid display option constants are listed in the [Drawing option constants](#) group.

#### See Also

[TDrawCircle](#)

#### Parameters

<i>_x</i>	The x value for the center of the circle.
<i>_y</i>	The y value for the center of the circle.
<i>_r</i>	The radius of the circle.
<i>_options</i>	The optional drawing options.

#### 5.38.2.3 #define ClearLine( *\_line* ) \_\_TextOutEx(0, *\_line*, \_\_BlankLine, 0)

Clear a line on the LCD screen.

This function lets you clear a single line on the NXT LCD.

#### Parameters

<i>_line</i>	The line you want to clear. See <a href="#">Line number constants</a> .
--------------	---

#### 5.38.2.4 #define ClearScreen( ) \_\_PointOutEx(200, 200, 1)

Clear LCD screen.

This function lets you clear the NXT LCD to a blank screen.

5.38.2.5 `#define EllipseOut( _x, _y, _rX, _rY ) __EllipseOutEx(_x,_y,_rX,_rY,0)`

Draw an ellipse.

This function lets you draw an ellipse on the screen with its center at the specified x and y location, using the specified radii.

#### See Also

[TDrawEllipse](#)

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<code>_x</code>	The x value for the center of the ellipse.
<code>_y</code>	The y value for the center of the ellipse.
<code>_rX</code>	The x axis radius.
<code>_rY</code>	The y axis radius.

5.38.2.6 `#define EllipseOutEx( _x, _y, _rX, _rY, _options ) __EllipseOutEx(_x,_y,_rX,_rY,_options)`

Draw an ellipse with drawing options.

This function lets you draw an ellipse on the screen with its center at the specified x and y location, using the specified radii. Also specify drawing options. Valid display option constants are listed in the [Drawing option constants](#) group.

#### See Also

[SysDrawEllipse](#), [DrawEllipseType](#)

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<code>_x</code>	The x value for the center of the ellipse.
<code>_y</code>	The y value for the center of the ellipse.
<code>_rX</code>	The x axis radius.
<code>_rY</code>	The y axis radius.
<code>_options</code>	The drawing options.

5.38.2.7 `#define FontNumOut( _x, _y, _fnt, _num ) __FontNumOutEx(_x,_y,_fnt,_num,0)`

Draw a number with font.

Draw a numeric value on the screen at the specified x and y location using a custom RIC font.

**See Also**[FontTextOut](#), [TDrawFont](#)**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<code>_x</code>	The x value for the start of the number output.
<code>_y</code>	The y value for the start of the number output.
<code>_fnt</code>	The filename of the RIC font.
<code>_num</code>	The value to output to the LCD screen. Any numeric type is supported.

**5.38.2.8 #define FontNumOutEx( `_x`, `_y`, `_fnt`, `_num`, `_options` ) \_\_FontNumOutEx(`_x`,`_y`,`_fnt`,`_num`,`_options`)**

Draw a number with font and drawing options.

Draw a numeric value on the screen at the specified x and y location using a custom RIC font. Also specify drawing options. Valid display option constants are listed in the [Drawing option constants](#) group. See the [Font drawing option constants](#) for options specific to the font drawing functions.

**See Also**[FontTextOut](#), [TDrawFont](#)**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<code>_x</code>	The x value for the start of the number output.
<code>_y</code>	The y value for the start of the number output.
<code>_fnt</code>	The filename of the RIC font.
<code>_num</code>	The value to output to the LCD screen. Any numeric type is supported.
<code>_options</code>	The optional drawing options.

**5.38.2.9 #define FontTextOut( `_x`, `_y`, `_fnt`, `_txt` ) \_\_FontTextOutEx(`_x`,`_y`,`_fnt`,`_txt`,0)**

Draw text with font.

Draw a text value on the screen at the specified x and y location using a custom RIC font.

**See Also**[FontNumOut](#), [TDrawFont](#)**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<i>_x</i>	The x value for the start of the text output.
<i>_y</i>	The y value for the start of the text output.
<i>_fnt</i>	The filename of the RIC font.
<i>_txt</i>	The text to output to the LCD screen.

**5.38.2.10 #define FontTextOutEx( *\_x*, *\_y*, *\_fnt*, *\_txt*, *\_options* ) \_\_FontTextOutEx(*\_x*,*\_y*,*\_fnt*,*\_txt*,*\_options*)**

Draw text with font and drawing options.

Draw a text value on the screen at the specified x and y location using a custom RIC font. Also specify drawing options. Valid display option constants are listed in the [Drawing option constants](#) group. See the [Font drawing option constants](#) for options specific to the font drawing functions.

**See Also**

[FontNumOut](#), [TDrawFont](#)

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<i>_x</i>	The x value for the start of the text output.
<i>_y</i>	The y value for the start of the text output.
<i>_fnt</i>	The filename of the RIC font.
<i>_txt</i>	The text to output to the LCD screen.
<i>_options</i>	The drawing options.

**5.38.2.11 #define GetDisplayContrast( *\_n* ) \_\_GetDisplayContrast(*\_n*)**

Read the display contrast setting.

This function lets you read the current display contrast setting.

**Parameters**

<i>_n</i>	The current display contrast (byte).
-----------	--------------------------------------

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**5.38.2.12 #define GetDisplayDisplay( *\_n* ) \_\_GetDisplayDisplay(*\_n*)**

Read the display memory address.

This function lets you read the current display memory address.

**Parameters**

<i>_n</i>	The current display memory address.
-----------	-------------------------------------

**5.38.2.13 #define GetDisplayEraseMask( \_n ) \_\_GetDisplayEraseMask(\_n)**

Read the display erase mask value.

This function lets you read the current display erase mask value.

**Parameters**

<code>_n</code>	The current display erase mask value.
-----------------	---------------------------------------

**5.38.2.14 #define GetDisplayFlags( \_n ) \_\_GetDisplayFlags(\_n)**

Read the display flags.

This function lets you read the current display flags. Valid flag values are listed in the [Display flags](#) group.

**Parameters**

<code>_n</code>	The current display flags.
-----------------	----------------------------

**5.38.2.15 #define GetDisplayFont( \_n ) \_\_GetDisplayFont(\_n)**

Read the display font memory address.

This function lets you read the current display font memory address.

**Parameters**

<code>_n</code>	The current display font memory address.
-----------------	--

**5.38.2.16 #define GetDisplayNormal( \_x, \_line, \_cnt, \_data ) \_\_getDisplayNormal(\_x, \_line, \_cnt, \_data)**

Read pixel data from the normal display buffer.

Read "cnt" bytes from the normal display memory into the data array. Start reading from the specified x, line coordinate. Each byte of data read from screen memory is a vertical strip of 8 bits at the desired location. Each bit represents a single pixel on the LCD screen. Use TEXTLINE\_1 through TEXTLINE\_8 for the "line" parameter.

**Parameters**

<code>_x</code>	The desired x position from which to read pixel data.
<code>_line</code>	The desired line from which to read pixel data.
<code>_cnt</code>	The number of bytes of pixel data to read.
<code>_data</code>	The array of bytes into which pixel data is read.

**5.38.2.17 #define GetDisplayPopup( \_x, \_line, \_cnt, \_data ) \_\_getDisplayPopup(\_x, \_line, \_cnt, \_data)**

Read pixel data from the popup display buffer.

Read "cnt" bytes from the popup display memory into the data array. Start reading from the specified x, line coordinate. Each byte of data read from screen memory is a vertical strip of 8 bits at the desired location. Each bit represents a single pixel on the LCD screen. Use TEXTLINE\_1 through TEXTLINE\_8 for the "line" parameter.

**Parameters**

<code>_x</code>	The desired x position from which to read pixel data.
<code>_line</code>	The desired line from which to read pixel data.

<i>_cnt</i>	The number of bytes of pixel data to read.
<i>_data</i>	The array of bytes into which pixel data is read.

5.38.2.18 #define GetDisplayTextLinesCenterFlags( *\_n* ) \_\_GetDisplayTextLinesCenterFlags(*\_n*)

Read the display text lines center flags.

This function lets you read the current display text lines center flags.

**Parameters**

<i>_n</i>	The current display text lines center flags.
-----------	--

5.38.2.19 #define GetDisplayUpdateMask( *\_n* ) \_\_GetDisplayUpdateMask(*\_n*)

Read the display update mask value.

This function lets you read the current display update mask value.

**Parameters**

<i>_n</i>	The current display update mask.
-----------	----------------------------------

5.38.2.20 #define GraphicArrayOut( *\_x*, *\_y*, *\_data* ) \_\_GraphicArrayOutEx(*\_x*,*\_y*,*\_data*,\_\_GraphicOutEmptyVars,0)

Draw a graphic image from byte array.

Draw a graphic image byte array on the screen at the specified x and y location. If the file cannot be found then nothing will be drawn and no errors will be reported.

**See Also**

[TDrawGraphicArray](#)

**Parameters**

<i>_x</i>	The x value for the position of the graphic image.
<i>_y</i>	The y value for the position of the graphic image.
<i>_data</i>	The byte array of the RIC graphic image.

5.38.2.21 #define GraphicArrayOutEx( *\_x*, *\_y*, *\_data*, *\_vars*, *\_options* ) \_\_GraphicArrayOutEx(*\_x*,*\_y*,*\_data*,*\_vars*,*\_options*)

Draw a graphic image from byte array with parameters and drawing options.

Draw a graphic image byte array on the screen at the specified x and y location using an array of parameters and drawing options. Valid display option constants are listed in the [Drawing option constants](#) group. If the file cannot be found then nothing will be drawn and no errors will be reported.

**See Also**

[TDrawGraphicArray](#)

**Parameters**

<i>_x</i>	The x value for the position of the graphic image.
<i>_y</i>	The y value for the position of the graphic image.
<i>_data</i>	The byte array of the RIC graphic image.
<i>_vars</i>	The byte array of parameters.
<i>_options</i>	The drawing options.

5.38.2.22 #define GraphicOut( *\_x*, *\_y*, *\_file* ) \_\_GraphicOutEx(*\_x*,*\_y*,*\_file*,\_\_GraphicOutEmptyVars,0)

Draw a graphic image.

Draw a graphic image file on the screen at the specified x and y location. If the file cannot be found then nothing will be drawn and no errors will be reported.

**See Also**

TDrawGraphic

**Parameters**

<i>_x</i>	The x value for the position of the graphic image.
<i>_y</i>	The y value for the position of the graphic image.
<i>_file</i>	The filename of the RIC graphic image.

5.38.2.23 #define GraphicOutEx( *\_x*, *\_y*, *\_file*, *\_vars*, *\_options* ) \_\_GraphicOutEx(*\_x*,*\_y*,*\_file*,*\_vars*,*\_options*)

Draw a graphic image with parameters and drawing options.

Draw a graphic image file on the screen at the specified x and y location using an array of parameters. Valid display option constants are listed in the [Drawing option constants](#) group. If the file cannot be found then nothing will be drawn and no errors will be reported.

**See Also**

TDrawGraphic

**Parameters**

<i>_x</i>	The x value for the position of the graphic image.
<i>_y</i>	The y value for the position of the graphic image.
<i>_file</i>	The filename of the RIC graphic image.
<i>_vars</i>	The byte array of parameters.
<i>_options</i>	The drawing options.

5.38.2.24 #define LineOut( *\_x1*, *\_y1*, *\_x2*, *\_y2* ) \_\_LineOutEx(*\_x1*,*\_y1*,*\_x2*,*\_y2*,0)

Draw a line.

This function lets you draw a line on the screen from x1, y1 to x2, y2.

**See Also**

TDrawLine

**Parameters**

<code>_x1</code>	The x value for the start of the line.
<code>_y1</code>	The y value for the start of the line.
<code>_x2</code>	The x value for the end of the line.
<code>_y2</code>	The y value for the end of the line.

**5.38.2.25 #define LineOutEx( `_x1`, `_y1`, `_x2`, `_y2`, `_options` ) \_\_LineOutEx(`x1`,`y1`,`x2`,`y2`,`options`)**

Draw a line with drawing options.

This function lets you draw a line on the screen from `x1`, `y1` to `x2`, `y2`. Also specify drawing options. Valid display option constants are listed in the [Drawing option constants](#) group.

**See Also**

[TDrawLine](#)

**Parameters**

<code>_x1</code>	The x value for the start of the line.
<code>_y1</code>	The y value for the start of the line.
<code>_x2</code>	The x value for the end of the line.
<code>_y2</code>	The y value for the end of the line.
<code>_options</code>	The optional drawing options.

**5.38.2.26 #define NumOut( `_x`, `_y`, `_num` ) \_\_NumOutEx(`x`,`y`,`num`,0)**

Draw a number.

Draw a numeric value on the screen at the specified `x` and `y` location. The `y` value must be a multiple of 8. Valid line number constants are listed in the [Line number constants](#) group.

**See Also**

[TDrawText](#)

**Parameters**

<code>_x</code>	The x value for the start of the number output.
<code>_y</code>	The text line number for the number output.
<code>_num</code>	The value to output to the LCD screen. Any numeric type is supported.

**5.38.2.27 #define NumOutEx( `_x`, `_y`, `_num`, `_options` ) \_\_NumOutEx(`x`,`y`,`num`,`options`)**

Draw a number with drawing options.

Draw a numeric value on the screen at the specified `x` and `y` location. The `y` value must be a multiple of 8. Valid line number constants are listed in the [Line number constants](#) group. Also specify drawing options. Valid display option constants are listed in the [Drawing option constants](#) group.

**See Also**

[TDrawText](#)

**Parameters**

<i>_x</i>	The x value for the start of the number output.
<i>_y</i>	The text line number for the number output.
<i>_num</i>	The value to output to the LCD screen. Any numeric type is supported.
<i>_options</i>	The optional drawing options.

**5.38.2.28 #define PointOut( *\_x*, *\_y* ) \_\_PointOutEx(*\_x*,*\_y*,0)**

Draw a point.

This function lets you draw a point on the screen at x, y.

**See Also**

[TDrawPoint](#)

**Parameters**

<i>_x</i>	The x value for the point.
<i>_y</i>	The y value for the point.

**5.38.2.29 #define PointOutEx( *\_x*, *\_y*, *\_options* ) \_\_PointOutEx(*\_x*,*\_y*,*\_options*)**

Draw a point with drawing options.

This function lets you draw a point on the screen at x, y. Also specify drawing options. Valid display option constants are listed in the [Drawing option constants](#) group.

**See Also**

[TDrawPoint](#)

**Parameters**

<i>_x</i>	The x value for the point.
<i>_y</i>	The y value for the point.
<i>_options</i>	The optional drawing options.

**5.38.2.30 #define PolyOut( *\_points* ) \_\_PolyOutEx(*\_points*,0)**

Draw a polygon.

This function lets you draw a polygon on the screen using an array of points.

**See Also**

[TDrawPolygon](#)

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<i>_points</i>	An array of LocationType points that define the polygon.
----------------	--

**5.38.2.31 #define PolyOutEx( *\_points*, *\_options* ) \_\_PolyOutEx(*\_points*,*\_options*)**

Draw a polygon with drawing options.

This function lets you draw a polygon on the screen using an array of points. Also specify drawing options. Valid display option constants are listed in the [Drawing option constants](#) group.

**See Also**

[TDrawPolygon](#)

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<i>_points</i>	An array of <a href="#">TLocation</a> points that define the polygon.
<i>_options</i>	The drawing options.

**5.38.2.32 #define RectOut( *\_x*, *\_y*, *\_w*, *\_h* ) \_\_RectOutEx(*\_x*,*\_y*,*\_w*,*\_h*,0)**

Draw a rectangle.

This function lets you draw a rectangle on the screen at x, y with the specified width and height.

**See Also**

[TDrawRect](#)

**Parameters**

<i>_x</i>	The x value for the top left corner of the rectangle.
<i>_y</i>	The y value for the top left corner of the rectangle.
<i>_w</i>	The width of the rectangle.
<i>_h</i>	The height of the rectangle.

**5.38.2.33 #define RectOutEx( *\_x*, *\_y*, *\_w*, *\_h*, *\_options* ) \_\_RectOutEx(*\_x*,*\_y*,*\_w*,*\_h*,*\_options*)**

Draw a rectangle with drawing options.

This function lets you draw a rectangle on the screen at x, y with the specified width and height. Also specify drawing options. Valid display option constants are listed in the [Drawing option constants](#) group.

**See Also**

[TDrawRect](#)

**Parameters**

<code>_x</code>	The x value for the top left corner of the rectangle.
<code>_y</code>	The y value for the top left corner of the rectangle.
<code>_w</code>	The width of the rectangle.
<code>_h</code>	The height of the rectangle.
<code>_options</code>	The optional drawing options.

**5.38.2.34 #define SetDisplayContrast( `_n` ) `_setDisplayContrast(_n)`**

Set the display contrast.

This function lets you set the display contrast setting.

**Parameters**

<code>_n</code>	The desired display contrast.
-----------------	-------------------------------

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**5.38.2.35 #define SetDisplayDisplay( `_n` ) `_setDisplayDisplay(_n)`**

Set the display memory address.

This function lets you set the current display memory address.

**Parameters**

<code>_n</code>	The new display memory address.
-----------------	---------------------------------

**5.38.2.36 #define SetDisplayEraseMask( `_n` ) `_setDisplayEraseMask(_n)`**

Set the display erase mask.

This function lets you set the current display erase mask.

**Parameters**

<code>_n</code>	The new display erase mask.
-----------------	-----------------------------

**5.38.2.37 #define SetDisplayFlags( `_n` ) `_setDisplayFlags(_n)`**

Set the display flags.

This function lets you set the current display flags.

**Parameters**

<code>_n</code>	The new display flags. See <a href="#">Display flags</a> .
-----------------	--

**5.38.2.38 #define SetDisplayFont( `_n` ) `_setDisplayFont(_n)`**

Set the display font memory address.

This function lets you set the current display font memory address.

Parameters

<code>_n</code>	The new display font memory address.
-----------------	--------------------------------------

**5.38.2.39 #define SetDisplayNormal( `_x`, `_line`, `_cnt`, `_data` ) \_\_setDisplayNormal(`_x`, `_line`, `_cnt`, `_data`)**

Write pixel data to the normal display buffer.

Write "cnt" bytes to the normal display memory from the data array. Start writing at the specified x, line coordinate. Each byte of data is a vertical strip of 8 bits at the desired location. Each bit represents a single pixel on the LCD screen. Use TEXTLINE\_1 through TEXTLINE\_8 for the "line" parameter.

Parameters

<code>_x</code>	The desired x position where you wish to write pixel data.
<code>_line</code>	The desired line where you wish to write pixel data.
<code>_cnt</code>	The number of bytes of pixel data to write.
<code>_data</code>	The array of bytes from which pixel data is read.

**5.38.2.40 #define SetDisplayPopup( `_x`, `_line`, `_cnt`, `_data` ) \_\_setDisplayPopup(`_x`, `_line`, `_cnt`, `_data`)**

Write pixel data to the popup display buffer.

Write "cnt" bytes to the popup display memory from the data array. Start writing at the specified x, line coordinate. Each byte of data is a vertical strip of 8 bits at the desired location. Each bit represents a single pixel on the LCD screen. Use TEXTLINE\_1 through TEXTLINE\_8 for the "line" parameter.

Parameters

<code>_x</code>	The desired x position where you wish to write pixel data.
<code>_line</code>	The desired line where you wish to write pixel data.
<code>_cnt</code>	The number of bytes of pixel data to write.
<code>_data</code>	The array of bytes from which pixel data is read.

**5.38.2.41 #define SetDisplayTextLinesCenterFlags( `_n` ) \_\_setDisplayTextLinesCenterFlags(`_n`)**

Set the display text lines center flags.

This function lets you set the current display text lines center flags.

Parameters

<code>_n</code>	The new display text lines center flags.
-----------------	--

**5.38.2.42 #define SetDisplayUpdateMask( `_n` ) \_\_setDisplayUpdateMask(`_n`)**

Set the display update mask.

This function lets you set the current display update mask.

Parameters

<code>_n</code>	The new display update mask.
-----------------	------------------------------

**5.38.2.43 #define TextOut( \_x, \_y, \_txt ) \_\_TextOutEx(\_x,\_y,\_txt,0)**

Draw text.

Draw a text value on the screen at the specified x and y location. The y value must be a multiple of 8. Valid line number constants are listed in the [Line number constants](#) group.

**See Also**

[TDrawText](#)

**Parameters**

<u>_x</u>	The x value for the start of the text output.
<u>_y</u>	The text line number for the text output.
<u>_txt</u>	The text to output to the LCD screen.

**5.38.2.44 #define TextOutEx( \_x, \_y, \_txt, \_options ) \_\_TextOutEx(\_x,\_y,\_txt,\_options)**

Draw text.

Draw a text value on the screen at the specified x and y location. The y value must be a multiple of 8. Valid line number constants are listed in the [Line number constants](#) group. Also specify drawing options. Valid display option constants are listed in the [Drawing option constants](#) group.

**See Also**

[TDrawText](#)

**Parameters**

<u>_x</u>	The x value for the start of the text output.
<u>_y</u>	The text line number for the text output.
<u>_txt</u>	The text to output to the LCD screen.
<u>_options</u>	The optional drawing options.

## 5.39 Sound module functions

Functions for accessing and modifying sound module features.

### Macros

- `#define PlayToneEx(_freq, _dur, _vol, _loop) __PlayToneEx(_freq,_dur,_vol,_loop)`  
*Play a tone with extra options.*
- `#define PlayTone(_freq, _dur) __PlayToneEx(_freq,_dur,4,0)`  
*Play a tone.*
- `#define PlayFile(_file) __PlayFileEx(_file,4,0,0)`  
*Play a file.*
- `#define PlayFileEx(_file, _vol, _loop, _sr) __PlayFileEx(_file,_vol,_loop,_sr)`  
*Play a file with extra options.*
- `#define GetSoundState(_state, _flags) __GetSoundState(_state, _flags)`  
*Get sound module state and flags.*
- `#define SetSoundState(_state, _flags, _result) __setSoundState(_state, _flags, _result)`  
*Set sound module state and flags.*
- `#define GetSoundFrequency(_n) __GetSoundFrequency(_n)`  
*Get sound frequency.*
- `#define GetSoundDuration(_n) __GetSoundDuration(_n)`  
*Get sound duration.*
- `#define GetSoundSampleRate(_n) __GetSoundSampleRate(_n)`  
*Get sample rate.*
- `#define GetSoundMode(_n) __GetSoundMode(_n)`  
*Get sound mode.*
- `#define GetSoundVolume(_n) __GetSoundVolume(_n)`  
*Get volume.*
- `#define SetSoundDuration(_n) __setSoundDuration(_n)`  
*Set sound duration.*
- `#define SetSoundFlags(_n) __setSoundFlags(_n)`  
*Set sound module flags.*
- `#define SetSoundFrequency(_n) __setSoundFrequency(_n)`  
*Set sound frequency.*
- `#define SetSoundMode(_n) __setSoundMode(_n)`  
*Set sound mode.*
- `#define SetSoundModuleState(_n) __setSoundModuleState(_n)`  
*Set sound module state.*
- `#define SetSoundSampleRate(_n) __setSoundSampleRate(_n)`  
*Set sample rate.*
- `#define SetSoundVolume(_n) __setSoundVolume(_n)`  
*Set sound volume.*

### 5.39.1 Detailed Description

Functions for accessing and modifying sound module features.

### 5.39.2 Macro Definition Documentation

#### 5.39.2.1 #define GetSoundDuration( \_n ) \_\_GetSoundDuration(\_n)

Get sound duration.

Return the current sound duration.

##### See Also

[SetSoundDuration](#)

##### Parameters

_n	The current sound duration.
----	-----------------------------

#### 5.39.2.2 #define GetSoundFrequency( \_n ) \_\_GetSoundFrequency(\_n)

Get sound frequency.

Return the current sound frequency.

##### See Also

[SetSoundFrequency](#)

##### Parameters

_n	The current sound frequency.
----	------------------------------

#### 5.39.2.3 #define GetSoundMode( \_n ) \_\_GetSoundMode(\_n)

Get sound mode.

Return the current sound mode. See the [SoundMode constants](#) group.

##### See Also

[SetSoundMode](#)

##### Parameters

_n	The current sound mode.
----	-------------------------

#### 5.39.2.4 #define GetSoundSampleRate( \_n ) \_\_GetSoundSampleRate(\_n)

Get sample rate.

Return the current sound sample rate.

##### See Also

[SetSoundSampleRate](#)

**Parameters**

<i>_n</i>	The current sound sample rate.
-----------	--------------------------------

5.39.2.5 #define **GetSoundState( \_state, \_flags )** \_\_GetSoundState(\_state, \_flags)

Get sound module state and flags.

Return the current sound module state and flags. See the [SoundState constants](#) group.

**See Also**

[SetSoundState](#)

**Parameters**

<i>_state</i>	The current sound module state.
<i>_flags</i>	The current sound module flags.

5.39.2.6 #define **GetSoundVolume( \_n )** \_\_GetSoundVolume(\_n)

Get volume.

Return the current sound volume.

**See Also**

[SetSoundVolume](#)

**Parameters**

<i>_n</i>	The current sound volume.
-----------	---------------------------

5.39.2.7 #define **PlayFile( \_file )** \_\_PlayFileEx(\_file,4,0,0)

Play a file.

Play the specified file. The filename may be any valid string expression. The sound file can either be an RSO file containing PCM or compressed ADPCM samples or it can be an NXT melody (RMD) file containing frequency and duration values.

**Parameters**

<i>_file</i>	The name of the sound or melody file to play.
--------------	---

5.39.2.8 #define **PlayFileEx( \_file, \_vol, \_loop, \_sr )** \_\_PlayFileEx(\_file,\_vol,\_loop,\_sr)

Play a file with extra options.

Play the specified file. The filename may be any valid string expression. Volume should be a number from 0 (silent) to 4 (loudest). Play the file repeatedly if loop is true. The sound file can either be an RSO file containing PCM or compressed ADPCM samples or it can be an NXT melody (RMD) file containing frequency and duration values.

**Parameters**

<i>_file</i>	The name of the sound or melody file to play.
<i>_vol</i>	The desired tone volume.

<i>_loop</i>	A boolean flag indicating whether to play the file repeatedly.
<i>_sr</i>	A sample rate at which to play the file.

5.39.2.9 #define PlayTone( *\_freq*, *\_dur* ) \_\_PlayToneEx(*\_freq*,*\_dur*,4,0)

Play a tone.

Play a single tone of the specified frequency and duration. The frequency is in Hz (see the [Tone constants](#) group). The duration is in 1000ths of a second (see the [Time constants](#) group). The tone is played at the loudest sound level supported by the firmware and it is not looped.

#### Parameters

<i>_freq</i>	The desired tone frequency, in Hz.
<i>_dur</i>	The desired tone duration, in ms.

5.39.2.10 #define PlayToneEx( *\_freq*, *\_dur*, *\_vol*, *\_loop* ) \_\_PlayToneEx(*\_freq*,*\_dur*,*\_vol*,*\_loop*)

Play a tone with extra options.

Play a single tone of the specified frequency, duration, and volume. The frequency is in Hz (see the [Tone constants](#) group). The duration is in 1000ths of a second (see the [Time constants](#) group). Volume should be a number from 0 (silent) to 4 (loudest). Play the tone repeatedly if loop is true.

#### Parameters

<i>_freq</i>	The desired tone frequency, in Hz.
<i>_dur</i>	The desired tone duration, in ms.
<i>_vol</i>	The desired tone volume.
<i>_loop</i>	A boolean flag indicating whether to play the tone repeatedly.

5.39.2.11 #define SetSoundDuration( *\_n* ) \_\_setSoundDuration(*\_n*)

Set sound duration.

Set the sound duration.

#### See Also

[GetSoundDuration](#)

#### Parameters

<i>_n</i>	The new sound duration
-----------	------------------------

5.39.2.12 #define SetSoundFlags( *\_n* ) \_\_setSoundFlags(*\_n*)

Set sound module flags.

Set the sound module flags. See the [SoundFlags constants](#) group.

#### Parameters

<i>_n</i>	The new sound module flags
-----------	----------------------------

**5.39.2.13 #define SetSoundFrequency( \_n ) \_\_setSoundFrequency(\_n)**

Set sound frequency.

Set the sound frequency.

**See Also**

[GetSoundFrequency](#)

**Parameters**

<code>_n</code>	The new sound frequency
-----------------	-------------------------

**5.39.2.14 #define SetSoundMode( \_n ) \_\_setSoundMode(\_n)**

Set sound mode.

Set the sound mode. See the [SoundMode constants](#) group.

**See Also**

[GetSoundMode](#)

**Parameters**

<code>_n</code>	The new sound mode
-----------------	--------------------

**5.39.2.15 #define SetSoundModuleState( \_n ) \_\_setSoundModuleState(\_n)**

Set sound module state.

Set the sound module state. See the [SoundState constants](#) group.

**See Also**

[GetSoundState](#)

**Parameters**

<code>_n</code>	The new sound state
-----------------	---------------------

**5.39.2.16 #define SetSoundSampleRate( \_n ) \_\_setSoundSampleRate(\_n)**

Set sample rate.

Set the sound sample rate.

**See Also**

[GetSoundSampleRate](#)

**Parameters**

<code>_n</code>	The new sample rate
-----------------	---------------------

**5.39.2.17 #define SetSoundState( *\_state*, *\_flags*, *\_result* ) \_\_setSoundState(*\_state*, *\_flags*, *\_result*)**

Set sound module state and flags.

Set the sound module state and flags. See the [SoundState constants](#) group.

**See Also**

[GetSoundState](#)

**Parameters**

<i>_state</i>	The sound module state.
<i>_flags</i>	The sound module flags.
<i>_result</i>	The function call result.

**5.39.2.18 #define SetSoundVolume( *\_n* ) \_\_setSoundVolume(*\_n*)**

Set sound volume.

Set the sound volume.

**See Also**

[GetSoundVolume](#)

**Parameters**

<i>_n</i>	The new volume
-----------	----------------

## 5.40 Command module functions

Functions for accessing and modifying Command module features.

### Macros

- #define `SetIOMapBytes(_modName, _offset, _cnt, _arrIn) __SetIOMapBytes(_modName, _offset, _cnt, _arrIn)`  
*Set IOMap bytes by name.*
- #define `SetIOMapValue(_modName, _offset, _n) __SetIOMapValue(_modName, _offset, _n)`  
*Set IOMap value by name.*
- #define `SetIOMapBytesByID(_modID, _offset, _cnt, _arrIn) __SetIOMapBytesByID(_modID, _offset, _cnt, _arrIn)`  
*Set IOMap bytes by ID.*
- #define `SetIOMapValueByID(_modID, _offset, _n) __SetIOMapValueByID(_modID, _offset, _n)`  
*Set IOMap value by ID.*
- #define `SetCommandModuleValue(_offset, _n) __SetIOMapValueByID(CommandModuleID, _offset, _n)`  
*Set Command module IOMap value.*
- #define `SetIOCtrlModuleValue(_offset, _n) __SetIOMapValueByID(IOCtrlModuleID, _offset, _n)`  
*Set IOCtrl module IOMap value.*
- #define `SetLoaderModuleValue(_offset, _n) __SetIOMapValueByID(LoaderModuleID, _offset, _n)`  
*Set Loader module IOMap value.*
- #define `SetUIModuleValue(_offset, _n) __SetIOMapValueByID(UIModuleID, _offset, _n)`  
*Set UI module IOMap value.*
- #define `SetSoundModuleValue(_offset, _n) __SetIOMapValueByID(SoundModuleID, _offset, _n)`  
*Set Sound module IOMap value.*
- #define `SetButtonModuleValue(_offset, _n) __SetIOMapValueByID(ButtonModuleID, _offset, _n)`  
*Set Button module IOMap value.*
- #define `SetInputModuleValue(_offset, _n) __SetIOMapValueByID(InputModuleID, _offset, _n)`  
*Set Input module IOMap value.*
- #define `SetOutputModuleValue(_offset, _n) __SetIOMapValueByID(OutputModuleID, _offset, _n)`  
*Set Output module IOMap value.*
- #define `SetLowSpeedModuleValue(_offset, _n) __SetIOMapValueByID(LowSpeedModuleID, _offset, _n)`  
*Set Lowspeed module IOMap value.*
- #define `SetDisplayModuleValue(_offset, _n) __SetIOMapValueByID(DisplayModuleID, _offset, _n)`  
*Set Display module IOMap value.*
- #define `SetCommModuleValue(_offset, _n) __SetIOMapValueByID(CommModuleID, _offset, _n)`  
*Set Comm module IOMap value.*
- #define `SetCommandModuleBytes(_offset, _cnt, _arrIn) __SetIOMapBytesByID(CommandModuleID, _offset, _cnt, _arrIn)`  
*Set Command module IOMap bytes.*
- #define `SetLowSpeedModuleBytes(_offset, _cnt, _arrIn) __SetIOMapBytesByID(LowSpeedModuleID, _offset, _cnt, _arrIn)`  
*Set Lowspeed module IOMap bytes.*
- #define `SetDisplayModuleBytes(_offset, _cnt, _arrIn) __SetIOMapBytesByID(DisplayModuleID, _offset, _cnt, _arrIn)`  
*Set Display module IOMap bytes.*
- #define `SetCommModuleBytes(_offset, _cnt, _arrIn) __SetIOMapBytesByID(CommModuleID, _offset, _cnt, _arrIn)`  
*Set Comm module IOMap bytes.*

- `#define SetSoundModuleBytes(_offset, _cnt, _arrIn) __SetIOMapBytesByID(SoundModuleID, _offset, _cnt, _arrIn)`
  - Set Comm module IOMap bytes.*
- `#define GetIOMapBytes(_modName, _offset, _cnt, _arrOut) __getIOMapBytes(_modName, _offset, _cnt, _arrOut)`
  - Set Sound module IOMap bytes.*
- `#define GetIOMapValue(_modName, _offset, _n) __getIOMapValue(_modName, _offset, _n)`
  - Get IOMap bytes by name.*
- `#define GetIOMapBytesByID(_modID, _offset, _cnt, _arrOut) __getIOMapBytesByID(_modID, _offset, _cnt, _arrOut)`
  - Get IOMap bytes by ID.*
- `#define GetIOMapValueByID(_modID, _offset, _n) __getIOMapValueByID(_modID, _offset, _n)`
  - Get IOMap value by ID.*
- `#define GetCommandModuleValue(_offset, _n) GetIOMapValueByID(CommandModuleID, _offset, _n)`
  - Get Command module IOMap value.*
- `#define GetLoaderModuleValue(_offset, _n) GetIOMapValueByID(LoaderModuleID, _offset, _n)`
  - Get Loader module IOMap value.*
- `#define GetSoundModuleValue(_offset, _n) GetIOMapValueByID(SoundModuleID, _offset, _n)`
  - Get Sound module IOMap value.*
- `#define GetButtonModuleValue(_offset, _n) GetIOMapValueByID(ButtonModuleID, _offset, _n)`
  - Get Button module IOMap value.*
- `#define GetUIModuleValue(_offset, _n) GetIOMapValueByID(UIModuleID, _offset, _n)`
  - Get Ui module IOMap value.*
- `#define GetInputModuleValue(_offset, _n) GetIOMapValueByID(InputModuleID, _offset, _n)`
  - Get Input module IOMap value.*
- `#define GetOutputModuleValue(_offset, _n) GetIOMapValueByID(OutputModuleID, _offset, _n)`
  - Get Output module IOMap value.*
- `#define GetLowSpeedModuleValue(_offset, _n) GetIOMapValueByID(LowSpeedModuleID, _offset, _n)`
  - Get LowSpeed module IOMap value.*
- `#define GetDisplayModuleValue(_offset, _n) GetIOMapValueByID(DisplayModuleID, _offset, _n)`
  - Get Display module IOMap value.*
- `#define GetCommModuleValue(_offset, _n) GetIOMapValueByID(CommModuleID, _offset, _n)`
  - Get Comm module IOMap value.*
- `#define GetLowSpeedModuleBytes(_offset, _cnt, _arrOut) __getLowSpeedModuleBytes(_offset, _cnt, _arrOut)`
  - Get Lowspeed module IOMap bytes.*
- `#define GetDisplayModuleBytes(_offset, _cnt, _arrOut) __getDisplayModuleBytes(_offset, _cnt, _arrOut)`
  - Get Display module IOMap bytes.*
- `#define GetCommModuleBytes(_offset, _cnt, _arrOut) __getCommModuleBytes(_offset, _cnt, _arrOut)`
  - Get Comm module IOMap bytes.*
- `#define GetCommandModuleBytes(_offset, _cnt, _arrOut) __getCommandModuleBytes(_offset, _cnt, _arrOut)`
  - Get Command module IOMap bytes.*
- `#define ResetSleepTimer syscall KeepAlive, __KeepAliveArgs`
  - Reset the sleep timer.*
- `#define GetFirstTick(_value) __GetFirstTick(_value)`
  - Get the first tick.*
- `#define Wait(_n) waitv _n`
  - Wait for \_n ticks.*

- *Wait some milliseconds.*
- `#define GetMemoryInfo(_Compact, _PoolSize, _DataspaceSize, _Result) __GetMemoryInfo(_Compact,_PoolSize,_DataspaceSize,_Result)`  
*Read memory information.*
- `#define GetLastResponseInfo(_Clear, _Length, _Command, _Buffer, _Result) __GetLastResponseInfo(_Clear,_Length,_Command,_Buffer,_Result)`  
*Read last response information.*

#### 5.40.1 Detailed Description

Functions for accessing and modifying Command module features.

#### 5.40.2 Macro Definition Documentation

##### 5.40.2.1 `#define GetButtonModuleValue( _offset, _n ) GetIOMapValueByID(ButtonModuleID, _offset, _n)`

Get Button module IOMap value.

Read a value from the Button module IOMap structure. You provide the offset into the Button module IOMap structure where you want to read the value from along with a variable that will store the value. The type of the variable determines how many bytes are read from the IOMap.

##### Parameters

<code>_offset</code>	The number of bytes offset from the start of the IOMap structure where the value should be read. See <a href="#">Button module IOMAP offsets</a> .
<code>_n</code>	A variable that will contain the value read from the IOMap.

##### 5.40.2.2 `#define GetCommandModuleBytes( _offset, _cnt, _arrOut ) __getCommandModuleBytes(_offset, _cnt, _arrOut)`

Get Command module IOMap bytes.

Read one or more bytes of data from Command module IOMap structure. You provide the offset into the Command module IOMap structure where you want to start reading, the number of bytes to read from that location, and a byte array where the data will be stored.

##### Parameters

<code>_offset</code>	The number of bytes offset from the start of the Command module IOMap structure where the data should be read. See <a href="#">Command module IOMAP offsets</a> .
<code>_cnt</code>	The number of bytes to read from the specified Command module IOMap offset.
<code>_arrOut</code>	A byte array that will contain the data read from the Command module IOMap.

##### 5.40.2.3 `#define GetCommandModuleValue( _offset, _n ) GetIOMapValueByID(CommandModuleID, _offset, _n)`

Get Command module IOMap value.

Read a value from the Command module IOMap structure. You provide the offset into the Command module IOMap structure where you want to read the value from along with a variable that will store the value. The type of the variable determines how many bytes are read from the IOMap.

**Parameters**

<code>_offset</code>	The number of bytes offset from the start of the IOMap structure where the value should be read. See <a href="#">Command module IOMAP offsets</a> .
<code>_n</code>	A variable that will contain the value read from the IOMap.

5.40.2.4 #define GetCommModuleBytes( `_offset`, `_cnt`, `_arrOut` ) `_getCommModuleBytes(_offset, _cnt, _arrOut)`

Get Comm module IOMap bytes.

Read one or more bytes of data from Comm module IOMap structure. You provide the offset into the Comm module IOMap structure where you want to start reading, the number of bytes to read from that location, and a byte array where the data will be stored.

**Parameters**

<code>_offset</code>	The number of bytes offset from the start of the Comm module IOMap structure where the data should be read. See <a href="#">Comm module IOMAP offsets</a> .
<code>_cnt</code>	The number of bytes to read from the specified Comm module IOMap offset.
<code>_arrOut</code>	A byte array that will contain the data read from the Comm module IOMap.

5.40.2.5 #define GetCommModuleValue( `_offset`, `_n` ) `GetIOMapValueByID(CommModuleID, _offset, _n)`

Get Comm module IOMap value.

Read a value from the Comm module IOMap structure. You provide the offset into the Comm module IOMap structure where you want to read the value from along with a variable that will store the value. The type of the variable determines how many bytes are read from the IOMap.

**Parameters**

<code>_offset</code>	The number of bytes offset from the start of the IOMap structure where the value should be read. See <a href="#">Comm module IOMAP offsets</a> .
<code>_n</code>	A variable that will contain the value read from the IOMap.

5.40.2.6 #define GetDisplayModuleBytes( `_offset`, `_cnt`, `_arrOut` ) `_getDisplayModuleBytes(_offset, _cnt, _arrOut)`

Get Display module IOMap bytes.

Read one or more bytes of data from Display module IOMap structure. You provide the offset into the Display module IOMap structure where you want to start reading, the number of bytes to read from that location, and a byte array where the data will be stored.

**Parameters**

<code>_offset</code>	The number of bytes offset from the start of the Display module IOMap structure where the data should be read. See <a href="#">Display module IOMAP offsets</a> .
<code>_cnt</code>	The number of bytes to read from the specified Display module IOMap offset.
<code>_arrOut</code>	A byte array that will contain the data read from the Display module IOMap.

5.40.2.7 #define GetDisplayModuleValue( `_offset`, `_n` ) `GetIOMapValueByID(DisplayModuleID, _offset, _n)`

Get Display module IOMap value.

Read a value from the Display module IOMap structure. You provide the offset into the Display module IOMap structure where you want to read the value from along with a variable that will store the value. The type of the variable determines

how many bytes are read from the IOMap.

#### Parameters

<code>_offset</code>	The number of bytes offset from the start of the IOMap structure where the value should be read. See <a href="#">Display module IOMAP offsets</a> .
<code>_n</code>	A variable that will contain the value read from the IOMap.

#### 5.40.2.8 #define GetFirstTick( `_value` ) `_GetFirstTick(_value)`

Get the first tick.

Return an unsigned 32-bit value, which is the system timing value (called a "tick") in milliseconds at the time that the program began running.

#### Parameters

<code>_value</code>	The tick count at the start of program execution.
---------------------	---

#### 5.40.2.9 #define GetInputModuleValue( `_offset`, `_n` ) `_GetIOMapValueByID(InputModuleID, _offset, _n)`

Get Input module IOMap value.

Read a value from the Input module IOMap structure. You provide the offset into the Input module IOMap structure where you want to read the value from along with a variable that will store the value. The type of the variable determines how many bytes are read from the IOMap.

#### Parameters

<code>_offset</code>	The number of bytes offset from the start of the IOMap structure where the value should be read. See <a href="#">Input module IOMAP offsets</a> .
<code>_n</code>	A variable that will contain the value read from the IOMap.

#### 5.40.2.10 #define GetIOMapBytes( `_modName`, `_offset`, `_cnt`, `_arrOut` ) `_getIOMapBytes(_modName, _offset, _cnt, _arrOut)`

Get IOMap bytes by name.

Read one or more bytes of data from an IOMap structure. The IOMap structure is specified by its module name. You also provide the offset into the IOMap structure where you want to start reading, the number of bytes to read from that location, and a byte array where the data will be stored.

#### Parameters

<code>_modName</code>	The module name of the IOMap. See <a href="#">NXT firmware module names</a> .
<code>_offset</code>	The number of bytes offset from the start of the IOMap structure where the data should be read
<code>_cnt</code>	The number of bytes to read from the specified IOMap offset.
<code>_arrOut</code>	A byte array that will contain the data read from the IOMap

#### 5.40.2.11 #define GetIOMapBytesByID( `_modID`, `_offset`, `_cnt`, `_arrOut` ) `_getIOMapBytesByID(_modID, _offset, _cnt, _arrOut)`

Get IOMap bytes by ID.

Read one or more bytes of data from an IOMap structure. The IOMap structure is specified by its Module ID. You also provide the offset into the IOMap structure where you want to start reading, the number of bytes to read from that location, and a byte array where the data will be stored.

**Parameters**

<i>_modID</i>	The module ID of the IOMap. See <a href="#">NXT firmware module IDs</a> .
<i>_offset</i>	The number of bytes offset from the start of the IOMap structure where the data should be read.
<i>_cnt</i>	The number of bytes to read from the specified IOMap offset.
<i>_arrOut</i>	A byte array that will contain the data read from the IOMap.

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.40.2.12 #define GetIOMapValue( *\_modName*, *\_offset*, *\_n* ) \_\_getIOMapValue(*\_modName*, *\_offset*, *\_n*)**

Get IOMap value by name.

Read a value from an IOMap structure. The IOMap structure is specified by its module name. You also provide the offset into the IOMap structure where you want to read the value along with a variable that will contain the IOMap value.

**Parameters**

<i>_modName</i>	The module name of the IOMap. See <a href="#">NXT firmware module names</a> .
<i>_offset</i>	The number of bytes offset from the start of the IOMap structure where the value should be read
<i>_n</i>	A variable that will contain the value read from the IOMap

**5.40.2.13 #define GetIOMapValueByID( *\_modID*, *\_offset*, *\_n* ) \_\_getIOMapValueByID(*\_modID*, *\_offset*, *\_n*)**

Get IOMap value by ID.

Read a value from an IOMap structure. The IOMap structure is specified by its Module ID. You also provide the offset into the IOMap structure where you want to read the value along with a variable that will contain the IOMap value.

**Parameters**

<i>_modID</i>	The module ID of the IOMap. See <a href="#">NXT firmware module IDs</a> .
<i>_offset</i>	The number of bytes offset from the start of the IOMap structure where the value should be read
<i>_n</i>	A variable that will contain the value read from the IOMap

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.40.2.14 #define GetLastResponseInfo( *\_Clear*, *\_Length*, *\_Command*, *\_Buffer*, *\_Result* ) \_\_GetLastResponseInfo(*\_Clear*,*\_Length*,*\_Command*,*\_Buffer*,*\_Result*)**

Read last response information.

Read the last direct or system command response packet received by the NXT. Optionally clear the response after retrieving the information.

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.31+.

**Parameters**

<i>_Clear</i>	A boolean value indicating whether to clear the response or not.
<i>_Length</i>	The response packet length.
<i>_Command</i>	The original command byte.
<i>_Buffer</i>	The response packet buffer.
<i>_Result</i>	The response status code.

#### 5.40.2.15 #define GetLoaderModuleValue( *\_offset*, *\_n* ) GetIOMapValueByID(LoaderModuleID, *\_offset*, *\_n*)

Get Loader module IOMap value.

Read a value from the Loader module IOMap structure. You provide the offset into the Loader module IOMap structure where you want to read the value from along with a variable that will store the value. The type of the variable determines how many bytes are read from the IOMap.

##### Parameters

<i>_offset</i>	The number of bytes offset from the start of the IOMap structure where the value should be read. See <a href="#">Loader module IOMAP offsets</a> .
<i>_n</i>	A variable that will contain the value read from the IOMap.

#### 5.40.2.16 #define GetLowSpeedModuleBytes( *\_offset*, *\_cnt*, *\_arrOut* ) \_\_getLowSpeedModuleBytes(*\_offset*, *\_cnt*, *\_arrOut*)

Get Lowspeed module IOMap bytes.

Read one or more bytes of data from Lowspeed module IOMap structure. You provide the offset into the Lowspeed module IOMap structure where you want to start reading, the number of bytes to read from that location, and a byte array where the data will be stored.

##### Parameters

<i>_offset</i>	The number of bytes offset from the start of the Lowspeed module IOMap structure where the data should be read. See <a href="#">Low speed module IOMAP offsets</a> .
<i>_cnt</i>	The number of bytes to read from the specified Lowspeed module IOMap offset.
<i>_arrOut</i>	A byte array that will contain the data read from the Lowspeed module IOMap.

#### 5.40.2.17 #define GetLowSpeedModuleValue( *\_offset*, *\_n* ) GetIOMapValueByID(LowSpeedModuleID, *\_offset*, *\_n*)

Get LowSpeed module IOMap value.

Read a value from the LowSpeed module IOMap structure. You provide the offset into the Command module IOMap structure where you want to read the value from along with a variable that will store the value. The type of the variable determines how many bytes are read from the IOMap.

##### Parameters

<i>_offset</i>	The number of bytes offset from the start of the IOMap structure where the value should be read. See <a href="#">Low speed module IOMAP offsets</a> .
<i>_n</i>	A variable that will contain the value read from the IOMap.

#### 5.40.2.18 #define GetMemoryInfo( *\_Compact*, *\_PoolSize*, *\_DataspaceSize*, *\_Result* ) \_\_GetMemoryInfo(*\_Compact*, *\_PoolSize*, *\_DataspaceSize*, *\_Result*)

Read memory information.

Read the current pool size and dataspace size. Optionally compact the dataspace before returning the information. Running programs have a maximum of 32k bytes of memory available. The amount of free RAM can be calculated by subtracting the value returned by this function from [POOL\\_MAX\\_SIZE](#).

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<i>_Compact</i>	A boolean value indicating whether to compact the dataspace or not.
<i>_PoolSize</i>	The current pool size.
<i>_DataspaceSize</i>	The current dataspace size.
<i>_Result</i>	The function call result. It will be <a href="#">NO_ERR</a> if the compact operation is not performed. Otherwise it will be the result of the compact operation.

#### 5.40.2.19 #define GetOutputModuleValue( *\_offset*, *\_n* ) GetIOMapValueByID(OutputModuleID, *\_offset*, *\_n*)

Get Output module IOMap value.

Read a value from the Output module IOMap structure. You provide the offset into the Output module IOMap structure where you want to read the value from along with a variable that will store the value. The type of the variable determines how many bytes are read from the IOMap.

#### Parameters

<i>_offset</i>	The number of bytes offset from the start of the IOMap structure where the value should be read. See <a href="#">Output module IOMAP offsets</a> .
<i>_n</i>	A variable that will contain the value read from the IOMap.

#### 5.40.2.20 #define GetSoundModuleValue( *\_offset*, *\_n* ) GetIOMapValueByID(SoundModuleID, *\_offset*, *\_n*)

Get Sound module IOMap value.

Read a value from the Sound module IOMap structure. You provide the offset into the Sound module IOMap structure where you want to read the value from along with a variable that will store the value. The type of the variable determines how many bytes are read from the IOMap.

#### Parameters

<i>_offset</i>	The number of bytes offset from the start of the IOMap structure where the value should be read. See <a href="#">Sound module IOMAP offsets</a> .
<i>_n</i>	A variable that will contain the value read from the IOMap.

#### 5.40.2.21 #define GetUIModuleValue( *\_offset*, *\_n* ) GetIOMapValueByID(UIModuleID, *\_offset*, *\_n*)

Get Ui module IOMap value.

Read a value from the Ui module IOMap structure. You provide the offset into the Ui module IOMap structure where you want to read the value from along with a variable that will store the value. The type of the variable determines how many bytes are read from the IOMap.

**Parameters**

<code>_offset</code>	The number of bytes offset from the start of the IOMap structure where the value should be read. See <a href="#">Ui module IOMAP offsets</a> .
<code>_n</code>	A variable that will contain the value read from the IOMap.

**5.40.2.22 #define ResetSleepTimer syscall KeepAlive, \_\_KeepAliveArgs**

Reset the sleep timer.

This function lets you reset the sleep timer.

**5.40.2.23 #define SetButtonModuleValue( \_offset, \_n ) \_\_SetIOMapValueByID(ButtonModuleID, \_offset, \_n)**

Set Button module IOMap value.

Set one of the fields of the Button module IOMap structure to a new value. You provide the offset into the Button module IOMap structure where you want to write the value along with a variable containing the new value.

**Parameters**

<code>_offset</code>	The number of bytes offset from the start of the Button module IOMap structure where the new value should be written. See <a href="#">Button module IOMAP offsets</a> .
<code>_n</code>	A variable containing the new value to write to the Button module IOMap.

**5.40.2.24 #define SetCommandModuleBytes( \_offset, \_cnt, \_arrIn ) \_\_SetIOMapBytesByID(CommandModuleID, \_offset, \_cnt, \_arrIn)**

Set Command module IOMap bytes.

Modify one or more bytes of data in the Command module IOMap structure. You provide the offset into the Command module IOMap structure where you want to start writing, the number of bytes to write at that location, and a byte array containing the new data.

**Parameters**

<code>_offset</code>	The number of bytes offset from the start of the Command module IOMap structure where the data should be written. See <a href="#">Command module IOMAP offsets</a> .
<code>_cnt</code>	The number of bytes to write at the specified Command module IOMap offset.
<code>_arrIn</code>	The byte array containing the data to write to the Command module IOMap.

**5.40.2.25 #define SetCommandModuleValue( \_offset, \_n ) \_\_SetIOMapValueByID(CommandModuleID, \_offset, \_n)**

Set Command module IOMap value.

Set one of the fields of the Command module IOMap structure to a new value. You provide the offset into the Command module IOMap structure where you want to write the value along with a variable containing the new value.

**Parameters**

<code>_offset</code>	The number of bytes offset from the start of the Command module IOMap structure where the new value should be written. See <a href="#">Command module IOMAP offsets</a> .
<code>_n</code>	A variable containing the new value to write to the Command module IOMap.

**5.40.2.26 #define SetCommModuleBytes( \_offset, \_cnt, \_arrIn ) \_\_SetIOMapBytesByID(CommModuleID, \_offset, \_cnt, \_arrIn)**

Set Comm module IOMap bytes.

Modify one or more bytes of data in an IOMap structure. You provide the offset into the Comm module IOMap structure where you want to start writing, the number of bytes to write at that location, and a byte array containing the new data.

**Parameters**

<i>_offset</i>	The number of bytes offset from the start of the Comm module IOMap structure where the data should be written. See <a href="#">Comm module IOMAP offsets</a> .
<i>_cnt</i>	The number of bytes to write at the specified Comm module IOMap offset.
<i>_arrIn</i>	The byte array containing the data to write to the Comm module IOMap.

**5.40.2.27 #define SetCommModuleValue( \_offset, \_n ) \_\_SetIOMapValueByID(CommModuleID, \_offset, \_n)**

Set Comm module IOMap value.

Set one of the fields of the Comm module IOMap structure to a new value. You provide the offset into the Comm module IOMap structure where you want to write the value along with a variable containing the new value.

**Parameters**

<i>_offset</i>	The number of bytes offset from the start of the Comm module IOMap structure where the new value should be written. See <a href="#">Comm module IOMAP offsets</a> .
<i>_n</i>	A variable containing the new value to write to the Comm module IOMap.

**5.40.2.28 #define SetDisplayModuleBytes( \_offset, \_cnt, \_arrIn ) \_\_SetIOMapBytesByID(DisplayModuleID, \_offset, \_cnt, \_arrIn)**

Set Display module IOMap bytes.

Modify one or more bytes of data in the Display module IOMap structure. You provide the offset into the Display module IOMap structure where you want to start writing, the number of bytes to write at that location, and a byte array containing the new data.

**Parameters**

<i>_offset</i>	The number of bytes offset from the start of the Display module IOMap structure where the data should be written. See <a href="#">Display module IOMAP offsets</a> .
<i>_cnt</i>	The number of bytes to write at the specified Display module IOMap offset.
<i>_arrIn</i>	The byte array containing the data to write to the Display module IOMap.

**5.40.2.29 #define SetDisplayModuleValue( \_offset, \_n ) \_\_SetIOMapValueByID(DisplayModuleID, \_offset, \_n)**

Set Display module IOMap value.

Set one of the fields of the Display module IOMap structure to a new value. You provide the offset into the Display module IOMap structure where you want to write the value along with a variable containing the new value.

**Parameters**

<i>_offset</i>	The number of bytes offset from the start of the Display module IOMap structure where the new value should be written. See <a href="#">Display module IOMAP offsets</a> .
<i>_n</i>	A variable containing the new value to write to the Display module IOMap.

**5.40.2.30 #define SetInputModuleValue( \_offset, \_n ) \_\_SetIOMapValueByID(InputModuleID, \_offset, \_n)**

Set Input module IOMap value.

Set one of the fields of the Input module IOMap structure to a new value. You provide the offset into the Input module IOMap structure where you want to write the value along with a variable containing the new value.

**Parameters**

<i>_offset</i>	The number of bytes offset from the start of the Input module IOMap structure where the new value should be written. See <a href="#">Input module IOMAP offsets</a> .
<i>_n</i>	A variable containing the new value to write to the Input module IOMap.

**5.40.2.31 #define SetIOCtrlModuleValue( \_offset, \_n ) \_\_SetIOMapValueByID(IOCtrlModuleID, \_offset, \_n)**

Set IOCtrl module IOMap value.

Set one of the fields of the IOCtrl module IOMap structure to a new value. You provide the offset into the IOCtrl module IOMap structure where you want to write the value along with a variable containing the new value.

**Parameters**

<i>_offset</i>	The number of bytes offset from the start of the IOCtrl module IOMap structure where the new value should be written. See <a href="#">IOCtrl module IOMAP offsets</a> .
<i>_n</i>	A variable containing the new value to write to the IOCtrl module IOMap.

**5.40.2.32 #define SetIOMapBytes( \_modName, \_offset, \_cnt, \_arrIn ) \_\_SetIOMapBytes(\_modName, \_offset, \_cnt, \_arrIn)**

Set IOMap bytes by name.

Modify one or more bytes of data in an IOMap structure. The IOMap structure is specified by its module name. You also provide the offset into the IOMap structure where you want to start writing, the number of bytes to write at that location, and a byte array containing the new data.

**Parameters**

<i>_modName</i>	The module name of the IOMap to modify. See <a href="#">NXT firmware module names</a> .
<i>_offset</i>	The number of bytes offset from the start of the IOMap structure where the data should be written
<i>_cnt</i>	The number of bytes to write at the specified IOMap offset.
<i>_arrIn</i>	The byte array containing the data to write to the IOMap

**5.40.2.33 #define SetIOMapBytesByID( \_modID, \_offset, \_cnt, \_arrIn ) \_\_SetIOMapBytesByID(\_modID, \_offset, \_cnt, \_arrIn)**

Set IOMap bytes by ID.

Modify one or more bytes of data in an IOMap structure. The IOMap structure is specified by its Module ID. You also provide the offset into the IOMap structure where you want to start writing, the number of bytes to write at that location, and a byte array containing the new data.

**Parameters**

<i>_modID</i>	The module ID of the IOMap to modify. See <a href="#">NXT firmware module IDs</a> .
<i>_offset</i>	The number of bytes offset from the start of the IOMap structure where the data should be written
<i>_cnt</i>	The number of bytes to write at the specified IOMap offset.
<i>_arrIn</i>	The byte array containing the data to write to the IOMap.

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.40.2.34 #define SetIOMapValue( \_modName, \_offset, \_n ) \_\_SetIOMapValue(\_modName, \_offset, \_n)**

Set IOMap value by name.

Set one of the fields of an IOMap structure to a new value. The IOMap structure is specified by its module name. You also provide the offset into the IOMap structure where you want to write the value along with a variable containing the new value.

**Parameters**

<i>_modName</i>	The module name of the IOMap to modify. See <a href="#">NXT firmware module names</a> .
<i>_offset</i>	The number of bytes offset from the start of the IOMap structure where the new value should be written
<i>_n</i>	A variable containing the new value to write to the IOMap

**5.40.2.35 #define SetIOMapValueByID( \_modID, \_offset, \_n ) \_\_SetIOMapValueByID(\_modID, \_offset, \_n)**

Set IOMap value by ID.

Set one of the fields of an IOMap structure to a new value. The IOMap structure is specified by its Module ID. You also provide the offset into the IOMap structure where you want to write the value along with a variable containing the new value.

**Parameters**

<i>_modID</i>	The module ID of the IOMap to modify. See <a href="#">NXT firmware module IDs</a> .
<i>_offset</i>	The number of bytes offset from the start of the IOMap structure where the new value should be written.
<i>_n</i>	A variable containing the new value to write to the IOMap.

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.40.2.36 #define SetLoaderModuleValue( \_offset, \_n ) \_\_SetIOMapValueByID(LoaderModuleID, \_offset, \_n)**

Set Loader module IOMap value.

Set one of the fields of the Loader module IOMap structure to a new value. You provide the offset into the Loader module IOMap structure where you want to write the value along with a variable containing the new value.

**Parameters**

<i>_offset</i>	The number of bytes offset from the start of the Loader module IOMap structure where the new value should be written. See <a href="#">Loader module IOMAP offsets</a> .
<i>_n</i>	A variable containing the new value to write to the Loader module IOMap.

**5.40.2.37 #define SetLowSpeedModuleBytes( \_offset, \_cnt, \_arrIn ) \_\_SetIOMapBytesByID(LowSpeedModuleID, \_offset, \_cnt, \_arrIn)**

Set Lowspeed module IOMap bytes.

Modify one or more bytes of data in the Lowspeed module IOMap structure. You provide the offset into the Lowspeed module IOMap structure where you want to start writing, the number of bytes to write at that location, and a byte array containing the new data.

#### Parameters

<code>_offset</code>	The number of bytes offset from the start of the Lowspeed module IOMap structure where the data should be written. See <a href="#">Low speed module IOMAP offsets</a> .
<code>_cnt</code>	The number of bytes to write at the specified Lowspeed module IOMap offset.
<code>_arrIn</code>	The byte array containing the data to write to the Lowspeed module IOMap.

#### 5.40.2.38 #define SetLowSpeedModuleValue( `_offset`, `_n` ) \_\_SetIOMapValueByID(`LowSpeedModuleID`, `_offset`, `_n`)

Set Lowspeed module IOMap value.

Set one of the fields of the Lowspeed module IOMap structure to a new value. You provide the offset into the Lowspeed module IOMap structure where you want to write the value along with a variable containing the new value.

#### Parameters

<code>_offset</code>	The number of bytes offset from the start of the Lowspeed module IOMap structure where the new value should be written. See <a href="#">Low speed module IOMAP offsets</a> .
<code>_n</code>	A variable containing the new value to write to the Lowspeed module IOMap.

#### 5.40.2.39 #define SetOutputModuleValue( `_offset`, `_n` ) \_\_SetIOMapValueByID(`OutputModuleID`, `_offset`, `_n`)

Set Output module IOMap value.

Set one of the fields of the Output module IOMap structure to a new value. You provide the offset into the Output module IOMap structure where you want to write the value along with a variable containing the new value.

#### Parameters

<code>_offset</code>	The number of bytes offset from the start of the Output module IOMap structure where the new value should be written. See <a href="#">Output module IOMAP offsets</a> .
<code>_n</code>	A variable containing the new value to write to the Output module IOMap.

#### 5.40.2.40 #define SetSoundModuleBytes( `_offset`, `_cnt`, `_arrIn` ) \_\_SetIOMapBytesByID(`SoundModuleID`, `_offset`, `_cnt`, `_arrIn`)

Set Sound module IOMap bytes.

Set one or more bytes of data in an IOMap structure. You provide the offset into the Sound module IOMap structure where you want to start writing, the number of bytes to write at that location, and a byte array containing the new data.

#### Parameters

<code>_offset</code>	The number of bytes offset from the start of the Sound module IOMap structure where the data should be written. See <a href="#">Sound module IOMAP offsets</a> .
<code>_cnt</code>	The number of bytes to write at the specified Sound module IOMap offset.
<code>_arrIn</code>	The byte array containing the data to write to the Sound module IOMap.

#### 5.40.2.41 #define SetSoundModuleValue( `_offset`, `_n` ) \_\_SetIOMapValueByID(`SoundModuleID`, `_offset`, `_n`)

Set Sound module IOMap value.

Set one of the fields of the Sound module IOMap structure to a new value. You provide the offset into the Sound module

IOMap structure where you want to write the value along with a variable containing the new value.

**Parameters**

<code>_offset</code>	The number of bytes offset from the start of the Sound module IOMap structure where the new value should be written. See <a href="#">Sound module IOMAP offsets</a> .
<code>_n</code>	A variable containing the new value to write to the Sound module IOMap.

**5.40.2.42 #define SetUIModuleValue( `_offset`, `_n` ) \_\_SetIOMapValueByID(UIModuleID, `_offset`, `_n`)**

Set Ui module IOMap value.

Set one of the fields of the Ui module IOMap structure to a new value. You provide the offset into the Ui module IOMap structure where you want to write the value along with a variable containing the new value.

**Parameters**

<code>_offset</code>	The number of bytes offset from the start of the Ui module IOMap structure where the new value should be written. See <a href="#">Ui module IOMAP offsets</a> .
<code>_n</code>	A variable containing the new value to write to the Ui module IOMap.

**5.40.2.43 #define Wait( `_n` ) waitv `_n`**

Wait some milliseconds.

Make a task sleep for specified amount of time (in 1000ths of a second).

**Parameters**

<code>_n</code>	The number of milliseconds to sleep.
-----------------	--------------------------------------

## 5.41 Button module functions

Functions for accessing and modifying Button module features.

### Macros

- `#define ReadButtonEx(_idx, _reset, _pressed, _count, _result) __ReadButtonEx(_idx, _reset, _pressed, _count, _result)`  
*Read button information.*
- `#define GetButtonPressCount(_b, _n) __GetButtonPressCount(_b, _n)`  
*Get button press count.*
- `#define GetButtonLongPressCount(_b, _n) __GetButtonLongPressCount(_b, _n)`  
*Get button long press count.*
- `#define GetButtonShortReleaseCount(_b, _n) __GetButtonShortReleaseCount(_b, _n)`  
*Get button short release count.*
- `#define GetButtonLongReleaseCount(_b, _n) __GetButtonLongReleaseCount(_b, _n)`  
*Get button long release count.*
- `#define GetButtonReleaseCount(_b, _n) __GetButtonReleaseCount(_b, _n)`  
*Get button release count.*
- `#define GetButtonState(_b, _n) __GetButtonState(_b, _n)`  
*Get button state.*
- `#define SetButtonPressCount(_b, _n) __setButtonPressCount(_b, _n)`  
*Set button press count.*
- `#define SetButtonLongPressCount(_b, _n) __setButtonLongPressCount(_b, _n)`  
*Set button long press count.*
- `#define SetButtonShortReleaseCount(_b, _n) __setButtonShortReleaseCount(_b, _n)`  
*Set button short release count.*
- `#define SetButtonLongReleaseCount(_b, _n) __setButtonLongReleaseCount(_b, _n)`  
*Set button long release count.*
- `#define SetButtonReleaseCount(_b, _n) __setButtonReleaseCount(_b, _n)`  
*Set button release count.*
- `#define SetButtonState(_b, _n) __setButtonState(_b, _n)`  
*Set button state.*

### 5.41.1 Detailed Description

Functions for accessing and modifying Button module features.

### 5.41.2 Macro Definition Documentation

#### 5.41.2.1 `#define GetButtonLongPressCount( _b, _n ) __GetButtonLongPressCount(_b, _n)`

Get button long press count.

Return the long press count of the specified button.

#### Parameters

<code>_b</code>	The button to check. See <a href="#">Button name constants</a> .
<code>_n</code>	The button long press count.

**5.41.2.2 #define GetButtonLongReleaseCount( \_b, \_n ) \_\_GetButtonLongReleaseCount(\_b, \_n)**

Get button long release count.

Return the long release count of the specified button.

**Parameters**

<u>_b</u>	The button to check. See <a href="#">Button name constants</a> .
<u>_n</u>	The button long release count.

**5.41.2.3 #define GetButtonPressCount( \_b, \_n ) \_\_GetButtonPressCount(\_b, \_n)**

Get button press count.

Return the press count of the specified button.

**Parameters**

<u>_b</u>	The button to check. See <a href="#">Button name constants</a> .
<u>_n</u>	The button press count.

**5.41.2.4 #define GetButtonReleaseCount( \_b, \_n ) \_\_GetButtonReleaseCount(\_b, \_n)**

Get button release count.

Return the release count of the specified button.

**Parameters**

<u>_b</u>	The button to check. See <a href="#">Button name constants</a> .
<u>_n</u>	The button release count.

**5.41.2.5 #define GetButtonShortReleaseCount( \_b, \_n ) \_\_GetButtonShortReleaseCount(\_b, \_n)**

Get button short release count.

Return the short release count of the specified button.

**Parameters**

<u>_b</u>	The button to check. See <a href="#">Button name constants</a> .
<u>_n</u>	The button short release count.

**5.41.2.6 #define GetButtonState( \_b, \_n ) \_\_GetButtonState(\_b, \_n)**

Get button state.

Return the state of the specified button. See [ButtonState constants](#).

**Parameters**

<u>_b</u>	The button to check. See <a href="#">Button name constants</a> .
<u>_n</u>	The button state.

**5.41.2.7 #define ReadButtonEx( \_idx, \_reset, \_pressed, \_count, \_result ) \_\_ReadButtonEx(\_idx, \_reset, \_pressed, \_count, \_result)**

Read button information.

Read the specified button. Set the pressed and count parameters with the current state of the button. Optionally reset the press count after reading it.

**Parameters**

<code>_idx</code>	The button to check. See <a href="#">Button name constants</a> .
<code>_reset</code>	Whether or not to reset the press counter.
<code>_pressed</code>	The button pressed state.
<code>_count</code>	The button press count.
<code>_result</code>	The function call result.

**5.41.2.8 #define SetButtonLongPressCount( \_b, \_n ) \_\_setButtonLongPressCount(\_b, \_n)**

Set button long press count.

Set the long press count of the specified button.

**Parameters**

<code>_b</code>	The button number. See <a href="#">Button name constants</a> .
<code>_n</code>	The new long press count value.

**5.41.2.9 #define SetButtonLongReleaseCount( \_b, \_n ) \_\_setButtonLongReleaseCount(\_b, \_n)**

Set button long release count.

Set the long release count of the specified button.

**Parameters**

<code>_b</code>	The button number. See <a href="#">Button name constants</a> .
<code>_n</code>	The new long release count value.

**5.41.2.10 #define SetButtonPressCount( \_b, \_n ) \_\_setButtonPressCount(\_b, \_n)**

Set button press count.

Set the press count of the specified button.

**Parameters**

<code>_b</code>	The button number. See <a href="#">Button name constants</a> .
<code>_n</code>	The new press count value.

**5.41.2.11 #define SetButtonReleaseCount( \_b, \_n ) \_\_setButtonReleaseCount(\_b, \_n)**

Set button release count.

Set the release count of the specified button.

**Parameters**

<code>_b</code>	The button number. See <a href="#">Button name constants</a> .
<code>_n</code>	The new release count value.

**5.41.2.12 #define SetButtonShortReleaseCount( `_b`, `_n` ) `_setButtonShortReleaseCount(_b, _n)`**

Set button short release count.

Set the short release count of the specified button.

**Parameters**

<code>_b</code>	The button number. See <a href="#">Button name constants</a> .
<code>_n</code>	The new short release count value.

**5.41.2.13 #define SetButtonState( `_b`, `_n` ) `_setButtonState(_b, _n)`**

Set button state.

Set the state of the specified button.

**Parameters**

<code>_b</code>	The button to check. See <a href="#">Button name constants</a> .
<code>_n</code>	The new button state. See <a href="#">ButtonState constants</a> .

## 5.42 Ui module functions

Functions for accessing and modifying Ui module features.

### Macros

- `#define SetCommandFlags(_n) __setCommandFlags(_n)`  
*Set command flags.*
- `#define SetUIState(_n) __setUIState(_n)`  
*Set UI state.*
- `#define SetUIButton(_n) __setUIButton(_n)`  
*Set UI button.*
- `#define SetVMRunState(_n) __setVMRunState(_n)`  
*Set VM run state.*
- `#define SetBatteryState(_n) __setBatteryState(_n)`  
*Set battery state.*
- `#define SetBluetoothState(_n) __setBluetoothState(_n)`  
*Set bluetooth state.*
- `#define SetUsbState(_n) __setUsbState(_n)`  
*Set Usb state.*
- `#define SetSleepTimeout(_n) __setSleepTimeout(_n)`  
*Set sleep timeout.*
- `#define SetSleepTimer(_n) __setSleepTimer(_n)`  
*Set the sleep timer.*
- `#define SetVolume(_n) __setVolume(_n)`  
*Set volume.*
- `#define SetOnBrickProgramPointer(_n) __setOnBrickProgramPointer(_n)`  
*Set on-brick program pointer.*
- `#define ForceOff(_n) __forceOff(_n)`  
*Turn off NXT.*
- `#define SetAbortFlag(_n) __setAbortFlag(_n)`  
*Set abort flag.*
- `#define GetBatteryLevel(_n) __GetBatteryLevel(_n)`  
*Get battery Level.*
- `#define GetCommandFlags(_n) __GetCommandFlags(_n)`  
*Get command flags.*
- `#define GetUIState(_n) __GetUIState(_n)`  
*Get UI module state.*
- `#define GetUIButton(_n) __GetUIButton(_n)`  
*Read UI button.*
- `#define GetVMRunState(_n) __GetVMRunState(_n)`  
*Read VM run state.*
- `#define GetBatteryState(_n) __GetBatteryState(_n)`  
*Get battery state.*
- `#define GetBluetoothState(_n) __GetBluetoothState(_n)`  
*Get bluetooth state.*
- `#define GetUsbState(_n) __GetUsbState(_n)`

- `#define GetSleepTimeout(_n) __GetSleepTimeout(_n)`  
*Read sleep timeout.*
- `#define GetSleepTimer(_n) __GetSleepTimer(_n)`  
*Read sleep timer.*
- `#define GetRechargeableBattery(_n) __GetRechargeableBattery(_n)`  
*Read battery type.*
- `#define GetVolume(_n) __GetVolume(_n)`  
*Read volume.*
- `#define GetOnBrickProgramPointer(_n) __GetOnBrickProgramPointer(_n)`  
*Read the on brick program pointer value.*
- `#define GetAbortFlag(_n) __GetAbortFlag(_n)`  
*Read abort flag.*

#### 5.42.1 Detailed Description

Functions for accessing and modifying Ui module features.

#### 5.42.2 Macro Definition Documentation

##### 5.42.2.1 `#define ForceOff( _n ) __forceOff(_n)`

Turn off NXT.

Force the NXT to turn off if the specified value is greater than zero.

##### Parameters

<code>_n</code>	If greater than zero the NXT will turn off.
-----------------	---

##### 5.42.2.2 `#define GetAbortFlag( _n ) __GetAbortFlag(_n)`

Read abort flag.

Return the enhanced NBC/NXC firmware's abort flag.

##### Parameters

<code>_n</code>	The current abort flag value. See <a href="#">ButtonState constants</a> .
-----------------	---

##### Warning

This function requires the enhanced NBC/NXC firmware.

##### 5.42.2.3 `#define GetBatteryLevel( _n ) __GetBatteryLevel(_n)`

Get battery Level.

Return the battery level in millivolts.

**Parameters**

_n	The battery level
----	-------------------

**5.42.2.4 #define GetBatteryState( \_n ) \_\_GetBatteryState(\_n)**

Get battery state.

Return battery state information (0..4).

**Parameters**

_n	The battery state (0..4)
----	--------------------------

**5.42.2.5 #define GetBluetoothState( \_n ) \_\_GetBluetoothState(\_n)**

Get bluetooth state.

Return the bluetooth state.

**Parameters**

_n	The bluetooth state. See <a href="#">BluetoothState constants</a> .
----	---

**5.42.2.6 #define GetCommandFlags( \_n ) \_\_GetCommandFlags(\_n)**

Get command flags.

Return the command flags.

**Parameters**

_n	Command flags. See <a href="#">CommandFlags constants</a>
----	---

**5.42.2.7 #define GetOnBrickProgramPointer( \_n ) \_\_GetOnBrickProgramPointer(\_n)**

Read the on brick program pointer value.

Return the current OBP (on-brick program) step

**Parameters**

_n	On brick program pointer (step).
----	----------------------------------

**5.42.2.8 #define GetRechargeableBattery( \_n ) \_\_GetRechargeableBattery(\_n)**

Read battery type.

Return whether the NXT has a rechargeable battery installed or not.

**Parameters**

_n	Whether the battery is rechargeable or not. (false = no, true = yes)
----	--

**5.42.2.9 #define GetSleepTimeout( \_n ) \_\_GetSleepTimeout(\_n)**

Read sleep timeout.

Return the number of minutes that the NXT will remain on before it automatically shuts down.

Parameters

_n	The sleep timeout value
----	-------------------------

**5.42.2.10 #define GetSleepTimer( \_n ) \_\_GetSleepTimer(\_n)**

Read sleep timer.

Return the number of minutes left in the countdown to zero from the original SleepTimeout value. When the SleepTimer value reaches zero the NXT will shutdown.

Parameters

_n	The sleep timer value
----	-----------------------

**5.42.2.11 #define GetUIButton( \_n ) \_\_GetUIButton(\_n)**

Read UI button.

Return user interface button information.

Parameters

_n	A UI button value. See <a href="#">UIButton constants</a> .
----	---

**5.42.2.12 #define GetUIState( \_n ) \_\_GetUIState(\_n)**

Get UI module state.

Return the user interface state.

Parameters

_n	The UI module state. See <a href="#">UIState constants</a> .
----	--

**5.42.2.13 #define GetUsbState( \_n ) \_\_GetUsbState(\_n)**

Get UI module USB state.

This method returns the UI module USB state.

Parameters

_n	The UI module USB state. (0=disconnected, 1=connected, 2=working)
----	---

**5.42.2.14 #define GetVMRunState( \_n ) \_\_GetVMRunState(\_n)**

Read VM run state.

Return VM run state information.

Parameters

_n	VM run state. See <a href="#">VM run state constants</a> .
----	--

**5.42.2.15 #define GetVolume( \_n ) \_\_GetVolume(\_n)**

Read volume.

Return the user interface volume level. Valid values are from 0 to 4.

**Parameters**

_n	The UI module volume. (0..4)
----	------------------------------

**5.42.2.16 #define SetAbortFlag( \_n ) \_\_setAbortFlag(\_n)**

Set abort flag.

Set the enhanced NBC/NXC firmware's program abort flag. By default the running program can be interrupted by a short press of the escape button. You can change this to any other button state flag.

**Parameters**

_n	The new abort flag value. See <a href="#">ButtonState constants</a>
----	---

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.42.2.17 #define SetBatteryState( \_n ) \_\_setBatteryState(\_n)**

Set battery state.

Set battery state information.

**Parameters**

_n	The desired battery state (0..4).
----	-----------------------------------

**5.42.2.18 #define SetBluetoothState( \_n ) \_\_setBluetoothState(\_n)**

Set bluetooth state.

Set the Bluetooth state.

**Parameters**

_n	The desired bluetooth state. See <a href="#">BluetoothState constants</a> .
----	---

**5.42.2.19 #define SetCommandFlags( \_n ) \_\_setCommandFlags(\_n)**

Set command flags.

Set the command flags.

**Parameters**

_n	The new command flags. See <a href="#">CommandFlags constants</a> .
----	---

**5.42.2.20 #define SetOnBrickProgramPointer( \_n ) \_\_setOnBrickProgramPointer(\_n)**

Set on-brick program pointer.

Set the current OBP (on-brick program) step.

**Parameters**

<code>_n</code>	The new on-brick program step.
-----------------	--------------------------------

**5.42.2.21 #define SetSleepTimeout( \_n ) \_\_setSleepTimeout(\_n)**

Set sleep timeout.

Set the NXT sleep timeout value to the specified number of minutes.

**Parameters**

<code>_n</code>	The minutes to wait before sleeping.
-----------------	--------------------------------------

**5.42.2.22 #define SetSleepTimer( \_n ) \_\_setSleepTimer(\_n)**

Set the sleep timer.

Set the system sleep timer to the specified number of minutes.

**Parameters**

<code>_n</code>	The minutes left on the timer.
-----------------	--------------------------------

**5.42.2.23 #define SetUIButton( \_n ) \_\_setUIButton(\_n)**

Set UI button.

Set user interface button information.

**Parameters**

<code>_n</code>	A user interface button value. See <a href="#">UIButton constants</a> .
-----------------	---

**5.42.2.24 #define SetUIState( \_n ) \_\_setUIState(\_n)**

Set UI state.

Set the user interface state.

**Parameters**

<code>_n</code>	A user interface state value. See <a href="#">UIState constants</a> .
-----------------	---

**5.42.2.25 #define SetUsbState( \_n ) \_\_setUsbState(\_n)**

Set Usb state.

This method sets the value of the Usb state.

**Parameters**

<code>_n</code>	The Usb state.
-----------------	----------------

**5.42.2.26 #define SetVMRunState( `_n` ) \_\_setVMRunState(`_n`)**

Set VM run state.

Set VM run state information.

**Parameters**

<code>_n</code>	The desired VM run state. See <a href="#">VM run state constants</a> .
-----------------	--

**5.42.2.27 #define SetVolume( `_n` ) \_\_setVolume(`_n`)**

Set volume.

Set the user interface volume level. Valid values are from 0 to 4.

**Parameters**

<code>_n</code>	The new volume level.
-----------------	-----------------------

## 5.43 Comm module functions

Functions for accessing and modifying Comm module features.

### Modules

- [Direct Command functions](#)

*Functions for sending direct commands to another NXT.*

- [System Command functions](#)

*Functions for sending system commands to another NXT.*

### Macros

- `#define SendMessage(_queue, _msg, _result) __sendMessage(_queue, _msg, _result)`  
*Send a message to a queue/mailbox.*
- `#define ReceiveMessage(_queue, _clear, _msg, _result) __receiveMessage(_queue, _clear, _msg, _result)`  
*Read a message from a queue/mailbox.*
- `#define ReceiveRemoteBool(_queue, _clear, _bval, _result) __receiveRemoteBool(_queue, _clear, _bval, _result)`  
*Read a boolean value from a queue/mailbox.*
- `#define ReceiveRemoteNumber(_queue, _clear, _val, _result) __receiveRemoteNumber(_queue, _clear, _val, _result)`  
*Read a numeric value from a queue/mailbox.*
- `#define ReceiveRemoteString(_queue, _clear, _str, _result) __receiveMessage(_queue, _clear, _str, _result)`  
*Read a string value from a queue/mailbox.*
- `#define ReceiveRemoteMessageEx(_queue, _clear, _str, _val, _bval, _result) __receiveRemoteMessageEx(_queue, _clear, _str, _val, _bval, _result)`  
*Read a value from a queue/mailbox.*
- `#define SendResponseString(_queue, _msg, _result) __sendResponseString(_queue, _msg, _result)`  
*Write a string value to a local response mailbox.*
- `#define SendResponseBool(_queue, _bval, _result) __sendResponseBool(_queue, _bval, _result)`  
*Write a boolean value to a local response mailbox.*
- `#define SendResponseNumber(_queue, _val, _result) __sendResponseNumber(_queue, _val, _result)`  
*Write a numeric value to a local response mailbox.*
- `#define BluetoothStatus(_conn, _result) __bluetoothStatus(_conn, _result)`  
*Check bluetooth status.*
- `#define BluetoothWrite(_conn, _buffer, _result) __bluetoothWrite(_conn, _buffer, _result)`  
*Write to a bluetooth connection.*
- `#define RemoteConnectionWrite(_conn, _buffer, _result) __connectionRawWrite(_conn, _buffer, _result)`  
*Write to a remote connection.*
- `#define RemoteConnectionIdle(_conn, _result) __remoteConnectionIdle(_conn, _result)`  
*Check if remote connection is idle.*
- `#define SendRemoteBool(_conn, _queue, _bval, _result) __sendRemoteBool(_conn, _queue, _bval, _result)`  
*Send a boolean value to a remote mailbox.*
- `#define SendRemoteNumber(_conn, _queue, _val, _result) __sendRemoteNumber(_conn, _queue, _val, _result)`  
*Send a numeric value to a remote mailbox.*
- `#define SendRemoteString(_conn, _queue, _str, _result) __sendRemoteString(_conn, _queue, _str, _result)`

- `#define UseRS485() __UseRS485()`  
*Send a string value to a remote mailbox.*
- `#define RS485Status(_sendingData, _dataAvail) __RS485Status(_sendingData, _dataAvail)`  
*Use the RS485 port.*
- `#define RS485Write(_buffer, _status) __RS485Write(_buffer, _status)`  
*Check RS485 status.*
- `#define RS485Read(_buffer, _status) __RS485Read(_buffer, _status)`  
*Write RS485 data.*
- `#define RS485ReadEx(_buffer, _buflen, _status) __RS485ReadEx(_buffer, _buflen, _status)`  
*Read RS485 data.*
- `#define RS485Control(_cmd, _baud, _mode, _result) __RS485Control(_cmd, _baud, _mode, _result)`  
*Read limited RS485 data.*
- `#define RS485Uart(_baud, _mode, _result) __RS485Control(HS_CTRL_UART, _baud, _mode, _result)`  
*Control the RS485 port.*
- `#define RS485Initialize(_result) __RS485Control(HS_CTRL_UART, HS_BAUD_DEFAULT, HS_MODE_DEFAULT, _result)`  
*Configure RS485 UART.*
- `#define RS485Enable(_result) __RS485Control(HS_CTRL_INIT, HS_BAUD_DEFAULT, HS_MODE_DEFAULT, _result)`  
*Initialize RS485 port.*
- `#define RS485Disable(_result) __RS485Control(HS_CTRL_EXIT, HS_BAUD_DEFAULT, HS_MODE_DEFAULT, _result)`  
*Enable RS485.*
- `#define SendRS485Bool(_bval, _status) __sendRS485Bool(_bval, _status)`  
*Disable RS485.*
- `#define SendRS485Number(_val, _status) __sendRS485Number(_val, _status)`  
*Write RS485 boolean.*
- `#define SendRS485String(_str, _status) __sendRS485String(_str, _status)`  
*Write RS485 numeric.*
- `#define GetBTDeviceName(_p, _str) __GetBTDeviceName(_p, _str)`  
*Write RS485 string.*
- `#define GetBTDeviceClass(_p, _n) __GetBTDeviceClass(_p, _n)`  
*Get bluetooth device name.*
- `#define GetBTDeviceAddress(_p, _btaddr) __getBTDeviceAddress(_p, _btaddr)`  
*Get bluetooth device class.*
- `#define GetBTDeviceStatus(_p, _n) __GetBTDeviceStatus(_p, _n)`  
*Get bluetooth device address.*
- `#define GetBTConnectionName(_p, _str) __GetBTConnectionName(_p, _str)`  
*Get bluetooth device status.*
- `#define GetBTConnectionClass(_p, _n) __GetBTConnectionClass(_p, _n)`  
*Get bluetooth device name.*
- `#define GetBTConnectionPinCode(_p, _code) __GetBTConnectionPinCode(_p, _code)`  
*Get bluetooth device class.*
- `#define GetBTConnectionAddress(_p, _btaddr) __getBTConnectionAddress(_p, _btaddr)`  
*Get bluetooth device pin code.*
- `#define GetBTConnectionHandleNum(_p, _n) __GetBTConnectionHandleNum(_p, _n)`  
*Get bluetooth device address.*

- `#define GetBTConnectionStreamStatus(_p, _n) __GetBTConnectionStreamStatus(_p, _n)`  
*Get bluetooth device stream status.*
- `#define GetBTConnectionLinkQuality(_p, _n) __GetBTConnectionLinkQuality(_p, _n)`  
*Get bluetooth device link quality.*
- `#define GetBrickDataName(_str) GetCommModuleBytes(CommOffsetBrickDataName, 16, _str)`  
*Get NXT name.*
- `#define GetBrickDataBluecoreVersion(_n)`  
*Get NXT bluecore version.*
- `#define GetBrickDataAddress(_btaddr) GetCommModuleBytes(CommOffsetBrickDataBdAddr, 7, _btaddr)`  
*Get NXT address.*
- `#define GetBrickDataBtStateStatus(_n)`  
*Get NXT bluetooth state status.*
- `#define GetBrickDataBtHardwareStatus(_n)`  
*Get NXT bluetooth hardware status.*
- `#define GetBrickDataTimeoutValue(_n)`  
*Get NXT bluetooth timeout value.*
- `#define GetBTInputBuffer(_offset, _cnt, _data) __getBTInputBuffer(_offset, _cnt, _data)`  
*Get bluetooth input buffer data.*
- `#define GetBTInputBufferInPtr(_n)`  
*Get bluetooth input buffer in-pointer.*
- `#define GetBTInputBufferOutPtr(_n)`  
*Get bluetooth input buffer out-pointer.*
- `#define GetBTOutputBuffer(_offset, _cnt, _data) __getBTOutputBuffer(_offset, _cnt, _data)`  
*Get bluetooth output buffer data.*
- `#define GetBTOutputBufferInPtr(_n)`  
*Get bluetooth output buffer in-pointer.*
- `#define GetBTOutputBufferOutPtr(_n)`  
*Get bluetooth output buffer out-pointer.*
- `#define GetHSInputBuffer(_offset, _cnt, _data) __getHSInputBuffer(_offset, _cnt, _data)`  
*Get hi-speed port input buffer data.*
- `#define GetHSInputBufferInPtr(_n)`  
*Get hi-speed port input buffer in-pointer.*
- `#define GetHSInputBufferOutPtr(_n)`  
*Get hi-speed port input buffer out-pointer.*
- `#define GetHSOutputBuffer(_offset, _cnt, _data) __getHSOutputBuffer(_offset, _cnt, _data)`  
*Get hi-speed port output buffer data.*
- `#define GetHSOutputBufferInPtr(_n)`  
*Get hi-speed port output buffer in-pointer.*
- `#define GetHSOutputBufferOutPtr(_n)`  
*Get hi-speed port output buffer out-pointer.*
- `#define GetUSBInputBuffer(_offset, _cnt, _data) __getUSBInputBuffer(_offset, _cnt, _data)`  
*Get usb input buffer data.*
- `#define GetUSBInputBufferInPtr(_n)`  
*Get usb port input buffer in-pointer.*
- `#define GetUSBInputBufferOutPtr(_n)`  
*Get usb port input buffer out-pointer.*

- #define `GetUSBOutputBuffer(_offset, _cnt, _data)` `__getUSBOutputBuffer(_offset, _cnt, _data)`  
*Get usb output buffer data.*
- #define `GetUSBOutputBufferInPtr(_n)`  
*Get usb port output buffer in-pointer.*
- #define `GetUSBOutputBufferOutPtr(_n)`  
*Get usb port output buffer out-pointer.*
- #define `GetUSBPollBuffer(_offset, _cnt, _data)` `__getUSBPollBuffer(_offset, _cnt, _data)`  
*Get usb poll buffer data.*
- #define `GetUSBPollBufferInPtr(_n)`  
*Get usb port poll buffer in-pointer.*
- #define `GetUSBPollBufferOutPtr(_n)`  
*Get usb port poll buffer out-pointer.*
- #define `GetBTDeviceCount(_n)`  
*Get bluetooth device count.*
- #define `GetBTDeviceNameCount(_n)`  
*Get bluetooth device name count.*
- #define `GetHSFlags(_n)`  
*Get hi-speed port flags.*
- #define `GetHSSpeed(_n)`  
*Get hi-speed port speed.*
- #define `GetHSState(_n)`  
*Get hi-speed port state.*
- #define `GetUSBState(_n)`  
*Get USB state.*
- #define `GetHSAddress(_n)`  
*Get hi-speed port address.*
- #define `GetHSMode(_n)`  
*Get hi-speed port mode.*
- #define `GetBTDataMode(_n)`  
*Get Bluetooth data mode.*
- #define `GetHSDataMode(_n)`  
*Get hi-speed port data mode.*
- #define `SetBTInputBuffer(_offset, _cnt, _data)` `__setBTInputBuffer(_offset, _cnt, _data)`  
*Set bluetooth input buffer data.*
- #define `SetBTInputBufferInPtr(_n) __setBTInputBufferInPtr(_n)`  
*Set bluetooth input buffer in-pointer.*
- #define `SetBTInputBufferOutPtr(_n) __setBTInputBufferOutPtr(_n)`  
*Set bluetooth input buffer out-pointer.*
- #define `SetBTOutputBuffer(_offset, _cnt, _data)` `__setBTOutputBuffer(_offset, _cnt, _data)`  
*Set bluetooth output buffer data.*
- #define `SetBTOutputBufferInPtr(_n) __setBTOutputBufferInPtr(_n)`  
*Set bluetooth output buffer in-pointer.*
- #define `SetBTOutputBufferOutPtr(_n) __setBTOutputBufferOutPtr(_n)`  
*Set bluetooth output buffer out-pointer.*
- #define `SetHSInputBuffer(_offset, _cnt, _data)` `__setHSInputBuffer(_offset, _cnt, _data)`  
*Set hi-speed port input buffer data.*
- #define `SetHSInputBufferInPtr(_n) __setHSInputBufferInPtr(_n)`

- `#define SetHSInputBufferOutPtr(_n) __setHSInputBufferOutPtr(_n)`  
*Set hi-speed port input buffer out-pointer.*
- `#define SetHSOutputBuffer(_offset, _cnt, _data) __setHSOutputBuffer(_offset, _cnt, _data)`  
*Set hi-speed port output buffer data.*
- `#define SetHSOutputBufferInPtr(_n) __setHSOutputBufferInPtr(_n)`  
*Set hi-speed port output buffer in-pointer.*
- `#define SetHSOutputBufferOutPtr(_n) __setHSOutputBufferOutPtr(_n)`  
*Set hi-speed port output buffer out-pointer.*
- `#define SetUSBInputBuffer(_offset, _cnt, _data) __setUSBInputBuffer(_offset, _cnt, _data)`  
*Set USB input buffer data.*
- `#define SetUSBInputBufferInPtr(_n) __setUSBInputBufferInPtr(_n)`  
*Set USB input buffer in-pointer.*
- `#define SetUSBInputBufferOutPtr(_n) __setUSBInputBufferOutPtr(_n)`  
*Set USB input buffer out-pointer.*
- `#define SetUSBOOutputBuffer(_offset, _cnt, _data) __setUSBOOutputBuffer(_offset, _cnt, _data)`  
*Set USB output buffer data.*
- `#define SetUSBOOutputBufferInPtr(_n) __setUSBOOutputBufferInPtr(_n)`  
*Set USB output buffer in-pointer.*
- `#define SetUSBOOutputBufferOutPtr(_n) __setUSBOOutputBufferOutPtr(_n)`  
*Set USB output buffer out-pointer.*
- `#define SetUSBPollBuffer(_offset, _cnt, _data) __setUSBPollBuffer(_offset, _cnt, _data)`  
*Set USB poll buffer data.*
- `#define SetUSBPollBufferInPtr(_n) __setUSBPollBufferInPtr(_n)`  
*Set USB poll buffer in-pointer.*
- `#define SetUSBPollBufferOutPtr(_n) __setUSBPollBufferOutPtr(_n)`  
*Set USB poll buffer out-pointer.*
- `#define SetHSFlags(_n) __setHSFlags(_n)`  
*Set hi-speed port flags.*
- `#define SetHSSpeed(_n) __setHSSpeed(_n)`  
*Set hi-speed port speed.*
- `#define SetHSState(_n) __setHSState(_n)`  
*Set hi-speed port state.*
- `#define SetUSBState(_n) __setUSBState(_n)`  
*Set USB state.*
- `#define SetHSAddress(_n) __setHSAddress(_n)`  
*Set hi-speed port address.*
- `#define SetHSMode(_n) __setHSMode(_n)`  
*Set hi-speed port mode.*
- `#define SetBTDataMode(_n) __setBTDataMode(_n)`  
*Set Bluetooth data mode.*
- `#define SetHSDataMode(_n) __setHSDataMode(_n)`  
*Set hi-speed port data mode.*

#### 5.43.1 Detailed Description

Functions for accessing and modifying Comm module features.

### 5.43.2 Macro Definition Documentation

#### 5.43.2.1 #define BluetoothStatus( *\_conn*, *\_result* ) \_\_bluetoothStatus(*\_conn*, *\_result*)

Check bluetooth status.

Check the status of the bluetooth subsystem for the specified connection slot.

##### Parameters

<i>_conn</i>	The connection slot (0..3). Connections 0 through 3 are for bluetooth connections. See <a href="#">Remote connection constants</a> .
<i>_result</i>	The bluetooth status for the specified connection.

#### 5.43.2.2 #define BluetoothWrite( *\_conn*, *\_buffer*, *\_result* ) \_\_bluetoothWrite(*\_conn*, *\_buffer*, *\_result*)

Write to a bluetooth connection.

This method tells the NXT firmware to write the data in the buffer to the device on the specified Bluetooth connection. Use [BluetoothStatus](#) to determine when this write request is completed.

##### Parameters

<i>_conn</i>	The connection slot (0..3). Connections 0 through 3 are for bluetooth connections. See <a href="#">Remote connection constants</a> .
<i>_buffer</i>	The data to be written (up to 128 bytes)
<i>_result</i>	A char value indicating whether the function call succeeded or not.

#### 5.43.2.3 #define GetBrickDataAddress( *\_btaddr* ) GetCommModuleBytes(CommOffsetBrickDataBdAddr, 7, *\_btaddr*)

Get NXT address.

This method reads the address of the NXT and stores it in the data buffer provided.

##### Parameters

<i>_btaddr</i>	The byte array reference that will contain the device address.
----------------	--

#### 5.43.2.4 #define GetBrickDataBluecoreVersion( *\_n* )

##### Value:

```
compchk EQ, sizeof(_n), 2 \
GetCommModuleValue(CommOffsetBrickDataBluecoreVersion
, _n)
```

Get NXT bluecore version.

This method returns the bluecore version of the NXT.

##### Parameters

<i>_n</i>	The NXT's bluecore version number.
-----------	------------------------------------

#### 5.43.2.5 #define GetBrickDataBtHardwareStatus( *\_n* )

##### Value:

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetBrickDataBtHwStatus
, _n)
```

Get NXT bluetooth hardware status.

This method returns the Bluetooth hardware status of the NXT.

**Parameters**

_n	The NXT's bluetooth hardware status.
----	--------------------------------------

#### 5.43.2.6 #define GetBrickDataBtStateStatus( \_n )

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetBrickDataBtStateStatus
, _n)
```

Get NXT bluetooth state status.

This method returns the Bluetooth state status of the NXT.

**Parameters**

_n	The NXT's bluetooth state status.
----	-----------------------------------

#### 5.43.2.7 #define GetBrickDataName( \_str ) GetCommModuleBytes(CommOffsetBrickDataName, 16, \_str)

Get NXT name.

This method returns the name of the NXT.

**Parameters**

_str	The NXT's bluetooth name.
------	---------------------------

#### 5.43.2.8 #define GetBrickDataTimeoutValue( \_n )

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetBrickDataTimeOutValue
, _n)
```

Get NXT bluetooth timeout value.

This method returns the Bluetooth timeout value of the NXT.

**Parameters**

_n	The NXT's bluetooth timeout value.
----	------------------------------------

#### 5.43.2.9 #define GetBTConnectionAddress( \_p, \_btaddr ) \_\_getBTConnectionAddress(\_p, \_btaddr)

Get bluetooth device address.

This method reads the address of the device at the specified index within the Bluetooth connection table and stores it in the data buffer provided.

#### Parameters

<code>_p</code>	The connection slot (0..3).
<code>_btaddr</code>	The byte array reference that will contain the device address.

#### 5.43.2.10 #define GetBTConnectionClass( `_p`, `_n` ) \_\_GetBTConnectionClass(`_p`, `_n`)

Get bluetooth device class.

This method returns the class of the device at the specified index within the Bluetooth connection table.

#### Parameters

<code>_p</code>	The connection slot (0..3).
<code>_n</code>	The class of the bluetooth device at the specified connection slot.

#### 5.43.2.11 #define GetBTConnectionHandleNum( `_p`, `_n` ) \_\_GetBTConnectionHandleNum(`_p`, `_n`)

Get bluetooth device handle number.

This method returns the handle number of the device at the specified index within the Bluetooth connection table.

#### Parameters

<code>_p</code>	The connection slot (0..3).
<code>_n</code>	The handle number of the bluetooth device at the specified connection slot.

#### 5.43.2.12 #define GetBTConnectionLinkQuality( `_p`, `_n` ) \_\_GetBTConnectionLinkQuality(`_p`, `_n`)

Get bluetooth device link quality.

This method returns the link quality of the device at the specified index within the Bluetooth connection table.

#### Parameters

<code>_p</code>	The connection slot (0..3).
<code>_n</code>	The link quality of the specified connection slot (unimplemented).

#### Warning

This function is not implemented at the firmware level.

#### 5.43.2.13 #define GetBTConnectionName( `_p`, `_str` ) \_\_GetBTConnectionName(`_p`, `_str`)

Get bluetooth device name.

This method returns the name of the device at the specified index in the Bluetooth connection table.

#### Parameters

<code>_p</code>	The connection slot (0..3).
<code>_str</code>	The name of the bluetooth device at the specified connection slot.

**5.43.2.14 #define GetBTConnectionPinCode( \_p, \_code ) \_\_GetBTConnectionPinCode(\_p, \_code)**

Get bluetooth device pin code.

This method returns the pin code of the device at the specified index in the Bluetooth connection table.

**Parameters**

<u>_p</u>	The connection slot (0..3).
<u>_code</u>	The pin code for the bluetooth device at the specified connection slot.

**5.43.2.15 #define GetBTConnectionStreamStatus( \_p, \_n ) \_\_GetBTConnectionStreamStatus(\_p, \_n)**

Get bluetooth device stream status.

This method returns the stream status of the device at the specified index within the Bluetooth connection table.

**Parameters**

<u>_p</u>	The connection slot (0..3).
<u>_n</u>	The stream status of the bluetooth device at the specified connection slot.

**5.43.2.16 #define GetBTDataMode( \_n )****Value:**

```
compchk EQ, sizeof(_n), 1 \
    GetCommModuleValue(CommOffsetBtDataMode
    , _n)
```

Get Bluetooth data mode.

This method returns the value of the Bluetooth data mode.

**Parameters**

<u>_n</u>	The Bluetooth data mode. See <a href="#">Data mode constants</a> .
-----------	--

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**5.43.2.17 #define GetBTDeviceAddress( \_p, \_btaddr ) \_\_getBTDeviceAddress(\_p, \_btaddr)**

Get bluetooth device address.

This method reads the address of the device at the specified index within the Bluetooth device table and stores it in the data buffer provided.

**Parameters**

<u>_p</u>	The device table index.
<u>_btaddr</u>	The byte array reference that will contain the device address.

## 5.43.2.18 #define GetBTDeviceClass( \_p, \_n ) \_\_GetBTDeviceClass(\_p, \_n)

Get bluetooth device class.

This method returns the class of the device at the specified index within the Bluetooth device table.

**Parameters**

<u>_p</u>	The device table index.
<u>_n</u>	The device class of the specified bluetooth device.

## 5.43.2.19 #define GetBTDeviceCount( \_n )

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetBtDeviceCnt
, _n)
```

Get bluetooth device count.

This method returns the number of devices defined within the Bluetooth device table.

**Returns**

The count of known bluetooth devices.

## 5.43.2.20 #define GetBTDeviceName( \_p, \_str ) \_\_GetBTDeviceName(\_p, \_str)

Get bluetooth device name.

This method returns the name of the device at the specified index in the Bluetooth device table.

**Parameters**

<u>_p</u>	The device table index.
<u>_str</u>	The device name of the specified bluetooth device.

## 5.43.2.21 #define GetBTDeviceNameCount( \_n )

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetBtDeviceNameCnt
, _n)
```

Get bluetooth device name count.

This method returns the number of device names defined within the Bluetooth device table. This usually has the same value as BTDeviceCount but it can differ in some instances.

**Parameters**

<u>_n</u>	The count of known bluetooth device names.
-----------	--

**5.43.2.22 #define GetBTDeviceStatus( \_p, \_n ) \_\_GetBTDeviceStatus(\_p, \_n)**

Get bluetooth device status.

This method returns the status of the device at the specified index within the Bluetooth device table.

**Parameters**

<u>_p</u>	The device table index.
<u>_n</u>	The status of the specified bluetooth device.

**5.43.2.23 #define GetBTInputBuffer( \_offset, \_cnt, \_data ) \_\_getBTInputBuffer(\_offset, \_cnt, \_data)**

Get bluetooth input buffer data.

This method reads count bytes of data from the Bluetooth input buffer and writes it to the buffer provided.

**Parameters**

<u>_offset</u>	A constant offset into the bluetooth input buffer.
<u>_cnt</u>	The number of bytes to read.
<u>_data</u>	The byte array reference which will contain the data read from the bluetooth input buffer.

**5.43.2.24 #define GetBTInputBufferInPtr( \_n )****Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetBtInBufInPtr
, _n)
```

Get bluetooth input buffer in-pointer.

This method returns the value of the input pointer of the Bluetooth input buffer.

**Parameters**

<u>_n</u>	The bluetooth input buffer's in-pointer value.
-----------	--

**5.43.2.25 #define GetBTInputBufferOutPtr( \_n )****Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetBtInBufOutPtr
, _n)
```

Get bluetooth input buffer out-pointer.

This method returns the value of the output pointer of the Bluetooth input buffer.

**Parameters**

<u>_n</u>	The bluetooth input buffer's out-pointer value.
-----------	---

**5.43.2.26 #define GetBTOutputBuffer( \_offset, \_cnt, \_data ) \_\_getBTOutputBuffer(\_offset, \_cnt, \_data)**

Get bluetooth output buffer data.

This method reads count bytes of data from the Bluetooth output buffer and writes it to the buffer provided.

**Parameters**

<code>_offset</code>	A constant offset into the bluetooth output buffer.
<code>_cnt</code>	The number of bytes to read.
<code>_data</code>	The byte array reference which will contain the data read from the bluetooth output buffer.

**5.43.2.27 #define GetBTOutputBufferInPtr( \_n )**

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetBtOutBufInPtr
, _n)
```

Get bluetooth output buffer in-pointer.

This method returns the value of the input pointer of the Bluetooth output buffer.

**Parameters**

<code>_n</code>	The bluetooth output buffer's in-pointer value.
-----------------	---

**5.43.2.28 #define GetBTOutputBufferOutPtr( \_n )**

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetBtOutBufOutPtr
, _n)
```

Get bluetooth output buffer out-pointer.

This method returns the value of the output pointer of the Bluetooth output buffer.

**Parameters**

<code>_n</code>	The bluetooth output buffer's out-pointer value.
-----------------	--

**5.43.2.29 #define GetHSAddress( \_n )**

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetHsAddress,
_n)
```

Get hi-speed port address.

This method returns the value of the hi-speed port address.

**Parameters**

<code>_n</code>	The hi-speed port address. See <a href="#">Hi-speed port address constants</a> .
-----------------	--

**5.43.2.30 #define GetHSDataMode( `_n` )****Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetHsDataMode
, _n)
```

Get hi-speed port data mode.

This method returns the value of the hi-speed port data mode.

**Parameters**

<code>_n</code>	The hi-speed port data mode. See <a href="#">Data mode constants</a> .
-----------------	--

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**5.43.2.31 #define GetHSFlags( `_n` )****Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetHsFlags, _n)
```

Get hi-speed port flags.

This method returns the value of the hi-speed port flags.

**Parameters**

<code>_n</code>	The hi-speed port flags. See <a href="#">Hi-speed port flags constants</a> .
-----------------	--

**5.43.2.32 #define GetHSInputBuffer( `_offset`, `_cnt`, `_data` ) \_\_getHSInputBuffer(`_offset`, `_cnt`, `_data`)**

Get hi-speed port input buffer data.

This method reads count bytes of data from the hi-speed port input buffer and writes it to the buffer provided.

**Parameters**

<code>_offset</code>	A constant offset into the hi-speed port input buffer.
<code>_cnt</code>	The number of bytes to read.
<code>_data</code>	The byte array reference which will contain the data read from the hi-speed port input buffer.

**5.43.2.33 #define GetHSInputBufferInPtr( `_n` )****Value:**

```
compchk EQ, sizeof(_n), 1 \
```

```
GetCommModuleValue(CommOffsetHsInBufInPtr
, _n)
```

Get hi-speed port input buffer in-pointer.

This method returns the value of the input pointer of the hi-speed port input buffer.

#### Parameters

_n	The hi-speed port input buffer's in-pointer value.
----	--

#### 5.43.2.34 #define GetHSInputBufferOutPtr( \_n )

##### Value:

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetHsInBufOutPtr
, _n)
```

Get hi-speed port input buffer out-pointer.

This method returns the value of the output pointer of the hi-speed port input buffer.

#### Parameters

_n	The hi-speed port input buffer's out-pointer value.
----	---

#### 5.43.2.35 #define GetHSMode( \_n )

##### Value:

```
compchk EQ, sizeof(_n), 2 \
GetCommModuleValue(CommOffsetHsMode, _n)
```

Get hi-speed port mode.

This method returns the value of the hi-speed port mode.

#### Parameters

_n	The hi-speed port mode (data bits, stop bits, parity). See <a href="#">Hi-speed port data bits constants</a> , <a href="#">Hi-speed port stop bits constants</a> , <a href="#">Hi-speed port parity constants</a> , and <a href="#">Hi-speed port combined UART constants</a> .
----	---

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### 5.43.2.36 #define GetHSOutputBuffer( \_offset, \_cnt, \_data ) \_\_getHSOutputBuffer(\_offset, \_cnt, \_data)

Get hi-speed port output buffer data.

This method reads count bytes of data from the hi-speed port output buffer and writes it to the buffer provided.

**Parameters**

<code>_offset</code>	A constant offset into the hi-speed port output buffer.
<code>_cnt</code>	The number of bytes to read.
<code>_data</code>	The byte array reference which will contain the data read from the hi-speed port output buffer.

5.43.2.37 #define GetHSOutputBufferInPtr( `_n` )

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetHsOutBufInPtr
, _n)
```

Get hi-speed port output buffer in-pointer.

This method returns the value of the input pointer of the hi-speed port output buffer.

**Parameters**

<code>_n</code>	The hi-speed port output buffer's in-pointer value.
-----------------	---

5.43.2.38 #define GetHSOutputBufferOutPtr( `_n` )

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetHsOutBufOutPtr
, _n)
```

Get hi-speed port output buffer out-pointer.

This method returns the value of the output pointer of the hi-speed port output buffer.

**Parameters**

<code>_n</code>	The hi-speed port output buffer's out-pointer value.
-----------------	--

5.43.2.39 #define GetHSSpeed( `_n` )

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetHsSpeed, _n)
```

Get hi-speed port speed.

This method returns the value of the hi-speed port speed (baud rate).

**Parameters**

<code>_n</code>	The hi-speed port speed (baud rate). See <a href="#">Hi-speed port baud rate constants</a> .
-----------------	--

5.43.2.40 #define GetHSState( `_n` )

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetHsState, _n)
```

Get hi-speed port state.

This method returns the value of the hi-speed port state.

#### Parameters

<code>_n</code>	The hi-speed port state. See <a href="#">Hi-speed port state constants</a> .
-----------------	--

5.43.2.41 #define GetUSBInputBuffer( `_offset`, `_cnt`, `_data` ) `_getUSBInputBuffer(_offset, _cnt, _data)`

Get usb input buffer data.

This method reads count bytes of data from the usb input buffer and writes it to the buffer provided.

#### Parameters

<code>_offset</code>	A constant offset into the usb input buffer.
<code>_cnt</code>	The number of bytes to read.
<code>_data</code>	The byte array reference which will contain the data read from the usb input buffer.

5.43.2.42 #define GetUSBInputBufferInPtr( `_n` )

#### Value:

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetUsbInBufInPtr
, _n)
```

Get usb port input buffer in-pointer.

This method returns the value of the input pointer of the usb port input buffer.

#### Parameters

<code>_n</code>	The USB port input buffer's in-pointer value.
-----------------	---

5.43.2.43 #define GetUSBInputBufferOutPtr( `_n` )

#### Value:

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetUsbInBufOutPtr
, _n)
```

Get usb port input buffer out-pointer.

This method returns the value of the output pointer of the usb port input buffer.

#### Parameters

<code>_n</code>	The USB port input buffer's out-pointer value.
-----------------	--

**5.43.2.44 #define GetUSBOOutputBuffer( \_offset, \_cnt, \_data ) \_\_getUSBOOutputBuffer(\_offset, \_cnt, \_data)**

Get usb output buffer data.

This method reads count bytes of data from the usb output buffer and writes it to the buffer provided.

**Parameters**

<i>_offset</i>	A constant offset into the usb output buffer.
<i>_cnt</i>	The number of bytes to read.
<i>_data</i>	The byte array reference which will contain the data read from the usb output buffer.

**5.43.2.45 #define GetUSBOOutputBufferInPtr( \_n )**

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetUsbOutBufInPtr
, _n)
```

Get usb port output buffer in-pointer.

This method returns the value of the input pointer of the usb port output buffer.

**Parameters**

<i>_n</i>	The USB port output buffer's in-pointer value.
-----------	--

**5.43.2.46 #define GetUSBOOutputBufferOutPtr( \_n )**

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetUsbOutBufOutPtr
, _n)
```

Get usb port output buffer out-pointer.

This method returns the value of the output pointer of the usb port output buffer.

**Parameters**

<i>_n</i>	The USB port output buffer's out-pointer value.
-----------	---

**5.43.2.47 #define GetUSBPollBuffer( \_offset, \_cnt, \_data ) \_\_getUSBPollBuffer(\_offset, \_cnt, \_data)**

Get usb poll buffer data.

This method reads count bytes of data from the usb poll buffer and writes it to the buffer provided.

**Parameters**

<i>_offset</i>	A constant offset into the usb poll buffer.
<i>_cnt</i>	The number of bytes to read.
<i>_data</i>	The byte array reference which will contain the data read from the usb poll buffer.

## 5.43.2.48 #define GetUSBPollBufferInPtr( \_n )

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetUsbPollBufInPtr
, _n)
```

Get usb port poll buffer in-pointer.

This method returns the value of the input pointer of the usb port poll buffer.

**Parameters**

_n	The USB port poll buffer's in-pointer value.
----	--

## 5.43.2.49 #define GetUSBPollBufferOutPtr( \_n )

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetUsbPollBufOutPtr
, _n)
```

Get usb port poll buffer out-pointer.

This method returns the value of the output pointer of the usb port poll buffer.

**Parameters**

_n	The USB port poll buffer's out-pointer value.
----	---

## 5.43.2.50 #define GetUSBState( \_n )

**Value:**

```
compchk EQ, sizeof(_n), 1 \
GetCommModuleValue(CommOffsetUsbState, _n
)
```

Get USB state.

This method returns the value of the USB state.

**Parameters**

_n	The USB state.
----	----------------

## 5.43.2.51 #define ReceiveMessage( \_queue, \_clear, \_msg, \_result ) \_\_receiveMessage(\_queue, \_clear, \_msg, \_result)

Read a message from a queue/mailbox.

Read a message from a mailbox and optionally remove it. If the local mailbox is empty and this NXT is the master then it attempts to poll one of its slave NXTs for a message from the response mailbox that corresponds to the specified local mailbox number.

**Parameters**

<code>_queue</code>	The mailbox number. See <a href="#">Mailbox constants</a> .
<code>_clear</code>	A flag indicating whether to remove the message from the mailbox after it has been read.
<code>_msg</code>	The message that is read from the mailbox.
<code>_result</code>	A char value indicating whether the function call succeeded or not.

**5.43.2.52 #define ReceiveRemoteBool( `_queue`, `_clear`, `_bval`, `_result` ) `_receiveRemoteBool(``_queue`, `_clear`, `_bval`, `_result`)**

Read a boolean value from a queue/mailbox.

Read a boolean value from a mailbox and optionally remove it. If the local mailbox is empty and this NXT is the master then it attempts to poll one of its slave NXTs for a message from the response mailbox that corresponds to the specified local mailbox number.

**Parameters**

<code>_queue</code>	The mailbox number. See <a href="#">Mailbox constants</a> .
<code>_clear</code>	A flag indicating whether to remove the message from the mailbox after it has been read.
<code>_bval</code>	The boolean value that is read from the mailbox.
<code>_result</code>	A char value indicating whether the function call succeeded or not.

**5.43.2.53 #define ReceiveRemoteMessageEx( `_queue`, `_clear`, `_str`, `_val`, `_bval`, `_result` ) `_receiveRemoteMessageEx(``_queue`, `_clear`, `_str`, `_val`, `_bval`, `_result`)**

Read a value from a queue/mailbox.

Read a value from a mailbox and optionally remove it. If the local mailbox is empty and this NXT is the master then it attempts to poll one of its slave NXTs for a message from the response mailbox that corresponds to the specified local mailbox number. Output the value in string, number, and boolean form.

**Parameters**

<code>_queue</code>	The mailbox number. See <a href="#">Mailbox constants</a> .
<code>_clear</code>	A flag indicating whether to remove the message from the mailbox after it has been read.
<code>_str</code>	The string value that is read from the mailbox.
<code>_val</code>	The numeric value that is read from the mailbox.
<code>_bval</code>	The boolean value that is read from the mailbox.
<code>_result</code>	A char value indicating whether the function call succeeded or not.

**5.43.2.54 #define ReceiveRemoteNumber( `_queue`, `_clear`, `_val`, `_result` ) `_receiveRemoteNumber(``_queue`, `_clear`, `_val`, `_result`)**

Read a numeric value from a queue/mailbox.

Read a numeric value from a mailbox and optionally remove it. If the local mailbox is empty and this NXT is the master then it attempts to poll one of its slave NXTs for a message from the response mailbox that corresponds to the specified local mailbox number.

**Parameters**

<code>_queue</code>	The mailbox number. See <a href="#">Mailbox constants</a> .
<code>_clear</code>	A flag indicating whether to remove the message from the mailbox after it has been read.
<code>_val</code>	The numeric value that is read from the mailbox.
<code>_result</code>	A char value indicating whether the function call succeeded or not.

**5.43.2.55 #define ReceiveRemoteString( \_queue, \_clear, \_str, \_result ) \_\_receiveMessage(\_queue, \_clear, \_str, \_result)**

Read a string value from a queue/mailbox.

Read a string value from a mailbox and optionally remove it. If the local mailbox is empty and this NXT is the master then it attempts to poll one of its slave NXTs for a message from the response mailbox that corresponds to the specified local mailbox number.

**Parameters**

<code>_queue</code>	The mailbox number. See <a href="#">Mailbox constants</a> .
<code>_clear</code>	A flag indicating whether to remove the message from the mailbox after it has been read.
<code>_str</code>	The string value that is read from the mailbox.
<code>_result</code>	A char value indicating whether the function call succeeded or not.

**5.43.2.56 #define RemoteConnectionIdle( \_conn, \_result ) \_\_remoteConnectionIdle(\_conn, \_result)**

Check if remote connection is idle.

Check whether a Bluetooth or RS485 hi-speed port connection is idle, i.e., not currently sending data.

**Parameters**

<code>_conn</code>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<code>_result</code>	A boolean value indicating whether the connection is idle or busy.

**Warning**

Checking the status of the RS485 hi-speed connection requires the enhanced NBC/NXC firmware

**5.43.2.57 #define RemoteConnectionWrite( \_conn, \_buffer, \_result ) \_\_connectionRawWrite(\_conn, \_buffer, \_result)**

Write to a remote connection.

This method tells the NXT firmware to write the data in the buffer to the device on the specified connection. Use [RemoteConnectionIdle](#) to determine when this write request is completed.

**Parameters**

<code>_conn</code>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<code>_buffer</code>	The data to be written (up to 128 bytes)
<code>_result</code>	A char value indicating whether the function call succeeded or not.

**Warning**

Writing to the RS485 hi-speed connection requires the enhanced NBC/NXC firmware

**5.43.2.58 #define RS485Control( \_cmd, \_baud, \_mode, \_result ) \_\_RS485Control(\_cmd, \_baud, \_mode, \_result)**

Control the RS485 port.

Control the RS485 hi-speed port using the specified parameters.

**Parameters**

<i>_cmd</i>	The control command to send to the port. See <a href="#">Hi-speed port SysCommHSControl constants</a> .
<i>_baud</i>	The baud rate for the RS485 port. See <a href="#">Hi-speed port baud rate constants</a> .
<i>_mode</i>	The RS485 port mode (data bits, stop bits, parity). See <a href="#">Hi-speed port data bits constants</a> , <a href="#">Hi-speed port stop bits constants</a> , <a href="#">Hi-speed port parity constants</a> , and <a href="#">Hi-speed port combined UART constants</a> .
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**5.43.2.59** `#define RS485Disable( _result ) __RS485Control(HS_CTRL_EXIT, HS_BAUD_DEFAULT, HS_MODE_DEFAULT, _result)`

Disable RS485.

Turn off the RS485 port.

**Parameters**

<i>_result</i>	A char value indicating whether the function call succeeded or not.
----------------	---

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**5.43.2.60** `#define RS485Enable( _result ) __RS485Control(HS_CTRL_INIT, HS_BAUD_DEFAULT, HS_MODE_DEFAULT, _result)`

Enable RS485.

Turn on the RS485 hi-speed port so that it can be used.

**Parameters**

<i>_result</i>	A char value indicating whether the function call succeeded or not.
----------------	---

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**5.43.2.61** `#define RS485Initialize( _result ) __RS485Control(HS_CTRL_UART, HS_BAUD_DEFAULT, HS_MODE_DEFAULT, _result)`

Initialize RS485 port.

Initialize the RS485 UART port to its default values. The baud rate is set to 921600 and the mode is set to 8N1 (8 data bits, no parity, 1 stop bit). Data cannot be sent or received over the RS485 port until the UART is initialized and the port has been configured for RS485 usage.

**Parameters**

<i>_result</i>	A char value indicating whether the function call succeeded or not.
----------------	---

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**5.43.2.62 #define RS485Read( *\_buffer*, *\_status* ) \_\_RS485Read(*\_buffer*, *\_status*)**

Read RS485 data.

Read data from the RS485 hi-speed port.

**Parameters**

<i>_buffer</i>	A byte array that will contain the data read from the RS485 port.
<i>_status</i>	A char value indicating whether the function call succeeded or not.

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.43.2.63 #define RS485ReadEx( *\_buffer*, *\_buflen*, *\_status* ) \_\_RS485ReadEx(*\_buffer*, *\_buflen*, *\_status*)**

Read limited RS485 data.

Read a limited number of bytes of data from the RS485 hi-speed port.

**Parameters**

<i>_buffer</i>	A byte array that will contain the data read from the RS485 port.
<i>_buflen</i>	The number of bytes you want to read.
<i>_status</i>	A char value indicating whether the function call succeeded or not.

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.31+.

**5.43.2.64 #define RS485Status( *\_sendingData*, *\_dataAvail* ) \_\_RS485Status(*\_sendingData*, *\_dataAvail*)**

Check RS485 status.

Check the status of the RS485 hi-speed port.

**Parameters**

<i>_sendingData</i>	A boolean value set to true on output if data is being sent.
<i>_dataAvail</i>	A boolean value set to true on output if data is available to be read.

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.43.2.65 #define RS485Uart( *\_baud*, *\_mode*, *\_result* ) \_\_RS485Control(HS\_CTRL\_UART, *\_baud*, *\_mode*, *\_result*)**

Configure RS485 UART.

Configure the RS485 UART parameters, including baud rate, data bits, stop bits, and parity.

**Parameters**

<i>_baud</i>	The baud rate for the RS485 port. See <a href="#">Hi-speed port baud rate constants</a> .
<i>_mode</i>	The RS485 port mode (data bits, stop bits, parity). See <a href="#">Hi-speed port data bits constants</a> , <a href="#">Hi-speed port stop bits constants</a> , <a href="#">Hi-speed port parity constants</a> , and <a href="#">Hi-speed port combined UART constants</a> .
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**5.43.2.66 #define RS485Write( *\_buffer*, *\_status* ) \_\_RS485Write(*\_buffer*, *\_status*)**

Write RS485 data.

Write data to the RS485 hi-speed port.

**Parameters**

<i>_buffer</i>	A byte array containing the data to write to the RS485 port.
<i>_status</i>	A char value indicating whether the function call succeeded or not.

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.43.2.67 #define SendMessage( *\_queue*, *\_msg*, *\_result* ) \_\_sendMessage(*\_queue*, *\_msg*, *\_result*)**

Send a message to a queue/mailbox.

Write a message into a local mailbox.

**Parameters**

<i>_queue</i>	The mailbox number. See <a href="#">Mailbox constants</a> .
<i>_msg</i>	The message to write to the mailbox.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.43.2.68 #define SendRemoteBool( *\_conn*, *\_queue*, *\_bval*, *\_result* ) \_\_sendRemoteBool(*\_conn*, *\_queue*, *\_bval*, *\_result*)**

Send a boolean value to a remote mailbox.

Send a boolean value on the specified connection to the specified remote mailbox number. Use [RemoteConnectionId](#) to determine when this write request is completed.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_queue</i>	The mailbox number. See <a href="#">Mailbox constants</a> .
<i>_bval</i>	The boolean value to send.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.43.2.69 #define SendRemoteNumber( \_conn, \_queue, \_val, \_result ) \_\_sendRemoteNumber(\_conn, \_queue, \_val, \_result)**

Send a numeric value to a remote mailbox.

Send a numeric value on the specified connection to the specified remote mailbox number. Use [RemoteConnectionId](#) to determine when this write request is completed.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_queue</i>	The mailbox number. See <a href="#">Mailbox constants</a> .
<i>_val</i>	The numeric value to send.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.43.2.70 #define SendRemoteString( \_conn, \_queue, \_str, \_result ) \_\_sendRemoteString(\_conn, \_queue, \_str, \_result)**

Send a string value to a remote mailbox.

Send a string value on the specified connection to the specified remote mailbox number. Use [RemoteConnectionId](#) to determine when this write request is completed.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_queue</i>	The mailbox number. See <a href="#">Mailbox constants</a> .
<i>_str</i>	The string value to send.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.43.2.71 #define SendResponseBool( \_queue, \_bval, \_result ) \_\_sendResponseBool(\_queue, \_bval, \_result)**

Write a boolean value to a local response mailbox.

Write a boolean value to a response mailbox (the mailbox number + 10).

**Parameters**

<i>_queue</i>	The mailbox number. See <a href="#">Mailbox constants</a> . This function shifts the specified value into the range of response mailbox numbers by adding 10.
<i>_bval</i>	The boolean value to write.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.43.2.72 #define SendResponseNumber( \_queue, \_val, \_result ) \_\_sendResponseNumber(\_queue, \_val, \_result)**

Write a numeric value to a local response mailbox.

Write a numeric value to a response mailbox (the mailbox number + 10).

**Parameters**

<i>_queue</i>	The mailbox number. See <a href="#">Mailbox constants</a> . This function shifts the specified value into the range of response mailbox numbers by adding 10.
<i>_val</i>	The numeric value to write.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.43.2.73 #define SendResponseString( *\_queue*, *\_msg*, *\_result* ) \_\_sendResponseString(*\_queue*, *\_msg*, *\_result*)**

Write a string value to a local response mailbox.

Write a string value to a response mailbox (the mailbox number + 10).

**Parameters**

<i>_queue</i>	The mailbox number. See <a href="#">Mailbox constants</a> . This function shifts the specified value into the range of response mailbox numbers by adding 10.
<i>_msg</i>	The string value to write.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.43.2.74 #define SendRS485Bool( *\_bval*, *\_status* ) \_\_sendRS485Bool(*\_bval*, *\_status*)**

Write RS485 boolean.

Write a boolean value to the RS485 hi-speed port.

**Parameters**

<i>_bval</i>	A boolean value to write over the RS485 port.
<i>_status</i>	A char value indicating whether the function call succeeded or not.

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.43.2.75 #define SendRS485Number( *\_val*, *\_status* ) \_\_sendRS485Number(*\_val*, *\_status*)**

Write RS485 numeric.

Write a numeric value to the RS485 hi-speed port.

**Parameters**

<i>_val</i>	A numeric value to write over the RS485 port.
<i>_status</i>	A char value indicating whether the function call succeeded or not.

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.43.2.76 #define SendRS485String( *\_str*, *\_status* ) \_\_sendRS485String(*\_str*, *\_status*)**

Write RS485 string.

Write a string value to the RS485 hi-speed port.

**Parameters**

<i>_str</i>	A string value to write over the RS485 port.
<i>_status</i>	A char value indicating whether the function call succeeded or not.

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.43.2.77 #define SetBTDataMode( \_n ) \_\_setBTDataMode(\_n)**

Set Bluetooth data mode.

This method sets the value of the Bluetooth data mode.

**Parameters**

<code>_n</code>	The Bluetooth data mode. See <a href="#">Data mode constants</a> .
-----------------	--

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**5.43.2.78 #define SetBTInputBuffer( \_offset, \_cnt, \_data ) \_\_setBTInputBuffer(\_offset, \_cnt, \_data)**

Set bluetooth input buffer data.

Write cnt bytes of data to the bluetooth input buffer at offset.

**Parameters**

<code>_offset</code>	A constant offset into the input buffer
<code>_cnt</code>	The number of bytes to write
<code>_data</code>	A byte array containing the data to write

**5.43.2.79 #define SetBTInputBufferInPtr( \_n ) \_\_setBTInputBufferInPtr(\_n)**

Set bluetooth input buffer in-pointer.

Set the value of the input buffer in-pointer.

**Parameters**

<code>_n</code>	The new in-pointer value (0..127).
-----------------	------------------------------------

**5.43.2.80 #define SetBTInputBufferOutPtr( \_n ) \_\_setBTInputBufferOutPtr(\_n)**

Set bluetooth input buffer out-pointer.

Set the value of the input buffer out-pointer.

**Parameters**

<code>_n</code>	The new out-pointer value (0..127).
-----------------	-------------------------------------

**5.43.2.81 #define SetBTOutputBuffer( \_offset, \_cnt, \_data ) \_\_setBTOutputBuffer(\_offset, \_cnt, \_data)**

Set bluetooth output buffer data.

Write cnt bytes of data to the bluetooth output buffer at offset.

**Parameters**

<i>_offset</i>	A constant offset into the output buffer
<i>_cnt</i>	The number of bytes to write
<i>_data</i>	A byte array containing the data to write

5.43.2.82 #define SetBTOutputBufferInPtr( *\_n* ) \_\_setBTOutputBufferInPtr(*\_n*)

Set bluetooth output buffer in-pointer.

Set the value of the output buffer in-pointer.

**Parameters**

<i>_n</i>	The new in-pointer value (0..127).
-----------	------------------------------------

5.43.2.83 #define SetBTOutputBufferOutPtr( *\_n* ) \_\_setBTOutputBufferOutPtr(*\_n*)

Set bluetooth output buffer out-pointer.

Set the value of the output buffer out-pointer.

**Parameters**

<i>_n</i>	The new out-pointer value (0..127).
-----------	-------------------------------------

5.43.2.84 #define SetHSAddress( *\_n* ) \_\_setHSAddress(*\_n*)

Set hi-speed port address.

This method sets the value of the hi-speed port address.

**Parameters**

<i>_n</i>	The hi-speed port address. See <a href="#">Hi-speed port address constants</a> .
-----------	--

5.43.2.85 #define SetHSDaDataMode( *\_n* ) \_\_setHSDaDataMode(*\_n*)

Set hi-speed port data mode.

This method sets the value of the hi-speed port data mode.

**Parameters**

<i>_n</i>	The hi-speed port data mode. See <a href="#">Data mode constants</a> .
-----------	--

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

5.43.2.86 #define SetHSFlags( *\_n* ) \_\_setHSFlags(*\_n*)

Set hi-speed port flags.

This method sets the value of the hi-speed port flags.

## Parameters

<i>_n</i>	The hi-speed port flags. See <a href="#">Hi-speed port flags constants</a> .
-----------	--

5.43.2.87 #define SetHSInputBuffer( *\_offset*, *\_cnt*, *\_data* ) \_\_setHSInputBuffer(*\_offset*, *\_cnt*, *\_data*)

Set hi-speed port input buffer data.

Write *cnt* bytes of data to the hi-speed port input buffer at *offset*.

## Parameters

<i>_offset</i>	A constant offset into the input buffer
<i>_cnt</i>	The number of bytes to write
<i>_data</i>	A byte array containing the data to write

5.43.2.88 #define SetHSInputBufferInPtr( *\_n* ) \_\_setHSInputBufferInPtr(*\_n*)

Set hi-speed port input buffer in-pointer.

Set the value of the input buffer in-pointer.

## Parameters

<i>_n</i>	The new in-pointer value (0..127).
-----------	------------------------------------

5.43.2.89 #define SetHSInputBufferOutPtr( *\_n* ) \_\_setHSInputBufferOutPtr(*\_n*)

Set hi-speed port input buffer out-pointer.

Set the value of the input buffer out-pointer.

## Parameters

<i>_n</i>	The new out-pointer value (0..127).
-----------	-------------------------------------

5.43.2.90 #define SetHSMode( *\_n* ) \_\_setHSMode(*\_n*)

Set hi-speed port mode.

This method sets the value of the hi-speed port mode.

## Parameters

<i>_n</i>	The hi-speed port mode (data bits, stop bits, parity). See <a href="#">Hi-speed port data bits constants</a> , <a href="#">Hi-speed port stop bits constants</a> , <a href="#">Hi-speed port parity constants</a> , and <a href="#">Hi-speed port combined UART constants</a> .
-----------	---

## Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

5.43.2.91 #define SetHSOutputBuffer( *\_offset*, *\_cnt*, *\_data* ) \_\_setHSOutputBuffer(*\_offset*, *\_cnt*, *\_data*)

Set hi-speed port output buffer data.

Write *cnt* bytes of data to the hi-speed port output buffer at *offset*.

**Parameters**

<i>_offset</i>	A constant offset into the output buffer
<i>_cnt</i>	The number of bytes to write
<i>_data</i>	A byte array containing the data to write

5.43.2.92 #define SetHSOutputBufferInPtr( *\_n* ) \_\_setHSOutputBufferInPtr(*\_n*)

Set hi-speed port output buffer in-pointer.

Set the value of the output buffer in-pointer.

**Parameters**

<i>_n</i>	The new in-pointer value (0..127).
-----------	------------------------------------

5.43.2.93 #define SetHSOutputBufferOutPtr( *\_n* ) \_\_setHSOutputBufferOutPtr(*\_n*)

Set hi-speed port output buffer out-pointer.

Set the value of the output buffer out-pointer.

**Parameters**

<i>_n</i>	The new out-pointer value (0..127).
-----------	-------------------------------------

5.43.2.94 #define SetHSSpeed( *\_n* ) \_\_setHSSpeed(*\_n*)

Set hi-speed port speed.

This method sets the value of the hi-speed port speed (baud rate).

**Parameters**

<i>_n</i>	The hi-speed port speed (baud rate). See <a href="#">Hi-speed port baud rate constants</a> .
-----------	--

5.43.2.95 #define SetHSState( *\_n* ) \_\_setHSState(*\_n*)

Set hi-speed port state.

This method sets the value of the hi-speed port state.

**Parameters**

<i>_n</i>	The hi-speed port state. See <a href="#">Hi-speed port state constants</a> .
-----------	--

5.43.2.96 #define SetUSBInputBuffer( *\_offset*, *\_cnt*, *\_data* ) \_\_setUSBInputBuffer(*\_offset*, *\_cnt*, *\_data*)

Set USB input buffer data.

Write cnt bytes of data to the USB input buffer at offset.

**Parameters**

<i>_offset</i>	A constant offset into the input buffer
<i>_cnt</i>	The number of bytes to write
<i>_data</i>	A byte array containing the data to write

**5.43.2.97 #define SetUSBInputBufferInPtr( \_n ) \_\_setUSBInputBufferInPtr(\_n)**

Set USB input buffer in-pointer.

Set the value of the input buffer in-pointer.

**Parameters**

<code>_n</code>	The new in-pointer value (0..63).
-----------------	-----------------------------------

**5.43.2.98 #define SetUSBInputBufferOutPtr( \_n ) \_\_setUSBInputBufferOutPtr(\_n)**

Set USB input buffer out-pointer.

Set the value of the input buffer out-pointer.

**Parameters**

<code>_n</code>	The new out-pointer value (0..63).
-----------------	------------------------------------

**5.43.2.99 #define SetUSBOOutputBuffer( \_offset, \_cnt, \_data ) \_\_setUSBOOutputBuffer(\_offset, \_cnt, \_data)**

Set USB output buffer data.

Write cnt bytes of data to the USB output buffer at offset.

**Parameters**

<code>_offset</code>	A constant offset into the output buffer
<code>_cnt</code>	The number of bytes to write
<code>_data</code>	A byte array containing the data to write

**5.43.2.100 #define SetUSBOOutputBufferInPtr( \_n ) \_\_setUSBOOutputBufferInPtr(\_n)**

Set USB output buffer in-pointer.

Set the value of the output buffer in-pointer.

**Parameters**

<code>_n</code>	The new in-pointer value (0..63).
-----------------	-----------------------------------

**5.43.2.101 #define SetUSBOOutputBufferOutPtr( \_n ) \_\_setUSBOOutputBufferOutPtr(\_n)**

Set USB output buffer out-pointer.

Set the value of the output buffer out-pointer.

**Parameters**

<code>_n</code>	The new out-pointer value (0..63).
-----------------	------------------------------------

**5.43.2.102 #define SetUSBPollBuffer( \_offset, \_cnt, \_data ) \_\_setUSBPollBuffer(\_offset, \_cnt, \_data)**

Set USB poll buffer data.

Write cnt bytes of data to the USB poll buffer at offset.

**Parameters**

<i>_offset</i>	A constant offset into the poll buffer
<i>_cnt</i>	The number of bytes to write
<i>_data</i>	A byte array containing the data to write

5.43.2.103 #define SetUSBPollBufferInPtr( *\_n* ) \_\_setUSBPollBufferInPtr(*\_n*)

Set USB poll buffer in-pointer.

Set the value of the poll buffer in-pointer.

**Parameters**

<i>_n</i>	The new in-pointer value (0..63).
-----------	-----------------------------------

5.43.2.104 #define SetUSBPollBufferOutPtr( *\_n* ) \_\_setUSBPollBufferOutPtr(*\_n*)

Set USB poll buffer out-pointer.

Set the value of the poll buffer out-pointer.

**Parameters**

<i>_n</i>	The new out-pointer value (0..63).
-----------	------------------------------------

5.43.2.105 #define SetUSBState( *\_n* ) \_\_setUSBState(*\_n*)

Set USB state.

This method sets the value of the USB state.

**Parameters**

<i>_n</i>	The USB state.
-----------	----------------

5.43.2.106 #define UseRS485( ) \_\_UseRS485()

Use the RS485 port.

Configure port 4 for RS485 usage.

## 5.44 Direct Command functions

Functions for sending direct commands to another NXT.

### Macros

- #define `RemoteMessageRead`(`_conn`, `_queue`, `_result`) `__remoteMessageRead`(`_conn`, `_queue`, `_result`)  
*Send a MessageRead message.*
- #define `RemoteMessageWrite`(`_conn`, `_queue`, `_msg`, `_result`) `__sendRemoteString`(`_conn`, `_queue`, `_msg`, `_result`)  
*Send a MessageWrite message.*
- #define `RemoteStartProgram`(`_conn`, `_filename`, `_result`) `__remoteStartProgram`(`_conn`, `_filename`, `_result`)  
*Send a StartProgram message.*
- #define `RemoteStopProgram`(`_conn`, `_result`) `__connectionSCDCWrite`(`_conn`, `__DCStopProgramPacket`, `_result`)  
*Send a StopProgram message.*
- #define `RemotePlaySoundFile`(`_conn`, `_filename`, `_bloop`, `_result`) `__remotePlaySoundFile`(`_conn`, `_filename`, `_bloop`, `_result`)  
*Send a PlaySoundFile message.*
- #define `RemotePlayTone`(`_conn`, `_frequency`, `_duration`, `_result`) `__remotePlayTone`(`_conn`, `_frequency`, `_duration`, `_result`)  
*Send a PlayTone message.*
- #define `RemoteStopSound`(`_conn`, `_result`) `__connectionSCDCWrite`(`_conn`, `__DCStopSoundPacket`, `_result`)  
*Send a StopSound message.*
- #define `RemoteKeepAlive`(`_conn`, `_result`) `__connectionSCDCWrite`(`_conn`, `__DCKeepAlivePacket`, `_result`)  
*Send a KeepAlive message.*
- #define `RemoteResetScaledValue`(`_conn`, `_port`, `_result`) `__remoteResetScaledValue`(`_conn`, `_port`, `_result`)  
*Send a ResetScaledValue message.*
- #define `RemoteResetMotorPosition`(`_conn`, `_port`, `_brelative`, `_result`) `__remoteResetMotorPosition`(`_conn`, `_port`, `_brelative`, `_result`)  
*Send a ResetMotorPosition message.*
- #define `RemoteSetInputMode`(`_conn`, `_port`, `_type`, `_mode`, `_result`) `__remoteSetInputMode`(`_conn`, `_port`, `_type`, `_mode`, `_result`)  
*Send a SetInputMode message.*
- #define `RemoteSetOutputState`(`_conn`, `_port`, `_speed`, `_mode`, `_regmode`, `_turnpct`, `_runstate`, `_tacholimit`, `_result`) `__remoteSetOutputState`(`_conn`, `_port`, `_speed`, `_mode`, `_regmode`, `_turnpct`, `_runstate`, `_tacholimit`, `_result`)  
*Send a SetOutputMode message.*
- #define `RemoteGetOutputState`(`_conn`, `_params`, `_result`)  
*Send a GetOutputState message.*
- #define `RemoteGetInputValues`(`_conn`, `_params`, `_result`)  
*Send a GetInputValues message.*
- #define `RemoteGetBatteryLevel`(`_conn`, `_value`, `_result`) `__remoteGetBatteryLevel`(`_conn`, `_value`, `_result`)  
*Send a GetBatteryLevel message.*
- #define `RemoteLowspeedGetStatus`(`_conn`, `_value`, `_result`) `__remoteLowspeedGetStatus`(`_conn`, `_value`, `_result`)  
*Send a LSGetStatus message.*
- #define `RemoteLowspeedRead`(`_conn`, `_port`, `_bread`, `_data`, `_result`) `__remoteLowspeedRead`(`_conn`, `_port`, `_bread`, `_data`, `_result`)

- `#define RemoteGetCurrentProgramName(_conn, _name, _result) __remoteGetCurrentProgramName(_conn, _name, _result)`
  - Send a LowspeedRead message.*
- `#define RemoteDatalogRead(_conn, _remove, _cnt, _log, _result) __remoteDatalogRead(_conn, _remove, _cnt, _log, _result)`
  - Send a GetCurrentProgramName message.*
- `#define RemoteGetContactCount(_conn, _cnt, _result) __remoteGetContactCount(_conn, _cnt, _result)`
  - Send a GetContactCount message.*
- `#define RemoteGetContactName(_conn, _idx, _name, _result) __remoteGetContactName(_conn, _idx, _name, _result)`
  - Send a GetContactName message.*
- `#define RemoteGetConnectionCount(_conn, _cnt, _result) __remoteGetConnectionCount(_conn, _cnt, _result)`
  - Send a GetConnectionCount message.*
- `#define RemoteGetConnectionName(_conn, _idx, _name, _result) __remoteGetConnectionName(_conn, _idx, _name, _result)`
  - Send a GetConnectionName message.*
- `#define RemoteResetTachoCount(_conn, _port, _result) __remoteResetTachoCount(_conn, _port, _result)`
  - Send a ResetTachoCount message.*
- `#define RemoteGetProperty(_conn, _property, _result) __remoteGetProperty(_conn, _property, _result)`
  - Send a GetProperty message.*
- `#define RemoteDatalogSetTimes(_conn, _syncTime, _result) __remoteDatalogSetTimes(_conn, _syncTime, _result)`
  - Send a DatalogSetTimes message.*
- `#define RemoteSetProperty(_conn, _prop, _value, _result) __remoteSetProperty(_conn, _prop, _value, _result)`
  - Send a SetProperty message.*
- `#define RemoteLowspeedWrite(_conn, _port, _txlen, _rxlen, _data, _result) __remoteLowspeedWrite(_conn, _port, _txlen, _rxlen, _data, _result)`
  - Send a LowspeedWrite message.*

#### 5.44.1 Detailed Description

Functions for sending direct commands to another NXT.

#### 5.44.2 Macro Definition Documentation

##### 5.44.2.1 `#define RemoteDatalogRead( _conn, _remove, _cnt, _log, _result ) __remoteDatalogRead(_conn, _remove, _cnt, _log, _result)`

Send a DatalogRead message.

Send the DatalogRead direct command on the specified connection slot.

#### Parameters

<code>_conn</code>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<code>_remove</code>	Remove the datalog message from the queue after reading it (true or false).
<code>_cnt</code>	The number of bytes read from the datalog.
<code>_log</code>	A byte array containing the datalog contents.
<code>_result</code>	A char value indicating whether the function call succeeded or not.

**5.44.2.2 #define RemoteDatalogSetTimes( \_conn, \_syncTime, \_result ) \_\_remoteDatalogSetTimes(\_conn, \_syncTime, \_result)**

Send a DatalogSetTimes message.

Send the DatalogSetTimes direct command on the specified connection slot. Use [RemoteConnectionId](#) to determine when this write request is completed.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_syncTime</i>	The datalog sync time.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.44.2.3 #define RemoteGetBatteryLevel( \_conn, \_value, \_result ) \_\_remoteGetBatteryLevel(\_conn, \_value, \_result)**

Send a GetBatteryLevel message.

This method sends a GetBatteryLevel direct command to the device on the specified connection.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_value</i>	The battery level value.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.44.2.4 #define RemoteGetConnectionCount( \_conn, \_cnt, \_result ) \_\_remoteGetConnectionCount(\_conn, \_cnt, \_result)**

Send a GetConnectionCount message.

This method sends a GetConnectionCount direct command to the device on the specified connection.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_cnt</i>	The number of connections.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.44.2.5 #define RemoteGetConnectionName( \_conn, \_idx, \_name, \_result ) \_\_remoteGetConnectionName(\_conn, \_idx, \_name, \_result)**

Send a GetConnectionName message.

Send the GetConnectionName direct command on the specified connection slot.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_idx</i>	The index of the connection.
<i>_name</i>	The name of the specified connection.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.44.2.6 #define RemoteGetContactCount( \_conn, \_cnt, \_result ) \_\_remoteGetContactCount(\_conn, \_cnt, \_result)**

Send a GetContactCount message.

This method sends a GetContactCount direct command to the device on the specified connection.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_cnt</i>	The number of contacts.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.44.2.7 #define RemoteGetContactName( \_conn, \_idx, \_name, \_result ) \_\_remoteGetContactName(\_conn, \_idx, \_name, \_result)**

Send a GetContactName message.

Send the GetContactName direct command on the specified connection slot.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_idx</i>	The index of the contact.
<i>_name</i>	The name of the specified contact.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.44.2.8 #define RemoteGetCurrentProgramName( \_conn, \_name, \_result ) \_\_remoteGetCurrentProgramName(\_conn, \_name, \_result)**

Send a GetCurrentProgramName message.

This method sends a GetCurrentProgramName direct command to the device on the specified connection.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_name</i>	The current program name.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.44.2.9 #define RemoteGetInputValues( \_conn, \_params, \_result )****Value:**

```
compchktype _params, TInputValues \
__remoteGetInputValues(_conn, _params, _result)
```

Send a GetInputValues message.

Send the GetInputValues direct command on the specified connection slot.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_params</i>	The input and output parameters for the function call.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.44.2.10 #define RemoteGetOutputState( *\_conn*, *\_params*, *\_result* )**Value:**

```
compchktype _params, TOutputState \
__remoteGetOutputState(_conn, _params, _result)
```

Send a GetOutputState message.

Send the GetOutputState direct command on the specified connection slot.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_params</i>	The input and output parameters for the function call.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.44.2.11 #define RemoteGetProperty( *\_conn*, *\_property*, *\_result* ) \_\_remoteGetProperty(*\_conn*, *\_property*, *\_result*)

Send a GetProperty message.

Send the GetProperty direct command on the specified connection slot. Use [RemoteConnectionId](#) to determine when this write request is completed.**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_property</i>	The property to read. See <a href="#">Property constants</a> .
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.44.2.12 #define RemoteKeepAlive( *\_conn*, *\_result* ) \_\_connectionSCDCWrite(*\_conn*, \_\_DCKeepAlivePacket, *\_result*)

Send a KeepAlive message.

This method sends a KeepAlive direct command to the device on the specified connection. Use [RemoteConnectionId](#) to determine when this write request is completed.**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.44.2.13 #define RemoteLowspeedGetStatus( *\_conn*, *\_value*, *\_result* ) \_\_remoteLowspeedGetStatus(*\_conn*, *\_value*, *\_result*)

Send a LSGetStatus message.

This method sends a LSGetStatus direct command to the device on the specified connection.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_value</i>	The count of available bytes to read.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.44.2.14 #define RemoteLowspeedRead( *\_conn*, *\_port*, *\_bread*, *\_data*, *\_result* ) \_\_remoteLowspeedRead(*\_conn*, *\_port*, *\_bread*, *\_data*, *\_result*)

Send a LowspeedRead message.

Send the LowspeedRead direct command on the specified connection slot.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_port</i>	The input port from which to read I2C data. See <a href="#">NBC Input port constants</a> .
<i>_bread</i>	The number of bytes read.
<i>_data</i>	A byte array containing the data read from the I2C device.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.44.2.15 #define RemoteLowspeedWrite( *\_conn*, *\_port*, *\_txlen*, *\_rxlen*, *\_data*, *\_result* ) \_\_remoteLowspeedWrite(*\_conn*, *\_port*, *\_txlen*, *\_rxlen*, *\_data*, *\_result*)

Send a LowspeedWrite message.

Send the LowspeedWrite direct command on the specified connection slot. Use [RemoteConnectionIdle](#) to determine when this write request is completed.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_port</i>	The I2C port. See <a href="#">NBC Input port constants</a> .
<i>_txlen</i>	The number of bytes you are writing to the I2C device.
<i>_rxlen</i>	The number of bytes want to read from the I2C device.
<i>_data</i>	A byte array containing the data you are writing to the device.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.44.2.16 #define RemoteMessageRead( *\_conn*, *\_queue*, *\_result* ) \_\_remoteMessageRead(*\_conn*, *\_queue*, *\_result*)

Send a MessageRead message.

This method sends a MessageRead direct command to the device on the specified connection. Use [RemoteConnectionIdle](#) to determine when this write request is completed.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_queue</i>	The mailbox to read. See <a href="#">Mailbox constants</a> .
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.44.2.17 #define RemoteMessageWrite( *\_conn*, *\_queue*, *\_msg*, *\_result* ) \_\_sendRemoteString(*\_conn*, *\_queue*, *\_msg*, *\_result*)

Send a MessageWrite message.

This method sends a MessageWrite direct command to the device on the specified connection. Use [RemoteConnectionIdle](#) to determine when this write request is completed.

## Parameters

<code>_conn</code>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<code>_queue</code>	The mailbox to write. See <a href="#">Mailbox constants</a> .
<code>_msg</code>	The message to write to the mailbox.
<code>_result</code>	A char value indicating whether the function call succeeded or not.

5.44.2.18 #define `RemotePlaySoundFile( _conn, _filename, _bloop, _result )` \_\_remotePlaySoundFile(`_conn, _filename, _bloop, _result`)

Send a PlaySoundFile message.

Send the PlaySoundFile direct command on the specified connection slot. Use [RemoteConnectionId](#) to determine when this write request is completed.

## Parameters

<code>_conn</code>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<code>_filename</code>	The name of the sound file to play.
<code>_bloop</code>	A boolean value indicating whether to loop the sound file or not.
<code>_result</code>	A char value indicating whether the function call succeeded or not.

5.44.2.19 #define `RemotePlayTone( _conn, _frequency, _duration, _result )` \_\_remotePlayTone(`_conn, _frequency, _duration, _result`)

Send a PlayTone message.

Send the PlayTone direct command on the specified connection slot. Use [RemoteConnectionId](#) to determine when this write request is completed.

## Parameters

<code>_conn</code>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<code>_frequency</code>	The frequency of the tone.
<code>_duration</code>	The duration of the tone.
<code>_result</code>	A char value indicating whether the function call succeeded or not.

5.44.2.20 #define `RemoteResetMotorPosition( _conn, _port, _brelative, _result )` \_\_remoteResetMotorPosition(`_conn, _port, _brelative, _result`)

Send a ResetMotorPosition message.

Send the ResetMotorPosition direct command on the specified connection slot. Use [RemoteConnectionId](#) to determine when this write request is completed.

## Parameters

<code>_conn</code>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<code>_port</code>	The output port to reset.
<code>_brelative</code>	A flag indicating whether the counter to reset is relative.
<code>_result</code>	A char value indicating whether the function call succeeded or not.

**5.44.2.21 #define RemoteResetScaledValue( *\_conn*, *\_port*, *\_result* ) \_\_remoteResetScaledValue(*\_conn*, *\_port*, *\_result*)**

Send a ResetScaledValue message.

Send the ResetScaledValue direct command on the specified connection slot. Use [RemoteConnectionId](#) to determine when this write request is completed.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_port</i>	The input port to reset.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.44.2.22 #define RemoteResetTachoCount( *\_conn*, *\_port*, *\_result* ) \_\_remoteResetTachoCount(*\_conn*, *\_port*, *\_result*)**

Send a ResetTachoCount message.

Send the ResetTachoCount direct command on the specified connection slot. Use [RemoteConnectionId](#) to determine when this write request is completed.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_port</i>	The output port to reset the tachometer count on. See <a href="#">Output port constants</a> .
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.44.2.23 #define RemoteSetInputMode( *\_conn*, *\_port*, *\_type*, *\_mode*, *\_result* ) \_\_remoteSetInputMode(*\_conn*, *\_port*, *\_type*, *\_mode*, *\_result*)**

Send a SetInputModule message.

Send the SetInputModule direct command on the specified connection slot. Use [RemoteConnectionId](#) to determine when this write request is completed.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_port</i>	The input port to configure. See <a href="#">NBC Input port constants</a> .
<i>_type</i>	The sensor type. See <a href="#">NBC sensor type constants</a> .
<i>_mode</i>	The sensor mode. See <a href="#">NBC sensor mode constants</a> .
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.44.2.24 #define RemoteSetOutputState( *\_conn*, *\_port*, *\_speed*, *\_mode*, *\_regmode*, *\_turnpct*, *\_runstate*, *\_tacholimit*, *\_result* ) \_\_remoteSetOutputState(*\_conn*, *\_port*, *\_speed*, *\_mode*, *\_regmode*, *\_turnpct*, *\_runstate*, *\_tacholimit*, *\_result*)**

Send a SetOutputMode message.

Send the SetOutputMode direct command on the specified connection slot. Use [RemoteConnectionId](#) to determine when this write request is completed.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
--------------	--

<code>_port</code>	The output port to configure. See <a href="#">Output port constants</a> .
<code>_speed</code>	The motor speed. (-100..100)
<code>_mode</code>	The motor mode. See <a href="#">Output port mode constants</a> .
<code>_regmode</code>	The motor regulation mode. See <a href="#">Output port regulation mode constants</a> .
<code>_turnpct</code>	The motor synchronized turn percentage. (-100..100)
<code>_runstate</code>	The motor run state. See <a href="#">Output port run state constants</a> .
<code>_tacholimit</code>	The motor tachometer limit.
<code>_result</code>	A char value indicating whether the function call succeeded or not.

5.44.2.25 #define `RemoteSetProperty( _conn, _prop, _value, _result ) __remote SetProperty(_conn, _prop, _value, _result)`

Send a SetProperty message.

Send the SetProperty direct command on the specified connection slot. Use [RemoteConnectionIdle](#) to determine when this write request is completed.

#### Parameters

<code>_conn</code>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<code>_prop</code>	The property to set. See <a href="#">Property constants</a> .
<code>_value</code>	The new property value.
<code>_result</code>	A char value indicating whether the function call succeeded or not.

5.44.2.26 #define `RemoteStartProgram( _conn, _filename, _result ) __remote StartProgram(_conn, _filename, _result)`

Send a StartProgram message.

Send the StartProgram direct command on the specified connection slot. Use [RemoteConnectionIdle](#) to determine when this write request is completed.

#### Parameters

<code>_conn</code>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<code>_filename</code>	The name of the program to start running.
<code>_result</code>	A char value indicating whether the function call succeeded or not.

5.44.2.27 #define `RemoteStopProgram( _conn, _result ) __connectionSCDCWrite(_conn, __DCStopProgramPacket, _result)`

Send a StopProgram message.

Send the StopProgram direct command on the specified connection slot. Use [RemoteConnectionIdle](#) to determine when this write request is completed.

#### Parameters

<code>_conn</code>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<code>_result</code>	A char value indicating whether the function call succeeded or not.

5.44.2.28 #define `RemoteStopSound( _conn, _result ) __connectionSCDCWrite(_conn, __DCStopSoundPacket, _result)`

Send a StopSound message.

Send the StopSound direct command on the specified connection slot. Use [RemoteConnectionId](#) to determine when this write request is completed.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_result</i>	A char value indicating whether the function call succeeded or not.

## 5.45 System Command functions

Functions for sending system commands to another NXT.

### Macros

- `#define RemoteOpenRead(_conn, _filename, _handle, _size, _result) __remoteOpenRead(_conn, _filename, _handle, _size, _result)`  
*Send an OpenRead message.*
- `#define RemoteOpenAppendData(_conn, _filename, _handle, _size, _result) __remoteOpenAppendData(_conn, _filename, _handle, _size, _result)`  
*Send an OpenAppendData message.*
- `#define RemoteDeleteFile(_conn, _filename, _result) __remoteDeleteFile(_conn, _filename, _result)`  
*Send a DeleteFile message.*
- `#define RemoteFindFirstFile(_conn, _mask, _handle, _name, _size, _result) __remoteFindFirstFile(_conn, _mask, _handle, _name, _size, _result)`  
*Send a FindFirstFile message.*
- `#define RemoteGetFirmwareVersion(_conn, _pmin, _pmaj, _fmin, _fmaj, _result) __remoteGetFirmwareVersion(-_conn, _pmin, _pmaj, _fmin, _fmaj, _result)`  
*Send a GetFirmwareVersion message.*
- `#define RemoteGetBluetoothAddress(_conn, _btaddr, _result) __remoteGetBluetoothAddress(_conn, _btaddr, -_result)`  
*Send a GetBluetoothAddress message.*
- `#define RemoteGetDeviceInfo(_conn, _name, _btaddr, _btsignal, _freemem, _result) __remoteGetDeviceInfo(_-conn, _name, _btaddr, _btsignal, _freemem, _result)`  
*Send a GetDeviceInfo message.*
- `#define RemoteDeleteUserFlash(_conn, _result) __remoteDeleteUserFlash(_conn, _result)`  
*Send a DeleteUserFlash message.*
- `#define RemoteOpenWrite(_conn, _filename, _size, _result) __remoteOpenWrite(_conn, _filename, _size, _-result)`  
*Send an OpenWrite message.*
- `#define RemoteOpenWriteLinear(_conn, _filename, _size, _result) __remoteOpenWriteLinear(_conn, _filename, _size, _result)`  
*Send an OpenWriteLinear message.*
- `#define RemoteOpenWriteData(_conn, _filename, _size, _result) __remoteOpenWriteData(_conn, _filename, _-size, _result)`  
*Send an OpenWriteData message.*
- `#define RemoteCloseFile(_conn, _handle, _result) __remoteCloseFile(_conn, _handle, _result)`  
*Send a CloseFile message.*
- `#define RemoteFindNextFile(_conn, _handle, _result) __remoteFindNextFile(_conn, _handle, _result)`  
*Send a FindNextFile message.*
- `#define RemotePollCommandLength(_conn, _bufnum, _result) __remotePollCommandLength(_conn, _bufnum, _result)`  
*Send a PollCommandLength message.*
- `#define RemoteWrite(_conn, _handle, _data, _result) __remoteWrite(_conn, _handle, _data, _result)`  
*Send a Write message.*
- `#define RemoteRead(_conn, _handle, _numbytes, _result) __remoteRead(_conn, _handle, _numbytes, _result)`  
*Send a Read message.*

- #define `RemoteIOMapRead(_conn, _id, _offset, _numbytes, _result)` \_\_remoteIOMapRead(\_conn, \_id, \_offset, \_numbytes, \_result)
 

*Send an IOMapRead message.*
- #define `RemotePollCommand(_conn, _bufnum, _len, _result)` \_\_remotePollCommand(\_conn, \_bufnum, \_len, \_result)
 

*Send a PollCommand message.*
- #define `RemoteRenameFile(_conn, _oldname, _newname, _result)` \_\_remoteRenameFile(\_conn, \_oldname, \_newname, \_result)
 

*Send a RenameFile message.*
- #define `RemoteBluetoothFactoryReset(_conn, _result)` \_\_connectionSCDCWrite(\_conn, \_\_SCBTFactoryResetPacket, \_result)
 

*Send a BluetoothFactoryReset message.*
- #define `RemoteIOMapWriteValue(_conn, _id, _offset, _value, _result)` \_\_remoteIOMapWriteValue(\_conn, \_id, \_offset, \_value, \_result)
 

*Send an IOMapWrite value message.*
- #define `RemoteIOMapWriteBytes(_conn, _id, _offset, _data, _result)` \_\_remoteIOMapWriteBytes(\_conn, \_id, \_offset, \_data, \_result)
 

*Send an IOMapWrite bytes message.*
- #define `RemoteSetBrickName(_conn, _name, _result)` \_\_remoteSetBrickName(\_conn, \_name, \_result)
 

*Send a SetBrickName message.*

#### 5.45.1 Detailed Description

Functions for sending system commands to another NXT.

#### 5.45.2 Macro Definition Documentation

##### 5.45.2.1 #define `RemoteBluetoothFactoryReset( _conn, _result )` \_\_connectionSCDCWrite(\_conn, \_\_SCBTFactoryResetPacket, \_result)

Send a BluetoothFactoryReset message.

This method sends a BluetoothFactoryReset system command to the device on the specified connection. Use `RemoteConnectionIdle` to determine when this write request is completed.

#### Parameters

<code>_conn</code>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<code>_result</code>	A char value indicating whether the function call succeeded or not.

##### 5.45.2.2 #define `RemoteCloseFile( _conn, _handle, _result )` \_\_remoteCloseFile(\_conn, \_handle, \_result)

Send a CloseFile message.

Send the CloseFile system command on the specified connection slot.

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_handle</i>	The handle of the file to close.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.3 #define RemoteDeleteFile( *\_conn*, *\_filename*, *\_result* ) \_\_remoteDeleteFile(*\_conn*, *\_filename*, *\_result*)

Send a DeleteFile message.

Send the DeleteFile system command on the specified connection slot.

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_filename</i>	The name of the program to delete.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.4 #define RemoteDeleteUserFlash( *\_conn*, *\_result* ) \_\_remoteDeleteUserFlash(*\_conn*, *\_result*)

Send a DeleteUserFlash message.

This method sends a DeleteUserFlash system command to the device on the specified connection.

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.5 #define RemoteFindFirstFile( *\_conn*, *\_mask*, *\_handle*, *\_name*, *\_size*, *\_result* ) \_\_remoteFindFirstFile(*\_conn*, *\_mask*, *\_handle*, *\_name*, *\_size*, *\_result*)

Send a FindFirstFile message.

Send the FindFirstFile system command on the specified connection slot.

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
--------------	--

<i>_mask</i>	The filename mask for the files you want to find.
<i>_handle</i>	The handle of the found file.
<i>_name</i>	The name of the found file.
<i>_size</i>	The size of the found file.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.6 #define RemoteFindNextFile( *\_conn*, *\_handle*, *\_result* ) \_\_remoteFindNextFile(*\_conn*, *\_handle*, *\_result*)

Send a FindNextFile message.

Send the FindNextFile system command on the specified connection slot.

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_handle</i>	The handle returned by the last <a href="#">FindFirstFile</a> or <a href="#">FindNextFile</a> call.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.7 #define RemoteGetBluetoothAddress( *\_conn*, *\_btaddr*, *\_result* ) \_\_remoteGetBluetoothAddress(*\_conn*, *\_btaddr*, *\_result*)

Send a GetBluetoothAddress message.

This method sends a GetBluetoothAddress system command to the device on the specified connection.

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_btaddr</i>	The bluetooth address of the remote device.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.8 #define RemoteGetDeviceInfo( *\_conn*, *\_name*, *\_btaddr*, *\_btsignal*, *\_freemem*, *\_result* ) \_\_remoteGetDeviceInfo(*\_conn*, *\_name*, *\_btaddr*, *\_btsignal*, *\_freemem*, *\_result*)

Send a GetDeviceInfo message.

This method sends a GetDeviceInfo system command to the device on the specified connection.

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_name</i>	The name of the remote device.
<i>_btaddr</i>	The bluetooth address of the remote device.
<i>_btsignal</i>	The signal strength of each connection on the remote device.
<i>_freemem</i>	The number of bytes of free flash memory on the remote device.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.9 #define **RemoteGetFirmwareVersion( *\_conn*, *\_pmin*, *\_pmaj*, *\_fmin*, *\_fmaj*, *\_result* )** \_\_remoteGetFirmwareVersion(*\_conn*, *\_pmin*, *\_pmaj*, *\_fmin*, *\_fmaj*, *\_result*)

Send a GetFirmwareVersion message.

This method sends a GetFirmwareVersion system command to the device on the specified connection.

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_pmin</i>	The protocol minor version byte.
<i>_pmaj</i>	The protocol major version byte.
<i>_fmin</i>	The firmware minor version byte.
<i>_fmaj</i>	The firmware major version byte.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.10 #define **RemoteIOMapRead( *\_conn*, *\_id*, *\_offset*, *\_numbytes*, *\_result* )** \_\_remoteIOMapRead(*\_conn*, *\_id*, *\_offset*, *\_numbytes*, *\_result*)

Send an IOMapRead message.

Send the IOMapRead system command on the specified connection slot.

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_id</i>	The ID of the module from which to read data.
<i>_offset</i>	The offset into the IOMap structure from which to read.
<i>_numbytes</i>	The number of bytes of data to read.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

---

5.45.2.11 #define RemoteIOMapWriteBytes( *\_conn*, *\_id*, *\_offset*, *\_data*, *\_result* ) \_\_remoteIOMapWriteBytes(*\_conn*, *\_id*, *\_offset*,  
*\_data*, *\_result*)

Send an IOMapWrite bytes message.

Send the IOMapWrite system command on the specified connection slot to write the data provided. Use [RemoteConnectionId](#) to determine when this write request is completed.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_id</i>	The ID of the module to which to write data.
<i>_offset</i>	The offset into the IOMap structure to which to write.
<i>_data</i>	A byte array containing the data you are writing to the IOMap structure.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.12 #define RemoteIOMapWriteValue( *\_conn*, *\_id*, *\_offset*, *\_value*, *\_result* ) \_\_remoteIOMapWriteValue(*\_conn*, *\_id*, *\_offset*,  
*\_value*, *\_result*)

Send an IOMapWrite value message.

Send the IOMapWrite system command on the specified connection slot to write the value provided. Use [RemoteConnectionId](#) to determine when this write request is completed.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_id</i>	The ID of the module to which to write data.
<i>_offset</i>	The offset into the IOMap structure to which to write.
<i>_value</i>	A scalar variable containing the value you are writing to the IOMap structure.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.13 #define RemoteOpenAppendData( *\_conn*, *\_filename*, *\_handle*, *\_size*, *\_result* ) \_\_remoteOpenAppendData(*\_conn*,  
*\_filename*, *\_handle*, *\_size*, *\_result*)

Send an OpenAppendData message.

Send the OpenAppendData system command on the specified connection slot.

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_filename</i>	The name of the file to open for appending.
<i>_handle</i>	The handle of the file.
<i>_size</i>	The size of the file.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

---

5.45.2.14 #define RemoteOpenRead( *\_conn*, *\_filename*, *\_handle*, *\_size*, *\_result* ) \_\_remoteOpenRead(*\_conn*, *\_filename*, *\_handle*,  
*\_size*, *\_result*)

Send an OpenRead message.

Send the OpenRead system command on the specified connection slot.

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_filename</i>	The name of the file to open for reading.
<i>_handle</i>	The handle of the file.
<i>_size</i>	The size of the file.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.15 #define RemoteOpenWrite( *\_conn*, *\_filename*, *\_size*, *\_result* ) \_\_remoteOpenWrite(*\_conn*, *\_filename*, *\_size*, *\_result*)

Send an OpenWrite message.

Send the OpenWrite system command on the specified connection slot.

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_filename</i>	The name of the program to open for writing (i.e., create the file).
<i>_size</i>	The size for the new file.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.16 #define RemoteOpenWriteData( *\_conn*, *\_filename*, *\_size*, *\_result* ) \_\_remoteOpenWriteData(*\_conn*, *\_filename*, *\_size*,  
*\_result*)

Send an OpenWriteData message.

Send the OpenWriteData system command on the specified connection slot.

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_filename</i>	The name of the program to open for writing (i.e., create the file).
<i>_size</i>	The size for the new file.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

---

5.45.2.17 #define RemoteOpenWriteLinear( *\_conn*, *\_filename*, *\_size*, *\_result* ) \_\_remoteOpenWriteLinear(*\_conn*, *\_filename*, *\_size*,  
*\_result*)

Send an OpenWriteLinear message.

Send the OpenWriteLinear system command on the specified connection slot.

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_filename</i>	The name of the program to open for writing (i.e., create the file).
<i>_size</i>	The size for the new file.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.18 #define RemotePollCommand( *\_conn*, *\_bufnum*, *\_len*, *\_result* ) \_\_remotePollCommand(*\_conn*, *\_bufnum*, *\_len*, *\_result*)

Send a PollCommand message.

Send the PollCommand system command on the specified connection slot to write the data provided.

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_bufnum</i>	The buffer from which to read data (0=USBPoll, 1=HiSpeed).
<i>_len</i>	The number of bytes to read.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

5.45.2.19 #define RemotePollCommandLength( *\_conn*, *\_bufnum*, *\_result* ) \_\_remotePollCommandLength(*\_conn*, *\_bufnum*, *\_result*)

Send a PollCommandLength message.

Send the PollCommandLength system command on the specified connection slot.

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_bufnum</i>	The poll buffer you want to query (0=USBPoll, 1=HiSpeed).
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.45.2.20 #define RemoteRead( \_conn, \_handle, \_numbytes, \_result ) \_\_remoteRead(\_conn, \_handle, \_numbytes, \_result)**

Send a Read message.

Send the Read system command on the specified connection slot.

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_handle</i>	The handle of the file you are reading from.
<i>_numbytes</i>	The number of bytes you want to read.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.45.2.21 #define RemoteRenameFile( \_conn, \_oldname, \_newname, \_result ) \_\_remoteRenameFile(\_conn, \_oldname, \_newname, \_result)**

Send a RenameFile message.

Send the RenameFile system command on the specified connection slot to write the data provided. Use [RemoteConnectionIdle](#) to determine when this write request is completed.

#### Warning

This function requires the enhanced NBC/NXC firmware version 1.28+.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_oldname</i>	The old filename.
<i>_newname</i>	The new filename.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.45.2.22 #define RemoteSetBrickName( \_conn, \_name, \_result ) \_\_remoteSetBrickName(\_conn, \_name, \_result)**

Send a SetBrickName message.

Send the SetBrickName system command on the specified connection slot to write the data provided. Use [RemoteConnectionIdle](#) to determine when this write request is completed.

#### Parameters

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_name</i>	The new brick name.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

**5.45.2.23 #define RemoteWrite( \_conn, \_handle, \_data, \_result ) \_\_remoteWrite(\_conn, \_handle, \_data, \_result)**

Send a Write message.

Send the Write system command on the specified connection slot.

**Warning**

This function requires the enhanced NBC/NXC firmware version 1.28+.

**Parameters**

<i>_conn</i>	The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See <a href="#">Remote connection constants</a> .
<i>_handle</i>	The handle of the file you are writing to.
<i>_data</i>	A byte array containing the data you are writing.
<i>_result</i>	A char value indicating whether the function call succeeded or not.

## 5.46 IOCtrl module functions

Functions for accessing and modifying IOCtrl module features.

### Macros

- `#define PowerDown SetIOCtrlModuleValue(IOCtrlOffsetPowerOn, IOCTL_POWERDOWN)`  
*Power down the NXT.*
- `#define RebootInFirmwareMode SetIOCtrlModuleValue(IOCtrlOffsetPowerOn, IOCTL_BOOT)`  
*Reboot the NXT in firmware download mode.*

### 5.46.1 Detailed Description

Functions for accessing and modifying IOCtrl module features.

### 5.46.2 Macro Definition Documentation

#### 5.46.2.1 `#define PowerDown SetIOCtrlModuleValue(IOCtrlOffsetPowerOn, IOCTL_POWERDOWN)`

Power down the NXT.

This function powers down the NXT. The running program will terminate as a result of this action.

#### 5.46.2.2 `#define RebootInFirmwareMode SetIOCtrlModuleValue(IOCtrlOffsetPowerOn, IOCTL_BOOT)`

Reboot the NXT in firmware download mode.

This function lets you reboot the NXT into SAMBA or firmware download mode. The running program will terminate as a result of this action.

## 5.47 Loader module functions

Functions for accessing and modifying Loader module features.

### Macros

- `#define GetFreeMemory(_value) __GetFreeMemory(_value)`  
*Get free flash memory.*
- `#define CreateFile(_fname, _fsize, _handle, _result) __createFile(_fname, _fsize, _handle, _result)`  
*Create a file.*
- `#define OpenFileAppend(_fname, _fsize, _handle, _result) __openFileAppend(_fname, _fsize, _handle, _result)`  
*Open a file for appending.*
- `#define OpenFileRead(_fname, _fsize, _handle, _result) __openFileRead(_fname, _fsize, _handle, _result)`  
*Open a file for reading.*
- `#define CloseFile(_handle, _result) __closeFile(_handle, _result)`  
*Close a file.*
- `#define ResolveHandle(_fname, _handle, _writable, _result) __resolveHandle(_fname, _handle, _writable, _result)`  
*Resolve a handle.*
- `#define RenameFile(_oldname, _newname, _result) __renameFile(_oldname, _newname, _result)`  
*Rename a file.*
- `#define DeleteFile(_fname, _result) __deleteFile(_fname, _result)`  
*Delete a file.*
- `#define ResizeFile(_fname, _newsize, _result) __fileResize(_fname, _newsize, _result)`  
*Resize a file.*
- `#define CreateFileLinear(_fname, _fsize, _handle, _result) __createFileLinear(_fname, _fsize, _handle, _result)`  
*Create a linear file.*
- `#define CreateFileNonLinear(_fname, _fsize, _handle, _result) __createFileNonLinear(_fname, _fsize, _handle, _result)`  
*Create a non-linear file.*
- `#define OpenFileReadLinear(_fname, _fsize, _handle, _result) __openFileReadLinear(_fname, _fsize, _handle, _result)`  
*Open a linear file for reading.*
- `#define FindFirstFile(_fname, _handle, _result) __findFirstFile(_fname, _handle, _result)`  
*Start searching for files.*
- `#define FindNextFile(_fname, _handle, _result) __findNextFile(_fname, _handle, _result)`  
*Continue searching for files.*
- `#define SizeOf(_n, _result) __sizeOF(_n, _result)`  
*Calculate the size of a variable.*
- `#define Read(_handle, _n, _result) __readValue(_handle, _n, _result)`  
*Read a value from a file.*
- `#define ReadLn(_handle, _n, _result) __readLnValue(_handle, _n, _result)`  
*Read a value from a file plus line ending.*
- `#define ReadBytes(_handle, _len, _buf, _result) __readBytes(_handle, _len, _buf, _result)`  
*Read bytes from a file.*
- `#define ReadLnString(_handle, _output, _result) __readLnString(_handle, _output, _result)`  
*Read a string from a file plus line ending.*

- #define [Write](#)(*\_handle*, *\_n*, *\_result*) \_\_writeValue(*\_handle*, *\_n*, *\_result*)  
*Write value to file.*
- #define [WriteLn](#)(*\_handle*, *\_n*, *\_result*) \_\_writeLnValue(*\_handle*, *\_n*, *\_result*)  
*Write a value and new line to a file.*
- #define [WriteString](#)(*\_handle*, *\_str*, *\_cnt*, *\_result*) \_\_writeString(*\_handle*, *\_str*, *\_cnt*, *\_result*)  
*Write string to a file.*
- #define [WriteLnString](#)(*\_handle*, *\_str*, *\_cnt*, *\_result*) \_\_writeLnString(*\_handle*, *\_str*, *\_cnt*, *\_result*)  
*Write string and new line to a file.*
- #define [WriteBytes](#)(*\_handle*, *\_buf*, *\_cnt*, *\_result*) \_\_writeBytes(*\_handle*, *\_buf*, *\_cnt*, *\_result*)  
*Write bytes to file.*
- #define [WriteBytesEx](#)(*\_handle*, *\_len*, *\_buf*, *\_result*) \_\_writeBytesEx(*\_handle*, *\_len*, *\_buf*, *\_result*)  
*Write bytes to a file with limit.*

#### 5.47.1 Detailed Description

Functions for accessing and modifying Loader module features.

#### 5.47.2 Macro Definition Documentation

##### 5.47.2.1 #define CloseFile( *\_handle*, *\_result* ) \_\_closeFile(*\_handle*, *\_result*)

Close a file.

Close the file associated with the specified file handle. The loader result code is returned as the value of the function call. The handle parameter must be a constant or a variable.

##### Parameters

<i>_handle</i>	The file handle.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

##### 5.47.2.2 #define CreateFile( *\_fname*, *\_fsize*, *\_handle*, *\_result* ) \_\_createFile(*\_fname*, *\_fsize*, *\_handle*, *\_result*)

Create a file.

Create a new file with the specified filename and size and open it for writing. The file handle is returned in the last parameter, which must be a variable. The loader result code is returned as the value of the function call. The filename and size parameters must be constants, constant expressions, or variables. A file created with a size of zero bytes cannot be written to since the NXC file writing functions do not grow the file if its capacity is exceeded during a write attempt.

##### Parameters

<i>_fname</i>	The name of the file to create.
<i>_fsize</i>	The size of the file.
<i>_handle</i>	The file handle output from the function call.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

##### 5.47.2.3 #define CreateFileLinear( *\_fname*, *\_fsize*, *\_handle*, *\_result* ) \_\_createFileLinear(*\_fname*, *\_fsize*, *\_handle*, *\_result*)

Create a linear file.

Create a new linear file with the specified filename and size and open it for writing. The file handle is returned in the last parameter, which must be a variable. The loader result code is returned as the value of the function call. The filename and size parameters must be constants, constant expressions, or variables. A file created with a size of zero bytes cannot be written to since the NXC file writing functions do not grow the file if its capacity is exceeded during a write attempt.

#### Parameters

<i>_fname</i>	The name of the file to create.
<i>_fsize</i>	The size of the file.
<i>_handle</i>	The file handle output from the function call.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

#### Warning

This function requires the enhanced NBC/NXC firmware.

#### 5.47.2.4 #define CreateFileNonLinear( *\_fname*, *\_fsize*, *\_handle*, *\_result* ) *\_createFileNonLinear*(*\_fname*, *\_fsize*, *\_handle*, *\_result*)

Create a non-linear file.

Create a new non-linear file with the specified filename and size and open it for writing. The file handle is returned in the last parameter, which must be a variable. The loader result code is returned as the value of the function call. The filename and size parameters must be constants, constant expressions, or variables. A file created with a size of zero bytes cannot be written to since the NXC file writing functions do not grow the file if its capacity is exceeded during a write attempt.

#### Parameters

<i>_fname</i>	The name of the file to create.
<i>_fsize</i>	The size of the file.
<i>_handle</i>	The file handle output from the function call.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

#### Warning

This function requires the enhanced NBC/NXC firmware.

#### 5.47.2.5 #define DeleteFile( *\_fname*, *\_result* ) *\_deleteFile*(*\_fname*, *\_result*)

Delete a file.

Delete the specified file. The loader result code is returned as the value of the function call. The filename parameter must be a constant or a variable.

#### Parameters

<i>_fname</i>	The name of the file to delete.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

#### 5.47.2.6 #define FindFirstFile( *\_fname*, *\_handle*, *\_result* ) *\_findFirstFile*(*\_fname*, *\_handle*, *\_result*)

Start searching for files.

This function lets you begin iterating through files stored on the NXT.

**Parameters**

<i>_fname</i>	On input this contains the filename pattern you are searching for. On output this contains the name of the first file found that matches the pattern.
<i>_handle</i>	The search handle input to and output from the function call.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.47.2.7 #define FindNextFile( *\_fname*, *\_handle*, *\_result* ) \_\_findNextFile(*\_fname*, *\_handle*, *\_result*)**

Continue searching for files.

This function lets you continue iterating through files stored on the NXT.

**Parameters**

<i>_fname</i>	On output this contains the name of the next file found that matches the pattern used when the search began by calling <a href="#">FindFirstFile</a> .
<i>_handle</i>	The search handle input to and output from the function call.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.47.2.8 #define GetFreeMemory( *\_value* ) \_\_GetFreeMemory(*\_value*)**

Get free flash memory.

Get the number of bytes of flash memory that are available for use.

**Parameters**

<i>_value</i>	The number of bytes of unused flash memory.
---------------	---

**5.47.2.9 #define OpenFileAppend( *\_fname*, *\_fsize*, *\_handle*, *\_result* ) \_\_openFileAppend(*\_fname*, *\_fsize*, *\_handle*, *\_result*)**

Open a file for appending.

Open an existing file with the specified filename for writing. The file size is returned in the second parameter, which must be a variable. The file handle is returned in the last parameter, which must be a variable. The loader result code is returned as the value of the function call. The filename parameter must be a constant or a variable.

**Parameters**

<i>_fname</i>	The name of the file to open.
<i>_fsize</i>	The size of the file returned by the function.
<i>_handle</i>	The file handle output from the function call.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

**5.47.2.10 #define OpenFileRead( \_fname, \_fsize, \_handle, \_result ) \_\_openFileRead(\_fname, \_fsize, \_handle, \_result)**

Open a file for reading.

Open an existing file with the specified filename for reading. The file size is returned in the second parameter, which must be a variable. The file handle is returned in the last parameter, which must be a variable. The loader result code is returned as the value of the function call. The filename parameter must be a constant or a variable.

**Parameters**

<u>_fname</u>	The name of the file to open.
<u>_fsize</u>	The size of the file returned by the function.
<u>_handle</u>	The file handle output from the function call.
<u>_result</u>	The function call result. See <a href="#">Loader module error codes</a> .

**5.47.2.11 #define OpenFileReadLinear( \_fname, \_fsize, \_handle, \_result ) \_\_openFileReadLinear(\_fname, \_fsize, \_handle, \_result)**

Open a linear file for reading.

Open an existing linear file with the specified filename for reading. The file size is returned in the second parameter, which must be a variable. The file handle is returned in the last parameter, which must be a variable. The loader result code is returned as the value of the function call. The filename parameter must be a constant or a variable.

**Parameters**

<u>_fname</u>	The name of the file to open.
<u>_fsize</u>	The size of the file returned by the function.
<u>_handle</u>	The file handle output from the function call.
<u>_result</u>	The function call result. See <a href="#">Loader module error codes</a> .

**Warning**

This function requires the enhanced NBC/NXC firmware.

**5.47.2.12 #define Read( \_handle, \_n, \_result ) \_\_readValue(\_handle, \_n, \_result)**

Read a value from a file.

Read a value from the file associated with the specified handle. The handle parameter must be a variable. The value parameter must be a variable. The type of the value parameter determines the number of bytes of data read.

**Parameters**

<u>_handle</u>	The file handle.
<u>_n</u>	The variable to store the data read from the file.
<u>_result</u>	The function call result. See <a href="#">Loader module error codes</a> .

**5.47.2.13 #define ReadBytes( \_handle, \_len, \_buf, \_result ) \_\_readBytes(\_handle, \_len, \_buf, \_result)**

Read bytes from a file.

Read the specified number of bytes from the file associated with the specified handle. The handle parameter must be a variable. The length parameter must be a variable. The buf parameter must be an array or a string variable. The actual number of bytes read is returned in the length parameter.

**Parameters**

<i>_handle</i>	The file handle.
<i>_len</i>	The number of bytes to read. Returns the number of bytes actually read.
<i>_buf</i>	The byte array where the data is stored on output.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

**5.47.2.14 #define ReadLn( *\_handle*, *\_n*, *\_result* ) \_\_readLnValue(*\_handle*, *\_n*, *\_result*)**

Read a value from a file plus line ending.

Read a value from the file associated with the specified handle. The handle parameter must be a variable. The value parameter must be a variable. The type of the value parameter determines the number of bytes of data read. The ReadLn function reads two additional bytes from the file which it assumes are a carriage return and line feed pair.

**Parameters**

<i>_handle</i>	The file handle.
<i>_n</i>	The variable to store the data read from the file.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

**5.47.2.15 #define ReadLnString( *\_handle*, *\_output*, *\_result* ) \_\_readLnString(*\_handle*, *\_output*, *\_result*)**

Read a string from a file plus line ending.

Read a string from the file associated with the specified handle. The handle parameter must be a variable. The output parameter must be a variable. Appends bytes to the output variable until a line ending (CRLF) is reached. The line ending is also read but it is not appended to the output parameter.

**Parameters**

<i>_handle</i>	The file handle.
<i>_output</i>	The variable to store the string read from the file.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

**5.47.2.16 #define RenameFile( *\_oldname*, *\_newname*, *\_result* ) \_\_renameFile(*\_oldname*, *\_newname*, *\_result*)**

Rename a file.

Rename a file from the old filename to the new filename. The loader param *\_result* code is returned as the value of the function call. The filename parameters must be constants or variables.

**Parameters**

<i>_oldname</i>	The old filename.
<i>_newname</i>	The new filename.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

**5.47.2.17 #define ResizeFile( *\_fname*, *\_newszie*, *\_result* ) \_\_fileResize(*\_fname*, *\_newszie*, *\_result*)**

Resize a file.

Resize the specified file. The loader result code is returned as the value of the function call. The filename parameter must be a constant or a variable.

## Parameters

<i>_fname</i>	The name of the file to resize.
<i>_newsize</i>	The new size for the file.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

5.47.2.18 #define ResolveHandle( *\_fname*, *\_handle*, *\_writable*, *\_result* ) \_\_resolveHandle(*\_fname*, *\_handle*, *\_writable*, *\_result*)

Resolve a handle.

Resolve a file handle from the specified filename. The file handle is returned in the second parameter, which must be a variable. A boolean value indicating whether the handle can be used to write to the file or not is returned in the last parameter, which must be a variable. The loader result code is returned as the value of the function call. The filename parameter must be a constant or a variable.

## Parameters

<i>_fname</i>	The name of the file for which to resolve a handle.
<i>_handle</i>	The file handle output from the function call.
<i>_writable</i>	A boolean flag indicating whether the handle is to a file open for writing (true) or reading (false).
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

5.47.2.19 #define SizeOf( *\_n*, *\_result* ) \_\_sizeOF(*\_n*, *\_result*)

Calculate the size of a variable.

Calculate the number of bytes required to store the contents of the variable passed into the function.

## Parameters

<i>_n</i>	The variable.
<i>_result</i>	The number of bytes occupied by the variable.

5.47.2.20 #define Write( *\_handle*, *\_n*, *\_result* ) \_\_writeValue(*\_handle*, *\_n*, *\_result*)

Write value to file.

Write a value to the file associated with the specified handle. The handle parameter must be a variable. The value parameter must be a constant, a constant expression, or a variable. The type of the value parameter determines the number of bytes of data written.

## Parameters

<i>_handle</i>	The file handle.
<i>_n</i>	The value to write to the file.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

5.47.2.21 #define WriteBytes( *\_handle*, *\_buf*, *\_cnt*, *\_result* ) \_\_writeBytes(*\_handle*, *\_buf*, *\_cnt*, *\_result*)

Write bytes to file.

Write the contents of the data array to the file associated with the specified handle. The handle parameter must be a variable. The cnt parameter must be a variable. The data parameter must be a byte array. The actual number of bytes written is returned in the cnt parameter.

**Parameters**

<i>_handle</i>	The file handle.
<i>_buf</i>	The byte array or string containing the data to write.
<i>_cnt</i>	The number of bytes actually written to the file.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

**5.47.2.22 #define WriteBytesEx( *\_handle*, *\_len*, *\_buf*, *\_result* ) \_\_writeBytesEx(*\_handle*, *\_len*, *\_buf*, *\_result*)**

Write bytes to a file with limit.

Write the specified number of bytes to the file associated with the specified handle. The handle parameter must be a variable. The len parameter must be a variable. The buf parameter must be a byte array or a string variable or string constant. The actual number of bytes written is returned in the len parameter.

**Parameters**

<i>_handle</i>	The file handle.
<i>_len</i>	The maximum number of bytes to write on input. Returns the actual number of bytes written.
<i>_buf</i>	The byte array or string containing the data to write.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

**5.47.2.23 #define WriteLn( *\_handle*, *\_n*, *\_result* ) \_\_writeLnValue(*\_handle*, *\_n*, *\_result*)**

Write a value and new line to a file.

Write a value to the file associated with the specified handle. The handle parameter must be a variable. The value parameter must be a constant, a constant expression, or a variable. The type of the value parameter determines the number of bytes of data written. This function also writes a carriage return and a line feed to the file following the numeric data.

**Parameters**

<i>_handle</i>	The file handle.
<i>_n</i>	The value to write to the file.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

**5.47.2.24 #define WriteLnString( *\_handle*, *\_str*, *\_cnt*, *\_result* ) \_\_writeLnString(*\_handle*, *\_str*, *\_cnt*, *\_result*)**

Write string and new line to a file.

Write the string to the file associated with the specified handle. The handle parameter must be a variable. The count parameter must be a variable. The str parameter must be a string variable or string constant. This function also writes a carriage return and a line feed to the file following the string data. The total number of bytes written is returned in the cnt parameter.

**Parameters**

<i>_handle</i>	The file handle.
<i>_str</i>	The string to write to the file.
<i>_cnt</i>	The number of bytes actually written to the file.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

**5.47.2.25 #define WriteString( *\_handle*, *\_str*, *\_cnt*, *\_result* ) \_\_writeString(*\_handle*, *\_str*, *\_cnt*, *\_result*)**

Write string to a file.

Write the string to the file associated with the specified handle. The handle parameter must be a variable. The count parameter must be a variable. The str parameter must be a string variable or string constant. The actual number of bytes written is returned in the cnt parameter.

**Parameters**

<i>_handle</i>	The file handle.
<i>_str</i>	The string to write to the file.
<i>_cnt</i>	The number of bytes actually written to the file.
<i>_result</i>	The function call result. See <a href="#">Loader module error codes</a> .

## 5.48 cstdlib API

Standard C cstdlib API functions.

### Macros

- `#define Random(_arg, _max) __Random(_arg,_max)`  
*Generate an unsigned random number.*
- `#define SignedRandom(_arg) __SignedRandom(_arg)`  
*Generate signed random number.*

### 5.48.1 Detailed Description

Standard C cstdlib API functions.

### 5.48.2 Macro Definition Documentation

#### 5.48.2.1 `#define Random( _arg, _max ) __Random(_arg,_max)`

Generate an unsigned random number.

Return an unsigned 16-bit random number. The returned value will range between 0 and n (exclusive).

##### Parameters

<code>_arg</code>	An unsigned random number.
<code>_max</code>	The maximum unsigned value desired.

#### 5.48.2.2 `#define SignedRandom( _arg ) __SignedRandom(_arg)`

Generate signed random number.

Return a signed 16-bit random number.

##### Parameters

<code>_arg</code>	A signed random number
-------------------	------------------------

## 5.49 cmath API

Standard C cmath API functions.

### Macros

- `#define bcd2dec(_bcd, _result) __bcd2dec(_bcd, _result)`

*Convert from BCD to decimal Return the decimal equivalent of the binary coded decimal value provided.*

### 5.49.1 Detailed Description

Standard C cmath API functions.

### 5.49.2 Macro Definition Documentation

#### 5.49.2.1 `#define bcd2dec( _bcd, _result ) __bcd2dec(_bcd, _result)`

Convert from BCD to decimal Return the decimal equivalent of the binary coded decimal value provided.

#### Parameters

<code>_bcd</code>	The value you want to convert from bcd to decimal.
<code>_result</code>	The decimal equivalent of the binary coded decimal byte.

## 5.50 Property constants

Use these constants for specifying the property for the GetProperty and SetProperty direct commands.

### Macros

- #define RC\_PROP\_BTONEOFF 0x0
- #define RC\_PROP\_SOUND\_LEVEL 0x1
- #define RC\_PROP\_SLEEP\_TIMEOUT 0x2
- #define RC\_PROP\_DEBUGGING 0xF

### 5.50.1 Detailed Description

Use these constants for specifying the property for the GetProperty and SetProperty direct commands.

### 5.50.2 Macro Definition Documentation

#### 5.50.2.1 #define RC\_PROP\_BTONEOFF 0x0

Set/get whether bluetooth is on or off

#### 5.50.2.2 #define RC\_PROP\_DEBUGGING 0xF

Set/get enhanced firmware debugging information (NBC/NXC)

#### 5.50.2.3 #define RC\_PROP\_SLEEP\_TIMEOUT 0x2

Set/get the NXT sleep timeout value (times 60000)

#### 5.50.2.4 #define RC\_PROP\_SOUND\_LEVEL 0x1

Set/get the NXT sound level

## 5.51 Variable type constants

Use these constants for testing a variable type.

### Macros

- #define VT\_UBYTE 0x01
- #define VT\_SBYTE 0x02
- #define VT\_UWORD 0x03
- #define VT\_SWORD 0x04
- #define VT ULONG 0x05
- #define VT\_SLONG 0x06
- #define VT\_STRUCT 0x08
- #define VT\_MUTEX 0x09
- #define VT\_FLOAT 0x0A
- #define VT\_A1\_UBYTE 0x11
- #define VT\_A1\_SBYTE 0x12
- #define VT\_A1\_UWORD 0x13
- #define VT\_A1\_SWORD 0x14
- #define VT\_A1 ULONG 0x15
- #define VT\_A1\_SLONG 0x16
- #define VT\_A1\_STRUCT 0x17
- #define VT\_A1\_FLOAT 0x1A
- #define VT\_A2\_UBYTE 0x21
- #define VT\_A2\_SBYTE 0x22
- #define VT\_A2\_UWORD 0x23
- #define VT\_A2\_SWORD 0x24
- #define VT\_A2 ULONG 0x25
- #define VT\_A2\_SLONG 0x26
- #define VT\_A2\_STRUCT 0x27
- #define VT\_A2\_FLOAT 0x2A
- #define VT\_ARRAY\_MASK 0xF0

### 5.51.1 Detailed Description

Use these constants for testing a variable type.

### 5.51.2 Macro Definition Documentation

#### 5.51.2.1 #define VT\_A1\_FLOAT 0x1A

Variable type 1d array of float

#### 5.51.2.2 #define VT\_A1\_SBYTE 0x12

Variable type 1d array of signed byte

#### 5.51.2.3 #define VT\_A1\_SLONG 0x16

Variable type 1d array of signed long

5.51.2.4 #define VT\_A1\_STRUCT 0x17

Variable type 1d array of structure

5.51.2.5 #define VT\_A1\_SWORD 0x14

Variable type 1d array of signed word

5.51.2.6 #define VT\_A1\_UBYTE 0x11

Variable type 1d array of unsigned byte

5.51.2.7 #define VT\_A1 ULONG 0x15

Variable type 1d array of unsigned long

5.51.2.8 #define VT\_A1\_UWORD 0x13

Variable type 1d array of unsigned word

5.51.2.9 #define VT\_A2\_FLOAT 0x2A

Variable type 2d array of float

5.51.2.10 #define VT\_A2\_SBYTE 0x22

Variable type 2d array of signed byte

5.51.2.11 #define VT\_A2\_SLONG 0x26

Variable type 2d array of signed long

5.51.2.12 #define VT\_A2\_STRUCT 0x27

Variable type 2d array of structure

5.51.2.13 #define VT\_A2\_SWORD 0x24

Variable type 2d array of signed word

5.51.2.14 #define VT\_A2\_UBYTE 0x21

Variable type 2d array of unsigned byte

5.51.2.15 #define VT\_A2 ULONG 0x25

Variable type 2d array of unsigned long

5.51.2.16 #define VT\_A2\_UWORD 0x23

Variable type 2d array of unsigned word

5.51.2.17 #define VT\_ARRAY\_MASK 0xF0

Variable type array mask

5.51.2.18 #define VT\_FLOAT 0x0A

Variable type float

5.51.2.19 #define VT\_MUTEX 0x09

Variable type mutex

5.51.2.20 #define VT\_SBYTE 0x02

Variable type signed byte

5.51.2.21 #define VT\_SLONG 0x06

Variable type signed long

5.51.2.22 #define VT\_STRUCT 0x08

Variable type structure

5.51.2.23 #define VT\_SWORD 0x04

Variable type signed word

5.51.2.24 #define VT\_UBYTE 0x01

Variable type unsigned byte

5.51.2.25 #define VT ULONG 0x05

Variable type unsigned long

5.51.2.26 #define VT\_UWORD 0x03

Variable type unsigned word

## 5.52 Array operation constants

Constants for use with the NXC ArrayOp function and the NBC arrop statement.

### Macros

- #define OPARR\_SUM 0x00
- #define OPARR\_MEAN 0x01
- #define OPARR\_SUMSQR 0x02
- #define OPARR\_STD 0x03
- #define OPARR\_MIN 0x04
- #define OPARR\_MAX 0x05
- #define OPARR\_SORT 0x06
- #define OPARR\_TOUPPER 0x07
- #define OPARR\_TOLOWER 0x08

### 5.52.1 Detailed Description

Constants for use with the NXC ArrayOp function and the NBC arrop statement.

### 5.52.2 Macro Definition Documentation

#### 5.52.2.1 #define OPARR\_MAX 0x05

Calculate the maximum value of the elements in the numeric input array

#### 5.52.2.2 #define OPARR\_MEAN 0x01

Calculate the mean value for the elements in the numeric input array

#### 5.52.2.3 #define OPARR\_MIN 0x04

Calculate the minimum value of the elements in the numeric input array

#### 5.52.2.4 #define OPARR\_SORT 0x06

Sort the elements in the numeric input array

#### 5.52.2.5 #define OPARR\_STD 0x03

Calculate the standard deviation of the elements in the numeric input array

#### 5.52.2.6 #define OPARR\_SUM 0x00

Calculate the sum of the elements in the numeric input array

#### 5.52.2.7 #define OPARR\_SUMSQR 0x02

Calculate the sum of the squares of the elements in the numeric input array

#### 5.52.2.8 #define OPARR\_TOLOWER 0x08

Lowercase the input string

## 5.52.2.9 #define OPARR\_TOUPPER 0x07

Uppercase the input string

## 5.53 System Call function constants

Constants for use in the SysCall() function or NBC syscall statement.

### Macros

- #define FileOpenRead 0
- #define FileOpenWrite 1
- #define FileOpenAppend 2
- #define FileRead 3
- #define FileWrite 4
- #define FileClose 5
- #define FileResolveHandle 6
- #define FileRename 7
- #define FileDelete 8
- #define SoundPlayFile 9
- #define SoundPlayTone 10
- #define SoundGetState 11
- #define SoundSetState 12
- #define DrawText 13
- #define DrawPoint 14
- #define DrawLine 15
- #define DrawCircle 16
- #define DrawRect 17
- #define DrawGraphic 18
- #define SetScreenMode 19
- #define ReadButton 20
- #define CommLSWrite 21
- #define CommLSRead 22
- #define CommLSCheckStatus 23
- #define RandomNumber 24
- #define GetStartTick 25
- #define MessageWrite 26
- #define MessageRead 27
- #define CommBTCheckStatus 28
- #define CommBTWrite 29
- #define CommBTRead 30
- #define KeepAlive 31
- #define IOMapRead 32
- #define IOMapWrite 33
- #define ColorSensorRead 34
- #define CommBTOffOn 35
- #define CommBTConnection 36
- #define CommHSWrite 37
- #define CommHSRead 38
- #define CommHSCheckStatus 39
- #define ReadSemData 40
- #define WriteSemData 41
- #define ComputeCalibValue 42
- #define UpdateCalibCacheInfo 43

- #define DatalogWrite 44
- #define DatalogGetTimes 45
- #define SetSleepTimeoutVal 46
- #define ListFiles 47
- #define InputPinFunction 77
- #define IOMapReadByID 78
- #define IOMapWriteByID 79
- #define DisplayExecuteFunction 80
- #define CommExecuteFunction 81
- #define LoaderExecuteFunction 82
- #define FileFindFirst 83
- #define FileFindNext 84
- #define FileOpenWriteLinear 85
- #define FileOpenWriteNonLinear 86
- #define FileOpenReadLinear 87
- #define CommHSCtrl 88
- #define CommLSWriteEx 89
- #define FileSeek 90
- #define FileResize 91
- #define DrawGraphicArray 92
- #define DrawPolygon 93
- #define DrawEllipse 94
- #define DrawFont 95
- #define MemoryManager 96
- #define ReadLastResponse 97
- #define FileTell 98
- #define RandomEx 99

### 5.53.1 Detailed Description

Constants for use in the SysCall() function or NBC syscall statement.

### 5.53.2 Macro Definition Documentation

#### 5.53.2.1 #define ColorSensorRead 34

Read data from the NXT 2.0 color sensor

#### 5.53.2.2 #define CommBTCheckStatus 28

Check the bluetooth status

#### 5.53.2.3 #define CommBTConnection 36

Connect or disconnect to a known bluetooth device

#### 5.53.2.4 #define CommBTOff 35

Turn the bluetooth radio on or off

#### 5.53.2.5 #define CommBTRead 30

Read from a bluetooth connection

5.53.2.6 #define CommBTWrite 29

Write to a bluetooth connections

5.53.2.7 #define CommExecuteFunction 81

Execute one of the Comm module's internal functions

5.53.2.8 #define CommHSCheckStatus 39

Check the status of the hi-speed port

5.53.2.9 #define CommHSControl 88

Control the hi-speed port

5.53.2.10 #define CommHSRead 38

Read data from the hi-speed port

5.53.2.11 #define CommHSWrite 37

Write data to the hi-speed port

5.53.2.12 #define CommLSCheckStatus 23

Check the status of a lowspeed (aka I2C) device

5.53.2.13 #define CommLSRead 22

Read from a lowspeed (aka I2C) device

5.53.2.14 #define CommLSWrite 21

Write to a lowspeed (aka I2C) device

5.53.2.15 #define CommLSWriteEx 89

Write to a lowspeed (aka I2C) device with optional restart on read

5.53.2.16 #define ComputeCalibValue 42

Compute a calibration value

5.53.2.17 #define DatalogGetTimes 45

Get datalog timing information

5.53.2.18 #define DatalogWrite 44

Write to the datalog

5.53.2.19 #define DisplayExecuteFunction 80

Execute one of the Display module's internal functions

5.53.2.20 #define DrawCircle 16

Draw a circle on the LCD screen

5.53.2.21 #define DrawEllipse 94

Draw an ellipse on the LCD screen

5.53.2.22 #define DrawFont 95

Draw text using a custom RIC-based font to the LCD screen

5.53.2.23 #define DrawGraphic 18

Draw a graphic image on the LCD screen

5.53.2.24 #define DrawGraphicArray 92

Draw a graphic image from a byte array to the LCD screen

5.53.2.25 #define DrawLine 15

Draw a line on the LCD screen

5.53.2.26 #define DrawPoint 14

Draw a single pixel on the LCD screen

5.53.2.27 #define DrawPolygon 93

Draw a polygon on the LCD screen

5.53.2.28 #define DrawRect 17

Draw a rectangle on the LCD screen

5.53.2.29 #define DrawText 13

Draw text to one of 8 LCD lines

5.53.2.30 #define FileClose 5

Close the specified file

5.53.2.31 #define FileDelete 8

Delete a file

5.53.2.32 #define FileFindFirst 83

Start a search for a file using a filename pattern

5.53.2.33 #define FileFindNext 84

Continue searching for a file

5.53.2.34 #define FileOpenAppend 2

Open a file for appending to the end of the file

5.53.2.35 #define FileOpenRead 0

Open a file for reading

5.53.2.36 #define FileOpenReadLinear 87

Open a linear file for reading

5.53.2.37 #define FileOpenWrite 1

Open a file for writing (creates a new file)

5.53.2.38 #define FileOpenWriteLinear 85

Open a linear file for writing

5.53.2.39 #define FileOpenWriteNonLinear 86

Open a non-linear file for writing

5.53.2.40 #define FileRead 3

Read from the specified file

5.53.2.41 #define FileRename 7

Rename a file

5.53.2.42 #define FileResize 91

Resize a file (not yet implemented)

5.53.2.43 #define FileResolveHandle 6

Get a file handle for the specified filename if it is already open

5.53.2.44 #define FileSeek 90

Seek to a specific position in an open file

5.53.2.45 #define FileTell 98

Return the current file position in an open file

5.53.2.46 #define FileWrite 4

Write to the specified file

5.53.2.47 #define GetStartTick 25

Get the current system tick count

**5.53.2.48 #define InputPinFunction 77**

Execute the Input module's pin function

**5.53.2.49 #define IOMapRead 32**

Read data from one of the firmware module's IOMap structures using the module's name

**5.53.2.50 #define IOMapReadByID 78**

Read data from one of the firmware module's IOMap structures using the module's ID

**5.53.2.51 #define IOMapWrite 33**

Write data to one of the firmware module's IOMap structures using the module's name

**5.53.2.52 #define IOMapWriteByID 79**

Write data to one of the firmware module's IOMap structures using the module's ID

**5.53.2.53 #define KeepAlive 31**

Reset the NXT sleep timer

**5.53.2.54 #define ListFiles 47**

List files that match the specified filename pattern

**5.53.2.55 #define LoaderExecuteFunction 82**

Execute one of the Loader module's internal functions

**5.53.2.56 #define MemoryManager 96**

Read memory manager information, optionally compacting the dataspace first

**5.53.2.57 #define MessageRead 27**

Read a message from a mailbox

**5.53.2.58 #define MessageWrite 26**

Write a message to a mailbox

**5.53.2.59 #define RandomEx 99**

Generate a random number or seed the RNG.

**5.53.2.60 #define RandomNumber 24**

Generate a random number

**5.53.2.61 #define ReadButton 20**

Read the current button state

**5.53.2.62 #define ReadLastResponse 97**

Read the last response packet received by the NXT. Optionally clear the value after reading it.

**5.53.2.63 #define ReadSemData 40**

Read motor semaphore data

**5.53.2.64 #define SetScreenMode 19**

Set the screen mode

**5.53.2.65 #define SetSleepTimeoutVal 46**

Set the NXT sleep timeout value

**5.53.2.66 #define SoundGetState 11**

Get the current sound module state

**5.53.2.67 #define SoundPlayFile 9**

Play a sound or melody file

**5.53.2.68 #define SoundPlayTone 10**

Play a simple tone with the specified frequency and duration

**5.53.2.69 #define SoundSetState 12**

Set the sound module state

**5.53.2.70 #define UpdateCalibCacheInfo 43**

Update sensor calibration cache information

**5.53.2.71 #define WriteSemData 41**

Write motor semaphore data

## 5.54 Line number constants

Line numbers for use with DrawText system function.

### Macros

- #define LCD\_LINE8 0
- #define LCD\_LINE7 8
- #define LCD\_LINE6 16
- #define LCD\_LINE5 24
- #define LCD\_LINE4 32
- #define LCD\_LINE3 40
- #define LCD\_LINE2 48
- #define LCD\_LINE1 56

### 5.54.1 Detailed Description

Line numbers for use with DrawText system function.

### See Also

[SysDrawText\(\)](#), [TextOut\(\)](#), [NumOut\(\)](#)

### 5.54.2 Macro Definition Documentation

#### 5.54.2.1 #define LCD\_LINE1 56

The 1st line of the LCD screen

#### 5.54.2.2 #define LCD\_LINE2 48

The 2nd line of the LCD screen

#### 5.54.2.3 #define LCD\_LINE3 40

The 3rd line of the LCD screen

#### 5.54.2.4 #define LCD\_LINE4 32

The 4th line of the LCD screen

#### 5.54.2.5 #define LCD\_LINE5 24

The 5th line of the LCD screen

#### 5.54.2.6 #define LCD\_LINE6 16

The 6th line of the LCD screen

#### 5.54.2.7 #define LCD\_LINE7 8

The 7th line of the LCD screen

#### 5.54.2.8 #define LCD\_LINE8 0

The 8th line of the LCD screen

## 5.55 Time constants

Constants for use with the [Wait\(\)](#) function.

### Macros

- #define MS\_1 1
- #define MS\_2 2
- #define MS\_3 3
- #define MS\_4 4
- #define MS\_5 5
- #define MS\_6 6
- #define MS\_7 7
- #define MS\_8 8
- #define MS\_9 9
- #define MS\_10 10
- #define MS\_20 20
- #define MS\_30 30
- #define MS\_40 40
- #define MS\_50 50
- #define MS\_60 60
- #define MS\_70 70
- #define MS\_80 80
- #define MS\_90 90
- #define MS\_100 100
- #define MS\_150 150
- #define MS\_200 200
- #define MS\_250 250
- #define MS\_300 300
- #define MS\_350 350
- #define MS\_400 400
- #define MS\_450 450
- #define MS\_500 500
- #define MS\_600 600
- #define MS\_700 700
- #define MS\_800 800
- #define MS\_900 900
- #define SEC\_1 1000
- #define SEC\_2 2000
- #define SEC\_3 3000
- #define SEC\_4 4000
- #define SEC\_5 5000
- #define SEC\_6 6000
- #define SEC\_7 7000
- #define SEC\_8 8000
- #define SEC\_9 9000
- #define SEC\_10 10000
- #define SEC\_15 15000
- #define SEC\_20 20000
- #define SEC\_30 30000

- #define MIN\_1 60000
- #define NOTE\_WHOLE 1000
- #define NOTE\_HALF (NOTE\_WHOLE/2)
- #define NOTE\_QUARTER (NOTE\_WHOLE/4)
- #define NOTE\_EIGHT (NOTE\_WHOLE/8)
- #define NOTE\_SIXTEEN (NOTE\_WHOLE/16)

### 5.55.1 Detailed Description

Constants for use with the [Wait\(\)](#) function.

#### See Also

[Wait\(\)](#)

### 5.55.2 Macro Definition Documentation

#### 5.55.2.1 #define MIN\_1 60000

1 minute

#### 5.55.2.2 #define MS\_1 1

1 millisecond

#### 5.55.2.3 #define MS\_10 10

10 milliseconds

#### 5.55.2.4 #define MS\_100 100

100 milliseconds

#### 5.55.2.5 #define MS\_150 150

150 milliseconds

#### 5.55.2.6 #define MS\_2 2

2 milliseconds

#### 5.55.2.7 #define MS\_20 20

20 milliseconds

#### 5.55.2.8 #define MS\_200 200

200 milliseconds

#### 5.55.2.9 #define MS\_250 250

250 milliseconds

#### 5.55.2.10 #define MS\_3 3

3 milliseconds

5.55.2.11 #define MS\_30 30

30 milliseconds

5.55.2.12 #define MS\_300 300

300 milliseconds

5.55.2.13 #define MS\_350 350

350 milliseconds

5.55.2.14 #define MS\_4 4

4 milliseconds

5.55.2.15 #define MS\_40 40

40 milliseconds

5.55.2.16 #define MS\_400 400

400 milliseconds

5.55.2.17 #define MS\_450 450

450 milliseconds

5.55.2.18 #define MS\_5 5

5 milliseconds

5.55.2.19 #define MS\_50 50

50 milliseconds

5.55.2.20 #define MS\_500 500

500 milliseconds

5.55.2.21 #define MS\_6 6

6 milliseconds

5.55.2.22 #define MS\_60 60

60 milliseconds

5.55.2.23 #define MS\_600 600

600 milliseconds

5.55.2.24 #define MS\_7 7

7 milliseconds

5.55.2.25 #define MS\_70 70

70 milliseconds

5.55.2.26 #define MS\_700 700

700 milliseconds

5.55.2.27 #define MS\_8 8

8 milliseconds

5.55.2.28 #define MS\_80 80

80 milliseconds

5.55.2.29 #define MS\_800 800

800 milliseconds

5.55.2.30 #define MS\_9 9

9 milliseconds

5.55.2.31 #define MS\_90 90

90 milliseconds

5.55.2.32 #define MS\_900 900

900 milliseconds

5.55.2.33 #define NOTE\_EIGHT (NOTE\_WHOLE/8)

The duration of an eighth note (ms)

5.55.2.34 #define NOTE\_HALF (NOTE\_WHOLE/2)

The duration of a half note (ms)

5.55.2.35 #define NOTE\_QUARTER (NOTE\_WHOLE/4)

The duration of a quarter note (ms)

5.55.2.36 #define NOTE\_SIXTEEN (NOTE\_WHOLE/16)

The duration of a sixteenth note (ms)

5.55.2.37 #define NOTE\_WHOLE 1000

The duration of a whole note (ms)

5.55.2.38 #define SEC\_1 1000

1 second

5.55.2.39 #define SEC\_10 10000

10 seconds

5.55.2.40 #define SEC\_15 15000

15 seconds

5.55.2.41 #define SEC\_2 2000

2 seconds

5.55.2.42 #define SEC\_20 20000

20 seconds

5.55.2.43 #define SEC\_3 3000

3 seconds

5.55.2.44 #define SEC\_30 30000

30 seconds

5.55.2.45 #define SEC\_4 4000

4 seconds

5.55.2.46 #define SEC\_5 5000

5 seconds

5.55.2.47 #define SEC\_6 6000

6 seconds

5.55.2.48 #define SEC\_7 7000

7 seconds

5.55.2.49 #define SEC\_8 8000

8 seconds

5.55.2.50 #define SEC\_9 9000

9 seconds

## 5.56 Mailbox constants

Mailbox number constants should be used to avoid confusing NXT-G users.

### Macros

- `#define MAILBOX1 0`
- `#define MAILBOX2 1`
- `#define MAILBOX3 2`
- `#define MAILBOX4 3`
- `#define MAILBOX5 4`
- `#define MAILBOX6 5`
- `#define MAILBOX7 6`
- `#define MAILBOX8 7`
- `#define MAILBOX9 8`
- `#define MAILBOX10 9`

### 5.56.1 Detailed Description

Mailbox number constants should be used to avoid confusing NXT-G users.

### See Also

[SysMessageWrite\(\)](#), [SysMessageRead\(\)](#), [SendMessage\(\)](#), [ReceiveMessage\(\)](#), [SendRemoteBool\(\)](#), [SendRemoteNumber\(\)](#), [SendRemoteString\(\)](#), [SendResponseBool\(\)](#), [SendResponseNumber\(\)](#), [SendResponseString\(\)](#), [ReceiveRemoteBool\(\)](#), [ReceiveRemoteNumber\(\)](#), [ReceiveRemoteString\(\)](#), [ReceiveRemoteMessageEx\(\)](#), [RemoteMessageRead\(\)](#), [RemoteMessageWrite\(\)](#)

### 5.56.2 Macro Definition Documentation

#### 5.56.2.1 `#define MAILBOX1 0`

Mailbox number 1

#### 5.56.2.2 `#define MAILBOX10 9`

Mailbox number 10

#### 5.56.2.3 `#define MAILBOX2 1`

Mailbox number 2

#### 5.56.2.4 `#define MAILBOX3 2`

Mailbox number 3

#### 5.56.2.5 `#define MAILBOX4 3`

Mailbox number 4

#### 5.56.2.6 `#define MAILBOX5 4`

Mailbox number 5

5.56.2.7 #define MAILBOX6 5

Mailbox number 6

5.56.2.8 #define MAILBOX7 6

Mailbox number 7

5.56.2.9 #define MAILBOX8 7

Mailbox number 8

5.56.2.10 #define MAILBOX9 8

Mailbox number 9

## 5.57 VM state constants

Constants defining possible VM states.

### Macros

- #define TIMES\_UP 6
- #define ROTATE\_QUEUE 5
- #define STOP\_REQ 4
- #define BREAKOUT\_REQ 3
- #define CLUMP\_SUSPEND 2
- #define CLUMP\_DONE 1

### 5.57.1 Detailed Description

Constants defining possible VM states.

### 5.57.2 Macro Definition Documentation

#### 5.57.2.1 #define BREAKOUT\_REQ 3

VM should break out of current thread

#### 5.57.2.2 #define CLUMP\_DONE 1

VM has finished executing thread

#### 5.57.2.3 #define CLUMP\_SUSPEND 2

VM should suspend thread

#### 5.57.2.4 #define ROTATE\_QUEUE 5

VM should rotate queue

#### 5.57.2.5 #define STOP\_REQ 4

VM should stop executing program

#### 5.57.2.6 #define TIMES\_UP 6

VM time is up

## 5.58 Fatal errors

Constants defining various fatal error conditions.

### Macros

- #define `ERR_ARG` -1
- #define `ERR_INSTR` -2
- #define `ERR_FILE` -3
- #define `ERR_VER` -4
- #define `ERR_MEM` -5
- #define `ERR_BAD_PTR` -6
- #define `ERR_CLUMP_COUNT` -7
- #define `ERR_NO_CODE` -8
- #define `ERR_INSANE_OFFSET` -9
- #define `ERR_BAD_POOL_SIZE` -10
- #define `ERR_LOADER_ERR` -11
- #define `ERR_SPOTCHECK_FAIL` -12
- #define `ERR_NO_ACTIVE_CLUMP` -13
- #define `ERR_DEFAULT_OFFSETS` -14
- #define `ERR_MEMMGR_FAIL` -15
- #define `ERR_NON_FATAL` -16

### 5.58.1 Detailed Description

Constants defining various fatal error conditions.

### 5.58.2 Macro Definition Documentation

#### 5.58.2.1 #define `ERR_ARG` -1

0xFF Bad arguments

#### 5.58.2.2 #define `ERR_BAD_POOL_SIZE` -10

0xF6 VarsCmd.PoolSize > POOL\_MAX\_SIZE

#### 5.58.2.3 #define `ERR_BAD_PTR` -6

0xFA Someone passed us a bad pointer!

#### 5.58.2.4 #define `ERR_CLUMP_COUNT` -7

0xF9 (FileClumpCount == 0 || FileClumpCount >= NOT\_A\_CLUMP)

#### 5.58.2.5 #define `ERR_DEFAULT_OFFSETS` -14

0xF2 (DefaultsOffset != FileOffsets.DynamicDefaults) || (DefaultsOffset + FileOffsets.DynamicDefaultsSize != FileOffsets.DSDefaultsSize)

#### 5.58.2.6 #define `ERR_FILE` -3

0xFD Malformed file contents

5.58.2.7 #define **ERR\_INSANE\_OFFSET** -9

0xF7 CurrOffset != (DataSize - VarsCmd.CodespaceCount \* 2)

5.58.2.8 #define **ERR\_INSTR** -2

0xFE Illegal bytecode instruction

5.58.2.9 #define **ERR\_LOADER\_ERR** -11

0xF5 LOADER\_ERR(LStatus) != SUCCESS || pData == NULL || DataSize == 0

5.58.2.10 #define **ERR\_MEM** -5

0xFB Insufficient memory available

5.58.2.11 #define **ERR\_MEMMGR\_FAIL** -15

0xF1 ((UBYTE \*)VarsCmd.MemMgr.pDopeVectorArray != VarsCmd.pDataspace + DV\_ARRAY[0].Offset

5.58.2.12 #define **ERR\_NO\_ACTIVE\_CLUMP** -13

0xF3 VarsCmd.RunQ.Head == NOT\_A\_CLUMP

5.58.2.13 #define **ERR\_NO\_CODE** -8

0xF8 VarsCmd.CodespaceCount == 0

5.58.2.14 #define **ERR\_NON\_FATAL** -16

Fatal errors are greater than this value

5.58.2.15 #define **ERR\_SPOTCHECK\_FAIL** -12

0xF4 ((UBYTE\*)(VarsCmd.pCodespace) < pData) (c\_cmd.c 1893)

5.58.2.16 #define **ERR\_VER** -4

0xFC Version mismatch between firmware and compiler

## 5.59 General errors

Constants defining general error conditions.

### Macros

- #define `ERR_INVALID_PORT` -16
- #define `ERR_INVALID_FIELD` -17
- #define `ERR_INVALID_QUEUE` -18
- #define `ERR_INVALID_SIZE` -19
- #define `ERR_NO_PROG` -20

### 5.59.1 Detailed Description

Constants defining general error conditions.

### 5.59.2 Macro Definition Documentation

#### 5.59.2.1 #define `ERR_INVALID_FIELD` -17

0xEF Attempted to access invalid field of a structure

#### 5.59.2.2 #define `ERR_INVALID_PORT` -16

0xF0 Bad input or output port specified

#### 5.59.2.3 #define `ERR_INVALID_QUEUE` -18

0xEE Illegal queue ID specified

#### 5.59.2.4 #define `ERR_INVALID_SIZE` -19

0xED Illegal size specified

#### 5.59.2.5 #define `ERR_NO_PROG` -20

0xEC No active program

## 5.60 Communications specific errors

Constants defining communication error conditions.

### Macros

- #define `ERR_COMM_CHAN_NOT_READY` -32
- #define `ERR_COMM_CHAN_INVALID` -33
- #define `ERR_COMM_BUFFER_FULL` -34
- #define `ERR_COMM_BUS_ERR` -35

### 5.60.1 Detailed Description

Constants defining communication error conditions.

### 5.60.2 Macro Definition Documentation

#### 5.60.2.1 #define `ERR_COMM_BUFFER_FULL` -34

0xDE No room in comm buffer

#### 5.60.2.2 #define `ERR_COMM_BUS_ERR` -35

0xDD Something went wrong on the communications bus

#### 5.60.2.3 #define `ERR_COMM_CHAN_INVALID` -33

0xDF Specified channel/connection is not valid

#### 5.60.2.4 #define `ERR_COMM_CHAN_NOT_READY` -32

0xE0 Specified channel/connection not configured or busy

## 5.61 Remote control (direct commands) errors

Constants defining errors that can occur during remote control (RC) direct command operations.

### Macros

- #define `ERR_RC_ILLEGAL_VAL` -64
- #define `ERR_RC_BAD_PACKET` -65
- #define `ERR_RC_UNKNOWN_CMD` -66
- #define `ERR_RC_FAILED` -67

### 5.61.1 Detailed Description

Constants defining errors that can occur during remote control (RC) direct command operations.

### 5.61.2 Macro Definition Documentation

#### 5.61.2.1 #define `ERR_RC_BAD_PACKET` -65

0xBF Clearly insane packet

#### 5.61.2.2 #define `ERR_RC_FAILED` -67

0xBD Request failed (i.e. specified file not found)

#### 5.61.2.3 #define `ERR_RC_ILLEGAL_VAL` -64

0xC0 Data contains out-of-range values

#### 5.61.2.4 #define `ERR_RC_UNKNOWN_CMD` -66

0xBE Unknown command opcode

## 5.62 Program status constants

Constants defining various states of the command module virtual machine.

### Macros

- #define PROG\_IDLE 0
- #define PROG\_OK 1
- #define PROG\_RUNNING 2
- #define PROG\_ERROR 3
- #define PROG\_ABORT 4
- #define PROG\_RESET 5

### 5.62.1 Detailed Description

Constants defining various states of the command module virtual machine.

### 5.62.2 Macro Definition Documentation

#### 5.62.2.1 #define PROG\_ABORT 4

Program has been aborted

#### 5.62.2.2 #define PROG\_ERROR 3

A program error has occurred

#### 5.62.2.3 #define PROG\_IDLE 0

Program state is idle

#### 5.62.2.4 #define PROG\_OK 1

Program state is okay

#### 5.62.2.5 #define PROG\_RESET 5

Program has been reset

#### 5.62.2.6 #define PROG\_RUNNING 2

Program is running

## 5.63 Command module IOMAP offsets

Constant offsets into the Command module IOMAP structure.

### Macros

- #define CommandOffsetFormatString 0
- #define CommandOffsetPRCHandler 16
- #define CommandOffsetTick 20
- #define CommandOffsetOffsetDS 24
- #define CommandOffsetOffsetDVA 26
- #define CommandOffsetProgStatus 28
- #define CommandOffsetAwake 29
- #define CommandOffsetActivateFlag 30
- #define CommandOffsetDeactivateFlag 31
- #define CommandOffsetFileName 32
- #define CommandOffsetMemoryPool 52
- #define CommandOffsetSyncTime 32820
- #define CommandOffsetSyncTick 32824

### 5.63.1 Detailed Description

Constant offsets into the Command module IOMAP structure.

### 5.63.2 Macro Definition Documentation

#### 5.63.2.1 #define CommandOffsetActivateFlag 30

Offset to the activate flag

#### 5.63.2.2 #define CommandOffsetAwake 29

Offset to the VM's awake state

#### 5.63.2.3 #define CommandOffsetDeactivateFlag 31

Offset to the deactivate flag

#### 5.63.2.4 #define CommandOffsetFileName 32

Offset to the running program's filename

#### 5.63.2.5 #define CommandOffsetFormatString 0

Offset to the format string

#### 5.63.2.6 #define CommandOffsetMemoryPool 52

Offset to the VM's memory pool

#### 5.63.2.7 #define CommandOffsetOffsetDS 24

Offset to the running program's data space (DS)

**5.63.2.8 #define CommandOffsetOffsetDVA 26**

Offset to the running program's DOPE vector address (DVA)

**5.63.2.9 #define CommandOffsetPRCHandler 16**

Offset to the RC Handler function pointer

**5.63.2.10 #define CommandOffsetProgStatus 28**

Offset to the running program's status

**5.63.2.11 #define CommandOffsetSyncTick 32824**

Offset to the VM sync tick

**5.63.2.12 #define CommandOffsetSyncTime 32820**

Offset to the VM sync time

**5.63.2.13 #define CommandOffsetTick 20**

Offset to the VM's current tick

## 5.64 IOCtrl module constants

Constants that are part of the NXT firmware's IOCtrl module.

### Modules

- [IOCtrl module IOMAP offsets](#)

*Constant offsets into the IOCtrl module IOMAP structure.*

- [PowerOn constants](#)

*Use these constants to power down the NXT or boot it into SAMBA (aka firmware download) mode.*

### 5.64.1 Detailed Description

Constants that are part of the NXT firmware's IOCtrl module.

## 5.65 PowerOn constants

Use these constants to power down the NXT or boot it into SAMBA (aka firmware download) mode.

### Macros

- `#define IOCTRL_POWERDOWN 0x5A00`
- `#define IOCTRL_BOOT 0xA55A`

#### 5.65.1 Detailed Description

Use these constants to power down the NXT or boot it into SAMBA (aka firmware download) mode.

#### 5.65.2 Macro Definition Documentation

##### 5.65.2.1 `#define IOCTRL_BOOT 0xA55A`

Reboot the NXT into SAMBA mode

##### 5.65.2.2 `#define IOCTRL_POWERDOWN 0x5A00`

Power down the NXT

## 5.66 IOCtrl module IOMAP offsets

Constant offsets into the IOCtrl module IOMAP structure.

### Macros

- #define IOCtrlOffsetPowerOn 0

#### 5.66.1 Detailed Description

Constant offsets into the IOCtrl module IOMAP structure.

#### 5.66.2 Macro Definition Documentation

##### 5.66.2.1 #define IOCtrlOffsetPowerOn 0

Offset to power on field

## 5.67 Loader module constants

Constants that are part of the NXT firmware's Loader module.

### Modules

- [Loader module IOMAP offsets](#)  
*Constant offsets into the Loader module IOMAP structure.*
- [Loader module error codes](#)  
*Error codes returned by functions in the Loader module (file access).*
- [Loader module function constants](#)  
*Constants defining the functions provided by the Loader module.*

### Macros

- [#define EOF -1](#)
- [#define NULL 0](#)

#### 5.67.1 Detailed Description

Constants that are part of the NXT firmware's Loader module.

#### 5.67.2 Macro Definition Documentation

##### 5.67.2.1 #define EOF -1

A constant representing end of file

##### 5.67.2.2 #define NULL 0

A constant representing NULL

## 5.68 Loader module IOMAP offsets

Constant offsets into the Loader module IOMAP structure.

### Macros

- `#define LoaderOffsetPFunc 0`
- `#define LoaderOffsetFreeUserFlash 4`

#### 5.68.1 Detailed Description

Constant offsets into the Loader module IOMAP structure.

#### 5.68.2 Macro Definition Documentation

##### 5.68.2.1 `#define LoaderOffsetFreeUserFlash 4`

Offset to the amount of free user flash

##### 5.68.2.2 `#define LoaderOffsetPFunc 0`

Offset to the Loader module function pointer

## 5.69 Loader module error codes

Error codes returned by functions in the Loader module (file access).

### Macros

- #define LDR\_SUCCESS 0x0000
- #define LDR\_INPROGRESS 0x0001
- #define LDR\_REQPIN 0x0002
- #define LDR\_NOMOREHANDLES 0x8100
- #define LDR\_NOSPACE 0x8200
- #define LDR\_NOMOREFILES 0x8300
- #define LDR\_EOFEXPECTED 0x8400
- #define LDR\_ENDOFFILE 0x8500
- #define LDR\_NOTLINEARFILE 0x8600
- #define LDR\_FILENOFOUND 0x8700
- #define LDR\_HANDLEALREADYCLOSED 0x8800
- #define LDR\_NOLINEARSPACE 0x8900
- #define LDR\_UNDEFINEDERROR 0x8A00
- #define LDR\_FILEISBUSY 0x8B00
- #define LDR\_NOWRITEBUFFERS 0x8C00
- #define LDR\_APPENDNOTPOSSIBLE 0x8D00
- #define LDR\_FILEISFULL 0x8E00
- #define LDR\_FILEEXISTS 0x8F00
- #define LDR\_MODULENOTFOUND 0x9000
- #define LDR\_OUTOFBOUNDARY 0x9100
- #define LDR\_ILLEGALFILENAME 0x9200
- #define LDR\_ILLEGALHANDLE 0x9300
- #define LDR\_BTBUSY 0x9400
- #define LDR\_BTCONNECTFAIL 0x9500
- #define LDR\_BT TIMEOUT 0x9600
- #define LDR\_FILETX\_TIMEOUT 0x9700
- #define LDR\_FILETX\_DSTEXISTS 0x9800
- #define LDR\_FILETX\_SRCMISSING 0x9900
- #define LDR\_FILETX\_STREAMERROR 0x9A00
- #define LDR\_FILETX\_CLOSEERROR 0x9B00
- #define LDR\_INVALIDSEEK 0x9C00

### 5.69.1 Detailed Description

Error codes returned by functions in the Loader module (file access).

### 5.69.2 Macro Definition Documentation

#### 5.69.2.1 #define LDR\_APPENDNOTPOSSIBLE 0x8D00

Only datafiles can be appended to.

#### 5.69.2.2 #define LDR\_BTBUSY 0x9400

The bluetooth system is busy.

5.69.2.3 #define LDR\_BTCONNECTFAIL 0x9500

Bluetooth connection attempt failed.

5.69.2.4 #define LDR\_BTTIMEOUT 0x9600

A timeout in the bluetooth system has occurred.

5.69.2.5 #define LDR\_ENDOFFILE 0x8500

The end of the file has been reached.

5.69.2.6 #define LDR\_EOFEXPECTED 0x8400

EOF expected.

5.69.2.7 #define LDR\_FILEEXISTS 0x8F00

A file with the same name already exists.

5.69.2.8 #define LDR\_FILEISBUSY 0x8B00

The file is already being used.

5.69.2.9 #define LDR\_FILEISFULL 0x8E00

The allocated file size has been filled.

5.69.2.10 #define LDR\_FILENOFOUND 0x8700

No files matched the search criteria.

5.69.2.11 #define LDR\_FILETX\_CLOSEERROR 0x9B00

Error transmitting file: attempt to close file failed.

5.69.2.12 #define LDR\_FILETX\_DSTEXISTS 0x9800

Error transmitting file: destination file exists.

5.69.2.13 #define LDR\_FILETX\_SRCMISSING 0x9900

Error transmitting file: source file is missing.

5.69.2.14 #define LDR\_FILETX\_STREAMERROR 0x9A00

Error transmitting file: a stream error occurred.

5.69.2.15 #define LDR\_FILETX\_TIMEOUT 0x9700

Error transmitting file: a timeout occurred.

5.69.2.16 #define LDR\_HANDLEALREADYCLOSED 0x8800

The file handle has already been closed.

5.69.2.17 #define LDR\_ILLEGALFILENAME 0x9200

Filename length to long or attempted open a system file (\*.rxe, \*.rtm, or \*.sys) for writing as a datafile.

5.69.2.18 #define LDR\_ILLEGALHANDLE 0x9300

Invalid file handle.

5.69.2.19 #define LDR\_INPROGRESS 0x0001

The function is executing but has not yet completed.

5.69.2.20 #define LDR\_INVALIDSEEK 0x9C00

Invalid file seek operation.

5.69.2.21 #define LDR\_MODULENOTFOUND 0x9000

No modules matched the specified search criteria.

5.69.2.22 #define LDR\_NOLINEARSPACE 0x8900

Not enough linear flash memory is available.

5.69.2.23 #define LDR\_NOMOREFILES 0x8300

The maximum number of files has been reached.

5.69.2.24 #define LDR\_NOMOREHANDLES 0x8100

All available file handles are in use.

5.69.2.25 #define LDR\_NOSPACE 0x8200

Not enough free flash memory for the specified file size.

5.69.2.26 #define LDR\_NOTLINEARFILE 0x8600

The specified file is not linear.

5.69.2.27 #define LDR\_NOWRITEBUFFERS 0x8C00

No more write buffers are available.

5.69.2.28 #define LDR\_OUTOFBOUNDARY 0x9100

Specified IOMap offset is outside the bounds of the IOMap.

5.69.2.29 #define LDR\_REQPIN 0x0002

A PIN exchange request is in progress.

5.69.2.30 #define LDR\_SUCCESS 0x0000

The function completed successfully.

## 5.69.2.31 #define LDR\_UNDEFINEDERROR 0x8A00

An undefined error has occurred.

## 5.70 Loader module function constants

Constants defining the functions provided by the Loader module.

### Macros

- #define LDR\_CMD\_OPENREAD 0x80
- #define LDR\_CMD\_OPENWRITE 0x81
- #define LDR\_CMD\_READ 0x82
- #define LDR\_CMD\_WRITE 0x83
- #define LDR\_CMD\_CLOSE 0x84
- #define LDR\_CMD\_DELETE 0x85
- #define LDR\_CMD\_FINDFIRST 0x86
- #define LDR\_CMD\_FINDNEXT 0x87
- #define LDR\_CMD VERSIONS 0x88
- #define LDR\_CMD\_OPENWITELINEAR 0x89
- #define LDR\_CMD\_OPENREADLINEAR 0x8A
- #define LDR\_CMD\_OPENWRITEDATA 0x8B
- #define LDR\_CMD\_OPENAPPENDDATA 0x8C
- #define LDR\_CMD\_CROPDATAFILE 0x8D
- #define LDR\_CMD\_FINDFIRSTMODULE 0x90
- #define LDR\_CMD\_FINDNEXTMODULE 0x91
- #define LDR\_CMD CLOSEMODHANDLE 0x92
- #define LDR\_CMD\_IOMAPREAD 0x94
- #define LDR\_CMD\_IOMAPWRITE 0x95
- #define LDR\_CMD\_BOOTCMD 0x97
- #define LDR\_CMD\_SETBRICKNAME 0x98
- #define LDR\_CMD\_BTGETADR 0x9A
- #define LDR\_CMD\_DEVICENFO 0x9B
- #define LDR\_CMD\_DELETEUSERFLASH 0xA0
- #define LDR\_CMD\_POLLCMDLEN 0xA1
- #define LDR\_CMD\_POLLCMD 0xA2
- #define LDR\_CMD\_RENAMEFILE 0xA3
- #define LDR\_CMD\_BTFACTORYRESET 0xA4
- #define LDR\_CMD\_RESIZEDATAFILE 0xD0
- #define LDR\_CMD\_SEEKFROMSTART 0xD1
- #define LDR\_CMD\_SEEKFROMCURRENT 0xD2
- #define LDR\_CMD\_SEEKFROMEND 0xD3

### 5.70.1 Detailed Description

Constants defining the functions provided by the Loader module.

### 5.70.2 Macro Definition Documentation

#### 5.70.2.1 #define LDR\_CMD\_BOOTCMD 0x97

Reboot the NXT into SAMBA mode

5.70.2.2 #define LDR\_CMD\_BTFACTORYRESET 0xA4

Reset bluetooth configuration to factory defaults

5.70.2.3 #define LDR\_CMD\_BTGETADR 0x9A

Get the NXT's bluetooth brick address

5.70.2.4 #define LDR\_CMD\_CLOSE 0x84

Close a file handle

5.70.2.5 #define LDR\_CMD\_CLOSEMODHANDLE 0x92

Close a module handle

5.70.2.6 #define LDR\_CMD\_CROPDATAFILE 0x8D

Crop a data file to its used space

5.70.2.7 #define LDR\_CMD\_DELETE 0x85

Delete a file

5.70.2.8 #define LDR\_CMD\_DELETEUSERFLASH 0xA0

Delete all files from user flash memory

5.70.2.9 #define LDR\_CMD\_DEVICEINFO 0x9B

Read device information

5.70.2.10 #define LDR\_CMD\_FINDFIRST 0x86

Find the first file matching the specified pattern

5.70.2.11 #define LDR\_CMD\_FINDFIRSTMODULE 0x90

Find the first module matching the specified pattern

5.70.2.12 #define LDR\_CMD\_FINDNEXT 0x87

Find the next file matching the specified pattern

5.70.2.13 #define LDR\_CMD\_FINDNEXTMODULE 0x91

Find the next module matching the specified pattern

5.70.2.14 #define LDR\_CMD\_IOMAPREAD 0x94

Read data from a module IOMAP

5.70.2.15 #define LDR\_CMD\_IOMAPWRITE 0x95

Write data to a module IOMAP

5.70.2.16 #define LDR\_CMD\_OPENAPPENDDATA 0x8C

Open a data file for appending

5.70.2.17 #define LDR\_CMD\_OPENREAD 0x80

Open a file for reading

5.70.2.18 #define LDR\_CMD\_OPENREADLINEAR 0x8A

Open a linear file for reading

5.70.2.19 #define LDR\_CMD\_OPENWRITE 0x81

Open a file for writing

5.70.2.20 #define LDR\_CMD\_OPENWRITEDATA 0x8B

Open a data file for writing

5.70.2.21 #define LDR\_CMD\_OPENWRITELINEAR 0x89

Open a linear file for writing

5.70.2.22 #define LDR\_CMD\_POLLCMD 0xA2

Poll command

5.70.2.23 #define LDR\_CMD\_POLLCMDLEN 0xA1

Read poll command length

5.70.2.24 #define LDR\_CMD\_READ 0x82

Read from a file

5.70.2.25 #define LDR\_CMD\_RENAMEFILE 0xA3

Rename a file

5.70.2.26 #define LDR\_CMD\_RESIZEDATAFILE 0xD0

Resize a data file

5.70.2.27 #define LDR\_CMD\_SEEKFROMCURRENT 0xD2

Seek from the current position

5.70.2.28 #define LDR\_CMD\_SEEKFROMEND 0xD3

Seek from the end of the file

5.70.2.29 #define LDR\_CMD\_SEEKFROMSTART 0xD1

Seek from the start of the file

5.70.2.30 #define LDR\_CMD\_SETBRICKNAME 0x98

Set the NXT's brick name

5.70.2.31 #define LDR\_CMD\_VERSIONS 0x88

Read firmware version information

5.70.2.32 #define LDR\_CMD\_WRITE 0x83

Write to a file

## 5.71 Sound module constants

Constants that are part of the NXT firmware's Sound module.

### Modules

- [Sound module IOMAP offsets](#)  
*Constant offsets into the Sound module IOMAP structure.*
- [Sound module miscellaneous constants](#)  
*Constants defining miscellaneous sound module aspects.*
- [SoundFlags constants](#)  
*Constants for use with the SoundFlags() function.*
- [SoundMode constants](#)  
*Constants for use with the SoundMode() function.*
- [SoundState constants](#)  
*Constants for use with the SoundState() function.*
- [Tone constants](#)  
*Constants for use in the [SoundPlayTone\(\)](#) API function.*

### 5.71.1 Detailed Description

Constants that are part of the NXT firmware's Sound module.

## 5.72 SoundFlags constants

Constants for use with the SoundFlags() function.

### Macros

- #define SOUND\_FLAGS\_IDLE 0x00
- #define SOUND\_FLAGS\_UPDATE 0x01
- #define SOUND\_FLAGS\_RUNNING 0x02

### 5.72.1 Detailed Description

Constants for use with the SoundFlags() function.

### See Also

[SoundFlags\(\)](#)

### 5.72.2 Macro Definition Documentation

#### 5.72.2.1 #define SOUND\_FLAGS\_IDLE 0x00

R - Sound is idle

#### 5.72.2.2 #define SOUND\_FLAGS\_RUNNING 0x02

R - Currently processing a tone or file

#### 5.72.2.3 #define SOUND\_FLAGS\_UPDATE 0x01

W - Make changes take effect

## 5.73 SoundState constants

Constants for use with the SoundState() function.

### Macros

- #define SOUND\_STATE\_IDLE 0x00
- #define SOUND\_STATE\_FILE 0x02
- #define SOUND\_STATE\_TONE 0x03
- #define SOUND\_STATE\_STOP 0x04

### 5.73.1 Detailed Description

Constants for use with the SoundState() function.

### See Also

[SoundState\(\)](#)

### 5.73.2 Macro Definition Documentation

#### 5.73.2.1 #define SOUND\_STATE\_FILE 0x02

R - Processing a file of sound/melody data

#### 5.73.2.2 #define SOUND\_STATE\_IDLE 0x00

R - Idle, ready for start sound (SOUND\_UPDATE)

#### 5.73.2.3 #define SOUND\_STATE\_STOP 0x04

W - Stop sound immediately and close hardware

#### 5.73.2.4 #define SOUND\_STATE\_TONE 0x03

R - Processing a play tone request

## 5.74 SoundMode constants

Constants for use with the SoundMode() function.

### Macros

- #define SOUND\_MODE\_ONCE 0x00
- #define SOUND\_MODE\_LOOP 0x01
- #define SOUND\_MODE\_TONE 0x02

### 5.74.1 Detailed Description

Constants for use with the SoundMode() function.

### See Also

[SoundMode\(\)](#)

### 5.74.2 Macro Definition Documentation

#### 5.74.2.1 #define SOUND\_MODE\_LOOP 0x01

W - Play file until writing SOUND\_STATE\_STOP into SoundState

#### 5.74.2.2 #define SOUND\_MODE\_ONCE 0x00

W - Only play file once

#### 5.74.2.3 #define SOUND\_MODE\_TONE 0x02

W - Play tone specified in Frequency for Duration ms

## 5.75 Sound module IOMAP offsets

Constant offsets into the Sound module IOMAP structure.

### Macros

- #define SoundOffsetFreq 0
- #define SoundOffsetDuration 2
- #define SoundOffsetSampleRate 4
- #define SoundOffsetSoundFilename 6
- #define SoundOffsetFlags 26
- #define SoundOffsetState 27
- #define SoundOffsetMode 28
- #define SoundOffsetVolume 29

### 5.75.1 Detailed Description

Constant offsets into the Sound module IOMAP structure.

### 5.75.2 Macro Definition Documentation

#### 5.75.2.1 #define SoundOffsetDuration 2

RW - Tone duration [mS] (2 bytes)

#### 5.75.2.2 #define SoundOffsetFlags 26

RW - Play flag - described above (1 byte) [SoundFlags constants](#)

#### 5.75.2.3 #define SoundOffsetFreq 0

RW - Tone frequency [Hz] (2 bytes)

#### 5.75.2.4 #define SoundOffsetMode 28

RW - Play mode - described above (1 byte) [SoundMode constants](#)

#### 5.75.2.5 #define SoundOffsetSampleRate 4

RW - Sound file sample rate [2000..16000] (2 bytes)

#### 5.75.2.6 #define SoundOffsetSoundFilename 6

RW - Sound/melody filename (20 bytes)

#### 5.75.2.7 #define SoundOffsetState 27

RW - Play state - described above (1 byte) [SoundState constants](#)

#### 5.75.2.8 #define SoundOffsetVolume 29

RW - Sound/melody volume [0..4] 0 = off (1 byte)

## 5.76 Sound module miscellaneous constants

Constants defining miscellaneous sound module aspects.

### Macros

- #define FREQUENCY\_MIN 220
- #define FREQUENCY\_MAX 14080
- #define SAMPLERATE\_MIN 2000
- #define SAMPLERATE\_DEFAULT 8000
- #define SAMPLERATE\_MAX 16000

### 5.76.1 Detailed Description

Constants defining miscellaneous sound module aspects.

### 5.76.2 Macro Definition Documentation

#### 5.76.2.1 #define FREQUENCY\_MAX 14080

Maximum frequency [Hz]

#### 5.76.2.2 #define FREQUENCY\_MIN 220

Minimum frequency [Hz]

#### 5.76.2.3 #define SAMPLERATE\_DEFAULT 8000

Default sample rate [sps]

#### 5.76.2.4 #define SAMPLERATE\_MAX 16000

Max sample rate [sps]

#### 5.76.2.5 #define SAMPLERATE\_MIN 2000

Min sample rate [sps]

## 5.77 Tone constants

Constants for use in the [SoundPlayTone\(\)](#) API function.

### Macros

- `#define TONE_C3 131`
- `#define TONE_CS3 139`
- `#define TONE_D3 147`
- `#define TONE_DS3 156`
- `#define TONE_E3 165`
- `#define TONE_F3 175`
- `#define TONE_FS3 185`
- `#define TONE_G3 196`
- `#define TONE_GS3 208`
- `#define TONE_A3 220`
- `#define TONE_AS3 233`
- `#define TONE_B3 247`
- `#define TONE_C4 262`
- `#define TONE_CS4 277`
- `#define TONE_D4 294`
- `#define TONE_DS4 311`
- `#define TONE_E4 330`
- `#define TONE_F4 349`
- `#define TONE_FS4 370`
- `#define TONE_G4 392`
- `#define TONE_GS4 415`
- `#define TONE_A4 440`
- `#define TONE_AS4 466`
- `#define TONE_B4 494`
- `#define TONE_C5 523`
- `#define TONE_CS5 554`
- `#define TONE_D5 587`
- `#define TONE_DS5 622`
- `#define TONE_E5 659`
- `#define TONE_F5 698`
- `#define TONE_FS5 740`
- `#define TONE_G5 784`
- `#define TONE_GS5 831`
- `#define TONE_A5 880`
- `#define TONE_AS5 932`
- `#define TONE_B5 988`
- `#define TONE_C6 1047`
- `#define TONE_CS6 1109`
- `#define TONE_D6 1175`
- `#define TONE_DS6 1245`
- `#define TONE_E6 1319`
- `#define TONE_F6 1397`
- `#define TONE_FS6 1480`
- `#define TONE_G6 1568`

- #define TONE\_GS6 1661
- #define TONE\_A6 1760
- #define TONE\_AS6 1865
- #define TONE\_B6 1976
- #define TONE\_C7 2093
- #define TONE\_CS7 2217
- #define TONE\_D7 2349
- #define TONE\_DS7 2489
- #define TONE\_E7 2637
- #define TONE\_F7 2794
- #define TONE\_FS7 2960
- #define TONE\_G7 3136
- #define TONE\_GS7 3322
- #define TONE\_A7 3520
- #define TONE\_AS7 3729
- #define TONE\_B7 3951

#### 5.77.1 Detailed Description

Constants for use in the [SoundPlayTone\(\)](#) API function.

#### See Also

[SoundPlayTone\(\)](#)

#### 5.77.2 Macro Definition Documentation

##### 5.77.2.1 #define TONE\_A3 220

Third octave A

##### 5.77.2.2 #define TONE\_A4 440

Fourth octave A

##### 5.77.2.3 #define TONE\_A5 880

Fifth octave A

##### 5.77.2.4 #define TONE\_A6 1760

Sixth octave A

##### 5.77.2.5 #define TONE\_A7 3520

Seventh octave A

##### 5.77.2.6 #define TONE\_AS3 233

Third octave A sharp

##### 5.77.2.7 #define TONE\_AS4 466

Fourth octave A sharp

5.77.2.8 #define TONE\_AS5 932

Fifth octave A sharp

5.77.2.9 #define TONE\_AS6 1865

Sixth octave A sharp

5.77.2.10 #define TONE\_AS7 3729

Seventh octave A sharp

5.77.2.11 #define TONE\_B3 247

Third octave B

5.77.2.12 #define TONE\_B4 494

Fourth octave B

5.77.2.13 #define TONE\_B5 988

Fifth octave B

5.77.2.14 #define TONE\_B6 1976

Sixth octave B

5.77.2.15 #define TONE\_B7 3951

Seventh octave B

5.77.2.16 #define TONE\_C3 131

Third octave C

5.77.2.17 #define TONE\_C4 262

Fourth octave C

5.77.2.18 #define TONE\_C5 523

Fifth octave C

5.77.2.19 #define TONE\_C6 1047

Sixth octave C

5.77.2.20 #define TONE\_C7 2093

Seventh octave C

5.77.2.21 #define TONE\_CS3 139

Third octave C sharp

5.77.2.22 #define TONE\_CS4 277

Fourth octave C sharp

5.77.2.23 #define TONE\_CS5 554

Fifth octave C sharp

5.77.2.24 #define TONE\_CS6 1109

Sixth octave C sharp

5.77.2.25 #define TONE\_CS7 2217

Seventh octave C sharp

5.77.2.26 #define TONE\_D3 147

Third octave D

5.77.2.27 #define TONE\_D4 294

Fourth octave D

5.77.2.28 #define TONE\_D5 587

Fifth octave D

5.77.2.29 #define TONE\_D6 1175

Sixth octave D

5.77.2.30 #define TONE\_D7 2349

Seventh octave D

5.77.2.31 #define TONE\_DS3 156

Third octave D sharp

5.77.2.32 #define TONE\_DS4 311

Fourth octave D sharp

5.77.2.33 #define TONE\_DS5 622

Fifth octave D sharp

5.77.2.34 #define TONE\_DS6 1245

Sixth octave D sharp

5.77.2.35 #define TONE\_DS7 2489

Seventh octave D sharp

5.77.2.36 #define TONE\_E3 165

Third octave E

5.77.2.37 #define TONE\_E4 330

Fourth octave E

5.77.2.38 #define TONE\_E5 659

Fifth octave E

5.77.2.39 #define TONE\_E6 1319

Sixth octave E

5.77.2.40 #define TONE\_E7 2637

Seventh octave E

5.77.2.41 #define TONE\_F3 175

Third octave F

5.77.2.42 #define TONE\_F4 349

Fourth octave F

5.77.2.43 #define TONE\_F5 698

Fifth octave F

5.77.2.44 #define TONE\_F6 1397

Sixth octave F

5.77.2.45 #define TONE\_F7 2794

Seventh octave F

5.77.2.46 #define TONE\_FS3 185

Third octave F sharp

5.77.2.47 #define TONE\_FS4 370

Fourth octave F sharp

5.77.2.48 #define TONE\_FS5 740

Fifth octave F sharp

5.77.2.49 #define TONE\_FS6 1480

Sixth octave F sharp

5.77.2.50 #define TONE\_FS7 2960

Seventh octave F sharp

5.77.2.51 #define TONE\_G3 196

Third octave G

5.77.2.52 #define TONE\_G4 392

Fourth octave G

5.77.2.53 #define TONE\_G5 784

Fifth octave G

5.77.2.54 #define TONE\_G6 1568

Sixth octave G

5.77.2.55 #define TONE\_G7 3136

Seventh octave G

5.77.2.56 #define TONE\_GS3 208

Third octave G sharp

5.77.2.57 #define TONE\_GS4 415

Fourth octave G sharp

5.77.2.58 #define TONE\_GS5 831

Fifth octave G sharp

5.77.2.59 #define TONE\_GS6 1661

Sixth octave G sharp

5.77.2.60 #define TONE\_GS7 3322

Seventh octave G sharp

## 5.78 Button module constants

Constants that are part of the NXT firmware's Button module.

### Modules

- [Button module IOMAP offsets](#)

*Constant offsets into the Button module IOMAP structure.*

- [Button name constants](#)

*Constants to specify which button to use with button module functions.*

- [ButtonState constants](#)

*Constants for use with the ButtonState() function.*

### 5.78.1 Detailed Description

Constants that are part of the NXT firmware's Button module.

## 5.79 Button name constants

Constants to specify which button to use with button module functions.

### Macros

- #define **BTN1** 0
- #define **BTN2** 1
- #define **BTN3** 2
- #define **BTN4** 3
- #define **BTNEXTIT** **BTN1**
- #define **BTNRIGHT** **BTN2**
- #define **BTNLEFT** **BTN3**
- #define **BTNCENTER** **BTN4**
- #define **NO\_OF\_BTNS** 4

### 5.79.1 Detailed Description

Constants to specify which button to use with button module functions.

### See Also

[ButtonPressed\(\)](#), [ButtonState\(\)](#), [ButtonCount\(\)](#), [ReadButtonEx\(\)](#), [SysReadButton\(\)](#), [ReadButtonType](#)

### 5.79.2 Macro Definition Documentation

#### 5.79.2.1 #define **BTN1** 0

The exit button.

#### 5.79.2.2 #define **BTN2** 1

The right button.

#### 5.79.2.3 #define **BTN3** 2

The left button.

#### 5.79.2.4 #define **BTN4** 3

The enter button.

#### 5.79.2.5 #define **BTNCENTER** **BTN4**

The enter button.

#### 5.79.2.6 #define **BTNEXTIT** **BTN1**

The exit button.

#### 5.79.2.7 #define **BTNLEFT** **BTN3**

The left button.

5.79.2.8 #define BTNRIGHT BTN2

The right button.

5.79.2.9 #define NO\_OF\_BTNS 4

The number of NXT buttons.

## 5.80 ButtonState constants

Constants for use with the ButtonState() function.

### Macros

- #define BTNSTATE\_PRESSED\_EV 0x01
- #define BTNSTATE\_SHORT\_RELEASED\_EV 0x02
- #define BTNSTATE\_LONG\_PRESSED\_EV 0x04
- #define BTNSTATE\_LONG\_RELEASED\_EV 0x08
- #define BTNSTATE\_PRESSED\_STATE 0x80
- #define BTNSTATE\_NONE 0x10

### 5.80.1 Detailed Description

Constants for use with the ButtonState() function. The \_EV values can be combined together using a bitwise OR operation.

### See Also

[ButtonState\(\)](#)

### 5.80.2 Macro Definition Documentation

#### 5.80.2.1 #define BTNSTATE\_LONG\_PRESSED\_EV 0x04

Button is in the long pressed state.

#### 5.80.2.2 #define BTNSTATE\_LONG\_RELEASED\_EV 0x08

Button is in the long released state.

#### 5.80.2.3 #define BTNSTATE\_NONE 0x10

The default button state.

#### 5.80.2.4 #define BTNSTATE\_PRESSED\_EV 0x01

Button is in the pressed state.

#### 5.80.2.5 #define BTNSTATE\_PRESSED\_STATE 0x80

A bitmask for the button pressed state

#### 5.80.2.6 #define BTNSTATE\_SHORT\_RELEASED\_EV 0x02

Button is in the short released state.

## 5.81 Button module IOMAP offsets

Constant offsets into the Button module IOMAP structure.

### Macros

- `#define ButtonOffsetPressedCnt(b) (((b)*8)+0)`
- `#define ButtonOffsetLongPressCnt(b) (((b)*8)+1)`
- `#define ButtonOffsetShortRelCnt(b) (((b)*8)+2)`
- `#define ButtonOffsetLongRelCnt(b) (((b)*8)+3)`
- `#define ButtonOffsetRelCnt(b) (((b)*8)+4)`
- `#define ButtonOffsetState(b) ((b)+32)`

### 5.81.1 Detailed Description

Constant offsets into the Button module IOMAP structure.

### 5.81.2 Macro Definition Documentation

#### 5.81.2.1 `#define ButtonOffsetLongPressCnt( b ) (((b)*8)+1)`

Offset to the LongPressCnt field. This field stores the long press count.

#### 5.81.2.2 `#define ButtonOffsetLongRelCnt( b ) (((b)*8)+3)`

Offset to the LongRelCnt field. This field stores the long release count.

#### 5.81.2.3 `#define ButtonOffsetPressedCnt( b ) (((b)*8)+0)`

Offset to the PressedCnt field. This field stores the press count.

#### 5.81.2.4 `#define ButtonOffsetRelCnt( b ) (((b)*8)+4)`

Offset to the RelCnt field. This field stores the release count.

#### 5.81.2.5 `#define ButtonOffsetShortRelCnt( b ) (((b)*8)+2)`

Offset to the ShortRelCnt field. This field stores the short release count.

#### 5.81.2.6 `#define ButtonOffsetState( b ) ((b)+32)`

Offset to the State field. This field stores the current button state.

## 5.82 Ui module constants

Constants that are part of the NXT firmware's Ui module.

### Modules

- [BluetoothState constants](#)  
*Constants for use with the BluetoothState() function.*
- [CommandFlags constants](#)  
*Constants for use with the CommandFlags() function.*
- [UIButton constants](#)  
*Constants for use with the UIButton() function.*
- [UIState constants](#)  
*Constants for use with the UIState() function.*
- [Ui module IOMAP offsets](#)  
*Constant offsets into the Ui module IOMAP structure.*
- [VM run state constants](#)  
*Constants for use with the VMRunState() function.*

### 5.82.1 Detailed Description

Constants that are part of the NXT firmware's Ui module.

## 5.83 CommandFlags constants

Constants for use with the CommandFlags() function.

### Macros

- #define UI\_FLAGS\_UPDATE 0x01
- #define UI\_FLAGS\_DISABLE\_LEFT\_RIGHT\_ENTER 0x02
- #define UI\_FLAGS\_DISABLE\_EXIT 0x04
- #define UI\_FLAGS\_REDRAW\_STATUS 0x08
- #define UI\_FLAGS\_RESET\_SLEEP\_TIMER 0x10
- #define UI\_FLAGS\_EXECUTE\_LMS\_FILE 0x20
- #define UI\_FLAGS\_BUSY 0x40
- #define UI\_FLAGS\_ENABLE\_STATUS\_UPDATE 0x80

### 5.83.1 Detailed Description

Constants for use with the CommandFlags() function.

### See Also

[CommandFlags\(\)](#)

### 5.83.2 Macro Definition Documentation

#### 5.83.2.1 #define UI\_FLAGS\_BUSY 0x40

R - UI busy running or datalogging (popup disabled)

#### 5.83.2.2 #define UI\_FLAGS\_DISABLE\_EXIT 0x04

RW - Disable exit button

#### 5.83.2.3 #define UI\_FLAGS\_DISABLE\_LEFT\_RIGHT\_ENTER 0x02

RW - Disable left, right and enter button

#### 5.83.2.4 #define UI\_FLAGS\_ENABLE\_STATUS\_UPDATE 0x80

W - Enable status line to be updated

#### 5.83.2.5 #define UI\_FLAGS\_EXECUTE\_LMS\_FILE 0x20

W - Execute LMS file in "LMSfilename" (Try It)

#### 5.83.2.6 #define UI\_FLAGS\_REDRAW\_STATUS 0x08

W - Redraw entire status line

#### 5.83.2.7 #define UI\_FLAGS\_RESET\_SLEEP\_TIMER 0x10

W - Reset sleep timeout timer

5.83.2.8 #define UI\_FLAGS\_UPDATE 0x01

W - Make changes take effect

## 5.84 UIState constants

Constants for use with the UIState() function.

### Macros

- #define UI\_STATE\_INIT\_DISPLAY 0
- #define UI\_STATE\_INIT\_LOW\_BATTERY 1
- #define UI\_STATE\_INIT\_INTRO 2
- #define UI\_STATE\_INIT\_WAIT 3
- #define UI\_STATE\_INIT\_MENU 4
- #define UI\_STATE\_NEXT\_MENU 5
- #define UI\_STATE\_DRAW\_MENU 6
- #define UI\_STATE\_TEST\_BUTTONS 7
- #define UI\_STATE\_LEFT\_PRESSED 8
- #define UI\_STATE\_RIGHT\_PRESSED 9
- #define UI\_STATE\_ENTER\_PRESSED 10
- #define UI\_STATE\_EXIT\_PRESSED 11
- #define UI\_STATE\_CONNECT\_REQUEST 12
- #define UI\_STATE\_EXECUTE\_FILE 13
- #define UI\_STATE\_EXECUTING\_FILE 14
- #define UI\_STATE\_LOW\_BATTERY 15
- #define UI\_STATE\_BT\_ERROR 16

### 5.84.1 Detailed Description

Constants for use with the UIState() function.

### See Also

UIState()

### 5.84.2 Macro Definition Documentation

#### 5.84.2.1 #define UI\_STATE\_BT\_ERROR 16

R - BT error

#### 5.84.2.2 #define UI\_STATE\_CONNECT\_REQUEST 12

RW - Request for connection accept

#### 5.84.2.3 #define UI\_STATE\_DRAW\_MENU 6

RW - Execute function and draw menu icons

#### 5.84.2.4 #define UI\_STATE\_ENTER\_PRESSED 10

RW - Load selected function and next menu id

#### 5.84.2.5 #define UI\_STATE\_EXECUTE\_FILE 13

RW - Execute file in "LMSfilename"

5.84.2.6 #define UI\_STATE\_EXECUTING\_FILE 14

R - Executing file in "LMSfilename"

5.84.2.7 #define UI\_STATE\_EXIT\_PRESSED 11

RW - Load selected function and next menu id

5.84.2.8 #define UI\_STATE\_INIT\_DISPLAY 0

RW - Init display and load font, menu etc.

5.84.2.9 #define UI\_STATE\_INIT\_INTRO 2

R - Display intro

5.84.2.10 #define UI\_STATE\_INIT\_LOW\_BATTERY 1

R - Low battery voltage at power on

5.84.2.11 #define UI\_STATE\_INIT\_MENU 4

RW - Init menu system

5.84.2.12 #define UI\_STATE\_INIT\_WAIT 3

RW - Wait for initialization end

5.84.2.13 #define UI\_STATE\_LEFT\_PRESSED 8

RW - Load selected function and next menu id

5.84.2.14 #define UI\_STATE\_LOW\_BATTERY 15

R - Low battery at runtime

5.84.2.15 #define UI\_STATE\_NEXT\_MENU 5

RW - Next menu icons ready for drawing

5.84.2.16 #define UI\_STATE\_RIGHT\_PRESSED 9

RW - Load selected function and next menu id

5.84.2.17 #define UI\_STATE\_TEST\_BUTTONS 7

RW - Wait for buttons to be pressed

## 5.85 UIButton constants

Constants for use with the UIButton() function.

### Macros

- #define UI\_BUTTON\_NONE 0
- #define UI\_BUTTON\_LEFT 1
- #define UI\_BUTTON\_ENTER 2
- #define UI\_BUTTON\_RIGHT 3
- #define UI\_BUTTON\_EXIT 4

### 5.85.1 Detailed Description

Constants for use with the UIButton() function.

### See Also

[UIButton\(\)](#)

### 5.85.2 Macro Definition Documentation

#### 5.85.2.1 #define UI\_BUTTON\_ENTER 2

W - Insert enter button

#### 5.85.2.2 #define UI\_BUTTON\_EXIT 4

W - Insert exit button

#### 5.85.2.3 #define UI\_BUTTON\_LEFT 1

W - Insert left arrow button

#### 5.85.2.4 #define UI\_BUTTON\_NONE 0

R - Button inserted are executed

#### 5.85.2.5 #define UI\_BUTTON\_RIGHT 3

W - Insert right arrow button

## 5.86 BluetoothState constants

Constants for use with the BluetoothState() function.

### Macros

- #define UI\_BT\_STATE\_VISIBLE 0x01
- #define UI\_BT\_STATE\_CONNECTED 0x02
- #define UI\_BT\_STATE\_OFF 0x04
- #define UI\_BT\_ERROR\_ATTENTION 0x08
- #define UI\_BT\_CONNECT\_REQUEST 0x40
- #define UI\_BT\_PIN\_REQUEST 0x80

### 5.86.1 Detailed Description

Constants for use with the BluetoothState() function.

### See Also

[BluetoothState\(\)](#)

### 5.86.2 Macro Definition Documentation

#### 5.86.2.1 #define UI\_BT\_CONNECT\_REQUEST 0x40

RW - BT get connect accept in progress

#### 5.86.2.2 #define UI\_BT\_ERROR\_ATTENTION 0x08

W - BT error attention

#### 5.86.2.3 #define UI\_BT\_PIN\_REQUEST 0x80

RW - BT get pin code

#### 5.86.2.4 #define UI\_BT\_STATE\_CONNECTED 0x02

RW - BT connected to something

#### 5.86.2.5 #define UI\_BT\_STATE\_OFF 0x04

RW - BT power off

#### 5.86.2.6 #define UI\_BT\_STATE\_VISIBLE 0x01

RW - BT visible

## 5.87 VM run state constants

Constants for use with the VMRunState() function.

### Macros

- #define UI\_VM\_IDLE 0
- #define UI\_VM\_RUN\_FREE 1
- #define UI\_VM\_RUN\_SINGLE 2
- #define UI\_VM\_RUN\_PAUSE 3
- #define UI\_VM\_RESET1 4
- #define UI\_VM\_RESET2 5

### 5.87.1 Detailed Description

Constants for use with the VMRunState() function.

### See Also

[VMRunState\(\)](#)

### 5.87.2 Macro Definition Documentation

#### 5.87.2.1 #define UI\_VM\_IDLE 0

VM\_IDLE: Just sitting around. Request to run program will lead to ONE of the VM\_RUN\* states.

#### 5.87.2.2 #define UI\_VM\_RESET1 4

VM\_RESET1: Initialize state variables and some I/O devices – executed when programs end

#### 5.87.2.3 #define UI\_VM\_RESET2 5

VM\_RESET2: Final clean up and return to IDLE

#### 5.87.2.4 #define UI\_VM\_RUN\_FREE 1

VM\_RUN\_FREE: Attempt to run as many instructions as possible within our timeslice

#### 5.87.2.5 #define UI\_VM\_RUN\_PAUSE 3

VM\_RUN\_PAUSE: Program still "active", but someone has asked us to pause

#### 5.87.2.6 #define UI\_VM\_RUN\_SINGLE 2

VM\_RUN\_SINGLE: Run exactly one instruction per timeslice

## 5.88 Ui module IOMAP offsets

Constant offsets into the Ui module IOMAP structure.

### Macros

- #define UIOffsetPMenu 0
- #define UIOffsetBatteryVoltage 4
- #define UIOffsetLMSfilename 6
- #define UIOffsetFlags 26
- #define UIOffsetState 27
- #define UIOffsetButton 28
- #define UIOffsetRunState 29
- #define UIOffsetBatteryState 30
- #define UIOffsetBluetoothState 31
- #define UIOffsetUsbState 32
- #define UIOffsetSleepTimeout 33
- #define UIOffsetSleepTimer 34
- #define UIOffsetRechargeable 35
- #define UIOffsetVolume 36
- #define UIOffsetError 37
- #define UIOffsetOBPPointer 38
- #define UIOffsetForceOff 39
- #define UIOffsetAbortFlag 40

### 5.88.1 Detailed Description

Constant offsets into the Ui module IOMAP structure.

### 5.88.2 Macro Definition Documentation

#### 5.88.2.1 #define UIOffsetAbortFlag 40

RW - Long Abort (true == use long press to abort) (1 byte)

#### 5.88.2.2 #define UIOffsetBatteryState 30

W - Battery state (0..4 capacity) (1 byte)

#### 5.88.2.3 #define UIOffsetBatteryVoltage 4

R - Battery voltage in millivolts (2 bytes)

#### 5.88.2.4 #define UIOffsetBluetoothState 31

W - Bluetooth state (0=on, 1=visible, 2=conn, 3=conn.visible, 4=off, 5=dfu) (1 byte)

#### 5.88.2.5 #define UIOffsetButton 28

RW - Insert button (buttons enumerated above) (1 byte)

5.88.2.6 #define UIOffsetError 37

W - Error code (1 byte)

5.88.2.7 #define UIOffsetFlags 26

RW - Update command flags (flags enumerated above) (1 byte)

5.88.2.8 #define UIOffsetForceOff 39

W - Force off (> 0 = off) (1 byte)

5.88.2.9 #define UIOffsetLMSfilename 6

W - LMS filename to execute (Try It) (20 bytes)

5.88.2.10 #define UIOffsetOBPPointer 38

W - Actual OBP step (0 - 4) (1 byte)

5.88.2.11 #define UIOffsetPMenu 0

W - Pointer to menu file (4 bytes)

5.88.2.12 #define UIOffsetRechargeable 35

R - Rechargeable battery (0 = no, 1 = yes) (1 byte)

5.88.2.13 #define UIOffsetRunState 29

W - VM Run state (0 = stopped, 1 = running) (1 byte)

5.88.2.14 #define UIOffsetSleepTimeout 33

RW - Sleep timeout time (min) (1 byte)

5.88.2.15 #define UIOffsetSleepTimer 34

RW - Sleep timer (min) (1 byte)

5.88.2.16 #define UIOffsetState 27

RW - UI state (states enumerated above) (1 byte)

5.88.2.17 #define UIOffsetUsbState 32

W - Usb state (0=disconnected, 1=connected, 2=working) (1 byte)

5.88.2.18 #define UIOffsetVolume 36

RW - Volume used in UI (0 - 4) (1 byte)

## 5.89 NBC Input port constants

Input port constants are used when calling sensor control API functions.

### Macros

- #define IN\_1 0x00
- #define IN\_2 0x01
- #define IN\_3 0x02
- #define IN\_4 0x03

### 5.89.1 Detailed Description

Input port constants are used when calling sensor control API functions. These constants are intended for use in NBC.

### See Also

[SetSensorType\(\)](#), [SetSensorMode\(\)](#), S1, S2, S3, S4

### 5.89.2 Macro Definition Documentation

#### 5.89.2.1 #define IN\_1 0x00

Input port 1

#### 5.89.2.2 #define IN\_2 0x01

Input port 2

#### 5.89.2.3 #define IN\_3 0x02

Input port 3

#### 5.89.2.4 #define IN\_4 0x03

Input port 4

## 5.90 NBC sensor type constants

Use sensor type constants to configure an input port for a specific type of sensor.

### Macros

- `#define IN_TYPE_NO_SENSOR 0x00`
- `#define IN_TYPE_SWITCH 0x01`
- `#define IN_TYPE_TEMPERATURE 0x02`
- `#define IN_TYPE_REFLECTION 0x03`
- `#define IN_TYPE_ANGLE 0x04`
- `#define IN_TYPE_LIGHT_ACTIVE 0x05`
- `#define IN_TYPE_LIGHT_INACTIVE 0x06`
- `#define IN_TYPE_SOUND_DB 0x07`
- `#define IN_TYPE_SOUND_DBA 0x08`
- `#define IN_TYPE_CUSTOM 0x09`
- `#define IN_TYPE_LOWSPEED 0x0A`
- `#define IN_TYPE_LOWSPEED_9V 0x0B`
- `#define IN_TYPE_HISPEED 0x0C`
- `#define IN_TYPE_COLORFULL 0x0D`
- `#define IN_TYPE_COLORRED 0x0E`
- `#define IN_TYPE_COLORGREEN 0x0F`
- `#define IN_TYPE_COLORBLUE 0x10`
- `#define IN_TYPE_COLORNONE 0x11`
- `#define IN_TYPE_COLOREXIT 0x12`

### 5.90.1 Detailed Description

Use sensor type constants to configure an input port for a specific type of sensor. These constants are intended for use in NBC.

### See Also

[SetSensorType\(\)](#)

### 5.90.2 Macro Definition Documentation

#### 5.90.2.1 `#define IN_TYPE_ANGLE 0x04`

RCX rotation sensor

#### 5.90.2.2 `#define IN_TYPE_COLORBLUE 0x10`

NXT 2.0 color sensor with blue light

#### 5.90.2.3 `#define IN_TYPE_COLOREXIT 0x12`

NXT 2.0 color sensor internal state

#### 5.90.2.4 `#define IN_TYPE_COLORFULL 0x0D`

NXT 2.0 color sensor in full color mode

5.90.2.5 `#define IN_TYPE_COLORGREEN 0x0F`

NXT 2.0 color sensor with green light

5.90.2.6 `#define IN_TYPE_COLORNONE 0x11`

NXT 2.0 color sensor with no light

5.90.2.7 `#define IN_TYPE_COLORRED 0x0E`

NXT 2.0 color sensor with red light

5.90.2.8 `#define IN_TYPE_CUSTOM 0x09`

NXT custom sensor

5.90.2.9 `#define IN_TYPE_HISPEED 0x0C`

NXT Hi-speed port (only S4)

5.90.2.10 `#define IN_TYPE_LIGHT_ACTIVE 0x05`

NXT light sensor with light

5.90.2.11 `#define IN_TYPE_LIGHT_INACTIVE 0x06`

NXT light sensor without light

5.90.2.12 `#define IN_TYPE_LOWSPEED 0x0A`

NXT I2C digital sensor

5.90.2.13 `#define IN_TYPE_LOWSPEED_9V 0x0B`

NXT I2C digital sensor with 9V power

5.90.2.14 `#define IN_TYPE_NO_SENSOR 0x00`

No sensor configured

5.90.2.15 `#define IN_TYPE_REFLECTION 0x03`

RCX light sensor

5.90.2.16 `#define IN_TYPE_SOUND_DB 0x07`

NXT sound sensor with dB scaling

5.90.2.17 `#define IN_TYPE_SOUND_DBA 0x08`

NXT sound sensor with dBA scaling

5.90.2.18 `#define IN_TYPE_SWITCH 0x01`

NXT or RCX touch sensor

5.90.2.19 #define IN\_TYPE\_TEMPERATURE 0x02

RCX temperature sensor

## 5.91 NBC sensor mode constants

Use sensor mode constants to configure an input port for the desired sensor mode.

### Macros

- `#define IN_MODE_RAW 0x00`
- `#define IN_MODE_BOOLEAN 0x20`
- `#define IN_MODE_TRANSITIONCNT 0x40`
- `#define IN_MODE_PERIODCOUNTER 0x60`
- `#define IN_MODE_PCTFULLSCALE 0x80`
- `#define IN_MODE_CELSIUS 0xA0`
- `#define IN_MODE_FAHRENHEIT 0xC0`
- `#define IN_MODE_ANGLESTEP 0xE0`
- `#define IN_MODE_SLOPEMASK 0x1F`
- `#define IN_MODE_MODEMASK 0xE0`

### 5.91.1 Detailed Description

Use sensor mode constants to configure an input port for the desired sensor mode. The constants are intended for use in NBC.

### See Also

[SetSensorMode\(\)](#)

### 5.91.2 Macro Definition Documentation

#### 5.91.2.1 `#define IN_MODE_ANGLESTEP 0xE0`

RCX rotation sensor (16 ticks per revolution)

#### 5.91.2.2 `#define IN_MODE_BOOLEAN 0x20`

Boolean value (0 or 1)

#### 5.91.2.3 `#define IN_MODE_CELSIUS 0xA0`

RCX temperature sensor value in degrees celcius

#### 5.91.2.4 `#define IN_MODE_FAHRENHEIT 0xC0`

RCX temperature sensor value in degrees fahrenheit

#### 5.91.2.5 `#define IN_MODE_MODEMASK 0xE0`

Mask for the mode without any slope value

#### 5.91.2.6 `#define IN_MODE_PCTFULLSCALE 0x80`

Scaled value from 0 to 100

5.91.2.7 #define IN\_MODE\_PERIODCOUNTER 0x60

Counts the number of boolean periods

5.91.2.8 #define IN\_MODE\_RAW 0x00

Raw value from 0 to 1023

5.91.2.9 #define IN\_MODE\_SLOPEMASK 0x1F

Mask for slope parameter added to mode

5.91.2.10 #define IN\_MODE\_TRANSITIONCNT 0x40

Counts the number of boolean transitions

## 5.92 Input field constants

Constants for use with SetInput() and GetInput().

### Macros

- #define TypeField 0
- #define InputModeField 1
- #define RawValueField 2
- #define NormalizedValueField 3
- #define ScaledValueField 4
- #define InvalidDataField 5

### 5.92.1 Detailed Description

Constants for use with SetInput() and GetInput(). Each sensor has six fields that are used to define its state.

### 5.92.2 Macro Definition Documentation

#### 5.92.2.1 #define InputModeField 1

Input mode field. Contains one of the sensor mode constants. Read/write.

#### 5.92.2.2 #define InvalidDataField 5

Invalid data field. Contains a boolean value indicating whether the sensor data is valid or not. Read/write.

#### 5.92.2.3 #define NormalizedValueField 3

Normalized value field. Contains the current normalized analog sensor value. Read only.

#### 5.92.2.4 #define RawValueField 2

Raw value field. Contains the current raw analog sensor value. Read only.

#### 5.92.2.5 #define ScaledValueField 4

Scaled value field. Contains the current scaled analog sensor value. Read/write.

#### 5.92.2.6 #define TypeField 0

Type field. Contains one of the sensor type constants. Read/write.

## 5.93 Input port digital pin constants

Constants for use when directly controlling or reading a port's digital pin state.

### Macros

- #define INPUT\_DIGI0 0x01
- #define INPUT\_DIGI1 0x02

#### 5.93.1 Detailed Description

Constants for use when directly controlling or reading a port's digital pin state.

#### 5.93.2 Macro Definition Documentation

##### 5.93.2.1 #define INPUT\_DIGI0 0x01

Digital pin 0

##### 5.93.2.2 #define INPUT\_DIGI1 0x02

Digital pin 1

## 5.94 Color sensor array indices

Constants for use with color sensor value arrays to index RGB and blank return values.

### Macros

- `#define INPUT_RED 0`
- `#define INPUT_GREEN 1`
- `#define INPUT_BLUE 2`
- `#define INPUT_BLANK 3`
- `#define INPUT_NO_OF_COLORS 4`

### 5.94.1 Detailed Description

Constants for use with color sensor value arrays to index RGB and blank return values.

### See Also

[ReadSensorColorEx\(\)](#), [ReadSensorColorRaw\(\)](#), [SysColorSensorRead\(\)](#), [ColorSensorReadType](#)

### 5.94.2 Macro Definition Documentation

#### 5.94.2.1 `#define INPUT_BLANK 3`

Access the blank value from color sensor value arrays

#### 5.94.2.2 `#define INPUT_BLUE 2`

Access the blue value from color sensor value arrays

#### 5.94.2.3 `#define INPUT_GREEN 1`

Access the green value from color sensor value arrays

#### 5.94.2.4 `#define INPUT_NO_OF_COLORS 4`

The number of entries in the color sensor value arrays

#### 5.94.2.5 `#define INPUT_RED 0`

Access the red value from color sensor value arrays

## 5.95 Color values

Constants for use with the ColorValue returned by the color sensor in full color mode.

### Macros

- #define INPUT\_BLACKCOLOR 1
- #define INPUT\_BLUECOLOR 2
- #define INPUT\_GREENCOLOR 3
- #define INPUT\_YELLOWCOLOR 4
- #define INPUT\_REDCOLOR 5
- #define INPUT\_WHITECOLOR 6

### 5.95.1 Detailed Description

Constants for use with the ColorValue returned by the color sensor in full color mode.

### See Also

SensorValue(), SysColorSensorRead(), ColorSensorReadType

### 5.95.2 Macro Definition Documentation

#### 5.95.2.1 #define INPUT\_BLACKCOLOR 1

The color value is black

#### 5.95.2.2 #define INPUT\_BLUECOLOR 2

The color value is blue

#### 5.95.2.3 #define INPUT\_GREENCOLOR 3

The color value is green

#### 5.95.2.4 #define INPUT\_REDCOLOR 5

The color value is red

#### 5.95.2.5 #define INPUT\_WHITECOLOR 6

The color value is white

#### 5.95.2.6 #define INPUT\_YELLOWCOLOR 4

The color value is yellow

## 5.96 Color calibration state constants

Constants for use with the color calibration state function.

### Macros

- #define INPUT\_SENSORCAL 0x01
- #define INPUT\_SENSOROFF 0x02
- #define INPUT\_RUNNINGCAL 0x20
- #define INPUT\_STARTCAL 0x40
- #define INPUT\_RESETCAL 0x80

### 5.96.1 Detailed Description

Constants for use with the color calibration state function.

#### See Also

[ColorCalibrationState\(\)](#)

### 5.96.2 Macro Definition Documentation

#### 5.96.2.1 #define INPUT\_RESETCAL 0x80

Unused calibration state constant

#### 5.96.2.2 #define INPUT\_RUNNINGCAL 0x20

Unused calibration state constant

#### 5.96.2.3 #define INPUT\_SENSORCAL 0x01

The state returned while the color sensor is calibrating

#### 5.96.2.4 #define INPUT\_SENSOROFF 0x02

The state returned once calibration has completed

#### 5.96.2.5 #define INPUT\_STARTCAL 0x40

Unused calibration state constant

## 5.97 Color calibration constants

Constants for use with the color calibration functions.

### Macros

- `#define INPUT_CAL_POINT_0 0`
- `#define INPUT_CAL_POINT_1 1`
- `#define INPUT_CAL_POINT_2 2`
- `#define INPUT_NO_OF_POINTS 3`

### 5.97.1 Detailed Description

Constants for use with the color calibration functions.

#### See Also

`ColorCalibration()`, `ColorCalLimits()`

### 5.97.2 Macro Definition Documentation

#### 5.97.2.1 `#define INPUT_CAL_POINT_0 0`

Calibration point 0

#### 5.97.2.2 `#define INPUT_CAL_POINT_1 1`

Calibration point 1

#### 5.97.2.3 `#define INPUT_CAL_POINT_2 2`

Calibration point 2

#### 5.97.2.4 `#define INPUT_NO_OF_POINTS 3`

The number of calibration points

## 5.98 Input module IOMAP offsets

Constant offsets into the Input module IOMAP structure.

### Macros

- `#define InputOffsetCustomZeroOffset(p) (((p)*20)+0)`
- `#define InputOffsetADRaw(p) (((p)*20)+2)`
- `#define InputOffsetSensorRaw(p) (((p)*20)+4)`
- `#define InputOffsetSensorValue(p) (((p)*20)+6)`
- `#define InputOffsetSensorType(p) (((p)*20)+8)`
- `#define InputOffsetSensorMode(p) (((p)*20)+9)`
- `#define InputOffsetSensorBoolean(p) (((p)*20)+10)`
- `#define InputOffsetDigiPinsDir(p) (((p)*20)+11)`
- `#define InputOffsetDigiPinsIn(p) (((p)*20)+12)`
- `#define InputOffsetDigiPinsOut(p) (((p)*20)+13)`
- `#define InputOffsetCustomPctFullScale(p) (((p)*20)+14)`
- `#define InputOffsetCustomActiveStatus(p) (((p)*20)+15)`
- `#define InputOffsetInvalidData(p) (((p)*20)+16)`
- `#define InputOffsetColorCalibration(p, np, nc) (80+((p)*84)+0+((np)*16)+((nc)*4))`
- `#define InputOffsetColorCalLimits(p, np) (80+((p)*84)+48+((np)*2))`
- `#define InputOffsetColorADRaw(p, nc) (80+((p)*84)+52+((nc)*2))`
- `#define InputOffsetColorSensorRaw(p, nc) (80+((p)*84)+60+((nc)*2))`
- `#define InputOffsetColorSensorValue(p, nc) (80+((p)*84)+68+((nc)*2))`
- `#define InputOffsetColorBoolean(p, nc) (80+((p)*84)+76+((nc)*2))`
- `#define InputOffsetColorCalibrationState(p) (80+((p)*84)+80)`

### 5.98.1 Detailed Description

Constant offsets into the Input module IOMAP structure.

### 5.98.2 Macro Definition Documentation

#### 5.98.2.1 `#define InputOffsetADRaw( p ) (((p)*20)+2)`

Read the AD raw sensor value (2 bytes) uword

#### 5.98.2.2 `#define InputOffsetColorADRaw( p, nc ) (80+((p)*84)+52+((nc)*2))`

Read AD raw color sensor values

#### 5.98.2.3 `#define InputOffsetColorBoolean( p, nc ) (80+((p)*84)+76+((nc)*2))`

Read color sensor boolean values

#### 5.98.2.4 `#define InputOffsetColorCalibration( p, np, nc ) (80+((p)*84)+0+((np)*16)+((nc)*4))`

Read/write color calibration point values

#### 5.98.2.5 `#define InputOffsetColorCalibrationState( p ) (80+((p)*84)+80)`

Read color sensor calibration state

5.98.2.6 #define InputOffsetColorCallLimits( *p, np* ) (80+((*p*)\*84)+48+((*np*)\*2))

Read/write color calibration limits

5.98.2.7 #define InputOffsetColorSensorRaw( *p, nc* ) (80+((*p*)\*84)+60+((*nc*)\*2))

Read raw color sensor values

5.98.2.8 #define InputOffsetColorSensorValue( *p, nc* ) (80+((*p*)\*84)+68+((*nc*)\*2))

Read scaled color sensor values

5.98.2.9 #define InputOffsetCustomActiveStatus( *p* ) (((*p*)\*20)+15)

Read/write the active or inactive state of the custom sensor

5.98.2.10 #define InputOffsetCustomPctFullScale( *p* ) (((*p*)\*20)+14)

Read/write the Pct full scale of the custom sensor

5.98.2.11 #define InputOffsetCustomZeroOffset( *p* ) (((*p*)\*20)+0)

Read/write the zero offset of a custom sensor (2 bytes) uword

5.98.2.12 #define InputOffsetDigiPinsDir( *p* ) (((*p*)\*20)+11)

Read/write the direction of the Digital pins (1 is output, 0 is input)

5.98.2.13 #define InputOffsetDigiPinsIn( *p* ) (((*p*)\*20)+12)

Read/write the status of the digital pins

5.98.2.14 #define InputOffsetDigiPinsOut( *p* ) (((*p*)\*20)+13)

Read/write the output level of the digital pins

5.98.2.15 #define InputOffsetInvalidData( *p* ) (((*p*)\*20)+16)

Indicates whether data is invalid (1) or valid (0)

5.98.2.16 #define InputOffsetSensorBoolean( *p* ) (((*p*)\*20)+10)

Read the sensor boolean value

5.98.2.17 #define InputOffsetSensorMode( *p* ) (((*p*)\*20)+9)

Read/write the sensor mode

5.98.2.18 #define InputOffsetSensorRaw( *p* ) (((*p*)\*20)+4)

Read the raw sensor value (2 bytes) uword

5.98.2.19 #define InputOffsetSensorType( *p* ) (((*p*)\*20)+8)

Read/write the sensor type

5.98.2.20 #define InputOffsetSensorValue( *p* ) (((*p*)\*20)+6)

Read/write the scaled sensor value (2 bytes) sword

## 5.99 Constants to use with the Input module's Pin function

Constants for use with the Input module's Pin function.

### Macros

- #define INPUT\_PINCMD\_DIR 0x00
- #define INPUT\_PINCMD\_SET 0x01
- #define INPUT\_PINCMD\_CLEAR 0x02
- #define INPUT\_PINCMD\_READ 0x03
- #define INPUT\_PINCMD\_MASK 0x03
- #define INPUT\_PINCMD\_WAIT(\_usec) ((.\_usec)<<2)
- #define INPUT\_PINDIR\_OUTPUT 0x00
- #define INPUT\_PINDIR\_INPUT 0x04

### 5.99.1 Detailed Description

Constants for use with the Input module's Pin function. These are the commands that you can pass into the pin function to change digital pin directions, set or clear pins, or read pin values. Also in this group are mask constants and a macro for ORing a microsecond wait onto the command byte which will occur after the command has been executed.

### 5.99.2 Macro Definition Documentation

#### 5.99.2.1 #define INPUT\_PINCMD\_CLEAR 0x02

Clear digital pin(s)

#### 5.99.2.2 #define INPUT\_PINCMD\_DIR 0x00

Set digital pin(s) direction

#### 5.99.2.3 #define INPUT\_PINCMD\_MASK 0x03

Mask for the two bits used by pin function commands

#### 5.99.2.4 #define INPUT\_PINCMD\_READ 0x03

Read digital pin(s)

#### 5.99.2.5 #define INPUT\_PINCMD\_SET 0x01

Set digital pin(s)

#### 5.99.2.6 #define INPUT\_PINCMD\_WAIT( \_usec ) ((.\_usec)<<2)

A wait value in microseconds that can be added after one of the above commands by ORing with the command

#### 5.99.2.7 #define INPUT\_PINDIR\_INPUT 0x04

Use with the direction command to set direction to output. OR this with the pin value.

#### 5.99.2.8 #define INPUT\_PINDIR\_OUTPUT 0x00

Use with the direction command to set direction to input. OR this with the pin value.

## 5.100 Output port constants

Output port constants are used when calling motor control API functions.

### Macros

- #define OUT\_A 0x00
- #define OUT\_B 0x01
- #define OUT\_C 0x02
- #define OUT\_AB 0x03
- #define OUT\_AC 0x04
- #define OUT\_BC 0x05
- #define OUT\_ABC 0x06

### 5.100.1 Detailed Description

Output port constants are used when calling motor control API functions.

### 5.100.2 Macro Definition Documentation

#### 5.100.2.1 #define OUT\_A 0x00

Output port A

#### 5.100.2.2 #define OUT\_AB 0x03

Output ports A and B

#### 5.100.2.3 #define OUT\_ABC 0x06

Output ports A, B, and C

#### 5.100.2.4 #define OUT\_AC 0x04

Output ports A and C

#### 5.100.2.5 #define OUT\_B 0x01

Output port B

#### 5.100.2.6 #define OUT\_BC 0x05

Output ports B and C

#### 5.100.2.7 #define OUT\_C 0x02

Output port C

## 5.101 PID constants

PID constants are for adjusting the Proportional, Integral, and Derivative motor controller parameters.

### Macros

- `#define PID_0 0`
- `#define PID_1 32`
- `#define PID_2 64`
- `#define PID_3 96`
- `#define PID_4 128`
- `#define PID_5 160`
- `#define PID_6 192`
- `#define PID_7 224`

### 5.101.1 Detailed Description

PID constants are for adjusting the Proportional, Integral, and Derivative motor controller parameters.

### See Also

[RotateMotorExPID\(\)](#), [RotateMotorPID\(\)](#), [OnFwdExPID\(\)](#), [OnRevExPID\(\)](#),  
[OnFwdRegExPID\(\)](#), [OnRevRegExPID\(\)](#), [OnFwdRegPID\(\)](#), [OnRevRegPID\(\)](#),  
[OnFwdSyncExPID\(\)](#), [OnRevSyncExPID\(\)](#), [OnFwdSyncPID\(\)](#), [OnRevSyncPID\(\)](#)

### 5.101.2 Macro Definition Documentation

#### 5.101.2.1 `#define PID_0 0`

PID zero

#### 5.101.2.2 `#define PID_1 32`

PID one

#### 5.101.2.3 `#define PID_2 64`

PID two

#### 5.101.2.4 `#define PID_3 96`

PID three

#### 5.101.2.5 `#define PID_4 128`

PID four

#### 5.101.2.6 `#define PID_5 160`

PID five

#### 5.101.2.7 `#define PID_6 192`

PID six

## 5.101.2.8 #define PID\_7 224

PID seven

## 5.102 Output port update flag constants

Use these constants to specify which motor values need to be updated.

### Macros

- `#define UF_UPDATE_MODE 0x01`
- `#define UF_UPDATE_SPEED 0x02`
- `#define UF_UPDATE_TACHO_LIMIT 0x04`
- `#define UF_UPDATE_RESET_COUNT 0x08`
- `#define UF_UPDATE_PID_VALUES 0x10`
- `#define UF_UPDATE_RESET_BLOCK_COUNT 0x20`
- `#define UF_UPDATE_RESET_ROTATION_COUNT 0x40`
- `#define UF_PENDING_UPDATES 0x80`

### 5.102.1 Detailed Description

Use these constants to specify which motor values need to be updated. Update flag constants can be combined with bitwise OR.

### See Also

[SetOutput\(\)](#)

### 5.102.2 Macro Definition Documentation

#### 5.102.2.1 `#define UF_PENDING_UPDATES 0x80`

Are there any pending motor updates?

#### 5.102.2.2 `#define UF_UPDATE_MODE 0x01`

Commits changes to the [OutputModeField](#) output property

#### 5.102.2.3 `#define UF_UPDATE_PID_VALUES 0x10`

Commits changes to the PID motor regulation properties

#### 5.102.2.4 `#define UF_UPDATE_RESET_BLOCK_COUNT 0x20`

Resets the NXT-G block-relative rotation counter

#### 5.102.2.5 `#define UF_UPDATE_RESET_COUNT 0x08`

Resets all rotation counters, cancels the current goal, and resets the rotation error-correction system

#### 5.102.2.6 `#define UF_UPDATE_RESET_ROTATION_COUNT 0x40`

Resets the program-relative (user) rotation counter

#### 5.102.2.7 `#define UF_UPDATE_SPEED 0x02`

Commits changes to the [PowerField](#) output property

## 5.102.2.8 #define UF\_UPDATE\_TACHO\_LIMIT 0x04

Commits changes to the [TachoLimitField](#) output property

## 5.103 Tachometer counter reset flags

Use these constants to specify which of the three tachometer counters should be reset.

### Macros

- #define RESET\_NONE 0x00
- #define RESET\_COUNT 0x08
- #define RESET\_BLOCK\_COUNT 0x20
- #define RESET\_ROTATION\_COUNT 0x40
- #define RESET\_BLOCKANDTACHO 0x28
- #define RESET\_ALL 0x68

### 5.103.1 Detailed Description

Use these constants to specify which of the three tachometer counters should be reset. Reset constants can be combined with bitwise OR.

### See Also

[OnFwdEx\(\)](#), [OnRevEx\(\)](#), etc...

### 5.103.2 Macro Definition Documentation

#### 5.103.2.1 #define RESET\_ALL 0x68

Reset all three tachometer counters

#### 5.103.2.2 #define RESET\_BLOCK\_COUNT 0x20

Reset the NXT-G block tachometer counter

#### 5.103.2.3 #define RESET\_BLOCKANDTACHO 0x28

Reset both the internal counter and the NXT-G block counter

#### 5.103.2.4 #define RESET\_COUNT 0x08

Reset the internal tachometer counter

#### 5.103.2.5 #define RESET\_NONE 0x00

No counters will be reset

#### 5.103.2.6 #define RESET\_ROTATION\_COUNT 0x40

Reset the rotation counter

## 5.104 Output port mode constants

Use these constants to configure the desired mode for the specified motor(s): coast, motoron, brake, or regulated.

### Macros

- `#define OUT_MODE_COAST 0x00`
- `#define OUT_MODE_MOTORON 0x01`
- `#define OUT_MODE BRAKE 0x02`
- `#define OUT_MODE_REGULATED 0x04`
- `#define OUT_MODE_REGMETHOD 0xF0`

### 5.104.1 Detailed Description

Use these constants to configure the desired mode for the specified motor(s): coast, motoron, brake, or regulated. Mode constants can be combined with bitwise OR.

### See Also

`SetOutput()`

### 5.104.2 Macro Definition Documentation

#### 5.104.2.1 `#define OUT_MODE_BRAKE 0x02`

Uses electronic braking to outputs

#### 5.104.2.2 `#define OUT_MODE_COAST 0x00`

No power and no braking so motors rotate freely.

#### 5.104.2.3 `#define OUT_MODE_MOTORON 0x01`

Enables PWM power to the outputs given the power setting

#### 5.104.2.4 `#define OUT_MODE_REGMETHOD 0xF0`

Mask for unimplemented regulation mode

#### 5.104.2.5 `#define OUT_MODE_REGULATED 0x04`

Enables active power regulation using the regulation mode value

## 5.105 Output port option constants

Use these constants to configure the desired options for the specified motor(s): hold at limit and ramp down to limit.

### Macros

- `#define OUT_OPTION_HOLDATLIMIT 0x10`
- `#define OUT_OPTION_RAMPDOWNTOLIMIT 0x20`

### 5.105.1 Detailed Description

Use these constants to configure the desired options for the specified motor(s): hold at limit and ramp down to limit. Option constants can be combined with bitwise OR.

### Warning

These options require the enhanced NBC/NXC firmware version 1.31+

### See Also

`SetOutput()`

### 5.105.2 Macro Definition Documentation

#### 5.105.2.1 `#define OUT_OPTION_HOLDATLIMIT 0x10`

Option to have the firmware hold the motor when it reaches the tachometer limit

#### 5.105.2.2 `#define OUT_OPTION_RAMPDOWNTOLIMIT 0x20`

Option to have the firmware rampdown the motor power as it approaches the tachometer limit

## 5.106 Output regulation option constants

Use these constants to configure the desired options for position regulation.

### Macros

- `#define OUT_REGOPTION_NO_SATURATION 0x01`

### 5.106.1 Detailed Description

Use these constants to configure the desired options for position regulation.

### Warning

These options require the enhanced NBC/NXC firmware version 1.31+

### 5.106.2 Macro Definition Documentation

#### 5.106.2.1 `#define OUT_REGOPTION_NO_SATURATION 0x01`

Do not limit intermediary regulation results

## 5.107 Output port run state constants

Use these constants to configure the desired run state for the specified motor(s): idle, rampup, running, rampdown, or hold.

### Macros

- `#define OUT_RUNSTATE_IDLE 0x00`
- `#define OUT_RUNSTATE_RAMPUP 0x10`
- `#define OUT_RUNSTATE_RUNNING 0x20`
- `#define OUT_RUNSTATE_RAMPDOWN 0x40`
- `#define OUT_RUNSTATE_HOLD 0x60`

### 5.107.1 Detailed Description

Use these constants to configure the desired run state for the specified motor(s): idle, rampup, running, rampdown, or hold.

### See Also

[SetOutput\(\)](#)

### 5.107.2 Macro Definition Documentation

#### 5.107.2.1 `#define OUT_RUNSTATE_HOLD 0x60`

Set motor run state to hold at the current position.

#### 5.107.2.2 `#define OUT_RUNSTATE_IDLE 0x00`

Disable all power to motors.

#### 5.107.2.3 `#define OUT_RUNSTATE_RAMPDOWN 0x40`

Enable ramping down from a current power to a new (lower) power over a specified [TachoLimitField](#) goal.

#### 5.107.2.4 `#define OUT_RUNSTATE_RAMPUP 0x10`

Enable ramping up from a current power to a new (higher) power over a specified [TachoLimitField](#) goal.

#### 5.107.2.5 `#define OUT_RUNSTATE_RUNNING 0x20`

Enable power to motors at the specified power level.

## 5.108 Output port regulation mode constants

Use these constants to configure the desired regulation mode for the specified motor(s): none, speed regulation, multi-motor synchronization, or position regulation (requires the enhanced NBC/NXC firmware version 1.31+).

### Macros

- `#define OUT_REGMODE_IDLE 0`
- `#define OUT_REGMODE_SPEED 1`
- `#define OUT_REGMODE_SYNC 2`
- `#define OUT_REGMODE_POS 4`

### 5.108.1 Detailed Description

Use these constants to configure the desired regulation mode for the specified motor(s): none, speed regulation, multi-motor synchronization, or position regulation (requires the enhanced NBC/NXC firmware version 1.31+).

### See Also

[SetOutput\(\)](#)

### 5.108.2 Macro Definition Documentation

#### 5.108.2.1 `#define OUT_REGMODE_IDLE 0`

No motor regulation.

#### 5.108.2.2 `#define OUT_REGMODE_POS 4`

Regulate a motor's position.

#### 5.108.2.3 `#define OUT_REGMODE_SPEED 1`

Regulate a motor's speed (aka power).

#### 5.108.2.4 `#define OUT_REGMODE_SYNC 2`

Synchronize the rotation of two motors.

## 5.109 Output field constants

Constants for use with SetOutput() and GetOutput().

### Macros

- #define UpdateFlagsField 0  
*Update flags field.*
- #define OutputModeField 1  
*Mode field.*
- #define PowerField 2  
*Power field.*
- #define ActualSpeedField 3  
*Actual speed field.*
- #define TachoCountField 4  
*Internal tachometer count field.*
- #define TachoLimitField 5  
*Tachometer limit field.*
- #define RunStateField 6  
*Run state field.*
- #define TurnRatioField 7  
*Turn ratio field.*
- #define RegModeField 8  
*Regulation mode field.*
- #define OverloadField 9  
*Overload field.*
- #define RegPValueField 10  
*Proportional field.*
- #define RegIValueField 11  
*Integral field.*
- #define RegDValueField 12  
*Derivative field.*
- #define BlockTachoCountField 13  
*NXT-G block tachometer count field.*
- #define RotationCountField 14  
*Rotation counter field.*
- #define OutputOptionsField 15  
*Options field.*
- #define MaxSpeedField 16  
*MaxSpeed field.*
- #define MaxAccelerationField 17  
*MaxAcceleration field.*

### 5.109.1 Detailed Description

Constants for use with SetOutput() and GetOutput().

### See Also

SetOutput(), GetOutput()

### 5.109.2 Macro Definition Documentation

#### 5.109.2.1 #define ActualSpeedField 3

Actual speed field.

Contains the actual power level (-100 to 100). Read only. Return the percent of full power the firmware is applying to the output. This may vary from the PowerField value when auto-regulation code in the firmware responds to a load on the output.

#### 5.109.2.2 #define BlockTachoCountField 13

NXT-G block tachometer count field.

Contains the current NXT-G block tachometer count. Read only. Return the block-relative position counter value for the specified port. Refer to the [UpdateFlagsField](#) description for information about how to use block-relative position counts. Set the [UF\\_UPDATE\\_RESET\\_BLOCK\\_COUNT](#) flag in [UpdateFlagsField](#) to request that the firmware reset the BlockTachoCountField. The sign of BlockTachoCountField indicates the direction of rotation. Positive values indicate forward rotation and negative values indicate reverse rotation. Forward and reverse depend on the orientation of the motor.

#### 5.109.2.3 #define MaxAccelerationField 17

MaxAcceleration field.

Contains the current max acceleration value. Read/write. Set the maximum acceleration to be used during position regulation.

#### Warning

This option requires the enhanced NBC/NXC firmware version 1.31+

#### 5.109.2.4 #define MaxSpeedField 16

MaxSpeed field.

Contains the current max speed value. Read/write. Set the maximum speed to be used during position regulation.

#### Warning

This option requires the enhanced NBC/NXC firmware version 1.31+

#### 5.109.2.5 #define OutputModeField 1

Mode field.

Contains a combination of the output mode constants. Read/write. The [OUT\\_MODE\\_MOTORON](#) bit must be set in order for power to be applied to the motors. Add [OUT\\_MODE\\_BRAKE](#) to enable electronic braking. Braking means that the output voltage is not allowed to float between active PWM pulses. It improves the accuracy of motor output but uses more battery power. To use motor regulation include [OUT\\_MODE\\_REGULATED](#) in the OutputModeField value. Use [UF\\_UPDATE\\_MODE](#) with [UpdateFlagsField](#) to commit changes to this field.

#### 5.109.2.6 #define OutputOptionsField 15

Options field.

Contains a combination of the output options constants. Read/write. Set options for how the output module will act when a tachometer limit is reached. Option constants can be combined with bitwise OR. Use [OUT\\_OPTION\\_HOLDATLIMIT](#)

to have the output module hold the motor when it reaches the tachometer limit. Use `OUT_OPTION_RAMPDOWNTOLIMIT` to have the output module ramp down the motor power as it approaches the tachometer limit.

#### Warning

This option requires the enhanced NBC/NXC firmware version 1.31+

##### 5.109.2.7 #define OverloadField 9

Overload field.

Contains a boolean value which is TRUE if the motor is overloaded. Read only. This field will have a value of 1 (true) if the firmware speed regulation cannot overcome a physical load on the motor. In other words, the motor is turning more slowly than expected. If the motor speed can be maintained in spite of loading then this field value is zero (false). In order to use this field the motor must have a non-idle `RunStateField`, an `OutputModeField` which includes `OUT_MODE_MOTORON` and `OUT_MODE_REGULATED`, and its `RegModeField` must be set to `OUT_REGMODE_SPEED`.

##### 5.109.2.8 #define PowerField 2

Power field.

Contains the desired power level (-100 to 100). Read/write. Specify the power level of the output. The absolute value of `PowerField` is a percentage of the full power of the motor. The sign of `PowerField` controls the rotation direction. Positive values tell the firmware to turn the motor forward, while negative values turn the motor backward. Use `UF_UPDATE_SPEED` with `UpdateFlagsField` to commit changes to this field.

##### 5.109.2.9 #define RegDValueField 12

Derivative field.

Contains the derivative constant for the PID motor controller. Read/write. This field specifies the derivative term used in the internal proportional-integral-derivative (PID) control algorithm. Set `UF_UPDATE_PID_VALUES` to commit changes to `RegPValue`, `RegIValue`, and `RegDValue` simultaneously.

##### 5.109.2.10 #define RegIValueField 11

Integral field.

Contains the integral constant for the PID motor controller. Read/write. This field specifies the integral term used in the internal proportional-integral-derivative (PID) control algorithm. Set `UF_UPDATE_PID_VALUES` to commit changes to `RegPValue`, `RegIValue`, and `RegDValue` simultaneously.

##### 5.109.2.11 #define RegModeField 8

Regulation mode field.

Contains one of the regulation mode constants. Read/write. This field specifies the regulation mode to use with the specified port(s). It is ignored if the `OUT_MODE_REGULATED` bit is not set in the `OutputModeField` field. Unlike `OutputModeField`, `RegModeField` is not a bitfield. Only one regulation mode value can be set at a time. Speed regulation means that the firmware tries to maintain a certain speed based on the `PowerField` setting. The firmware adjusts the PWM duty cycle if the motor is affected by a physical load. This adjustment is reflected by the value of the `ActualSpeedField` property. When using speed regulation, do not set `PowerField` to its maximum value since the firmware cannot adjust to higher power levels in that situation. Synchronization means the firmware tries to keep two motors in sync regardless of physical loads. Use this mode to maintain a straight path for a mobile robot automatically. Also use this mode with the `TurnRatioField` property to provide proportional turning. Set `OUT_REGMODE_SYNC` on at least two motor ports in order for synchronization to function. Setting `OUT_REGMODE_SYNC` on all three motor ports will result in only the first two (`OUT_A` and `OUT_B`) being synchronized.

**5.109.2.12 #define RegPValueField 10**

Proportional field.

Contains the proportional constant for the PID motor controller. Read/write. This field specifies the proportional term used in the internal proportional-integral-derivative (PID) control algorithm. Set [UF\\_UPDATE\\_PID\\_VALUES](#) to commit changes to RegPValue, RegIValue, and RegDValue simultaneously.

**5.109.2.13 #define RotationCountField 14**

Rotation counter field.

Contains the current rotation count. Read only. Return the program-relative position counter value for the specified port. Refer to the [UpdateFlagsField](#) description for information about how to use program-relative position counts. Set the [UF\\_UPDATE\\_RESET\\_ROTATION\\_COUNT](#) flag in [UpdateFlagsField](#) to request that the firmware reset the RotationCountField. The sign of RotationCountField indicates the direction of rotation. Positive values indicate forward rotation and negative values indicate reverse rotation. Forward and reverse depend on the orientation of the motor.

**5.109.2.14 #define RunStateField 6**

Run state field.

Contains one of the run state constants. Read/write. Use this field to specify the running state of an output. Set the RunStateField to [OUT\\_RUNSTATE\\_RUNNING](#) to enable power to any output. Use [OUT\\_RUNSTATE\\_RAMPUP](#) to enable automatic ramping to a new [PowerField](#) level greater than the current [PowerField](#) level. Use [OUT\\_RUNSTATE\\_RAMPDOWN](#) to enable automatic ramping to a new [PowerField](#) level less than the current [PowerField](#) level. Both the rampup and rampdown bits must be used in conjunction with appropriate [TachoLimitField](#) and [PowerField](#) values. In this case the firmware smoothly increases or decreases the actual power to the new [PowerField](#) level over the total number of degrees of rotation specified in [TachoLimitField](#).

**5.109.2.15 #define TachoCountField 4**

Internal tachometer count field.

Contains the current internal tachometer count. Read only. Return the internal position counter value for the specified output. The internal count is reset automatically when a new goal is set using the [TachoLimitField](#) and the [UF\\_UPDATE\\_TACHO\\_LIMIT](#) flag. Set the [UF\\_UPDATE\\_RESET\\_COUNT](#) flag in [UpdateFlagsField](#) to reset TachoCountField and cancel any [TachoLimitField](#). The sign of TachoCountField indicates the motor rotation direction.

**5.109.2.16 #define TachoLimitField 5**

Tachometer limit field.

Contains the current tachometer limit. Read/write. Specify the number of degrees the motor should rotate. Use [UF\\_UPDATE\\_TACHO\\_LIMIT](#) with the [UpdateFlagsField](#) field to commit changes to the TachoLimitField. The value of this field is a relative distance from the current motor position at the moment when the [UF\\_UPDATE\\_TACHO\\_LIMIT](#) flag is processed.

**5.109.2.17 #define TurnRatioField 7**

Turn ratio field.

Contains the current turn ratio. Only applicable when synchronizing multiple motors. Read/write. Use this field to specify a proportional turning ratio. This field must be used in conjunction with other field values: [OutputModeField](#) must include [OUT\\_MODE\\_MOTORON](#) and [OUT\\_MODE\\_REGULATED](#), [RegModeField](#) must be set to [OUT\\_REGMODE\\_SYNC](#), [RunStateField](#) must not be [OUT\\_RUNSTATE\\_IDLE](#), and [PowerField](#) must be non-zero. There are only three valid combinations of left and right motors for use with TurnRatioField: [OUT\\_AB](#), [OUT\\_BC](#), and [OUT\\_AC](#). In each of these three options the first motor listed is considered to be the left motor and the second motor is the right motor, regardless of the physical configuration of the robot. Negative turn ratio values shift power toward the left motor while

positive values shift power toward the right motor. An absolute value of 50 usually results in one motor stopping. An absolute value of 100 usually results in two motors turning in opposite directions at equal power.

#### 5.109.2.18 #define UpdateFlagsField 0

Update flags field.

Contains a combination of the update flag constants. Read/write. Use [UF\\_UPDATE\\_MODE](#), [UF\\_UPDATE\\_SPEED](#), [UF\\_UPDATE\\_TACHO\\_LIMIT](#), and [UF\\_UPDATE\\_PID\\_VALUES](#) along with other fields to commit changes to the state of outputs. Set the appropriate flags after setting one or more of the output fields in order for the changes to actually go into affect.

## 5.110 Output module IOMAP offsets

Constant offsets into the Output module IOMAP structure.

### Macros

- `#define OutputOffsetTachoCount(p) (((p)*32)+0)`
- `#define OutputOffsetBlockTachoCount(p) (((p)*32)+4)`
- `#define OutputOffsetRotationCount(p) (((p)*32)+8)`
- `#define OutputOffsetTachoLimit(p) (((p)*32)+12)`
- `#define OutputOffsetMotorRPM(p) (((p)*32)+16)`
- `#define OutputOffsetFlags(p) (((p)*32)+18)`
- `#define OutputOffsetMode(p) (((p)*32)+19)`
- `#define OutputOffsetSpeed(p) (((p)*32)+20)`
- `#define OutputOffsetActualSpeed(p) (((p)*32)+21)`
- `#define OutputOffsetRegPParameter(p) (((p)*32)+22)`
- `#define OutputOffsetRegIParameter(p) (((p)*32)+23)`
- `#define OutputOffsetRegDParameter(p) (((p)*32)+24)`
- `#define OutputOffsetRunState(p) (((p)*32)+25)`
- `#define OutputOffsetRegMode(p) (((p)*32)+26)`
- `#define OutputOffsetOverloaded(p) (((p)*32)+27)`
- `#define OutputOffsetSyncTurnParameter(p) (((p)*32)+28)`
- `#define OutputOffsetOptions(p) (((p)*32)+29)`
- `#define OutputOffsetMaxSpeed(p) (((p)*32)+30)`
- `#define OutputOffsetMaxAccel(p) (((p)*32)+31)`
- `#define OutputOffsetRegulationTime 96`
- `#define OutputOffsetRegulationOptions 97`

### 5.110.1 Detailed Description

Constant offsets into the Output module IOMAP structure.

### 5.110.2 Macro Definition Documentation

#### 5.110.2.1 `#define OutputOffsetActualSpeed( p ) (((p)*32)+21)`

R - Holds the current motor speed (1 byte) sbyte

#### 5.110.2.2 `#define OutputOffsetBlockTachoCount( p ) (((p)*32)+4)`

R - Holds current number of counts for the current output block (4 bytes) slong

#### 5.110.2.3 `#define OutputOffsetFlags( p ) (((p)*32)+18)`

RW - Holds flags for which data should be updated (1 byte) ubyte

#### 5.110.2.4 `#define OutputOffsetMaxAccel( p ) (((p)*32)+31)`

RW - holds the maximum acceleration for position regulation (1 byte) sbyte (enhanced NBC/NXC firmware only)

5.110.2.5 `#define OutputOffsetMaxSpeed( p ) (((p)*32)+30)`

RW - holds the maximum speed for position regulation (1 byte) sbyte (enhanced NBC/NXC firmware only)

5.110.2.6 `#define OutputOffsetMode( p ) (((p)*32)+19)`

RW - Holds motor mode: Run, Break, regulated, ... (1 byte) ubyte

5.110.2.7 `#define OutputOffsetMotorRPM( p ) (((p)*32)+16)`

Not updated, will be removed later !! (2 bytes) sword

5.110.2.8 `#define OutputOffsetOptions( p ) (((p)*32)+29)`

RW - holds extra motor options related to the tachometer limit (1 byte) ubyte (enhanced NBC/NXC firmware only)

5.110.2.9 `#define OutputOffsetOverloaded( p ) (((p)*32)+27)`

R - True if the motor has been overloaded within speed control regulation (1 byte) ubyte

5.110.2.10 `#define OutputOffsetRegDParameter( p ) (((p)*32)+24)`

RW - Holds the D-constant used in the regulation (1 byte) ubyte

5.110.2.11 `#define OutputOffsetRegIParameter( p ) (((p)*32)+23)`

RW - Holds the I-constant used in the regulation (1 byte) ubyte

5.110.2.12 `#define OutputOffsetRegMode( p ) (((p)*32)+26)`

RW - Tells which regulation mode should be used (1 byte) ubyte

5.110.2.13 `#define OutputOffsetRegPParameter( p ) (((p)*32)+22)`

RW - Holds the P-constant used in the regulation (1 byte) ubyte

5.110.2.14 `#define OutputOffsetRegulationOptions 97`

use for position regulation options (1 byte) ubyte (enhanced NBC/NXC firmware only)

5.110.2.15 `#define OutputOffsetRegulationTime 96`

use for frequency of checking regulation mode (1 byte) ubyte (enhanced NBC/NXC firmware only)

5.110.2.16 `#define OutputOffsetRotationCount( p ) (((p)*32)+8)`

R - Holds current number of counts for the rotation counter to the output (4 bytes) slong

5.110.2.17 `#define OutputOffsetRunState( p ) (((p)*32)+25)`

RW - Holds the current motor run state in the output module (1 byte) ubyte

5.110.2.18 `#define OutputOffsetSpeed( p ) (((p)*32)+20)`

RW - Holds the wanted speed (1 byte) sbyte

5.110.2.19 #define OutputOffsetSyncTurnParameter( p ) (((p)\*32)+28)

RW - Holds the turning parameter need within MoveBlock (1 byte) sbyte

5.110.2.20 #define OutputOffsetTachoCount( p ) (((p)\*32)+0)

R - Holds current number of counts, since last reset, updated every 1 mS (4 bytes) slong

5.110.2.21 #define OutputOffsetTachoLimit( p ) (((p)\*32)+12)

RW - Holds number of counts to travel, 0 => Run forever (4 bytes) ulong

## 5.111 LowSpeed module constants

Constants that are part of the NXT firmware's LowSpeed module.

### Modules

- [E-Meter sensor constants](#)

*Constants for use with the e-meter sensor.*

- [I2C option constants](#)

*Constants for the SetI2COPTIONS function.*

- [LEGO I2C address constants](#)

*Constants for LEGO I2C device addresses.*

- [LEGO temperature sensor constants](#)

*Constants for use with the LEGO temperature sensor.*

- [LSChannelState constants](#)

*Constants for the low speed module LSChannelState function.*

- [LSErrorType constants](#)

*Constants for the low speed module LSErrorType function.*

- [LSMode constants](#)

*Constants for the low speed module LSMode function.*

- [LSNoRestartOnRead constants](#)

*Constants for the low speed module LSNoRestartOnRead and SetLSNoRestartOnRead functions.*

- [LSState constants](#)

*Constants for the low speed module LSState function.*

- [Low speed module IOMAP offsets](#)

*Constant offsets into the low speed module IOMAP structure.*

- [Standard I2C constants](#)

*Constants for use with standard I2C devices.*

- [Ultrasonic sensor constants](#)

*Constants for use with the ultrasonic sensor.*

### 5.111.1 Detailed Description

Constants that are part of the NXT firmware's LowSpeed module.

## 5.112 LSState constants

Constants for the low speed module LSState function.

### Macros

- #define COM\_CHANNEL\_NONE\_ACTIVE 0x00
- #define COM\_CHANNEL\_ONE\_ACTIVE 0x01
- #define COM\_CHANNEL\_TWO\_ACTIVE 0x02
- #define COM\_CHANNEL\_THREE\_ACTIVE 0x04
- #define COM\_CHANNEL\_FOUR\_ACTIVE 0x08

### 5.112.1 Detailed Description

Constants for the low speed module LSState function. These values are combined together using a bitwise OR operation.

### See Also

[LSState\(\)](#)

### 5.112.2 Macro Definition Documentation

#### 5.112.2.1 #define COM\_CHANNEL\_FOUR\_ACTIVE 0x08

Low speed channel 4 is active

#### 5.112.2.2 #define COM\_CHANNEL\_NONE\_ACTIVE 0x00

None of the low speed channels are active

#### 5.112.2.3 #define COM\_CHANNEL\_ONE\_ACTIVE 0x01

Low speed channel 1 is active

#### 5.112.2.4 #define COM\_CHANNEL\_THREE\_ACTIVE 0x04

Low speed channel 3 is active

#### 5.112.2.5 #define COM\_CHANNEL\_TWO\_ACTIVE 0x02

Low speed channel 2 is active

## 5.113 LSChannelState constants

Constants for the low speed module LSChannelState function.

### Macros

- #define LOWSPEED\_IDLE 0
- #define LOWSPEED\_INIT 1
- #define LOWSPEED\_LOAD\_BUFFER 2
- #define LOWSPEED\_COMMUNICATING 3
- #define LOWSPEED\_ERROR 4
- #define LOWSPEED\_DONE 5

### 5.113.1 Detailed Description

Constants for the low speed module LSChannelState function.

### See Also

[LSChannelState\(\)](#)

### 5.113.2 Macro Definition Documentation

#### 5.113.2.1 #define LOWSPEED\_COMMUNICATING 3

Channel is actively communicating

#### 5.113.2.2 #define LOWSPEED\_DONE 5

Channel is done communicating

#### 5.113.2.3 #define LOWSPEED\_ERROR 4

Channel is in an error state

#### 5.113.2.4 #define LOWSPEED\_IDLE 0

Channel is idle

#### 5.113.2.5 #define LOWSPEED\_INIT 1

Channel is being initialized

#### 5.113.2.6 #define LOWSPEED\_LOAD\_BUFFER 2

Channel buffer is loading

## 5.114 LSMode constants

Constants for the low speed module LSMode function.

### Macros

- #define LOWSPEED\_TRANSMITTING 1
- #define LOWSPEED RECEIVING 2
- #define LOWSPEED\_DATA RECEIVED 3

### 5.114.1 Detailed Description

Constants for the low speed module LSMode function.

### See Also

[LSMode\(\)](#)

### 5.114.2 Macro Definition Documentation

#### 5.114.2.1 #define LOWSPEED\_DATA RECEIVED 3

Lowspeed port is in data received mode

#### 5.114.2.2 #define LOWSPEED RECEIVING 2

Lowspeed port is in receiving mode

#### 5.114.2.3 #define LOWSPEED\_TRANSMITTING 1

Lowspeed port is in transmitting mode

## 5.115 LSErrorType constants

Constants for the low speed module LSErrorType function.

### Macros

- #define LOWSPEED\_NO\_ERROR 0
- #define LOWSPEED\_CH\_NOT\_READY 1
- #define LOWSPEED\_TX\_ERROR 2
- #define LOWSPEED\_RX\_ERROR 3

### 5.115.1 Detailed Description

Constants for the low speed module LSErrorType function.

### See Also

[LSErrorType\(\)](#)

### 5.115.2 Macro Definition Documentation

#### 5.115.2.1 #define LOWSPEED\_CH\_NOT\_READY 1

Lowspeed port is not ready

#### 5.115.2.2 #define LOWSPEED\_NO\_ERROR 0

Lowspeed port has no error

#### 5.115.2.3 #define LOWSPEED\_RX\_ERROR 3

Lowspeed port encountered an error while receiving data

#### 5.115.2.4 #define LOWSPEED\_TX\_ERROR 2

Lowspeed port encountered an error while transmitting data

## 5.116 Low speed module IOMAP offsets

Constant offsets into the low speed module IOMAP structure.

### Macros

- #define LowSpeedOffsetInBufBuf(p) (((p)\*19)+0)
- #define LowSpeedOffsetInBufInPtr(p) (((p)\*19)+16)
- #define LowSpeedOffsetInBufOutPtr(p) (((p)\*19)+17)
- #define LowSpeedOffsetInBufBytesToRx(p) (((p)\*19)+18)
- #define LowSpeedOffsetOutBufBuf(p) (((p)\*19)+76)
- #define LowSpeedOffsetOutBufInPtr(p) (((p)\*19)+92)
- #define LowSpeedOffsetOutBufOutPtr(p) (((p)\*19)+93)
- #define LowSpeedOffsetOutBufBytesToRx(p) (((p)\*19)+94)
- #define LowSpeedOffsetMode(p) ((p)+152)
- #define LowSpeedOffsetChannelState(p) ((p)+156)
- #define LowSpeedOffsetErrorType(p) ((p)+160)
- #define LowSpeedOffsetState 164
- #define LowSpeedOffsetSpeed 165
- #define LowSpeedOffsetNoRestartOnRead 166

### 5.116.1 Detailed Description

Constant offsets into the low speed module IOMAP structure.

### 5.116.2 Macro Definition Documentation

#### 5.116.2.1 #define LowSpeedOffsetChannelState( p ) ((p)+156)

R - Lowspeed channel state (1 byte)

#### 5.116.2.2 #define LowSpeedOffsetErrorType( p ) ((p)+160)

R - Lowspeed port error type (1 byte)

#### 5.116.2.3 #define LowSpeedOffsetInBufBuf( p ) (((p)\*19)+0)

RW - Input buffer data buffer field offset (16 bytes)

#### 5.116.2.4 #define LowSpeedOffsetInBufBytesToRx( p ) (((p)\*19)+18)

RW - Input buffer bytes to receive field offset (1 byte)

#### 5.116.2.5 #define LowSpeedOffsetInBufInPtr( p ) (((p)\*19)+16)

RW - Input buffer in pointer field offset (1 byte)

#### 5.116.2.6 #define LowSpeedOffsetInBufOutPtr( p ) (((p)\*19)+17)

RW - Input buffer out pointer field offset (1 byte)

5.116.2.7 #define LowSpeedOffsetMode( *p* ) ((*p*)+152)

R - Lowspeed port mode (1 byte)

5.116.2.8 #define LowSpeedOffsetNoRestartOnRead 166

RW - Lowspeed option for no restart on read (all channels) (NBC/NXC)

5.116.2.9 #define LowSpeedOffsetOutBufBuf( *p* ) (((*p*)\*19)+76)

RW - Output buffer data buffer field offset (16 bytes)

5.116.2.10 #define LowSpeedOffsetOutBufBytesToRx( *p* ) (((*p*)\*19)+94)

RW - Output buffer bytes to receive field offset (1 byte)

5.116.2.11 #define LowSpeedOffsetOutBufInPtr( *p* ) (((*p*)\*19)+92)

RW - Output buffer in pointer field offset (1 byte)

5.116.2.12 #define LowSpeedOffsetOutBufOutPtr( *p* ) (((*p*)\*19)+93)

RW - Output buffer out pointer field offset (1 byte)

5.116.2.13 #define LowSpeedOffsetSpeed 165

R - Lowspeed speed (unused)

5.116.2.14 #define LowSpeedOffsetState 164

R - Lowspeed state (all channels)

## 5.117 LSNoRestartOnRead constants

Constants for the low speed module LSNoRestartOnRead and SetLSNoRestartOnRead functions.

### Macros

- #define LSREAD\_RESTART\_ALL 0x00
- #define LSREAD\_NO\_RESTART\_1 0x01
- #define LSREAD\_NO\_RESTART\_2 0x02
- #define LSREAD\_NO\_RESTART\_3 0x04
- #define LSREAD\_NO\_RESTART\_4 0x08
- #define LSREAD\_RESTART\_NONE 0x0F
- #define LSREAD\_NO\_RESTART\_MASK 0x10

### 5.117.1 Detailed Description

Constants for the low speed module LSNoRestartOnRead and SetLSNoRestartOnRead functions. These values are combined with a bitwise OR operation.

### See Also

[LSNoRestartOnRead\(\)](#), [SetLSNoRestartOnRead\(\)](#)

### Warning

These options require the enhanced NBC/NXC firmware

### 5.117.2 Macro Definition Documentation

#### 5.117.2.1 #define LSREAD\_NO\_RESTART\_1 0x01

No restart on read for channel 1

#### 5.117.2.2 #define LSREAD\_NO\_RESTART\_2 0x02

No restart on read for channel 2

#### 5.117.2.3 #define LSREAD\_NO\_RESTART\_3 0x04

No restart on read for channel 3

#### 5.117.2.4 #define LSREAD\_NO\_RESTART\_4 0x08

No restart on read for channel 4

#### 5.117.2.5 #define LSREAD\_NO\_RESTART\_MASK 0x10

No restart mask

#### 5.117.2.6 #define LSREAD\_RESTART\_ALL 0x00

Restart on read for all channels (default)

## 5.117.2.7 #define LSREAD\_RESTART\_NONE 0x0F

No restart on read for all channels

## 5.118 Standard I2C constants

Constants for use with standard I2C devices.

### Macros

- `#define I2C_ADDR_DEFAULT 0x02`
- `#define I2C_REG_VERSION 0x00`
- `#define I2C_REG_VENDOR_ID 0x08`
- `#define I2C_REG_DEVICE_ID 0x10`
- `#define I2C_REG_CMD 0x41`

### 5.118.1 Detailed Description

Constants for use with standard I2C devices.

### 5.118.2 Macro Definition Documentation

#### 5.118.2.1 `#define I2C_ADDR_DEFAULT 0x02`

Standard NXT I2C device address

#### 5.118.2.2 `#define I2C_REG_CMD 0x41`

Standard NXT I2C device command register

#### 5.118.2.3 `#define I2C_REG_DEVICE_ID 0x10`

Standard NXT I2C device ID register

#### 5.118.2.4 `#define I2C_REG_VENDOR_ID 0x08`

Standard NXT I2C vendor ID register

#### 5.118.2.5 `#define I2C_REG_VERSION 0x00`

Standard NXT I2C version register

## 5.119 LEGO I2C address constants

Constants for LEGO I2C device addresses.

### Macros

- `#define LEGO_ADDR_US 0x02`
- `#define LEGO_ADDR_TEMP 0x98`
- `#define LEGO_ADDR_EMETER 0x04`

### 5.119.1 Detailed Description

Constants for LEGO I2C device addresses.

### 5.119.2 Macro Definition Documentation

#### 5.119.2.1 `#define LEGO_ADDR_EMETER 0x04`

The LEGO e-meter sensor's I2C address

#### 5.119.2.2 `#define LEGO_ADDR_TEMP 0x98`

The LEGO temperature sensor's I2C address

#### 5.119.2.3 `#define LEGO_ADDR_US 0x02`

The LEGO ultrasonic sensor's I2C address

## 5.120 Ultrasonic sensor constants

Constants for use with the ultrasonic sensor.

### Macros

- `#define US_CMD_OFF 0x00`
- `#define US_CMD_SINGLESHOT 0x01`
- `#define US_CMD_CONTINUOUS 0x02`
- `#define US_CMD_EVENTCAPTURE 0x03`
- `#define US_CMD_WARMRESET 0x04`
- `#define US_REG_CM_INTERVAL 0x40`
- `#define US_REG_ACTUAL_ZERO 0x50`
- `#define US_REG_SCALE_FACTOR 0x51`
- `#define US_REG_SCALE_DIVISOR 0x52`
- `#define US_REG_FACTORY_ACTUAL_ZERO 0x11`
- `#define US_REG_FACTORY_SCALE_FACTOR 0x12`
- `#define US_REG_FACTORY_SCALE_DIVISOR 0x13`
- `#define US_REG_MEASUREMENT_UNITS 0x14`

### 5.120.1 Detailed Description

Constants for use with the ultrasonic sensor.

### 5.120.2 Macro Definition Documentation

#### 5.120.2.1 `#define US_CMD_CONTINUOUS 0x02`

Command to put the ultrasonic sensor into continuous polling mode (default)

#### 5.120.2.2 `#define US_CMD_EVENTCAPTURE 0x03`

Command to put the ultrasonic sensor into event capture mode

#### 5.120.2.3 `#define US_CMD_OFF 0x00`

Command to turn off the ultrasonic sensor

#### 5.120.2.4 `#define US_CMD_SINGLESHOT 0x01`

Command to put the ultrasonic sensor into single shot mode

#### 5.120.2.5 `#define US_CMD_WARMRESET 0x04`

Command to warm reset the ultrasonic sensor

#### 5.120.2.6 `#define US_REG_ACTUAL_ZERO 0x50`

The register address used to store the actual zero value

#### 5.120.2.7 `#define US_REG_CM_INTERVAL 0x40`

The register address used to store the CM interval

5.120.2.8 #define US\_REG\_FACTORY\_ACTUAL\_ZERO 0x11

The register address containing the factory setting for the actual zero value

5.120.2.9 #define US\_REG\_FACTORY\_SCALE\_DIVISOR 0x13

The register address containing the factory setting for the scale divisor value

5.120.2.10 #define US\_REG\_FACTORY\_SCALE\_FACTOR 0x12

The register address containing the factory setting for the scale factor value

5.120.2.11 #define US\_REG\_MEASUREMENT\_UNITS 0x14

The register address containing the measurement units (degrees C or F)

5.120.2.12 #define US\_REG\_SCALE\_DIVISOR 0x52

The register address used to store the scale divisor value

5.120.2.13 #define US\_REG\_SCALE\_FACTOR 0x51

The register address used to store the scale factor value

## 5.121 LEGO temperature sensor constants

Constants for use with the LEGO temperature sensor.

### Macros

- #define TEMP\_RES\_9BIT 0x00
- #define TEMP\_RES\_10BIT 0x20
- #define TEMP\_RES\_11BIT 0x40
- #define TEMP\_RES\_12BIT 0x60
- #define TEMP\_SD\_CONTINUOUS 0x00
- #define TEMP\_SD\_SHUTDOWN 0x01
- #define TEMP\_TM\_COMPARATOR 0x00
- #define TEMP\_TM\_INTERRUPT 0x02
- #define TEMP\_OS\_ONESHOT 0x80
- #define TEMP\_FQ\_1 0x00
- #define TEMP\_FQ\_2 0x08
- #define TEMP\_FQ\_4 0x10
- #define TEMP\_FQ\_6 0x18
- #define TEMP\_POL\_LOW 0x00
- #define TEMP\_POL\_HIGH 0x04
- #define TEMP\_REG\_TEMP 0x00
- #define TEMP\_REG\_CONFIG 0x01
- #define TEMP\_REG\_TLOW 0x02
- #define TEMP\_REG\_THIGH 0x03

### 5.121.1 Detailed Description

Constants for use with the LEGO temperature sensor.

### 5.121.2 Macro Definition Documentation

#### 5.121.2.1 #define TEMP\_FQ\_1 0x00

Set fault queue to 1 fault before alert

#### 5.121.2.2 #define TEMP\_FQ\_2 0x08

Set fault queue to 2 faults before alert

#### 5.121.2.3 #define TEMP\_FQ\_4 0x10

Set fault queue to 4 faults before alert

#### 5.121.2.4 #define TEMP\_FQ\_6 0x18

Set fault queue to 6 faults before alert

#### 5.121.2.5 #define TEMP\_OS\_ONESHOT 0x80

Set the sensor into oneshot mode. When the device is in shutdown mode this will start a single temperature conversion. The device returns to shutdown mode when it completes.

5.121.2.6 #define TEMP\_POL\_HIGH 0x04

Set polarity of ALERT pin to be active HIGH

5.121.2.7 #define TEMP\_POL\_LOW 0x00

Set polarity of ALERT pin to be active LOW

5.121.2.8 #define TEMP\_REG\_CONFIG 0x01

The register for reading/writing sensor configuration values

5.121.2.9 #define TEMP\_REG\_TEMP 0x00

The register where temperature values can be read

5.121.2.10 #define TEMP\_REG\_THIGH 0x03

The register for reading/writing a user-defined high temperature limit

5.121.2.11 #define TEMP\_REG\_TLOW 0x02

The register for reading/writing a user-defined low temperature limit

5.121.2.12 #define TEMP\_RES\_10BIT 0x20

Set the temperature conversion resolution to 10 bit

5.121.2.13 #define TEMP\_RES\_11BIT 0x40

Set the temperature conversion resolution to 11 bit

5.121.2.14 #define TEMP\_RES\_12BIT 0x60

Set the temperature conversion resolution to 12 bit

5.121.2.15 #define TEMP\_RES\_9BIT 0x00

Set the temperature conversion resolution to 9 bit

5.121.2.16 #define TEMP\_SD\_CONTINUOUS 0x00

Set the sensor mode to continuous

5.121.2.17 #define TEMP\_SD\_SHUTDOWN 0x01

Set the sensor mode to shutdown. The device will shut down after the current conversion is completed.

5.121.2.18 #define TEMP\_TM\_COMPARATOR 0x00

Set the thermostat mode to comparator

5.121.2.19 #define TEMP\_TM\_INTERRUPT 0x02

Set the thermostat mode to interrupt

## 5.122 E-Meter sensor constants

Constants for use with the e-meter sensor.

### Macros

- #define EMETER\_REG\_VIN 0x0a
- #define EMETER\_REG\_AIN 0x0c
- #define EMETER\_REG\_VOUT 0x0e
- #define EMETER\_REG\_AOUT 0x10
- #define EMETER\_REG\_JOULES 0x12
- #define EMETER\_REG\_WIN 0x14
- #define EMETER\_REG\_WOUT 0x16

### 5.122.1 Detailed Description

Constants for use with the e-meter sensor.

### 5.122.2 Macro Definition Documentation

#### 5.122.2.1 #define EMETER\_REG\_AIN 0x0c

The register address for amps in

#### 5.122.2.2 #define EMETER\_REG\_AOUT 0x10

The register address for amps out

#### 5.122.2.3 #define EMETER\_REG\_JOULES 0x12

The register address for joules

#### 5.122.2.4 #define EMETER\_REG\_VIN 0x0a

The register address for voltage in

#### 5.122.2.5 #define EMETER\_REG\_VOUT 0x0e

The register address for voltage out

#### 5.122.2.6 #define EMETER\_REG\_WIN 0x14

The register address for watts in

#### 5.122.2.7 #define EMETER\_REG\_WOUT 0x16

The register address for watts out

## 5.123 I2C option constants

Constants for the SetI2COOptions function.

### Macros

- #define I2C\_OPTION\_STANDARD 0x00
- #define I2C\_OPTION\_NORESTART 0x04
- #define I2C\_OPTION\_FAST 0x08

### 5.123.1 Detailed Description

Constants for the SetI2COOptions function. These values are combined with a bitwise OR operation.

### See Also

[SetI2COOptions\(\)](#)

### Warning

These options require the enhanced NBC/NXC firmware

### 5.123.2 Macro Definition Documentation

#### 5.123.2.1 #define I2C\_OPTION\_FAST 0x08

Fast I2C speed

#### 5.123.2.2 #define I2C\_OPTION\_NORESTART 0x04

Use no restart on I2C read

#### 5.123.2.3 #define I2C\_OPTION\_STANDARD 0x00

Standard I2C speed

## 5.124 Display module constants

Constants that are part of the NXT firmware's Display module.

### Modules

- [Display contrast constants](#)

*Constants that are for use with the display contrast API functions.*

- [Display flags](#)

*Constants that are for use with the display flags functions.*

- [Display module IOMAP offsets](#)

*Constant offsets into the display module IOMAP structure.*

- [DisplayExecuteFunction constants](#)

*Constants that are for use with the DisplayExecuteFunction system call.*

- [Drawing option constants](#)

*Constants that are for specifying drawing options in several display module API functions.*

- [Line number constants](#)

*Line numbers for use with DrawText system function.*

- [Text line constants](#)

*Constants that are for use with getting/setting display data.*

### Macros

- #define SCREEN\_MODE\_RESTORE 0x00
- #define SCREEN\_MODE\_CLEAR 0x01
- #define DISPLAY\_HEIGHT 64
- #define DISPLAY\_WIDTH 100
- #define DISPLAY\_MENUICONS\_Y 40
- #define DISPLAY\_MENUICONS\_X\_OFFSET 7
- #define DISPLAY\_MENUICONS\_X\_DIFF 31
- #define MENUICON\_LEFT 0
- #define MENUICON\_CENTER 1
- #define MENUICON\_RIGHT 2
- #define MENUICONS 3
- #define FRAME\_SELECT 0
- #define STATUSTEXT 1
- #define MENUTEXT 2
- #define STEPLINE 3
- #define TOPLINE 4
- #define SPECIALS 5
- #define STATUSICON\_BLUETOOTH 0
- #define STATUSICON\_USB 1
- #define STATUSICON\_VM 2
- #define STATUSICON\_BATTERY 3
- #define STATUSICONS 4
- #define SCREEN\_BACKGROUND 0
- #define SCREEN\_LARGE 1
- #define SCREEN\_SMALL 2
- #define SCREENS 3

- #define BITMAP\_1 0
- #define BITMAP\_2 1
- #define BITMAP\_3 2
- #define BITMAP\_4 3
- #define BITMAPS 4
- #define STEPICON\_1 0
- #define STEPICON\_2 1
- #define STEPICON\_3 2
- #define STEPICON\_4 3
- #define STEPICON\_5 4
- #define STEPICONS 5

#### 5.124.1 Detailed Description

Constants that are part of the NXT firmware's Display module.

#### 5.124.2 Macro Definition Documentation

##### 5.124.2.1 #define BITMAP\_1 0

Bitmap 1

##### 5.124.2.2 #define BITMAP\_2 1

Bitmap 2

##### 5.124.2.3 #define BITMAP\_3 2

Bitmap 3

##### 5.124.2.4 #define BITMAP\_4 3

Bitmap 4

##### 5.124.2.5 #define BITMAPS 4

The number of bitmap bits

##### 5.124.2.6 #define DISPLAY\_HEIGHT 64

The height of the LCD screen in pixels

##### 5.124.2.7 #define DISPLAY\_MENUICONS\_X\_DIFF 31

Display menu icons x delta

##### 5.124.2.8 #define DISPLAY\_MENUICONS\_X\_OFFSET 7

Display menu icons x offset

##### 5.124.2.9 #define DISPLAY\_MENUICONS\_Y 40

Display menu icons y value

5.124.2.10 #define DISPLAY\_WIDTH 100

The width of the LCD screen in pixels

5.124.2.11 #define FRAME\_SELECT 0

Center icon select frame

5.124.2.12 #define MENUICON\_CENTER 1

Center icon

5.124.2.13 #define MENUICON\_LEFT 0

Left icon

5.124.2.14 #define MENUICON\_RIGHT 2

Right icon

5.124.2.15 #define MENUICONS 3

The number of menu icons

5.124.2.16 #define MENUTEXT 2

Center icon text

5.124.2.17 #define SCREEN\_BACKGROUND 0

Entire screen

5.124.2.18 #define SCREEN\_LARGE 1

Entire screen except status line

5.124.2.19 #define SCREEN\_MODE\_CLEAR 0x01

Clear the screen

#### See Also

[SetScreenMode\(\)](#)

5.124.2.20 #define SCREEN\_MODE\_RESTORE 0x00

Restore the screen

#### See Also

[SetScreenMode\(\)](#)

5.124.2.21 #define SCREEN\_SMALL 2

Screen between menu icons and status line

5.124.2.22 #define SCREENS 3

The number of screen bits

5.124.2.23 #define SPECIALS 5

The number of special bit values

5.124.2.24 #define STATUSICON\_BATTERY 3

Battery status icon collection

5.124.2.25 #define STATUSICON\_BLUETOOTH 0

BlueTooth status icon collection

5.124.2.26 #define STATUSICON\_USB 1

USB status icon collection

5.124.2.27 #define STATUSICON\_VM 2

VM status icon collection

5.124.2.28 #define STATUSICONS 4

The number of status icons

5.124.2.29 #define STATUSTEXT 1

Status text (BT name)

5.124.2.30 #define STEPICON\_1 0

Left most step icon

5.124.2.31 #define STEPICON\_2 1

Step icon #2

5.124.2.32 #define STEPICON\_3 2

Step icon #3

5.124.2.33 #define STEPICON\_4 3

Step icon #4

5.124.2.34 #define STEPICON\_5 4

Right most step icon

5.124.2.35 #define STEPICONS 5

The number of step icons

5.124.2.36 #define STEPLINE 3

Step collection lines

5.124.2.37 #define TOPLINE 4

Top status underline

## 5.125 DisplayExecuteFunction constants

Constants that are for use with the DisplayExecuteFunction system call.

### Macros

- #define DISPLAY\_ERASE\_ALL 0x00
- #define DISPLAY\_PIXEL 0x01
- #define DISPLAY\_HORIZONTAL\_LINE 0x02
- #define DISPLAY\_VERTICAL\_LINE 0x03
- #define DISPLAY\_CHAR 0x04
- #define DISPLAY\_ERASE\_LINE 0x05
- #define DISPLAY\_FILL\_REGION 0x06
- #define DISPLAY\_FRAME 0x07

### 5.125.1 Detailed Description

Constants that are for use with the DisplayExecuteFunction system call.

### Warning

These options require the enhanced NBC/NXC firmware

### 5.125.2 Macro Definition Documentation

#### 5.125.2.1 #define DISPLAY\_CHAR 0x04

W - draw char (actual font) (CMD,TRUE,X1,Y1,Char,x)

#### 5.125.2.2 #define DISPLAY\_ERASE\_ALL 0x00

W - erase entire screen (CMD,x,x,x,x,x)

#### 5.125.2.3 #define DISPLAY\_ERASE\_LINE 0x05

W - erase a single line (CMD,x,LINE,x,x,x)

#### 5.125.2.4 #define DISPLAY\_FILL\_REGION 0x06

W - fill screen region (CMD,TRUE/FALSE,X1,Y1,X2,Y2)

#### 5.125.2.5 #define DISPLAY\_FRAME 0x07

W - draw a frame (on/off) (CMD,TRUE/FALSE,X1,Y1,X2,Y2)

#### 5.125.2.6 #define DISPLAY\_HORIZONTAL\_LINE 0x02

W - draw horizontal line (CMD,TRUE/FALSE,X1,Y1,X2,x)

#### 5.125.2.7 #define DISPLAY\_PIXEL 0x01

W - set pixel (on/off) (CMD,TRUE/FALSE,X,Y,x,x)

## 5.125.2.8 #define DISPLAY\_VERTICAL\_LINE 0x03

W - draw vertical line (CMD,TRUE/FALSE,X1,Y1,x,Y2)

## 5.126 Drawing option constants

Constants that are for specifying drawing options in several display module API functions.

### Modules

- [Font drawing option constants](#)

*These addition drawing option constants are only for use when drawing text and numbers on the LCD using an RIC-based font.*

### Macros

- `#define DRAW_OPT_NORMAL (0x0000)`
- `#define DRAW_OPT_CLEAR_WHOLE_SCREEN (0x0001)`
- `#define DRAW_OPT_CLEAR_EXCEPT_STATUS_SCREEN (0x0002)`
- `#define DRAW_OPT_CLEAR_PIXELS (0x0004)`
- `#define DRAW_OPT_CLEAR (0x0004)`
- `#define DRAW_OPT_INVERT (0x0004)`
- `#define DRAW_OPT_LOGICAL_COPY (0x0000)`
- `#define DRAW_OPT_LOGICAL_AND (0x0008)`
- `#define DRAW_OPT_LOGICAL_OR (0x0010)`
- `#define DRAW_OPT_LOGICAL_XOR (0x0018)`
- `#define DRAW_OPT_FILL_SHAPE (0x0020)`
- `#define DRAW_OPT_CLEAR_SCREEN_MODES (0x0003)`
- `#define DRAW_OPT_LOGICAL_OPERATIONS (0x0018)`
- `#define DRAW_OPT_POLYGON_POLYLINE (0x0400)`
- `#define DRAW_OPT_CLEAR_LINE (0x0800)`
- `#define DRAW_OPT_CLEAR_EOL (0x1000)`

### 5.126.1 Detailed Description

Constants that are for specifying drawing options in several display module API functions. Bits 0 & 1 (values 0,1,2,3) control screen clearing behaviour (Not within RIC files). Bit 2 (value 4) controls the NOT operation, i.e. draw in white or invert text/graphics. Bits 3 & 4 (values 0,8,16,24) control pixel logical combinations (COPY/AND/OR/XOR). Bit 5 (value 32) controls shape filling, or overrides text/graphic bitmaps with set pixels. These may be ORed together for the full instruction (e.g., DRAW\_OPT\_NORMAL|DRAW\_OPT\_LOGICAL\_XOR) These operations are resolved into the separate, common parameters defined in 'c\_display.iom' before any drawing function is called. Note that when drawing a RIC file, the initial 'DrawingOptions' parameter supplied in the drawing instruction controls screen clearing, but nothing else. The 'CopyOptions' parameter from each instruction in the RIC file then controls graphic operations, but the screen-clearing bits are ignored.

### See Also

[TextOut\(\)](#), [NumOut\(\)](#), [PointOut\(\)](#), [LineOut\(\)](#), [CircleOut\(\)](#), [RectOut\(\)](#), [PolyOut\(\)](#), [EllipseOut\(\)](#), [FontTextOut\(\)](#), [FontNumOut\(\)](#), [GraphicOut\(\)](#), [GraphicArrayOut\(\)](#)

### Warning

These options require the enhanced NBC/NXC firmware

## 5.126.2 Macro Definition Documentation

5.126.2.1 `#define DRAW_OPT_CLEAR (0x0004)`

Clear pixels while drawing (aka draw in white)

5.126.2.2 `#define DRAW_OPT_CLEAR_EOL (0x1000)`

When drawing text, clear to the end of the line after drawing the text

5.126.2.3 `#define DRAW_OPT_CLEAR_EXCEPT_STATUS_SCREEN (0x0002)`

Clear the screen except for the status line before drawing

5.126.2.4 `#define DRAW_OPT_CLEAR_LINE (0x0800)`

When drawing text, clear the entire line before drawing the text

5.126.2.5 `#define DRAW_OPT_CLEAR_PIXELS (0x0004)`

Clear pixels while drawing (aka draw in white)

5.126.2.6 `#define DRAW_OPT_CLEAR_SCREEN_MODES (0x0003)`

Bit mask for the clear screen modes

5.126.2.7 `#define DRAW_OPT_CLEAR_WHOLE_SCREEN (0x0001)`

Clear the entire screen before drawing

5.126.2.8 `#define DRAW_OPT_FILL_SHAPE (0x0020)`

Fill the shape while drawing (rectangle, circle, ellipses, and polygon)

5.126.2.9 `#define DRAW_OPT_INVERT (0x0004)`

Invert text or graphics

5.126.2.10 `#define DRAW_OPT_LOGICAL_AND (0x0008)`

Draw pixels using a logical AND operation

5.126.2.11 `#define DRAW_OPT_LOGICAL_COPY (0x0000)`

Draw pixels using a logical copy operation

5.126.2.12 `#define DRAW_OPT_LOGICAL_OPERATIONS (0x0018)`

Bit mask for the logical drawing operations

5.126.2.13 `#define DRAW_OPT_LOGICAL_OR (0x0010)`

Draw pixels using a logical OR operation

5.126.2.14 `#define DRAW_OPT_LOGICAL_XOR (0x0018)`

Draw pixels using a logical XOR operation

5.126.2.15 #define DRAW\_OPT\_NORMAL (0x0000)

Normal drawing

5.126.2.16 #define DRAW\_OPT\_POLYGON\_POLYLINE (0x0400)

When drawing polygons, do not close (i.e., draw a polyline instead)

## 5.127 Font drawing option constants

These addition drawing option constants are only for use when drawing text and numbers on the LCD using an RIC-based font.

### Macros

- `#define DRAW_OPT_FONT_DIRECTIONS (0x01C0)`
- `#define DRAW_OPT_FONT_WRAP (0x0200)`
- `#define DRAW_OPT_FONT_DIR_L2RB (0x0000)`
- `#define DRAW_OPT_FONT_DIR_L2RT (0x0040)`
- `#define DRAW_OPT_FONT_DIR_R2LB (0x0080)`
- `#define DRAW_OPT_FONT_DIR_R2LT (0x00C0)`
- `#define DRAW_OPT_FONT_DIR_B2TL (0x0100)`
- `#define DRAW_OPT_FONT_DIR_B2TR (0x0140)`
- `#define DRAW_OPT_FONT_DIR_T2BL (0x0180)`
- `#define DRAW_OPT_FONT_DIR_T2BR (0x01C0)`

### 5.127.1 Detailed Description

These addition drawing option constants are only for use when drawing text and numbers on the LCD using an RIC-based font.

### See Also

[FontTextOut\(\)](#), [FontNumOut\(\)](#)

### Warning

These options require the enhanced NBC/NXC firmware

### 5.127.2 Macro Definition Documentation

#### 5.127.2.1 `#define DRAW_OPT_FONT_DIR_B2TL (0x0100)`

Font bottom to top left align

#### 5.127.2.2 `#define DRAW_OPT_FONT_DIR_B2TR (0x0140)`

Font bottom to top right align

#### 5.127.2.3 `#define DRAW_OPT_FONT_DIR_L2RB (0x0000)`

Font left to right bottom align

#### 5.127.2.4 `#define DRAW_OPT_FONT_DIR_L2RT (0x0040)`

Font left to right top align

#### 5.127.2.5 `#define DRAW_OPT_FONT_DIR_R2LB (0x0080)`

Font right to left bottom align

5.127.2.6 #define DRAW\_OPT\_FONT\_DIR\_R2LT (0x00C0)

Font right to left top align

5.127.2.7 #define DRAW\_OPT\_FONT\_DIR\_T2BL (0x0180)

Font top to bottom left align

5.127.2.8 #define DRAW\_OPT\_FONT\_DIR\_T2BR (0x01C0)

Font top to bottom right align

5.127.2.9 #define DRAW\_OPT\_FONT\_DIRECTIONS (0x01C0)

Bit mask for the font direction bits

5.127.2.10 #define DRAW\_OPT\_FONT\_WRAP (0x0200)

Option to have text wrap in [FontNumOut](#) and [FontTextOut](#) calls

## 5.128 Display flags

Constants that are for use with the display flags functions.

### Macros

- #define DISPLAY\_ON 0x01
- #define DISPLAY\_REFRESH 0x02
- #define DISPLAY\_POPUP 0x08
- #define DISPLAY\_REFRESH\_DISABLED 0x40
- #define DISPLAY\_BUSY 0x80

### 5.128.1 Detailed Description

Constants that are for use with the display flags functions.

### See Also

[SetDisplayFlags\(\)](#), [DisplayFlags\(\)](#)

### 5.128.2 Macro Definition Documentation

#### 5.128.2.1 #define DISPLAY\_BUSY 0x80

R - Refresh in progress

#### 5.128.2.2 #define DISPLAY\_ON 0x01

W - Display on

#### 5.128.2.3 #define DISPLAY\_POPUP 0x08

W - Use popup display memory

#### 5.128.2.4 #define DISPLAY\_REFRESH 0x02

W - Enable refresh

#### 5.128.2.5 #define DISPLAY\_REFRESH\_DISABLED 0x40

R - Refresh disabled

## 5.129 Display contrast constants

Constants that are for use with the display contrast API functions.

### Macros

- #define DISPLAY\_CONTRAST\_DEFAULT 0x5A
- #define DISPLAY\_CONTRAST\_MAX 0x7F

### 5.129.1 Detailed Description

Constants that are for use with the display contrast API functions.

### See Also

[SetDisplayContrast\(\)](#), [DisplayContrast\(\)](#)

### Warning

These options require the enhanced NBC/NXC firmware

### 5.129.2 Macro Definition Documentation

#### 5.129.2.1 #define DISPLAY\_CONTRAST\_DEFAULT 0x5A

Default display contrast value

#### 5.129.2.2 #define DISPLAY\_CONTRAST\_MAX 0x7F

Maximum display contrast value

## 5.130 Text line constants

Constants that are for use with getting/setting display data.

### Macros

- `#define TEXTLINE_1 0`
- `#define TEXTLINE_2 1`
- `#define TEXTLINE_3 2`
- `#define TEXTLINE_4 3`
- `#define TEXTLINE_5 4`
- `#define TEXTLINE_6 5`
- `#define TEXTLINE_7 6`
- `#define TEXTLINE_8 7`
- `#define TEXTLINES 8`

### 5.130.1 Detailed Description

Constants that are for use with getting/setting display data.

#### See Also

[SetDisplayNormal\(\)](#), [GetDisplayNormal\(\)](#), [SetDisplayPopup\(\)](#), [GetDisplayPopup\(\)](#)

### 5.130.2 Macro Definition Documentation

#### 5.130.2.1 `#define TEXTLINE_1 0`

Text line 1

#### 5.130.2.2 `#define TEXTLINE_2 1`

Text line 2

#### 5.130.2.3 `#define TEXTLINE_3 2`

Text line 3

#### 5.130.2.4 `#define TEXTLINE_4 3`

Text line 4

#### 5.130.2.5 `#define TEXTLINE_5 4`

Text line 5

#### 5.130.2.6 `#define TEXTLINE_6 5`

Text line 6

#### 5.130.2.7 `#define TEXTLINE_7 6`

Text line 7

**5.130.2.8 #define TEXTLINE\_8 7**

Text line 8

**5.130.2.9 #define TEXTLINES 8**

The number of text lines on the LCD

## 5.131 Display module IOMAP offsets

Constant offsets into the display module IOMAP structure.

### Macros

- #define `DisplayOffsetPFunc` 0
- #define `DisplayOffsetEraseMask` 4
- #define `DisplayOffsetUpdateMask` 8
- #define `DisplayOffsetPFont` 12
- #define `DisplayOffsetPTextLines(p)`  $((p)*4)+16$
- #define `DisplayOffsetPStatusText` 48
- #define `DisplayOffsetPStatusIcons` 52
- #define `DisplayOffsetPScreens(p)`  $((p)*4)+56$
- #define `DisplayOffsetPBitmaps(p)`  $((p)*4)+68$
- #define `DisplayOffsetPMenuText` 84
- #define `DisplayOffsetPMenuIcons(p)`  $((p)*4)+88$
- #define `DisplayOffsetPStepIcons` 100
- #define `DisplayOffsetDisplay` 104
- #define `DisplayOffsetStatusIcons(p)`  $((p)+108)$
- #define `DisplayOffsetStepIcons(p)`  $((p)+112)$
- #define `DisplayOffsetFlags` 117
- #define `DisplayOffsetTextLinesCenterFlags` 118
- #define `DisplayOffsetNormal(l, w)`  $((l)*100)+(w)+119$
- #define `DisplayOffsetPopup(l, w)`  $((l)*100)+(w)+919$
- #define `DisplayOffsetContrast` 1719

### 5.131.1 Detailed Description

Constant offsets into the display module IOMAP structure.

### 5.131.2 Macro Definition Documentation

#### 5.131.2.1 #define `DisplayOffsetContrast` 1719

Adjust the display contrast with this field (NBC/NXC)

#### 5.131.2.2 #define `DisplayOffsetDisplay` 104

Display content copied to physical display every 17 mS

#### 5.131.2.3 #define `DisplayOffsetEraseMask` 4

Section erase mask (executed first)

#### 5.131.2.4 #define `DisplayOffsetFlags` 117

Update flags enumerated above

#### 5.131.2.5 #define `DisplayOffsetNormal( l, w )` $((l)*100)+(w)+119$

Raw display memory for normal screen

5.131.2.6 #define DisplayOffsetPBitmaps( *p* ) (((*p*)\*4)+68)

Pointer to free bitmap files

5.131.2.7 #define DisplayOffsetPFont 12

Pointer to font file

5.131.2.8 #define DisplayOffsetPFunc 0

Simple draw entry

5.131.2.9 #define DisplayOffsetPMenuIcons( *p* ) (((*p*)\*4)+88)

Pointer to menu icon images (NULL == none)

5.131.2.10 #define DisplayOffsetPMenuText 84

Pointer to menu icon text (NULL == none)

5.131.2.11 #define DisplayOffsetPopup( *l*, *w* ) (((*l*)\*100)+(*w*)+919)

Raw display memory for popup screen

5.131.2.12 #define DisplayOffsetPScreens( *p* ) (((*p*)\*4)+56)

Pointer to screen bitmap file

5.131.2.13 #define DisplayOffsetPStatusIcons 52

Pointer to status icon collection file

5.131.2.14 #define DisplayOffsetPStatusText 48

Pointer to status text string

5.131.2.15 #define DisplayOffsetPStepIcons 100

Pointer to step icon collection file

5.131.2.16 #define DisplayOffsetPTextLines( *p* ) (((*p*)\*4)+16)

Pointer to text strings

5.131.2.17 #define DisplayOffsetStatusIcons( *p* ) ((*p*)+108)

Index in status icon collection file (index = 0 -> none)

5.131.2.18 #define DisplayOffsetStepIcons( *p* ) ((*p*)+112)

Index in step icon collection file (index = 0 -> none)

5.131.2.19 #define DisplayOffsetTextLinesCenterFlags 118

Mask to center TextLines

5.131.2.20 #define DisplayOffsetUpdateMask 8

Section update mask (executed next)

## 5.132 Comm module constants

Constants that are part of the NXT firmware's Comm module.

### Modules

- [Bluetooth State constants](#)  
*Constants related to the bluetooth state.*
- [Bluetooth hardware status constants](#)  
*Constants related to the bluetooth hardware status.*
- [Bluetooth state status constants](#)  
*Constants related to the bluetooth state status.*
- [Comm module IOMAP offsets](#)  
*Constant offsets into the Comm module IOMAP structure.*
- [Comm module interface function constants](#)  
*Constants for all the Comm module interface functions executable via `SysCommExecuteFunction`.*
- [Comm module status code constants](#)  
*Constants for Comm module status codes.*
- [Data mode constants](#)  
*Constants related to the bluetooth and hi-speed data modes.*
- [Device status constants](#)  
*Constants referring to `DeviceStatus` within `DeviceTable`.*
- [Hi-speed port constants](#)  
*Constants related to the hi-speed port.*
- [Joystick message constants](#)  
*Constants for reading joystick information.*
- [Mailbox constants](#)  
*Mailbox number constants should be used to avoid confusing NXT-G users.*
- [Miscellaneous Comm module constants](#)  
*Miscellaneous constants related to the Comm module.*
- [Remote connection constants](#)  
*Constants for specifying remote connection slots.*

### 5.132.1 Detailed Description

Constants that are part of the NXT firmware's Comm module.

## 5.133 Miscellaneous Comm module constants

Miscellaneous constants related to the Comm module.

### Macros

- `#define SIZE_OF_USBBUF 64`
- `#define USB_PROTOCOL_OVERHEAD 2`
- `#define SIZE_OF_USBDATA 62`
- `#define SIZE_OF_HSBUF 128`
- `#define SIZE_OF_BTBUF 128`
- `#define BT_CMD_BYTE 1`
- `#define SIZE_OF_BT_DEVICE_TABLE 30`
- `#define SIZE_OF_BT_CONNECT_TABLE 4`
- `#define SIZE_OF_BT_NAME 16`
- `#define SIZE_OF_BRICK_NAME 8`
- `#define SIZE_OF_CLASS_OF_DEVICE 4`
- `#define SIZE_OF_BT_PINCODE 16`
- `#define SIZE_OF_BDADDR 7`
- `#define MAX_BT_MSG_SIZE 60000`
- `#define BT_DEFAULT_INQUIRY_MAX 0`
- `#define BT_DEFAULT_INQUIRY_TIMEOUT_LO 15`

### 5.133.1 Detailed Description

Miscellaneous constants related to the Comm module.

### 5.133.2 Macro Definition Documentation

#### 5.133.2.1 `#define BT_CMD_BYTE 1`

Size of Bluetooth command

#### 5.133.2.2 `#define BT_DEFAULT_INQUIRY_MAX 0`

Bluetooth default inquiry Max (0 == unlimited)

#### 5.133.2.3 `#define BT_DEFAULT_INQUIRY_TIMEOUT_LO 15`

Bluetooth inquiry timeout (15\*1.28 sec = 19.2 sec)

#### 5.133.2.4 `#define MAX_BT_MSG_SIZE 60000`

Max Bluetooth Message Size

#### 5.133.2.5 `#define SIZE_OF_BDADDR 7`

Size of Bluetooth Address

#### 5.133.2.6 `#define SIZE_OF_BRICK_NAME 8`

Size of NXT Brick name

5.133.2.7 #define SIZE\_OF\_BT\_CONNECT\_TABLE 4

Size of Bluetooth connection table – Index 0 is always incoming connection

5.133.2.8 #define SIZE\_OF\_BT\_DEVICE\_TABLE 30

Size of Bluetooth device table

5.133.2.9 #define SIZE\_OF\_BT\_NAME 16

Size of Bluetooth name

5.133.2.10 #define SIZE\_OF\_BT\_PINCODE 16

Size of Bluetooth PIN

5.133.2.11 #define SIZE\_OF\_BTBUF 128

Size of Bluetooth buffer

5.133.2.12 #define SIZE\_OF\_CLASS\_OF\_DEVICE 4

Size of class of device

5.133.2.13 #define SIZE\_OF\_HSBUF 128

Size of High Speed Port 4 buffer

5.133.2.14 #define SIZE\_OF\_USBBUF 64

Size of USB Buffer in bytes

5.133.2.15 #define SIZE\_OF\_USBDATA 62

Size of USB Buffer available for data

5.133.2.16 #define USB\_PROTOCOL\_OVERHEAD 2

Size of USB Overhead in bytes – Command type byte + Command

## 5.134 Joystick message constants

Constants for reading joystick information.

### Macros

- #define JOY\_BTN\_01 0x00000001
- #define JOY\_BTN\_02 0x00000002
- #define JOY\_BTN\_03 0x00000004
- #define JOY\_BTN\_04 0x00000008
- #define JOY\_BTN\_05 0x00000010
- #define JOY\_BTN\_06 0x00000020
- #define JOY\_BTN\_07 0x00000040
- #define JOY\_BTN\_08 0x00000080
- #define JOY\_BTN\_09 0x00000100
- #define JOY\_BTN\_10 0x00000200
- #define JOY\_BTN\_11 0x00000400
- #define JOY\_BTN\_12 0x00000800
- #define JOY\_BTN\_13 0x00001000
- #define JOY\_BTN\_14 0x00002000
- #define JOY\_BTN\_15 0x00004000
- #define JOY\_BTN\_16 0x00008000
- #define JOY\_BTN\_17 0x00010000
- #define JOY\_BTN\_18 0x00020000
- #define JOY\_BTN\_19 0x00040000
- #define JOY\_BTN\_20 0x00080000
- #define JOY\_BTN\_21 0x00100000
- #define JOY\_BTN\_22 0x00200000
- #define JOY\_BTN\_23 0x00400000
- #define JOY\_BTN\_24 0x00800000
- #define JOY\_BTN\_25 0x01000000
- #define JOY\_BTN\_26 0x02000000
- #define JOY\_BTN\_27 0x04000000
- #define JOY\_BTN\_28 0x08000000
- #define JOY\_BTN\_29 0x10000000
- #define JOY\_BTN\_30 0x20000000
- #define JOY\_BTN\_31 0x40000000
- #define JOY\_BTN\_32 0x80000000
- #define JOY\_POV\_FORWARD 0
- #define JOY\_POV\_TOPRIGHT 4500
- #define JOY\_POV\_RIGHT 9000
- #define JOY\_POV\_BOTRIGHT 13500
- #define JOY\_POV\_BACKWARD 18000
- #define JOY\_POV\_BOTLEFT 22500
- #define JOY\_POV\_LEFT 27000
- #define JOY\_POV\_TOPLEFT 31500
- #define JOY\_POV\_CENTERED 65535

### 5.134.1 Detailed Description

Constants for reading joystick information.

## 5.134.2 Macro Definition Documentation

5.134.2.1 `#define JOY_BTN_01 0x00000001`

Joystick button 1

5.134.2.2 `#define JOY_BTN_02 0x00000002`

Joystick button 2

5.134.2.3 `#define JOY_BTN_03 0x00000004`

Joystick button 3

5.134.2.4 `#define JOY_BTN_04 0x00000008`

Joystick button 4

5.134.2.5 `#define JOY_BTN_05 0x00000010`

Joystick button 5

5.134.2.6 `#define JOY_BTN_06 0x00000020`

Joystick button 6

5.134.2.7 `#define JOY_BTN_07 0x00000040`

Joystick button 7

5.134.2.8 `#define JOY_BTN_08 0x00000080`

Joystick button 8

5.134.2.9 `#define JOY_BTN_09 0x00000100`

Joystick button 9

5.134.2.10 `#define JOY_BTN_10 0x00000200`

Joystick button 10

5.134.2.11 `#define JOY_BTN_11 0x00000400`

Joystick button 11

5.134.2.12 `#define JOY_BTN_12 0x00000800`

Joystick button 12

5.134.2.13 `#define JOY_BTN_13 0x00001000`

Joystick button 13

5.134.2.14 `#define JOY_BTN_14 0x00002000`

Joystick button 14

5.134.2.15 #define JOY\_BTN\_15 0x00004000

Joystick button 15

5.134.2.16 #define JOY\_BTN\_16 0x00008000

Joystick button 16

5.134.2.17 #define JOY\_BTN\_17 0x00010000

Joystick button 17

5.134.2.18 #define JOY\_BTN\_18 0x00020000

Joystick button 18

5.134.2.19 #define JOY\_BTN\_19 0x00040000

Joystick button 19

5.134.2.20 #define JOY\_BTN\_20 0x00080000

Joystick button 20

5.134.2.21 #define JOY\_BTN\_21 0x00100000

Joystick button 21

5.134.2.22 #define JOY\_BTN\_22 0x00200000

Joystick button 22

5.134.2.23 #define JOY\_BTN\_23 0x00400000

Joystick button 23

5.134.2.24 #define JOY\_BTN\_24 0x00800000

Joystick button 24

5.134.2.25 #define JOY\_BTN\_25 0x01000000

Joystick button 25

5.134.2.26 #define JOY\_BTN\_26 0x02000000

Joystick button 26

5.134.2.27 #define JOY\_BTN\_27 0x04000000

Joystick button 27

5.134.2.28 #define JOY\_BTN\_28 0x08000000

Joystick button 28

5.134.2.29 #define JOY\_BTN\_29 0x10000000

Joystick button 29

5.134.2.30 #define JOY\_BTN\_30 0x20000000

Joystick button 30

5.134.2.31 #define JOY\_BTN\_31 0x40000000

Joystick button 31

5.134.2.32 #define JOY\_BTN\_32 0x80000000

Joystick button 32

5.134.2.33 #define JOY\_POV\_BACKWARD 18000

Joystick POV backward

5.134.2.34 #define JOY\_POV\_BOTLEFT 22500

Joystick POV bottom left

5.134.2.35 #define JOY\_POV\_BOTRIGHT 13500

Joystick POV bottom right

5.134.2.36 #define JOY\_POV\_CENTERED 65535

Joystick POV centered

5.134.2.37 #define JOY\_POV\_FORWARD 0

Joystick POV forward

5.134.2.38 #define JOY\_POV\_LEFT 27000

Joystick POV left

5.134.2.39 #define JOY\_POV\_RIGHT 9000

Joystick POV right

5.134.2.40 #define JOY\_POV\_TOPLEFT 31500

Joystick POV top left

5.134.2.41 #define JOY\_POV\_TOPRIGHT 4500

Joystick POV top right

## 5.135 Bluetooth State constants

Constants related to the bluetooth state.

### Macros

- #define BT\_ARM\_OFF 0
- #define BT\_ARM\_CMD\_MODE 1
- #define BT\_ARM\_DATA\_MODE 2

#### 5.135.1 Detailed Description

Constants related to the bluetooth state.

#### 5.135.2 Macro Definition Documentation

##### 5.135.2.1 #define BT\_ARM\_CMD\_MODE 1

BtState constant bluetooth command mode

##### 5.135.2.2 #define BT\_ARM\_DATA\_MODE 2

BtState constant bluetooth data mode

##### 5.135.2.3 #define BT\_ARM\_OFF 0

BtState constant bluetooth off

## 5.136 Data mode constants

Constants related to the bluetooth and hi-speed data modes.

### Macros

- #define DATA\_MODE\_NXT 0x00
- #define DATA\_MODE\_GPS 0x01
- #define DATA\_MODE\_RAW 0x02
- #define DATA\_MODE\_MASK 0x07
- #define DATA\_MODE\_UPDATE 0x08

### 5.136.1 Detailed Description

Constants related to the bluetooth and hi-speed data modes.

#### Warning

These options require the enhanced NBC/NXC firmware

### 5.136.2 Macro Definition Documentation

#### 5.136.2.1 #define DATA\_MODE\_GPS 0x01

Use GPS data mode

#### 5.136.2.2 #define DATA\_MODE\_MASK 0x07

A mask for the data mode bits.

#### 5.136.2.3 #define DATA\_MODE\_NXT 0x00

Use NXT data mode

#### 5.136.2.4 #define DATA\_MODE\_RAW 0x02

Use RAW data mode

#### 5.136.2.5 #define DATA\_MODE\_UPDATE 0x08

Indicates that the data mode has been changed.

## 5.137 Bluetooth state status constants

Constants related to the bluetooth state status.

### Macros

- `#define BT_BRICK_VISIBILITY 0x01`
- `#define BT_BRICK_PORT_OPEN 0x02`
- `#define BT_CONNECTION_0_ENABLE 0x10`
- `#define BT_CONNECTION_1_ENABLE 0x20`
- `#define BT_CONNECTION_2_ENABLE 0x40`
- `#define BT_CONNECTION_3_ENABLE 0x80`

### 5.137.1 Detailed Description

Constants related to the bluetooth state status.

### 5.137.2 Macro Definition Documentation

#### 5.137.2.1 `#define BT_BRICK_PORT_OPEN 0x02`

BtStateStatus port open bit

#### 5.137.2.2 `#define BT_BRICK_VISIBILITY 0x01`

BtStateStatus brick visibility bit

#### 5.137.2.3 `#define BT_CONNECTION_0_ENABLE 0x10`

BtStateStatus connection 0 enable/disable bit

#### 5.137.2.4 `#define BT_CONNECTION_1_ENABLE 0x20`

BtStateStatus connection 1 enable/disable bit

#### 5.137.2.5 `#define BT_CONNECTION_2_ENABLE 0x40`

BtStateStatus connection 2 enable/disable bit

#### 5.137.2.6 `#define BT_CONNECTION_3_ENABLE 0x80`

BtStateStatus connection 3 enable/disable bit

## 5.138 Remote connection constants

Constants for specifying remote connection slots.

### Macros

- `#define CONN_BT0 0x0`
- `#define CONN_BT1 0x1`
- `#define CONN_BT2 0x2`
- `#define CONN_BT3 0x3`
- `#define CONN_HS4 0x4`
- `#define CONN_HS_ALL 0x4`
- `#define CONN_HS_1 0x5`
- `#define CONN_HS_2 0x6`
- `#define CONN_HS_3 0x7`
- `#define CONN_HS_4 0x8`
- `#define CONN_HS_5 0x9`
- `#define CONN_HS_6 0xa`
- `#define CONN_HS_7 0xb`
- `#define CONN_HS_8 0xc`

### 5.138.1 Detailed Description

Constants for specifying remote connection slots.

### 5.138.2 Macro Definition Documentation

#### 5.138.2.1 `#define CONN_BT0 0x0`

Bluetooth connection 0

#### 5.138.2.2 `#define CONN_BT1 0x1`

Bluetooth connection 1

#### 5.138.2.3 `#define CONN_BT2 0x2`

Bluetooth connection 2

#### 5.138.2.4 `#define CONN_BT3 0x3`

Bluetooth connection 3

#### 5.138.2.5 `#define CONN_HS4 0x4`

RS485 (hi-speed) connection (port 4, all devices)

#### 5.138.2.6 `#define CONN_HS_1 0x5`

RS485 (hi-speed) connection (port 4, device address 1)

5.138.2.7 #define CONN\_HS\_2 0x6

RS485 (hi-speed) connection (port 4, device address 2)

5.138.2.8 #define CONN\_HS\_3 0x7

RS485 (hi-speed) connection (port 4, device address 3)

5.138.2.9 #define CONN\_HS\_4 0x8

RS485 (hi-speed) connection (port 4, device address 4)

5.138.2.10 #define CONN\_HS\_5 0x9

RS485 (hi-speed) connection (port 4, device address 5)

5.138.2.11 #define CONN\_HS\_6 0xa

RS485 (hi-speed) connection (port 4, device address 6)

5.138.2.12 #define CONN\_HS\_7 0xb

RS485 (hi-speed) connection (port 4, device address 7)

5.138.2.13 #define CONN\_HS\_8 0xc

RS485 (hi-speed) connection (port 4, device address 8)

5.138.2.14 #define CONN\_HS\_ALL 0x4

RS485 (hi-speed) connection (port 4, all devices)

## 5.139 Bluetooth hardware status constants

Constants related to the bluetooth hardware status.

### Macros

- #define BT\_ENABLE 0x00
- #define BT\_DISABLE 0x01

### 5.139.1 Detailed Description

Constants related to the bluetooth hardware status.

### 5.139.2 Macro Definition Documentation

#### 5.139.2.1 #define BT\_DISABLE 0x01

BtHwStatus bluetooth disable

#### 5.139.2.2 #define BT\_ENABLE 0x00

BtHwStatus bluetooth enable

## 5.140 Hi-speed port constants

Constants related to the hi-speed port.

### Modules

- [Hi-speed port SysCommHSControl constants](#)

*Constants for use with the SysCommHSControl API function.*

- [Hi-speed port UART mode constants](#)

*Constants referring to HsMode UART configuration settings.*

- [Hi-speed port address constants](#)

*Constants that are used to specify the Hi-speed (RS-485) port device address.*

- [Hi-speed port baud rate constants](#)

*Constants for configuring the hi-speed port baud rate (HsSpeed).*

- [Hi-speed port flags constants](#)

*Constants related to the hi-speed port flags.*

- [Hi-speed port state constants](#)

*Constants related to the hi-speed port state.*

### 5.140.1 Detailed Description

Constants related to the hi-speed port.

## 5.141 Hi-speed port flags constants

Constants related to the hi-speed port flags.

### Macros

- #define HS\_UPDATE 1

#### 5.141.1 Detailed Description

Constants related to the hi-speed port flags.

#### 5.141.2 Macro Definition Documentation

##### 5.141.2.1 #define HS\_UPDATE 1

HsFlags high speed update required

## 5.142 Hi-speed port state constants

Constants related to the hi-speed port state.

### Macros

- `#define HS_INITIALISE 1`
- `#define HS_INIT_RECEIVER 2`
- `#define HS_SEND_DATA 3`
- `#define HS_DISABLE 4`
- `#define HS_ENABLE 5`
- `#define HS_DEFAULT 6`
- `#define HS_BYTES_REMAINING 16`

### 5.142.1 Detailed Description

Constants related to the hi-speed port state.

### 5.142.2 Macro Definition Documentation

#### 5.142.2.1 `#define HS_BYTES_REMAINING 16`

HsState bytes remaining to be sent

#### 5.142.2.2 `#define HS_DEFAULT 6`

HsState default

#### 5.142.2.3 `#define HS_DISABLE 4`

HsState disable

#### 5.142.2.4 `#define HS_ENABLE 5`

HsState enable

#### 5.142.2.5 `#define HS_INIT_RECEIVER 2`

HsState initialize receiver

#### 5.142.2.6 `#define HS_INITIALISE 1`

HsState initialize

#### 5.142.2.7 `#define HS_SEND_DATA 3`

HsState send data

## 5.143 Hi-speed port SysCommHSControl constants

Constants for use with the SysCommHSControl API function.

### Macros

- #define HS\_CTRL\_INIT 0
- #define HS\_CTRL\_UART 1
- #define HS\_CTRL\_EXIT 2

### 5.143.1 Detailed Description

Constants for use with the SysCommHSControl API function.

### See Also

SysCommHSControl()

### Warning

These options require the enhanced NBC/NXC firmware

### 5.143.2 Macro Definition Documentation

#### 5.143.2.1 #define HS\_CTRL\_EXIT 2

Ddisable the high speed port

#### 5.143.2.2 #define HS\_CTRL\_INIT 0

Enable the high speed port

#### 5.143.2.3 #define HS\_CTRL\_UART 1

Setup the high speed port UART configuration

## 5.144 Hi-speed port baud rate constants

Constants for configuring the hi-speed port baud rate (HsSpeed).

### Macros

- #define HS\_BAUD\_1200 0
- #define HS\_BAUD\_2400 1
- #define HS\_BAUD\_3600 2
- #define HS\_BAUD\_4800 3
- #define HS\_BAUD\_7200 4
- #define HS\_BAUD\_9600 5
- #define HS\_BAUD\_14400 6
- #define HS\_BAUD\_19200 7
- #define HS\_BAUD\_28800 8
- #define HS\_BAUD\_38400 9
- #define HS\_BAUD\_57600 10
- #define HS\_BAUD\_76800 11
- #define HS\_BAUD\_115200 12
- #define HS\_BAUD\_230400 13
- #define HS\_BAUD\_460800 14
- #define HS\_BAUD\_921600 15
- #define HS\_BAUD\_DEFAULT 15

### 5.144.1 Detailed Description

Constants for configuring the hi-speed port baud rate (HsSpeed).

#### Warning

These options require the enhanced NBC/NXC firmware

### 5.144.2 Macro Definition Documentation

#### 5.144.2.1 #define HS\_BAUD\_115200 12

HsSpeed 115200 Baud

#### 5.144.2.2 #define HS\_BAUD\_1200 0

HsSpeed 1200 Baud

#### 5.144.2.3 #define HS\_BAUD\_14400 6

HsSpeed 14400 Baud

#### 5.144.2.4 #define HS\_BAUD\_19200 7

HsSpeed 19200 Baud

#### 5.144.2.5 #define HS\_BAUD\_230400 13

HsSpeed 230400 Baud

5.144.2.6 #define HS\_BAUD\_2400 1

HsSpeed 2400 Baud

5.144.2.7 #define HS\_BAUD\_28800 8

HsSpeed 28800 Baud

5.144.2.8 #define HS\_BAUD\_3600 2

HsSpeed 3600 Baud

5.144.2.9 #define HS\_BAUD\_38400 9

HsSpeed 38400 Baud

5.144.2.10 #define HS\_BAUD\_460800 14

HsSpeed 460800 Baud

5.144.2.11 #define HS\_BAUD\_4800 3

HsSpeed 4800 Baud

5.144.2.12 #define HS\_BAUD\_57600 10

HsSpeed 57600 Baud

5.144.2.13 #define HS\_BAUD\_7200 4

HsSpeed 7200 Baud

5.144.2.14 #define HS\_BAUD\_76800 11

HsSpeed 76800 Baud

5.144.2.15 #define HS\_BAUD\_921600 15

HsSpeed 921600 Baud

5.144.2.16 #define HS\_BAUD\_9600 5

HsSpeed 9600 Baud

5.144.2.17 #define HS\_BAUD\_DEFAULT 15

HsSpeed default Baud (921600)

## 5.145 Hi-speed port UART mode constants

Constants referring to HsMode UART configuration settings.

### Modules

- [Hi-speed port combined UART constants](#)

*Constants that combine data bits, parity, and stop bits into a single value.*

- [Hi-speed port data bits constants](#)

*Constants referring to HsMode (number of data bits)*

- [Hi-speed port parity constants](#)

*Constants referring to HsMode (parity)*

- [Hi-speed port stop bits constants](#)

*Constants referring to HsMode (number of stop bits)*

### Macros

- `#define HS_MODE_UART_RS485 0x0`
- `#define HS_MODE_UART_RS232 0x1`
- `#define HS_MODE_MASK 0x3EC0`
- `#define HS_UART_MASK 0x000F`
- `#define HS_MODE_DEFAULT HS_MODE_8N1`

### 5.145.1 Detailed Description

Constants referring to HsMode UART configuration settings.

#### Warning

These options require the enhanced NBC/NXC firmware

### 5.145.2 Macro Definition Documentation

#### 5.145.2.1 `#define HS_MODE_DEFAULT HS_MODE_8N1`

HsMode default mode (8 data bits, no parity, 1 stop bit)

#### 5.145.2.2 `#define HS_MODE_MASK 0x3EC0`

HsMode mode mask

#### 5.145.2.3 `#define HS_MODE_UART_RS232 0x1`

HsMode UART in normal or RS232 mode

#### 5.145.2.4 `#define HS_MODE_UART_RS485 0x0`

HsMode UART in default or RS485 mode

#### 5.145.2.5 `#define HS_UART_MASK 0x000F`

HsMode UART mask

## 5.146 Hi-speed port data bits constants

Constants referring to HsMode (number of data bits)

### Macros

- #define HS\_MODE\_5\_DATA 0x0000
- #define HS\_MODE\_6\_DATA 0x0040
- #define HS\_MODE\_7\_DATA 0x0080
- #define HS\_MODE\_8\_DATA 0x00C0

### 5.146.1 Detailed Description

Constants referring to HsMode (number of data bits)

#### Warning

These options require the enhanced NBC/NXC firmware

### 5.146.2 Macro Definition Documentation

#### 5.146.2.1 #define HS\_MODE\_5\_DATA 0x0000

HsMode 5 data bits

#### 5.146.2.2 #define HS\_MODE\_6\_DATA 0x0040

HsMode 6 data bits

#### 5.146.2.3 #define HS\_MODE\_7\_DATA 0x0080

HsMode 7 data bits

#### 5.146.2.4 #define HS\_MODE\_8\_DATA 0x00C0

HsMode 8 data bits

## 5.147 Hi-speed port stop bits constants

Constants referring to HsMode (number of stop bits)

### Macros

- #define HS\_MODE\_10\_STOP 0x0000
- #define HS\_MODE\_15\_STOP 0x1000
- #define HS\_MODE\_20\_STOP 0x2000

### 5.147.1 Detailed Description

Constants referring to HsMode (number of stop bits)

### Warning

These options require the enhanced NBC/NXC firmware

### 5.147.2 Macro Definition Documentation

#### 5.147.2.1 #define HS\_MODE\_10\_STOP 0x0000

HsMode 1 stop bit

#### 5.147.2.2 #define HS\_MODE\_15\_STOP 0x1000

HsMode 1.5 stop bits

#### 5.147.2.3 #define HS\_MODE\_20\_STOP 0x2000

HsMode 2 stop bits

## 5.148 Hi-speed port parity constants

Constants referring to HsMode (parity)

### Macros

- #define HS\_MODE\_E\_PARITY 0x0000
- #define HS\_MODE\_O\_PARITY 0x0200
- #define HS\_MODE\_S\_PARITY 0x0400
- #define HS\_MODE\_M\_PARITY 0x0600
- #define HS\_MODE\_N\_PARITY 0x0800

### 5.148.1 Detailed Description

Constants referring to HsMode (parity)

#### Warning

These options require the enhanced NBC/NXC firmware

### 5.148.2 Macro Definition Documentation

#### 5.148.2.1 #define HS\_MODE\_E\_PARITY 0x0000

HsMode Even parity

#### 5.148.2.2 #define HS\_MODE\_M\_PARITY 0x0600

HsMode Mark parity

#### 5.148.2.3 #define HS\_MODE\_N\_PARITY 0x0800

HsMode No parity

#### 5.148.2.4 #define HS\_MODE\_O\_PARITY 0x0200

HsMode Odd parity

#### 5.148.2.5 #define HS\_MODE\_S\_PARITY 0x0400

HsMode Space parity

## 5.149 Hi-speed port combined UART constants

Constants that combine data bits, parity, and stop bits into a single value.

### Macros

- `#define HS_MODE_8N1 (HS_MODE_8_DATA|HS_MODE_N_PARITY|HS_MODE_10_STOP)`
- `#define HS_MODE_7E1 (HS_MODE_7_DATA|HS_MODE_E_PARITY|HS_MODE_10_STOP)`

### 5.149.1 Detailed Description

Constants that combine data bits, parity, and stop bits into a single value.

### Warning

These options require the enhanced NBC/NXC firmware

### 5.149.2 Macro Definition Documentation

#### 5.149.2.1 `#define HS_MODE_7E1 (HS_MODE_7_DATA|HS_MODE_E_PARITY|HS_MODE_10_STOP)`

HsMode 7 data bits, even parity, 1 stop bit

#### 5.149.2.2 `#define HS_MODE_8N1 (HS_MODE_8_DATA|HS_MODE_N_PARITY|HS_MODE_10_STOP)`

HsMode 8 data bits, no parity, 1 stop bit

## 5.150 Hi-speed port address constants

Constants that are used to specify the Hi-speed (RS-485) port device address.

### Macros

- #define HS\_ADDRESS\_ALL 0
- #define HS\_ADDRESS\_1 1
- #define HS\_ADDRESS\_2 2
- #define HS\_ADDRESS\_3 3
- #define HS\_ADDRESS\_4 4
- #define HS\_ADDRESS\_5 5
- #define HS\_ADDRESS\_6 6
- #define HS\_ADDRESS\_7 7
- #define HS\_ADDRESS\_8 8

### 5.150.1 Detailed Description

Constants that are used to specify the Hi-speed (RS-485) port device address.

#### Warning

These options require the enhanced NBC/NXC firmware

### 5.150.2 Macro Definition Documentation

#### 5.150.2.1 #define HS\_ADDRESS\_1 1

HsAddress device address 1

#### 5.150.2.2 #define HS\_ADDRESS\_2 2

HsAddress device address 2

#### 5.150.2.3 #define HS\_ADDRESS\_3 3

HsAddress device address 3

#### 5.150.2.4 #define HS\_ADDRESS\_4 4

HsAddress device address 4

#### 5.150.2.5 #define HS\_ADDRESS\_5 5

HsAddress device address 5

#### 5.150.2.6 #define HS\_ADDRESS\_6 6

HsAddress device address 6

#### 5.150.2.7 #define HS\_ADDRESS\_7 7

HsAddress device address 7

5.150.2.8 #define HS\_ADDRESS\_8 8

HsAddress device address 8

5.150.2.9 #define HS\_ADDRESS\_ALL 0

HsAddress all devices

## 5.151 Device status constants

Constants referring to DeviceStatus within DeviceTable.

### Macros

- #define BT\_DEVICE\_EMPTY 0x00
- #define BT\_DEVICE\_UNKNOWN 0x01
- #define BT\_DEVICE\_KNOWN 0x02
- #define BT\_DEVICE\_NAME 0x40
- #define BT\_DEVICE\_AWAY 0x80

### 5.151.1 Detailed Description

Constants referring to DeviceStatus within DeviceTable.

### 5.151.2 Macro Definition Documentation

#### 5.151.2.1 #define BT\_DEVICE\_AWAY 0x80

Bluetooth device away

#### 5.151.2.2 #define BT\_DEVICE\_EMPTY 0x00

Bluetooth device table empty

#### 5.151.2.3 #define BT\_DEVICE\_KNOWN 0x02

Bluetooth device known

#### 5.151.2.4 #define BT\_DEVICE\_NAME 0x40

Bluetooth device name

#### 5.151.2.5 #define BT\_DEVICE\_UNKNOWN 0x01

Bluetooth device unknown

## 5.152 Comm module interface function constants

Constants for all the Comm module interface functions executable via SysCommExecuteFunction.

### Macros

- #define INTF\_SENDFILE 0
- #define INTF\_SEARCH 1
- #define INTF\_STOPSEARCH 2
- #define INTF\_CONNECT 3
- #define INTF\_DISCONNECT 4
- #define INTF\_DISCONNECTALL 5
- #define INTF\_REMOVEDevice 6
- #define INTF\_VISIBILITY 7
- #define INTF\_SETCMDMODE 8
- #define INTF\_OPENSTREAM 9
- #define INTF\_SENDDATA 10
- #define INTF\_FACTORYRESET 11
- #define INTF\_BTON 12
- #define INTF\_BTOFF 13
- #define INTF\_SETBTNAME 14
- #define INTF\_EXTREAD 15
- #define INTF\_PINREQ 16
- #define INTF\_CONNECTREQ 17
- #define INTF\_CONNECTBYNAME 18

### 5.152.1 Detailed Description

Constants for all the Comm module interface functions executable via SysCommExecuteFunction.

### See Also

SysCommExecuteFunction()

### 5.152.2 Macro Definition Documentation

#### 5.152.2.1 #define INTF\_BTOFF 13

Turn off the bluetooth radio

#### 5.152.2.2 #define INTF\_BTON 12

Turn on the bluetooth radio

#### 5.152.2.3 #define INTF\_CONNECT 3

Connect to one of the known devices

#### 5.152.2.4 #define INTF\_CONNECTBYNAME 18

Connect to a bluetooth device by name

5.152.2.5 #define INTF\_CONNECTREQ 17

Connection request from another device

5.152.2.6 #define INTF\_DISCONNECT 4

Disconnect from one of the connected devices

5.152.2.7 #define INTF\_DISCONNECTALL 5

Disconnect all devices

5.152.2.8 #define INTF\_EXTREAD 15

External read request

5.152.2.9 #define INTF\_FACTORYRESET 11

Reset bluetooth settings to factory values

5.152.2.10 #define INTF\_OPENSTREAM 9

Open a bluetooth stream

5.152.2.11 #define INTF\_PINREQ 16

Bluetooth PIN request

5.152.2.12 #define INTF\_REMOVEDEVICE 6

Remove a device from the known devices table

5.152.2.13 #define INTF\_SEARCH 1

Search for bluetooth devices

5.152.2.14 #define INTF\_SENDDATA 10

Send data over a bluetooth connection

5.152.2.15 #define INTF\_SENDFILE 0

Send a file via bluetooth to another device

5.152.2.16 #define INTF\_SETBTNAME 14

Set the bluetooth name

5.152.2.17 #define INTF\_SETCMDMODE 8

Set bluetooth into command mode

5.152.2.18 #define INTF\_STOPSEARCH 2

Stop searching for bluetooth devices

## 5.152.2.19 #define INTF\_VISIBILITY 7

Set the bluetooth visibility on or off

## 5.153 Comm module status code constants

Constants for Comm module status codes.

### Macros

- #define LR\_SUCCESS 0x50
- #define LR\_COULD\_NOT\_SAVE 0x51
- #define LR\_STORE\_IS\_FULL 0x52
- #define LR\_ENTRY\_REMOVED 0x53
- #define LR\_UNKNOWN\_ADDR 0x54
- #define USB\_CMD\_READY 0x01
- #define BT\_CMD\_READY 0x02
- #define HS\_CMD\_READY 0x04

### 5.153.1 Detailed Description

Constants for Comm module status codes.

### 5.153.2 Macro Definition Documentation

#### 5.153.2.1 #define BT\_CMD\_READY 0x02

A constant representing bluetooth direct command

#### 5.153.2.2 #define HS\_CMD\_READY 0x04

A constant representing high speed direct command

#### 5.153.2.3 #define LR\_COULD\_NOT\_SAVE 0x51

Bluetooth list result could not save

#### 5.153.2.4 #define LR\_ENTRY\_REMOVED 0x53

Bluetooth list result entry removed

#### 5.153.2.5 #define LR\_STORE\_IS\_FULL 0x52

Bluetooth list result store is full

#### 5.153.2.6 #define LR\_SUCCESS 0x50

Bluetooth list result success

#### 5.153.2.7 #define LR\_UNKNOWN\_ADDR 0x54

Bluetooth list result unknown address

#### 5.153.2.8 #define USB\_CMD\_READY 0x01

A constant representing usb direct command

## 5.154 Comm module IOMAP offsets

Constant offsets into the Comm module IOMAP structure.

### Macros

- #define CommOffsetPFunc 0
- #define CommOffsetPFuncTwo 4
- #define CommOffsetBtDeviceTableName(p) (((p)\*31)+8)
- #define CommOffsetBtDeviceTableClassOfDevice(p) (((p)\*31)+24)
- #define CommOffsetBtDeviceTableBdAddr(p) (((p)\*31)+28)
- #define CommOffsetBtDeviceTableDeviceStatus(p) (((p)\*31)+35)
- #define CommOffsetBtConnectTableName(p) (((p)\*47)+938)
- #define CommOffsetBtConnectTableClassOfDevice(p) (((p)\*47)+954)
- #define CommOffsetBtConnectTablePinCode(p) (((p)\*47)+958)
- #define CommOffsetBtConnectTableBdAddr(p) (((p)\*47)+974)
- #define CommOffsetBtConnectTableHandleNr(p) (((p)\*47)+981)
- #define CommOffsetBtConnectTableStreamStatus(p) (((p)\*47)+982)
- #define CommOffsetBtConnectTableLinkQuality(p) (((p)\*47)+983)
- #define CommOffsetBrickDataName 1126
- #define CommOffsetBrickDataBluecore\_Version 1142
- #define CommOffsetBrickDataBdAddr 1144
- #define CommOffsetBrickDataBtStateStatus 1151
- #define CommOffsetBrickDataBtHwStatus 1152
- #define CommOffsetBrickDataTimeOutValue 1153
- #define CommOffsetBtInBufBuf 1157
- #define CommOffsetBtInBufInPtr 1285
- #define CommOffsetBtInBufOutPtr 1286
- #define CommOffsetBtOutBufBuf 1289
- #define CommOffsetBtOutBufInPtr 1417
- #define CommOffsetBtOutBufOutPtr 1418
- #define CommOffsetHsInBufBuf 1421
- #define CommOffsetHsInBufInPtr 1549
- #define CommOffsetHsInBufOutPtr 1550
- #define CommOffsetHsOutBufBuf 1553
- #define CommOffsetHsOutBufInPtr 1681
- #define CommOffsetHsOutBufOutPtr 1682
- #define CommOffsetUsbInBufBuf 1685
- #define CommOffsetUsbInBufInPtr 1749
- #define CommOffsetUsbInBufOutPtr 1750
- #define CommOffsetUsbOutBufBuf 1753
- #define CommOffsetUsbOutBufInPtr 1817
- #define CommOffsetUsbOutBufOutPtr 1818
- #define CommOffsetUsbPollBufBuf 1821
- #define CommOffsetUsbPollBufInPtr 1885
- #define CommOffsetUsbPollBufOutPtr 1886
- #define CommOffsetBtDeviceCnt 1889
- #define CommOffsetBtDeviceNameCnt 1890
- #define CommOffsetHsFlags 1891
- #define CommOffsetHsSpeed 1892

- #define CommOffsetHsState 1893
- #define CommOffsetUsbState 1894
- #define CommOffsetHsAddress 1895
- #define CommOffsetHsMode 1896
- #define CommOffsetBtDataMode 1898
- #define CommOffsetHsDataMode 1899

#### 5.154.1 Detailed Description

Constant offsets into the Comm module IOMAP structure.

#### 5.154.2 Macro Definition Documentation

##### 5.154.2.1 #define CommOffsetBrickDataBdAddr 1144

Offset to Bluetooth address (7 bytes)

##### 5.154.2.2 #define CommOffsetBrickDataBluecoreVersion 1142

Offset to Bluecore version (2 bytes)

##### 5.154.2.3 #define CommOffsetBrickDataBtHwStatus 1152

Offset to BtHwStatus (1 byte)

##### 5.154.2.4 #define CommOffsetBrickDataBtStateStatus 1151

Offset to BtStateStatus (1 byte)

##### 5.154.2.5 #define CommOffsetBrickDataName 1126

Offset to brick name (16 bytes)

##### 5.154.2.6 #define CommOffsetBrickDataTimeOutValue 1153

Offset to data timeout value (1 byte)

##### 5.154.2.7 #define CommOffsetBtConnectTableBdAddr( p ) (((p)\*47)+974)

Offset to Bluetooth connect table address (7 bytes)

##### 5.154.2.8 #define CommOffsetBtConnectTableClassOfDevice( p ) (((p)\*47)+954)

Offset to Bluetooth connect table device class (4 bytes)

##### 5.154.2.9 #define CommOffsetBtConnectTableHandleNr( p ) (((p)\*47)+981)

Offset to Bluetooth connect table handle (1 byte)

##### 5.154.2.10 #define CommOffsetBtConnectTableLinkQuality( p ) (((p)\*47)+983)

Offset to Bluetooth connect table link quality (1 byte)

5.154.2.11 #define CommOffsetBtConnectTableName( *p* ) (((*p*)\*47)+938)

Offset to Bluetooth connect table name (16 bytes)

5.154.2.12 #define CommOffsetBtConnectTablePinCode( *p* ) (((*p*)\*47)+958)

Offset to Bluetooth connect table pin code (16 bytes)

5.154.2.13 #define CommOffsetBtConnectTableStreamStatus( *p* ) (((*p*)\*47)+982)

Offset to Bluetooth connect table stream status (1 byte)

5.154.2.14 #define CommOffsetBtDataMode 1898

Offset to Bluetooth data mode (1 byte)

5.154.2.15 #define CommOffsetBtDeviceCnt 1889

Offset to Bluetooth device count (1 byte)

5.154.2.16 #define CommOffsetBtDeviceNameCnt 1890

Offset to Bluetooth device name count (1 byte)

5.154.2.17 #define CommOffsetBtDeviceTableBdAddr( *p* ) (((*p*)\*31)+28)

Offset to Bluetooth device table address (7 bytes)

5.154.2.18 #define CommOffsetBtDeviceTableClassOfDevice( *p* ) (((*p*)\*31)+24)

Offset to Bluetooth device table device class (4 bytes)

5.154.2.19 #define CommOffsetBtDeviceTableDeviceStatus( *p* ) (((*p*)\*31)+35)

Offset to Bluetooth device table status (1 byte)

5.154.2.20 #define CommOffsetBtDeviceTableName( *p* ) (((*p*)\*31)+8)

Offset to BT device table name (16 bytes)

5.154.2.21 #define CommOffsetBtInBufBuf 1157

Offset to Bluetooth input buffer data (128 bytes)

5.154.2.22 #define CommOffsetBtInBufInPtr 1285

Offset to Bluetooth input buffer front pointer (1 byte)

5.154.2.23 #define CommOffsetBtInBufOutPtr 1286

Offset to Bluetooth output buffer back pointer (1 byte)

5.154.2.24 #define CommOffsetBtOutBufBuf 1289

Offset to Bluetooth output buffer offset data (128 bytes)

5.154.2.25 #define CommOffsetBtOutBufInPtr 1417

Offset to Bluetooth output buffer front pointer (1 byte)

5.154.2.26 #define CommOffsetBtOutBufOutPtr 1418

Offset to Bluetooth output buffer back pointer (1 byte)

5.154.2.27 #define CommOffsetHsAddress 1895

Offset to High Speed address (1 byte)

5.154.2.28 #define CommOffsetHsDataMode 1899

Offset to High Speed data mode (1 byte)

5.154.2.29 #define CommOffsetHsFlags 1891

Offset to High Speed flags (1 byte)

5.154.2.30 #define CommOffsetHsInBufBuf 1421

Offset to High Speed input buffer data (128 bytes)

5.154.2.31 #define CommOffsetHsInBufInPtr 1549

Offset to High Speed input buffer front pointer (1 byte)

5.154.2.32 #define CommOffsetHsInBufOutPtr 1550

Offset to High Speed input buffer back pointer (1 byte)

5.154.2.33 #define CommOffsetHsMode 1896

Offset to High Speed mode (2 bytes)

5.154.2.34 #define CommOffsetHsOutBufBuf 1553

Offset to High Speed output buffer data (128 bytes)

5.154.2.35 #define CommOffsetHsOutBufInPtr 1681

Offset to High Speed output buffer front pointer (1 byte)

5.154.2.36 #define CommOffsetHsOutBufOutPtr 1682

Offset to High Speed output buffer back pointer (1 byte)

5.154.2.37 #define CommOffsetHsSpeed 1892

Offset to High Speed speed (1 byte)

5.154.2.38 #define CommOffsetHsState 1893

Offset to High Speed state (1 byte)

5.154.2.39 #define CommOffsetPFunc 0

Offset to the Comm module first function pointer (4 bytes)

5.154.2.40 #define CommOffsetPFuncTwo 4

Offset to the Comm module second function pointer (4 bytes)

5.154.2.41 #define CommOffsetUsbInBufBuf 1685

Offset to Usb input buffer data (64 bytes)

5.154.2.42 #define CommOffsetUsbInBufInPtr 1749

Offset to Usb input buffer front pointer (1 byte)

5.154.2.43 #define CommOffsetUsbInBufOutPtr 1750

Offset to Usb input buffer back pointer (1 byte)

5.154.2.44 #define CommOffsetUsbOutBufBuf 1753

Offset to Usb output buffer data (64 bytes)

5.154.2.45 #define CommOffsetUsbOutBufInPtr 1817

Offset to Usb output buffer front pointer (1 byte)

5.154.2.46 #define CommOffsetUsbOutBufOutPtr 1818

Offset to Usb output buffer back pointer (1 byte)

5.154.2.47 #define CommOffsetUsbPollBufBuf 1821

Offset to Usb Poll buffer data (64 bytes)

5.154.2.48 #define CommOffsetUsbPollBufInPtr 1885

Offset to Usb Poll buffer front pointer (1 byte)

5.154.2.49 #define CommOffsetUsbPollBufOutPtr 1886

Offset to Usb Poll buffer back pointer (1 byte)

5.154.2.50 #define CommOffsetUsbState 1894

Offset to Usb State (1 byte)

## 5.155 RCX constants

Constants that are for use with devices that communicate with the RCX or Scout programmable bricks via IR such as the HiTechnic IRLink or the MindSensors nRLink.

### Modules

- [RCX IR remote constants](#)

*Constants for use when simulating RCX IR remote messages.*

- [RCX and Scout opcode constants](#)

*Constants for use when specifying RCX and Scout opcodes.*

- [RCX and Scout sound constants](#)

*Constants for use when playing standard RCX and Scout sounds.*

- [RCX and Scout source constants](#)

*Constants for use when specifying RCX and Scout sources.*

- [RCX output constants](#)

*Constants for use when choosing RCX outputs.*

- [RCX output direction constants](#)

*Constants for use when configuring RCX output direction.*

- [RCX output mode constants](#)

*Constants for use when configuring RCX output mode.*

- [RCX output power constants](#)

*Constants for use when configuring RCX output power.*

- [Scout constants](#)

*Constants for use when controlling the Scout brick.*

### 5.155.1 Detailed Description

Constants that are for use with devices that communicate with the RCX or Scout programmable bricks via IR such as the HiTechnic IRLink or the MindSensors nRLink.

## 5.156 RCX output constants

Constants for use when choosing RCX outputs.

### Macros

- `#define RCX_OUT_A 0x01`
- `#define RCX_OUT_B 0x02`
- `#define RCX_OUT_C 0x04`
- `#define RCX_OUT_AB 0x03`
- `#define RCX_OUT_AC 0x05`
- `#define RCX_OUT_BC 0x06`
- `#define RCX_OUT_ABC 0x07`

### 5.156.1 Detailed Description

Constants for use when choosing RCX outputs.

### 5.156.2 Macro Definition Documentation

#### 5.156.2.1 `#define RCX_OUT_A 0x01`

RCX Output A

#### 5.156.2.2 `#define RCX_OUT_AB 0x03`

RCX Outputs A and B

#### 5.156.2.3 `#define RCX_OUT_ABC 0x07`

RCX Outputs A, B, and C

#### 5.156.2.4 `#define RCX_OUT_AC 0x05`

RCX Outputs A and C

#### 5.156.2.5 `#define RCX_OUT_B 0x02`

RCX Output B

#### 5.156.2.6 `#define RCX_OUT_BC 0x06`

RCX Outputs B and C

#### 5.156.2.7 `#define RCX_OUT_C 0x04`

RCX Output C

## 5.157 RCX output mode constants

Constants for use when configuring RCX output mode.

### Macros

- `#define RCX_OUT_FLOAT 0`
- `#define RCX_OUT_OFF 0x40`
- `#define RCX_OUT_ON 0x80`

### 5.157.1 Detailed Description

Constants for use when configuring RCX output mode.

### 5.157.2 Macro Definition Documentation

#### 5.157.2.1 `#define RCX_OUT_FLOAT 0`

Set RCX output to float

#### 5.157.2.2 `#define RCX_OUT_OFF 0x40`

Set RCX output to off

#### 5.157.2.3 `#define RCX_OUT_ON 0x80`

Set RCX output to on

## 5.158 RCX output direction constants

Constants for use when configuring RCX output direction.

### Macros

- #define RCX\_OUT\_REV 0
- #define RCX\_OUT\_TOGGLE 0x40
- #define RCX\_OUT\_FWD 0x80

#### 5.158.1 Detailed Description

Constants for use when configuring RCX output direction.

#### 5.158.2 Macro Definition Documentation

##### 5.158.2.1 #define RCX\_OUT\_FWD 0x80

Set RCX output direction to forward

##### 5.158.2.2 #define RCX\_OUT\_REV 0

Set RCX output direction to reverse

##### 5.158.2.3 #define RCX\_OUT\_TOGGLE 0x40

Set RCX output direction to toggle

## 5.159 RCX output power constants

Constants for use when configuring RCX output power.

### Macros

- `#define RCX_OUT_LOW 0`
- `#define RCX_OUT_HALF 3`
- `#define RCX_OUT_FULL 7`

#### 5.159.1 Detailed Description

Constants for use when configuring RCX output power.

#### 5.159.2 Macro Definition Documentation

##### 5.159.2.1 `#define RCX_OUT_FULL 7`

Set RCX output power level to full

##### 5.159.2.2 `#define RCX_OUT_HALF 3`

Set RCX output power level to half

##### 5.159.2.3 `#define RCX_OUT_LOW 0`

Set RCX output power level to low

## 5.160 RCX IR remote constants

Constants for use when simulating RCX IR remote messages.

### Macros

- #define RCX\_RemoteKeysReleased 0x0000
- #define RCX\_RemotePBMessag1 0x0100
- #define RCX\_RemotePBMessag2 0x0200
- #define RCX\_RemotePBMessag3 0x0400
- #define RCX\_RemoteOutAForward 0x0800
- #define RCX\_RemoteOutBForward 0x1000
- #define RCX\_RemoteOutCForward 0x2000
- #define RCX\_RemoteOutABackward 0x4000
- #define RCX\_RemoteOutBBackward 0x8000
- #define RCX\_RemoteOutCBackward 0x0001
- #define RCX\_RemoteSelProgram1 0x0002
- #define RCX\_RemoteSelProgram2 0x0004
- #define RCX\_RemoteSelProgram3 0x0008
- #define RCX\_RemoteSelProgram4 0x0010
- #define RCX\_RemoteSelProgram5 0x0020
- #define RCX\_RemoteStopOutOff 0x0040
- #define RCX\_RemotePlayASound 0x0080

### 5.160.1 Detailed Description

Constants for use when simulating RCX IR remote messages.

### 5.160.2 Macro Definition Documentation

#### 5.160.2.1 #define RCX\_RemoteKeysReleased 0x0000

All remote keys have been released

#### 5.160.2.2 #define RCX\_RemoteOutABackward 0x4000

Set output A backward

#### 5.160.2.3 #define RCX\_RemoteOutAForward 0x0800

Set output A forward

#### 5.160.2.4 #define RCX\_RemoteOutBBackward 0x8000

Set output B backward

#### 5.160.2.5 #define RCX\_RemoteOutBForward 0x1000

Set output B forward

#### 5.160.2.6 #define RCX\_RemoteOutCBackward 0x0001

Set output C backward

5.160.2.7 #define RCX\_RemoteOutCForward 0x2000

Set output C forward

5.160.2.8 #define RCX\_RemotePBMessagel 0x0100

Send PB message 1

5.160.2.9 #define RCX\_RemotePBMessage2 0x0200

Send PB message 2

5.160.2.10 #define RCX\_RemotePBMessage3 0x0400

Send PB message 3

5.160.2.11 #define RCX\_RemotePlayASound 0x0080

Play a sound

5.160.2.12 #define RCX\_RemoteSelProgram1 0x0002

Select program 1

5.160.2.13 #define RCX\_RemoteSelProgram2 0x0004

Select program 2

5.160.2.14 #define RCX\_RemoteSelProgram3 0x0008

Select program 3

5.160.2.15 #define RCX\_RemoteSelProgram4 0x0010

Select program 4

5.160.2.16 #define RCX\_RemoteSelProgram5 0x0020

Select program 5

5.160.2.17 #define RCX\_RemoteStopOutOff 0x0040

Stop and turn off outputs

## 5.161 RCX and Scout sound constants

Constants for use when playing standard RCX and Scout sounds.

### Macros

- #define SOUND\_CLICK 0
- #define SOUND\_DOUBLE\_BEEP 1
- #define SOUND\_DOWN 2
- #define SOUND\_UP 3
- #define SOUND\_LOW\_BEEP 4
- #define SOUND\_FAST\_UP 5

### 5.161.1 Detailed Description

Constants for use when playing standard RCX and Scout sounds.

### 5.161.2 Macro Definition Documentation

#### 5.161.2.1 #define SOUND\_CLICK 0

Play the standard key click sound

#### 5.161.2.2 #define SOUND\_DOUBLE\_BEEP 1

Play the standard double beep sound

#### 5.161.2.3 #define SOUND\_DOWN 2

Play the standard sweep down sound

#### 5.161.2.4 #define SOUND\_FAST\_UP 5

Play the standard fast up sound

#### 5.161.2.5 #define SOUND\_LOW\_BEEP 4

Play the standard low beep sound

#### 5.161.2.6 #define SOUND\_UP 3

Play the standard sweep up sound

## 5.162 Scout constants

Constants for use when controlling the Scout brick.

### Modules

- [Scout light constants](#)

*Constants for use when controlling the Scout light settings.*

- [Scout light rule constants](#)

*Constants for use when setting the scout light rule.*

- [Scout mode constants](#)

*Constants for use when setting the scout mode.*

- [Scout motion rule constants](#)

*Constants for use when setting the scout motion rule.*

- [Scout sound constants](#)

*Constants for use when playing standard Scout sounds.*

- [Scout sound set constants](#)

*Constants for use when choosing standard Scout sound sets.*

- [Scout special effect constants](#)

*Constants for use when setting the scout special effect.*

- [Scout touch rule constants](#)

*Constants for use when setting the scout touch rule.*

- [Scout transmit rule constants](#)

*Constants for use when setting the scout transmit rule.*

### 5.162.1 Detailed Description

Constants for use when controlling the Scout brick.

## 5.163 Scout light constants

Constants for use when controlling the Scout light settings.

### Macros

- #define SCOUT\_LIGHT\_ON 0x80
- #define SCOUT\_LIGHT\_OFF 0

### 5.163.1 Detailed Description

Constants for use when controlling the Scout light settings.

### 5.163.2 Macro Definition Documentation

#### 5.163.2.1 #define SCOUT\_LIGHT\_OFF 0

Turn off the scout light

#### 5.163.2.2 #define SCOUT\_LIGHT\_ON 0x80

Turn on the scout light

## 5.164 Scout sound constants

Constants for use when playing standard Scout sounds.

### Macros

- #define SCOUT\_SOUND\_REMOTE 6
- #define SCOUT\_SOUND\_ENTERSA 7
- #define SCOUT\_SOUND\_KEYERROR 8
- #define SCOUT\_SOUND\_NONE 9
- #define SCOUT\_SOUND\_TOUCH1\_PRES 10
- #define SCOUT\_SOUND\_TOUCH1\_REL 11
- #define SCOUT\_SOUND\_TOUCH2\_PRES 12
- #define SCOUT\_SOUND\_TOUCH2\_REL 13
- #define SCOUT\_SOUND\_ENTER\_BRIGHT 14
- #define SCOUT\_SOUND\_ENTER\_NORMAL 15
- #define SCOUT\_SOUND\_ENTER\_DARK 16
- #define SCOUT\_SOUND\_1\_BLINK 17
- #define SCOUT\_SOUND\_2\_BLINK 18
- #define SCOUT\_SOUND\_COUNTER1 19
- #define SCOUT\_SOUND\_COUNTER2 20
- #define SCOUT\_SOUND\_TIMER1 21
- #define SCOUT\_SOUND\_TIMER2 22
- #define SCOUT\_SOUND\_TIMER3 23
- #define SCOUT\_SOUND\_MAIL\_RECEIVED 24
- #define SCOUT\_SOUND\_SPECIAL1 25
- #define SCOUT\_SOUND\_SPECIAL2 26
- #define SCOUT\_SOUND\_SPECIAL3 27

### 5.164.1 Detailed Description

Constants for use when playing standard Scout sounds.

### 5.164.2 Macro Definition Documentation

#### 5.164.2.1 #define SCOUT\_SOUND\_1\_BLINK 17

Play the Scout 1 blink sound

#### 5.164.2.2 #define SCOUT\_SOUND\_2\_BLINK 18

Play the Scout 2 blink sound

#### 5.164.2.3 #define SCOUT\_SOUND\_COUNTER1 19

Play the Scout counter 1 sound

#### 5.164.2.4 #define SCOUT\_SOUND\_COUNTER2 20

Play the Scout counter 2 sound

5.164.2.5 #define SCOUT\_SOUND\_ENTER\_BRIGHT 14

Play the Scout enter bright sound

5.164.2.6 #define SCOUT\_SOUND\_ENTER\_DARK 16

Play the Scout enter dark sound

5.164.2.7 #define SCOUT\_SOUND\_ENTER\_NORMAL 15

Play the Scout enter normal sound

5.164.2.8 #define SCOUT\_SOUND\_ENTERSA 7

Play the Scout enter standalone sound

5.164.2.9 #define SCOUT\_SOUND\_KEYERROR 8

Play the Scout key error sound

5.164.2.10 #define SCOUT\_SOUND\_MAIL\_RECEIVED 24

Play the Scout mail received sound

5.164.2.11 #define SCOUT\_SOUND\_NONE 9

Play the Scout none sound

5.164.2.12 #define SCOUT\_SOUND\_REMOTE 6

Play the Scout remote sound

5.164.2.13 #define SCOUT\_SOUND\_SPECIAL1 25

Play the Scout special 1 sound

5.164.2.14 #define SCOUT\_SOUND\_SPECIAL2 26

Play the Scout special 2 sound

5.164.2.15 #define SCOUT\_SOUND\_SPECIAL3 27

Play the Scout special 3 sound

5.164.2.16 #define SCOUT\_SOUND\_TIMER1 21

Play the Scout timer 1 sound

5.164.2.17 #define SCOUT\_SOUND\_TIMER2 22

Play the Scout timer 2 sound

5.164.2.18 #define SCOUT\_SOUND\_TIMER3 23

Play the Scout timer 3 sound

5.164.2.19 #define SCOUT\_SOUND\_TOUCH1\_PRES 10

Play the Scout touch 1 pressed sound

5.164.2.20 #define SCOUT\_SOUND\_TOUCH1\_REL 11

Play the Scout touch 1 released sound

5.164.2.21 #define SCOUT\_SOUND\_TOUCH2\_PRES 12

Play the Scout touch 2 pressed sound

5.164.2.22 #define SCOUT\_SOUND\_TOUCH2\_REL 13

Play the Scout touch 2 released sound

## 5.165 Scout sound set constants

Constants for use when choosing standard Scout sound sets.

### Macros

- #define SCOUT\_SNDSET\_NONE 0
- #define SCOUT\_SNDSET\_BASIC 1
- #define SCOUT\_SNDSET\_BUG 2
- #define SCOUT\_SNDSET\_ALARM 3
- #define SCOUT\_SNDSET\_RANDOM 4
- #define SCOUT\_SNDSET\_SCIENCE 5

### 5.165.1 Detailed Description

Constants for use when choosing standard Scout sound sets.

### 5.165.2 Macro Definition Documentation

#### 5.165.2.1 #define SCOUT\_SNDSET\_ALARM 3

Set sound set to alarm

#### 5.165.2.2 #define SCOUT\_SNDSET\_BASIC 1

Set sound set to basic

#### 5.165.2.3 #define SCOUT\_SNDSET\_BUG 2

Set sound set to bug

#### 5.165.2.4 #define SCOUT\_SNDSET\_NONE 0

Set sound set to none

#### 5.165.2.5 #define SCOUT\_SNDSET\_RANDOM 4

Set sound set to random

#### 5.165.2.6 #define SCOUT\_SNDSET\_SCIENCE 5

Set sound set to science

## 5.166 Scout mode constants

Constants for use when setting the scout mode.

### Macros

- #define SCOUT\_MODE\_STANDALONE 0
- #define SCOUT\_MODE\_POWER 1

### 5.166.1 Detailed Description

Constants for use when setting the scout mode.

### 5.166.2 Macro Definition Documentation

#### 5.166.2.1 #define SCOUT\_MODE\_POWER 1

Enter power mode

#### 5.166.2.2 #define SCOUT\_MODE\_STANDALONE 0

Enter stand alone mode

## 5.167 Scout motion rule constants

Constants for use when setting the scout motion rule.

### Macros

- #define SCOUT\_MR\_NO\_MOTION 0
- #define SCOUT\_MR\_FORWARD 1
- #define SCOUT\_MR\_ZIGZAG 2
- #define SCOUT\_MR\_CIRCLE\_RIGHT 3
- #define SCOUT\_MR\_CIRCLE\_LEFT 4
- #define SCOUT\_MR\_LOOP\_A 5
- #define SCOUT\_MR\_LOOP\_B 6
- #define SCOUT\_MR\_LOOP\_AB 7

### 5.167.1 Detailed Description

Constants for use when setting the scout motion rule.

### 5.167.2 Macro Definition Documentation

#### 5.167.2.1 #define SCOUT\_MR\_CIRCLE\_LEFT 4

Motion rule circle left

#### 5.167.2.2 #define SCOUT\_MR\_CIRCLE\_RIGHT 3

Motion rule circle right

#### 5.167.2.3 #define SCOUT\_MR\_FORWARD 1

Motion rule forward

#### 5.167.2.4 #define SCOUT\_MR\_LOOP\_A 5

Motion rule loop A

#### 5.167.2.5 #define SCOUT\_MR\_LOOP\_AB 7

Motion rule loop A then B

#### 5.167.2.6 #define SCOUT\_MR\_LOOP\_B 6

Motion rule loop B

#### 5.167.2.7 #define SCOUT\_MR\_NO\_MOTION 0

Motion rule none

#### 5.167.2.8 #define SCOUT\_MR\_ZIGZAG 2

Motion rule zigzag

## 5.168 Scout touch rule constants

Constants for use when setting the scout touch rule.

### Macros

- #define SCOUT\_TR\_IGNORE 0
- #define SCOUT\_TR\_REVERSE 1
- #define SCOUT\_TR\_AVOID 2
- #define SCOUT\_TR\_WAIT\_FOR 3
- #define SCOUT\_TR\_OFF\_WHEN 4

### 5.168.1 Detailed Description

Constants for use when setting the scout touch rule.

### 5.168.2 Macro Definition Documentation

#### 5.168.2.1 #define SCOUT\_TR\_AVOID 2

Touch rule avoid

#### 5.168.2.2 #define SCOUT\_TR\_IGNORE 0

Touch rule ignore

#### 5.168.2.3 #define SCOUT\_TR\_OFF\_WHEN 4

Touch rule off when

#### 5.168.2.4 #define SCOUT\_TR\_REVERSE 1

Touch rule reverse

#### 5.168.2.5 #define SCOUT\_TR\_WAIT\_FOR 3

Touch rule wait for

## 5.169 Scout light rule constants

Constants for use when setting the scout light rule.

### Macros

- #define SCOUT\_LR\_IGNORE 0
- #define SCOUT\_LR\_SEEK\_LIGHT 1
- #define SCOUT\_LR\_SEEK\_DARK 2
- #define SCOUT\_LR\_AVOID 3
- #define SCOUT\_LR\_WAIT\_FOR 4
- #define SCOUT\_LR\_OFF\_WHEN 5

### 5.169.1 Detailed Description

Constants for use when setting the scout light rule.

### 5.169.2 Macro Definition Documentation

#### 5.169.2.1 #define SCOUT\_LR\_AVOID 3

Light rule avoid

#### 5.169.2.2 #define SCOUT\_LR\_IGNORE 0

Light rule ignore

#### 5.169.2.3 #define SCOUT\_LR\_OFF\_WHEN 5

Light rule off when

#### 5.169.2.4 #define SCOUT\_LR\_SEEK\_DARK 2

Light rule seek dark

#### 5.169.2.5 #define SCOUT\_LR\_SEEK\_LIGHT 1

Light rule seek light

#### 5.169.2.6 #define SCOUT\_LR\_WAIT\_FOR 4

Light rule wait for

## 5.170 Scout transmit rule constants

Constants for use when setting the scout transmit rule.

### Macros

- #define SCOUT\_TGS\_SHORT 0
- #define SCOUT\_TGS\_MEDIUM 1
- #define SCOUT\_TGS\_LONG 2

### 5.170.1 Detailed Description

Constants for use when setting the scout transmit rule.

### 5.170.2 Macro Definition Documentation

#### 5.170.2.1 #define SCOUT\_TGS\_LONG 2

Transmit level long

#### 5.170.2.2 #define SCOUT\_TGS\_MEDIUM 1

Transmit level medium

#### 5.170.2.3 #define SCOUT\_TGS\_SHORT 0

Transmit level short

## 5.171 Scout special effect constants

Constants for use when setting the scout special effect.

### Macros

- #define SCOUT\_FXR\_NONE 0
- #define SCOUT\_FXR\_BUG 1
- #define SCOUT\_FXR\_ALARM 2
- #define SCOUT\_FXR\_RANDOM 3
- #define SCOUT\_FXR\_SCIENCE 4

### 5.171.1 Detailed Description

Constants for use when setting the scout special effect.

### 5.171.2 Macro Definition Documentation

#### 5.171.2.1 #define SCOUT\_FXR\_ALARM 2

Alarm special effects

#### 5.171.2.2 #define SCOUT\_FXR\_BUG 1

Bug special effects

#### 5.171.2.3 #define SCOUT\_FXR\_NONE 0

No special effects

#### 5.171.2.4 #define SCOUT\_FXR\_RANDOM 3

Random special effects

#### 5.171.2.5 #define SCOUT\_FXR\_SCIENCE 4

Science special effects

## 5.172 RCX and Scout source constants

Constants for use when specifying RCX and Scout sources.

### Macros

- #define RCX\_VariableSrc 0
- #define RCX\_TimerSrc 1
- #define RCX\_ConstantSrc 2
- #define RCX\_OutputStatusSrc 3
- #define RCX\_RandomSrc 4
- #define RCX\_ProgramSlotSrc 8
- #define RCX\_InputValueSrc 9
- #define RCX\_InputTypeSrc 10
- #define RCX\_InputModeSrc 11
- #define RCX\_InputRawSrc 12
- #define RCX\_InputBooleanSrc 13
- #define RCX\_WatchSrc 14
- #define RCX\_MessageSrc 15
- #define RCX\_GlobalMotorStatusSrc 17
- #define RCX\_ScoutRulesSrc 18
- #define RCX\_ScoutLightParamsSrc 19
- #define RCX\_ScoutTimerLimitSrc 20
- #define RCX\_CounterSrc 21
- #define RCX\_ScoutCounterLimitSrc 22
- #define RCX\_TaskEventsSrc 23
- #define RCX\_ScoutEventFBSrc 24
- #define RCX\_EventStateSrc 25
- #define RCX\_TenMSTimerSrc 26
- #define RCX\_ClickCounterSrc 27
- #define RCX\_UpperThresholdSrc 28
- #define RCX\_LowerThresholdSrc 29
- #define RCX\_HysteresisSrc 30
- #define RCX\_DurationSrc 31
- #define RCX\_UARTSetupSrc 33
- #define RCX\_BatteryLevelSrc 34
- #define RCX\_FirmwareVersionSrc 35
- #define RCX\_IndirectVarSrc 36
- #define RCX\_DatalogSrcIndirectSrc 37
- #define RCX\_DatalogSrcDirectSrc 38
- #define RCX\_DatalogValueIndirectSrc 39
- #define RCX\_DatalogValueDirectSrc 40
- #define RCX\_DatalogRawIndirectSrc 41
- #define RCX\_DatalogRawDirectSrc 42

### 5.172.1 Detailed Description

Constants for use when specifying RCX and Scout sources.

### 5.172.2 Macro Definition Documentation

#### 5.172.2.1 #define RCX\_BatteryLevelSrc 34

The RCX battery level source

#### 5.172.2.2 #define RCX\_ClickCounterSrc 27

The RCX event click counter source

#### 5.172.2.3 #define RCX\_ConstantSrc 2

The RCX constant value source

#### 5.172.2.4 #define RCX\_CounterSrc 21

The RCX counter source

#### 5.172.2.5 #define RCX\_DatalogRawDirectSrc 42

The RCX direct datalog raw source

#### 5.172.2.6 #define RCX\_DatalogRawIndirectSrc 41

The RCX indirect datalog raw source

#### 5.172.2.7 #define RCX\_DatalogSrcDirectSrc 38

The RCX direct datalog source source

#### 5.172.2.8 #define RCX\_DatalogSrcIndirectSrc 37

The RCX indirect datalog source source

#### 5.172.2.9 #define RCX\_DatalogValueDirectSrc 40

The RCX direct datalog value source

#### 5.172.2.10 #define RCX\_DatalogValueIndirectSrc 39

The RCX indirect datalog value source

#### 5.172.2.11 #define RCX\_DurationSrc 31

The RCX event duration source

#### 5.172.2.12 #define RCX\_EventStateSrc 25

The RCX event static source

#### 5.172.2.13 #define RCX\_FirmwareVersionSrc 35

The RCX firmware version source

#### 5.172.2.14 #define RCX\_GlobalMotorStatusSrc 17

The RCX global motor status source

5.172.2.15 #define RCX\_HysteresisSrc 30

The RCX event hysteresis source

5.172.2.16 #define RCX\_IndirectVarSrc 36

The RCX indirect variable source

5.172.2.17 #define RCX\_InputBooleanSrc 13

The RCX input boolean source

5.172.2.18 #define RCX\_InputModeSrc 11

The RCX input mode source

5.172.2.19 #define RCX\_InputRawSrc 12

The RCX input raw source

5.172.2.20 #define RCX\_InputTypeSrc 10

The RCX input type source

5.172.2.21 #define RCX\_InputValueSrc 9

The RCX input value source

5.172.2.22 #define RCX\_LowerThresholdSrc 29

The RCX event lower threshold source

5.172.2.23 #define RCX\_MessageSrc 15

The RCX message source

5.172.2.24 #define RCX\_OutputStatusSrc 3

The RCX output status source

5.172.2.25 #define RCX\_ProgramSlotSrc 8

The RCX program slot source

5.172.2.26 #define RCX\_RandomSrc 4

The RCX random number source

5.172.2.27 #define RCX\_ScoutCounterLimitSrc 22

The Scout counter limit source

5.172.2.28 #define RCX\_ScoutEventFBSrc 24

The Scout event feedback source

5.172.2.29 #define RCX\_ScoutLightParamsSrc 19

The Scout light parameters source

5.172.2.30 #define RCX\_ScoutRulesSrc 18

The Scout rules source

5.172.2.31 #define RCX\_ScoutTimerLimitSrc 20

The Scout timer limit source

5.172.2.32 #define RCX\_TaskEventsSrc 23

The RCX task events source

5.172.2.33 #define RCX\_TenMSTimerSrc 26

The RCX 10ms timer source

5.172.2.34 #define RCX\_TimerSrc 1

The RCX timer source

5.172.2.35 #define RCX\_UARTSetupSrc 33

The RCX UART setup source

5.172.2.36 #define RCX\_UpperThresholdSrc 28

The RCX event upper threshold source

5.172.2.37 #define RCX\_VariableSrc 0

The RCX variable source

5.172.2.38 #define RCX\_WatchSrc 14

The RCX watch source

## 5.173 RCX and Scout opcode constants

Constants for use when specifying RCX and Scout opcodes.

### Macros

- #define RCX\_PingOp 0x10
- #define RCX\_BatteryLevelOp 0x30
- #define RCX\_DeleteTasksOp 0x40
- #define RCX\_StopAllTasksOp 0x50
- #define RCX\_PBTurnOffOp 0x60
- #define RCX\_DeleteSubsOp 0x70
- #define RCX\_ClearSoundOp 0x80
- #define RCX\_ClearMsgOp 0x90
- #define RCX\_LSCalibrateOp 0xc0
- #define RCX\_MuteSoundOp 0xd0
- #define RCX\_UnmuteSoundOp 0xe0
- #define RCX\_ClearAllEventsOp 0x06
- #define RCX\_OnOffFloatOp 0x21
- #define RCX\_IRModeOp 0x31
- #define RCX\_PlaySoundOp 0x51
- #define RCX\_DeleteTaskOp 0x61
- #define RCX\_StartTaskOp 0x71
- #define RCX\_StopTaskOp 0x81
- #define RCX\_SelectProgramOp 0x91
- #define RCX\_ClearTimerOp 0xa1
- #define RCX\_AutoOffOp 0xb1
- #define RCX\_DeleteSubOp 0xc1
- #define RCX\_ClearSensorOp 0xd1
- #define RCX\_OutputDirOp 0xe1
- #define RCX\_PlayToneVarOp 0x02
- #define RCX\_PollOp 0x12
- #define RCX\_SetWatchOp 0x22
- #define RCX\_InputTypeOp 0x32
- #define RCX\_InputModeOp 0x42
- #define RCX\_SetDatalogOp 0x52
- #define RCX\_DatalogOp 0x62
- #define RCX\_SendUARTDataOp 0xc2
- #define RCX\_RemoteOp 0xd2
- #define RCX\_VLLOp 0xe2
- #define RCX\_DirectEventOp 0x03
- #define RCX\_OutputPowerOp 0x13
- #define RCX\_PlayToneOp 0x23
- #define RCX\_DisplayOp 0x33
- #define RCX\_PollMemoryOp 0x63
- #define RCX\_SetFeedbackOp 0x83
- #define RCX\_SetEventOp 0x93
- #define RCX\_GOutputPowerOp 0xa3
- #define RCX\_LSSupperThreshOp 0xb3
- #define RCX\_LSLowerThreshOp 0xc3

- #define RCX\_LSHysteresisOp 0xd3
- #define RCX\_LSBlinkTimeOp 0xe3
- #define RCX\_CalibrateEventOp 0x04
- #define RCX\_SetVarOp 0x14
- #define RCX\_SumVarOp 0x24
- #define RCX\_SubVarOp 0x34
- #define RCX\_DivVarOp 0x44
- #define RCX\_MulVarOp 0x54
- #define RCX\_SgnVarOp 0x64
- #define RCX\_AbsVarOp 0x74
- #define RCX\_AndVarOp 0x84
- #define RCX\_OrVarOp 0x94
- #define RCX\_UploadDatalogOp 0xa4
- #define RCX\_SetTimerLimitOp 0xc4
- #define RCX\_SetCounterOp 0xd4
- #define RCX\_SetSourceValueOp 0x05
- #define RCX\_UnlockOp 0x15
- #define RCX\_BootModeOp 0x65
- #define RCX\_UnlockFirmOp 0xa5
- #define RCX\_ScoutRulesOp 0xd5
- #define RCX\_ViewSourceValOp 0xe5
- #define RCX\_ScoutOp 0x47
- #define RCX\_SoundOp 0x57
- #define RCX\_GOutputModeOp 0x67
- #define RCX\_GOutputDirOp 0x77
- #define RCX\_LightOp 0x87
- #define RCX\_IncCounterOp 0x97
- #define RCX\_DecCounterOp 0xa7
- #define RCX\_ClearCounterOp 0xb7
- #define RCX\_SetPriorityOp 0xd7
- #define RCX\_MessageOp 0xf7

### 5.173.1 Detailed Description

Constants for use when specifying RCX and Scout opcodes.

### 5.173.2 Macro Definition Documentation

#### 5.173.2.1 #define RCX\_AbsVarOp 0x74

Absolute value function

#### 5.173.2.2 #define RCX\_AndVarOp 0x84

AND function

#### 5.173.2.3 #define RCX\_AutoOffOp 0xb1

Set auto off timer

5.173.2.4 #define RCX\_BatteryLevelOp 0x30

Read the battery level

5.173.2.5 #define RCX\_BootModeOp 0x65

Set into boot mode

5.173.2.6 #define RCX\_CalibrateEventOp 0x04

Calibrate event

5.173.2.7 #define RCX\_ClearAllEventsOp 0x06

Clear all events

5.173.2.8 #define RCX\_ClearCounterOp 0xb7

Clear a counter

5.173.2.9 #define RCX\_ClearMsgOp 0x90

Clear message

5.173.2.10 #define RCX\_ClearSensorOp 0xd1

Clear a sensor

5.173.2.11 #define RCX\_ClearSoundOp 0x80

Clear sound

5.173.2.12 #define RCX\_ClearTimerOp 0xa1

Clear a timer

5.173.2.13 #define RCX\_DatalogOp 0x62

Datalog the specified source/value

5.173.2.14 #define RCX\_DecCounterOp 0xa7

Decrement a counter

5.173.2.15 #define RCX\_DeleteSubOp 0xc1

Delete a subroutine

5.173.2.16 #define RCX\_DeleteSubsOp 0x70

Delete subroutines

5.173.2.17 #define RCX\_DeleteTaskOp 0x61

Delete a task

5.173.2.18 #define RCX\_DeleteTasksOp 0x40

Delete tasks

5.173.2.19 #define RCX\_DirectEventOp 0x03

Fire an event

5.173.2.20 #define RCX\_DisplayOp 0x33

Set LCD display value

5.173.2.21 #define RCX\_DivVarOp 0x44

Divide function

5.173.2.22 #define RCX\_GOutputDirOp 0x77

Set global motor direction

5.173.2.23 #define RCX\_GOutputModeOp 0x67

Set global motor mode

5.173.2.24 #define RCX\_GOutputPowerOp 0xa3

Set global motor power levels

5.173.2.25 #define RCX\_IncCounterOp 0x97

Increment a counter

5.173.2.26 #define RCX\_InputModeOp 0x42

Set the input mode

5.173.2.27 #define RCX\_InputTypeOp 0x32

Set the input type

5.173.2.28 #define RCX\_IRModeOp 0x31

Set the IR transmit mode

5.173.2.29 #define RCX\_LightOp 0x87

Light opcode

5.173.2.30 #define RCX\_LSBlinkTimeOp 0xe3

Set the light sensor blink time

5.173.2.31 #define RCX\_LSCalibrateOp 0xc0

Calibrate the light sensor

5.173.2.32 #define RCX\_LSHysteresisOp 0xd3

Set the light sensor hysteresis

5.173.2.33 #define RCX\_LSLowerThreshOp 0xc3

Set the light sensor lower threshold

5.173.2.34 #define RCX\_LSSUpperThreshOp 0xb3

Set the light sensor upper threshold

5.173.2.35 #define RCX\_MessageOp 0xf7

Set message

5.173.2.36 #define RCX\_MulVarOp 0x54

Multiply function

5.173.2.37 #define RCX\_MuteSoundOp 0xd0

Mute sound

5.173.2.38 #define RCX\_OnOffFloatOp 0x21

Control motor state - on, off, float

5.173.2.39 #define RCX\_OrVarOp 0x94

OR function

5.173.2.40 #define RCX\_OutputDirOp 0xe1

Set the motor direction

5.173.2.41 #define RCX\_OutputPowerOp 0x13

Set the motor power level

5.173.2.42 #define RCX\_PBTurnOffOp 0x60

Turn off the brick

5.173.2.43 #define RCX\_PingOp 0x10

Ping the brick

5.173.2.44 #define RCX\_PlaySoundOp 0x51

Play a sound

5.173.2.45 #define RCX\_PlayToneOp 0x23

Play a tone

5.173.2.46 #define RCX\_PlayToneVarOp 0x02

Play a tone using a variable

5.173.2.47 #define RCX\_PollMemoryOp 0x63

Poll a memory location

5.173.2.48 #define RCX\_PollOp 0x12

Poll a source/value combination

5.173.2.49 #define RCX\_RemoteOp 0xd2

Execute simulated remote control buttons

5.173.2.50 #define RCX\_ScoutOp 0x47

Scout opcode

5.173.2.51 #define RCX\_ScoutRulesOp 0xd5

Set Scout rules

5.173.2.52 #define RCX\_SelectProgramOp 0x91

Select a program slot

5.173.2.53 #define RCX\_SendUARTDataOp 0xc2

Send data via IR using UART settings

5.173.2.54 #define RCX\_SetCounterOp 0xd4

Set counter value

5.173.2.55 #define RCX\_SetDatalogOp 0x52

Set the datalog size

5.173.2.56 #define RCX\_SetEventOp 0x93

Set an event

5.173.2.57 #define RCX\_SetFeedbackOp 0x83

Set Scout feedback

5.173.2.58 #define RCX\_SetPriorityOp 0xd7

Set task priority

5.173.2.59 #define RCX\_SetSourceValueOp 0x05

Set a source/value

5.173.2.60 #define RCX\_SetTimerLimitOp 0xc4

Set timer limit

5.173.2.61 #define RCX\_SetVarOp 0x14

Set function

5.173.2.62 #define RCX\_SetWatchOp 0x22

Set the watch source/value

5.173.2.63 #define RCX\_SgnVarOp 0x64

Sign function

5.173.2.64 #define RCX\_SoundOp 0x57

Sound opcode

5.173.2.65 #define RCX\_StartTaskOp 0x71

Start a task

5.173.2.66 #define RCX\_StopAllTasksOp 0x50

Stop all tasks

5.173.2.67 #define RCX\_StopTaskOp 0x81

Stop a task

5.173.2.68 #define RCX\_SubVarOp 0x34

Subtract function

5.173.2.69 #define RCX\_SumVarOp 0x24

Sum function

5.173.2.70 #define RCX\_UnlockFirmOp 0xa5

Unlock the firmware

5.173.2.71 #define RCX\_UnlockOp 0x15

Unlock the brick

5.173.2.72 #define RCX\_UnmuteSoundOp 0xe0

Unmute sound

5.173.2.73 #define RCX\_UploadDatalogOp 0xa4

Upload datalog contents

5.173.2.74 #define RCX\_ViewSourceValOp 0xe5

View a source/value

5.173.2.75 #define RCX\_VLLOp 0xe2

Send visual light link (VLL) data

## 5.174 HiTechnic/mindsensors Power Function/IR Train constants

Constants that are for use with the HiTechnic IRLink or mindsensors nRLink in Power Function or IR Train mode.

### Modules

- [IR Train channel constants](#)

*Constants that are for specifying IR Train channels.*

- [PF/IR Train function constants](#)

*Constants that are for sending PF/IR Train functions.*

- [Power Function CST options constants](#)

*Constants that are for specifying Power Function CST options.*

- [Power Function PWM option constants](#)

*Constants that are for specifying Power Function PWM options.*

- [Power Function channel constants](#)

*Constants that are for specifying Power Function channels.*

- [Power Function command constants](#)

*Constants that are for sending Power Function commands.*

- [Power Function mode constants](#)

*Constants that are for choosing Power Function modes.*

- [Power Function output constants](#)

*Constants that are for choosing a Power Function output.*

- [Power Function pin constants](#)

*Constants that are for choosing a Power Function pin.*

- [Power Function single pin function constants](#)

*Constants that are for sending Power Function single pin functions.*

### 5.174.1 Detailed Description

Constants that are for use with the HiTechnic IRLink or mindsensors nRLink in Power Function or IR Train mode.

## 5.175 Power Function command constants

Constants that are for sending Power Function commands.

### Macros

- #define PF\_CMD\_STOP 0
- #define PF\_CMD\_FLOAT 0
- #define PF\_CMD\_FWD 1
- #define PF\_CMD\_REV 2
- #define PF\_CMD\_BRAKE 3

### 5.175.1 Detailed Description

Constants that are for sending Power Function commands.

### 5.175.2 Macro Definition Documentation

#### 5.175.2.1 #define PF\_CMD\_BRAKE 3

Power function command brake

#### 5.175.2.2 #define PF\_CMD\_FLOAT 0

Power function command float (same as stop)

#### 5.175.2.3 #define PF\_CMD\_FWD 1

Power function command forward

#### 5.175.2.4 #define PF\_CMD\_REV 2

Power function command reverse

#### 5.175.2.5 #define PF\_CMD\_STOP 0

Power function command stop

## 5.176 Power Function channel constants

Constants that are for specifying Power Function channels.

### Macros

- #define PF\_CHANNEL\_1 0
- #define PF\_CHANNEL\_2 1
- #define PF\_CHANNEL\_3 2
- #define PF\_CHANNEL\_4 3

### 5.176.1 Detailed Description

Constants that are for specifying Power Function channels.

### 5.176.2 Macro Definition Documentation

#### 5.176.2.1 #define PF\_CHANNEL\_1 0

Power function channel 1

#### 5.176.2.2 #define PF\_CHANNEL\_2 1

Power function channel 2

#### 5.176.2.3 #define PF\_CHANNEL\_3 2

Power function channel 3

#### 5.176.2.4 #define PF\_CHANNEL\_4 3

Power function channel 4

## 5.177 Power Function mode constants

Constants that are for choosing Power Function modes.

### Macros

- #define PF\_MODE\_TRAIN 0
- #define PF\_MODE\_COMBO\_DIRECT 1
- #define PF\_MODE\_SINGLE\_PIN\_CONT 2
- #define PF\_MODE\_SINGLE\_PIN\_TIME 3
- #define PF\_MODE\_COMBO\_PWM 4
- #define PF\_MODE\_SINGLE\_OUTPUT\_PWM 4
- #define PF\_MODE\_SINGLE\_OUTPUT\_CST 6

### 5.177.1 Detailed Description

Constants that are for choosing Power Function modes.

### 5.177.2 Macro Definition Documentation

#### 5.177.2.1 #define PF\_MODE\_COMBO\_DIRECT 1

Power function mode combo direct

#### 5.177.2.2 #define PF\_MODE\_COMBO\_PWM 4

Power function mode combo pulse width modulation (PWM)

#### 5.177.2.3 #define PF\_MODE\_SINGLE\_OUTPUT\_CST 6

Power function mode single output clear, set, toggle (CST)

#### 5.177.2.4 #define PF\_MODE\_SINGLE\_OUTPUT\_PWM 4

Power function mode single output pulse width modulation (PWM)

#### 5.177.2.5 #define PF\_MODE\_SINGLE\_PIN\_CONT 2

Power function mode single pin continuous

#### 5.177.2.6 #define PF\_MODE\_SINGLE\_PIN\_TIME 3

Power function mode single pin timed

#### 5.177.2.7 #define PF\_MODE\_TRAIN 0

Power function mode IR Train

## 5.178 PF/IR Train function constants

Constants that are for sending PF/IR Train functions.

### Macros

- #define TRAIN\_FUNC\_STOP 0
- #define TRAIN\_FUNC\_INCR\_SPEED 1
- #define TRAIN\_FUNC\_DECR\_SPEED 2
- #define TRAIN\_FUNC\_TOGGLE\_LIGHT 4

### 5.178.1 Detailed Description

Constants that are for sending PF/IR Train functions.

### 5.178.2 Macro Definition Documentation

#### 5.178.2.1 #define TRAIN\_FUNC\_DECR\_SPEED 2

PF/IR Train function decrement speed

#### 5.178.2.2 #define TRAIN\_FUNC\_INCR\_SPEED 1

PF/IR Train function increment speed

#### 5.178.2.3 #define TRAIN\_FUNC\_STOP 0

PF/IR Train function stop

#### 5.178.2.4 #define TRAIN\_FUNC\_TOGGLE\_LIGHT 4

PF/IR Train function toggle light

## 5.179 IR Train channel constants

Constants that are for specifying IR Train channels.

### Macros

- `#define TRAIN_CHANNEL_1 0`
- `#define TRAIN_CHANNEL_2 1`
- `#define TRAIN_CHANNEL_3 2`
- `#define TRAIN_CHANNEL_ALL 3`

### 5.179.1 Detailed Description

Constants that are for specifying IR Train channels.

### 5.179.2 Macro Definition Documentation

#### 5.179.2.1 `#define TRAIN_CHANNEL_1 0`

IR Train channel 1

#### 5.179.2.2 `#define TRAIN_CHANNEL_2 1`

IR Train channel 2

#### 5.179.2.3 `#define TRAIN_CHANNEL_3 2`

IR Train channel 3

#### 5.179.2.4 `#define TRAIN_CHANNEL_ALL 3`

IR Train channel all

## 5.180 Power Function output constants

Constants that are for choosing a Power Function output.

### Macros

- `#define PF_OUT_A 0`
- `#define PF_OUT_B 1`

#### 5.180.1 Detailed Description

Constants that are for choosing a Power Function output.

#### 5.180.2 Macro Definition Documentation

##### 5.180.2.1 `#define PF_OUT_A 0`

Power function output A

##### 5.180.2.2 `#define PF_OUT_B 1`

Power function output B

## 5.181 Power Function pin constants

Constants that are for choosing a Power Function pin.

### Macros

- #define PF\_PIN\_C1 0
- #define PF\_PIN\_C2 1

### 5.181.1 Detailed Description

Constants that are for choosing a Power Function pin.

### 5.181.2 Macro Definition Documentation

#### 5.181.2.1 #define PF\_PIN\_C1 0

Power function pin C1

#### 5.181.2.2 #define PF\_PIN\_C2 1

Power function pin C2

## 5.182 Power Function single pin function constants

Constants that are for sending Power Function single pin functions.

### Macros

- #define PF\_FUNC\_NOCHANGE 0
- #define PF\_FUNC\_CLEAR 1
- #define PF\_FUNC\_SET 2
- #define PF\_FUNC\_TOGGLE 3

### 5.182.1 Detailed Description

Constants that are for sending Power Function single pin functions.

### 5.182.2 Macro Definition Documentation

#### 5.182.2.1 #define PF\_FUNC\_CLEAR 1

Power function single pin - clear

#### 5.182.2.2 #define PF\_FUNC\_NOCHANGE 0

Power function single pin - no change

#### 5.182.2.3 #define PF\_FUNC\_SET 2

Power function single pin - set

#### 5.182.2.4 #define PF\_FUNC\_TOGGLE 3

Power function single pin - toggle

## 5.183 Power Function CST options constants

Constants that are for specifying Power Function CST options.

### Macros

- #define PF\_CST\_CLEAR1\_CLEAR2 0
- #define PF\_CST\_SET1\_CLEAR2 1
- #define PF\_CST\_CLEAR1\_SET2 2
- #define PF\_CST\_SET1\_SET2 3
- #define PF\_CST\_INCREMENT\_PWM 4
- #define PF\_CST\_DECREMENT\_PWM 5
- #define PF\_CST\_FULL\_FWD 6
- #define PF\_CST\_FULL\_REV 7
- #define PF\_CST\_TOGGLE\_DIR 8

### 5.183.1 Detailed Description

Constants that are for specifying Power Function CST options.

### 5.183.2 Macro Definition Documentation

#### 5.183.2.1 #define PF\_CST\_CLEAR1\_CLEAR2 0

Power function CST clear 1 and clear 2

#### 5.183.2.2 #define PF\_CST\_CLEAR1\_SET2 2

Power function CST clear 1 and set 2

#### 5.183.2.3 #define PF\_CST\_DECREMENT\_PWM 5

Power function CST decrement PWM

#### 5.183.2.4 #define PF\_CST\_FULL\_FWD 6

Power function CST full forward

#### 5.183.2.5 #define PF\_CST\_FULL\_REV 7

Power function CST full reverse

#### 5.183.2.6 #define PF\_CST\_INCREMENT\_PWM 4

Power function CST increment PWM

#### 5.183.2.7 #define PF\_CST\_SET1\_CLEAR2 1

Power function CST set 1 and clear 2

#### 5.183.2.8 #define PF\_CST\_SET1\_SET2 3

Power function CST set 1 and set 2

## 5.183.2.9 #define PF\_CST\_TOGGLE\_DIR 8

Power function CST toggle direction

## 5.184 Power Function PWM option constants

Constants that are for specifying Power Function PWM options.

### Macros

- #define PF\_PWM\_FLOAT 0
- #define PF\_PWM\_FWD1 1
- #define PF\_PWM\_FWD2 2
- #define PF\_PWM\_FWD3 3
- #define PF\_PWM\_FWD4 4
- #define PF\_PWM\_FWD5 5
- #define PF\_PWM\_FWD6 6
- #define PF\_PWM\_FWD7 7
- #define PF\_PWM\_BRAKE 8
- #define PF\_PWM\_REV7 9
- #define PF\_PWM\_REV6 10
- #define PF\_PWM\_REV5 11
- #define PF\_PWM\_REV4 12
- #define PF\_PWM\_REV3 13
- #define PF\_PWM\_REV2 14
- #define PF\_PWM\_REV1 15

### 5.184.1 Detailed Description

Constants that are for specifying Power Function PWM options.

### 5.184.2 Macro Definition Documentation

#### 5.184.2.1 #define PF\_PWM\_BRAKE 8

Power function PWM brake

#### 5.184.2.2 #define PF\_PWM\_FLOAT 0

Power function PWM float

#### 5.184.2.3 #define PF\_PWM\_FWD1 1

Power function PWM foward level 1

#### 5.184.2.4 #define PF\_PWM\_FWD2 2

Power function PWM foward level 2

#### 5.184.2.5 #define PF\_PWM\_FWD3 3

Power function PWM foward level 3

#### 5.184.2.6 #define PF\_PWM\_FWD4 4

Power function PWM foward level 4

5.184.2.7 #define PF\_PWM\_FWD5 5

Power function PWM foward level 5

5.184.2.8 #define PF\_PWM\_FWD6 6

Power function PWM foward level 6

5.184.2.9 #define PF\_PWM\_FWD7 7

Power function PWM foward level 7

5.184.2.10 #define PF\_PWM\_REV1 15

Power function PWM reverse level 1

5.184.2.11 #define PF\_PWM\_REV2 14

Power function PWM reverse level 2

5.184.2.12 #define PF\_PWM\_REV3 13

Power function PWM reverse level 3

5.184.2.13 #define PF\_PWM\_REV4 12

Power function PWM reverse level 4

5.184.2.14 #define PF\_PWM\_REV5 11

Power function PWM reverse level 5

5.184.2.15 #define PF\_PWM\_REV6 10

Power function PWM reverse level 6

5.184.2.16 #define PF\_PWM\_REV7 9

Power function PWM reverse level 7

## 5.185 HiTechnic device constants

Constants that are for use with HiTechnic devices.

### Modules

- [HiTechnic Angle sensor constants](#)

*Constants that are for use with the HiTechnic Angle sensor device.*

- [HiTechnic Barometric sensor constants](#)

*Constants that are for use with the HiTechnic Barometric sensor device.*

- [HiTechnic Color2 constants](#)

*Constants that are for use with the HiTechnic Color2 device.*

- [HiTechnic IRReceiver constants](#)

*Constants that are for use with the HiTechnic IRReceiver device.*

- [HiTechnic IRSeeker2 constants](#)

*Constants that are for use with the HiTechnic IRSeeker2 device.*

- [HiTechnic PIR sensor constants](#)

*Constants that are for use with the HiTechnic PIR sensor device.*

- [HiTechnic Prototype board constants](#)

*Constants that are for use with the HiTechnic Prototype board.*

- [HiTechnic SuperPro constants](#)

*Constants that are for use with the HiTechnic SuperPro board.*

### Macros

- `#define HT_ADDR_IRSEEKER 0x02`
- `#define HT_ADDR_IRSEEKER2 0x10`
- `#define HT_ADDR_IRRECEIVER 0x02`
- `#define HT_ADDR_COMPASS 0x02`
- `#define HT_ADDR_ACCEL 0x02`
- `#define HT_ADDR_COLOR 0x02`
- `#define HT_ADDR_COLOR2 0x02`
- `#define HT_ADDR_IRLINK 0x02`
- `#define HT_ADDR_ANGLE 0x02`
- `#define HT_ADDR_BAROMETRIC 0x02`
- `#define HT_ADDR_PROTOBOARD 0x02`
- `#define HT_ADDR_SUPERPRO 0x10`
- `#define HT_ADDR_PIR 0x02`

#### 5.185.1 Detailed Description

Constants that are for use with HiTechnic devices.

#### 5.185.2 Macro Definition Documentation

##### 5.185.2.1 `#define HT_ADDR_ACCEL 0x02`

HiTechnic Accel I2C address

5.185.2.2 #define HT\_ADDR\_ANGLE 0x02

HiTechnic Angle I2C address

5.185.2.3 #define HT\_ADDR\_BAROMETRIC 0x02

HiTechnic Barometric I2C address

5.185.2.4 #define HT\_ADDR\_COLOR 0x02

HiTechnic Color I2C address

5.185.2.5 #define HT\_ADDR\_COLOR2 0x02

HiTechnic Color2 I2C address

5.185.2.6 #define HT\_ADDR\_COMPASS 0x02

HiTechnic Compass I2C address

5.185.2.7 #define HT\_ADDR\_IRLINK 0x02

HiTechnic IRLink I2C address

5.185.2.8 #define HT\_ADDR\_IRRECEIVER 0x02

HiTechnic IRReceiver I2C address

5.185.2.9 #define HT\_ADDR\_IRSEEKER 0x02

HiTechnic IRSeeker I2C address

5.185.2.10 #define HT\_ADDR\_IRSEEKER2 0x10

HiTechnic IRSeeker2 I2C address

5.185.2.11 #define HT\_ADDR\_PIR 0x02

HiTechnic PIR (Passive Infrared) I2C address

5.185.2.12 #define HT\_ADDR\_PROTOBOARD 0x02

HiTechnic Prototype board I2C address

5.185.2.13 #define HT\_ADDR\_SUPERPRO 0x10

HiTechnic SuperPro board I2C address

## 5.186 HiTechnic IRSeeker2 constants

Constants that are for use with the HiTechnic IRSeeker2 device.

### Macros

- `#define HTIR2_MODE_1200 0`
- `#define HTIR2_MODE_600 1`
- `#define HTIR2_REG_MODE 0x41`
- `#define HTIR2_REG_DCDIR 0x42`
- `#define HTIR2_REG_DC01 0x43`
- `#define HTIR2_REG_DC02 0x44`
- `#define HTIR2_REG_DC03 0x45`
- `#define HTIR2_REG_DC04 0x46`
- `#define HTIR2_REG_DC05 0x47`
- `#define HTIR2_REG_DCAVG 0x48`
- `#define HTIR2_REG_ACDIR 0x49`
- `#define HTIR2_REG_AC01 0x4A`
- `#define HTIR2_REG_AC02 0x4B`
- `#define HTIR2_REG_AC03 0x4C`
- `#define HTIR2_REG_AC04 0x4D`
- `#define HTIR2_REG_AC05 0x4E`

### 5.186.1 Detailed Description

Constants that are for use with the HiTechnic IRSeeker2 device.

### 5.186.2 Macro Definition Documentation

#### 5.186.2.1 `#define HTIR2_MODE_1200 0`

Set IRSeeker2 to 1200 mode

#### 5.186.2.2 `#define HTIR2_MODE_600 1`

Set IRSeeker2 to 600 mode

#### 5.186.2.3 `#define HTIR2_REG_AC01 0x4A`

IRSeeker2 AC 01 register

#### 5.186.2.4 `#define HTIR2_REG_AC02 0x4B`

IRSeeker2 AC 02 register

#### 5.186.2.5 `#define HTIR2_REG_AC03 0x4C`

IRSeeker2 AC 03 register

#### 5.186.2.6 `#define HTIR2_REG_AC04 0x4D`

IRSeeker2 AC 04 register

5.186.2.7 #define HTIR2\_REG\_AC05 0x4E

IRSeeker2 AC 05 register

5.186.2.8 #define HTIR2\_REG\_ACDIR 0x49

IRSeeker2 AC direction register

5.186.2.9 #define HTIR2\_REG\_DC01 0x43

IRSeeker2 DC 01 register

5.186.2.10 #define HTIR2\_REG\_DC02 0x44

IRSeeker2 DC 02 register

5.186.2.11 #define HTIR2\_REG\_DC03 0x45

IRSeeker2 DC 03 register

5.186.2.12 #define HTIR2\_REG\_DC04 0x46

IRSeeker2 DC 04 register

5.186.2.13 #define HTIR2\_REG\_DCAVG 0x47

IRSeeker2 DC average register

5.186.2.14 #define HTIR2\_REG\_DCDIR 0x42

IRSeeker2 DC direction register

5.186.2.16 #define HTIR2\_REG\_MODE 0x41

IRSeeker2 mode register

## 5.187 HiTechnic IRReceiver constants

Constants that are for use with the HiTechnic IRReceiver device.

### Macros

- `#define HT_CH1_A 0`
- `#define HT_CH1_B 1`
- `#define HT_CH2_A 2`
- `#define HT_CH2_B 3`
- `#define HT_CH3_A 4`
- `#define HT_CH3_B 5`
- `#define HT_CH4_A 6`
- `#define HT_CH4_B 7`

### 5.187.1 Detailed Description

Constants that are for use with the HiTechnic IRReceiver device.

### 5.187.2 Macro Definition Documentation

#### 5.187.2.1 `#define HT_CH1_A 0`

Use IRReceiver channel 1 output A

#### 5.187.2.2 `#define HT_CH1_B 1`

Use IRReceiver channel 1 output B

#### 5.187.2.3 `#define HT_CH2_A 2`

Use IRReceiver channel 2 output A

#### 5.187.2.4 `#define HT_CH2_B 3`

Use IRReceiver channel 2 output B

#### 5.187.2.5 `#define HT_CH3_A 4`

Use IRReceiver channel 3 output A

#### 5.187.2.6 `#define HT_CH3_B 5`

Use IRReceiver channel 3 output B

#### 5.187.2.7 `#define HT_CH4_A 6`

Use IRReceiver channel 4 output A

#### 5.187.2.8 `#define HT_CH4_B 7`

Use IRReceiver channel 4 output B

## 5.188 HiTechnic Color2 constants

Constants that are for use with the HiTechnic Color2 device.

### Macros

- `#define HT_CMD_COLOR2_ACTIVE 0x00`
- `#define HT_CMD_COLOR2_PASSIVE 0x01`
- `#define HT_CMD_COLOR2_RAW 0x03`
- `#define HT_CMD_COLOR2_50HZ 0x35`
- `#define HT_CMD_COLOR2_60HZ 0x36`
- `#define HT_CMD_COLOR2_BLCAL 0x42`
- `#define HT_CMD_COLOR2_WBCAL 0x43`
- `#define HT_CMD_COLOR2_FAR 0x46`
- `#define HT_CMD_COLOR2_LED_HI 0x48`
- `#define HT_CMD_COLOR2_LED_LOW 0x4C`
- `#define HT_CMD_COLOR2_NEAR 0x4E`

### 5.188.1 Detailed Description

Constants that are for use with the HiTechnic Color2 device.

### 5.188.2 Macro Definition Documentation

#### 5.188.2.1 `#define HT_CMD_COLOR2_50HZ 0x35`

Set the Color2 sensor to 50Hz mode

#### 5.188.2.2 `#define HT_CMD_COLOR2_60HZ 0x36`

Set the Color2 sensor to 60Hz mode

#### 5.188.2.3 `#define HT_CMD_COLOR2_ACTIVE 0x00`

Set the Color2 sensor to active mode

#### 5.188.2.4 `#define HT_CMD_COLOR2_BLCAL 0x42`

Set the Color2 sensor to black level calibration mode

#### 5.188.2.5 `#define HT_CMD_COLOR2_FAR 0x46`

Set the Color2 sensor to far mode

#### 5.188.2.6 `#define HT_CMD_COLOR2_LED_HI 0x48`

Set the Color2 sensor to LED high mode

#### 5.188.2.7 `#define HT_CMD_COLOR2_LED_LOW 0x4C`

Set the Color2 sensor to LED low mode

5.188.2.8 #define HT\_CMD\_COLOR2\_NEAR 0x4E

Set the Color2 sensor to near mode

5.188.2.9 #define HT\_CMD\_COLOR2\_PASSIVE 0x01

Set the Color2 sensor to passive mode

5.188.2.10 #define HT\_CMD\_COLOR2\_RAW 0x03

Set the Color2 sensor to raw mode

5.188.2.11 #define HT\_CMD\_COLOR2\_WBCAL 0x43

Set the Color2 sensor to white level calibration mode

## 5.189 HiTechnic Angle sensor constants

Constants that are for use with the HiTechnic Angle sensor device.

### Macros

- `#define HTANGLE_MODE_NORMAL 0x00`
- `#define HTANGLE_MODE_CALIBRATE 0x43`
- `#define HTANGLE_MODE_RESET 0x52`
- `#define HTANGLE_REG_MODE 0x41`
- `#define HTANGLE_REG_DCDIR 0x42`
- `#define HTANGLE_REG_DC01 0x43`
- `#define HTANGLE_REG_DC02 0x44`
- `#define HTANGLE_REG_DC03 0x45`
- `#define HTANGLE_REG_DC04 0x46`
- `#define HTANGLE_REG_DC05 0x47`
- `#define HTANGLE_REG_DCAVG 0x48`
- `#define HTANGLE_REG_ACDIR 0x49`

### 5.189.1 Detailed Description

Constants that are for use with the HiTechnic Angle sensor device.

### 5.189.2 Macro Definition Documentation

#### 5.189.2.1 `#define HTANGLE_MODE_CALIBRATE 0x43`

Resets 0 degree position to current shaft angle

#### 5.189.2.2 `#define HTANGLE_MODE_NORMAL 0x00`

Normal angle measurement mode

#### 5.189.2.3 `#define HTANGLE_MODE_RESET 0x52`

Resets the accumulated angle

#### 5.189.2.4 `#define HTANGLE_REG_ACDIR 0x49`

Angle 16 bit revolutions per minute, low byte register

#### 5.189.2.5 `#define HTANGLE_REG_DC01 0x43`

Angle current angle (1 degree adder) register

#### 5.189.2.6 `#define HTANGLE_REG_DC02 0x44`

Angle 32 bit accumulated angle, high byte register

#### 5.189.2.7 `#define HTANGLE_REG_DC03 0x45`

Angle 32 bit accumulated angle, mid byte register

5.189.2.8 #define HTANGLE\_REG\_DC04 0x46

Angle 32 bit accumulated angle, mid byte register

5.189.2.9 #define HTANGLE\_REG\_DC05 0x47

Angle 32 bit accumulated angle, low byte register

5.189.2.10 #define HTANGLE\_REG\_DCAVG 0x48

Angle 16 bit revolutions per minute, high byte register

5.189.2.11 #define HTANGLE\_REG\_DCDIR 0x42

Angle current angle (2 degree increments) register

5.189.2.12 #define HTANGLE\_REG\_MODE 0x41

Angle mode register

## 5.190 HiTechnic Barometric sensor constants

Constants that are for use with the HiTechnic Barometric sensor device.

### Macros

- #define HTBAR\_REG\_COMMAND 0x40
- #define HTBAR\_REG\_TEMPERATURE 0x42
- #define HTBAR\_REG\_PRESSURE 0x44
- #define HTBAR\_REG\_CALIBRATION 0x46

### 5.190.1 Detailed Description

Constants that are for use with the HiTechnic Barometric sensor device.

### 5.190.2 Macro Definition Documentation

#### 5.190.2.1 #define HTBAR\_REG\_CALIBRATION 0x46

Barometric sensor calibration register (2 bytes msb/lsb)

#### 5.190.2.2 #define HTBAR\_REG\_COMMAND 0x40

Barometric sensor command register

#### 5.190.2.3 #define HTBAR\_REG\_PRESSURE 0x44

Barometric sensor pressure register (2 bytes msb/lsb)

#### 5.190.2.4 #define HTBAR\_REG\_TEMPERATURE 0x42

Barometric sensor temperature register (2 bytes msb/lsb)

## 5.191 HiTechnic Prototype board constants

Constants that are for use with the HiTechnic Prototype board.

### Modules

- [HiTechnic Prototype board analog input constants](#)

*Constants that are for use with reading the HiTechnic Prototype board analog input values.*

### Macros

- `#define HTPROTO_REG_A0 0x42`
- `#define HTPROTO_REG_A1 0x44`
- `#define HTPROTO_REG_A2 0x46`
- `#define HTPROTO_REG_A3 0x48`
- `#define HTPROTO_REG_A4 0x4A`
- `#define HTPROTO_REG_DIN 0x4C`
- `#define HTPROTO_REG_DOUT 0x4D`
- `#define HTPROTO_REG_DCTRL 0x4E`
- `#define HTPROTO_REG_SRATE 0x4F`

### 5.191.1 Detailed Description

Constants that are for use with the HiTechnic Prototype board.

### 5.191.2 Macro Definition Documentation

#### 5.191.2.1 `#define HTPROTO_REG_A0 0x42`

Prototype board analog 0 register (2 bytes msb/lsb)

#### 5.191.2.2 `#define HTPROTO_REG_A1 0x44`

Prototype board analog 1 register (2 bytes msb/lsb)

#### 5.191.2.3 `#define HTPROTO_REG_A2 0x46`

Prototype board analog 2 register (2 bytes msb/lsb)

#### 5.191.2.4 `#define HTPROTO_REG_A3 0x48`

Prototype board analog 3 register (2 bytes msb/lsb)

#### 5.191.2.5 `#define HTPROTO_REG_A4 0x4A`

Prototype board analog 4 register (2 bytes msb/lsb)

#### 5.191.2.6 `#define HTPROTO_REG_DCTRL 0x4E`

Prototype board digital pin control register (6 bits)

5.191.2.7 #define HTPROTO\_REG\_DIN 0x4C

Prototype board digital pin input register (6 bits)

5.191.2.8 #define HTPROTO\_REG\_DOUT 0x4D

Prototype board digital pin output register (6 bits)

5.191.2.9 #define HTPROTO\_REG\_SRATE 0x4F

Prototype board sample rate register

## 5.192 HiTechnic Prototype board analog input constants

Constants that are for use with reading the HiTechnic Prototype board analog input values.

### Macros

- `#define HTPROTO_A0 0x42`
- `#define HTPROTO_A1 0x44`
- `#define HTPROTO_A2 0x46`
- `#define HTPROTO_A3 0x48`
- `#define HTPROTO_A4 0x4A`

### 5.192.1 Detailed Description

Constants that are for use with reading the HiTechnic Prototype board analog input values.

### 5.192.2 Macro Definition Documentation

#### 5.192.2.1 `#define HTPROTO_A0 0x42`

Read Prototype board analog input 0

#### 5.192.2.2 `#define HTPROTO_A1 0x44`

Read Prototype board analog input 1

#### 5.192.2.3 `#define HTPROTO_A2 0x46`

Read Prototype board analog input 2

#### 5.192.2.4 `#define HTPROTO_A3 0x48`

Read Prototype board analog input 3

#### 5.192.2.5 `#define HTPROTO_A4 0x4A`

Read Prototype board analog input 4

## 5.193 HiTechnic SuperPro constants

Constants that are for use with the HiTechnic SuperPro board.

### Modules

- [HiTechnic SuperPro analog input index constants](#)

*Constants that are for use with reading the HiTechnic SuperPro analog input values.*

- [HiTechnic SuperPro analog output index constants](#)

*Constants that are for use with configuraing the HiTechnic SuperPro analog outputs.*

- [SuperPro LED control constants](#)

*Constants for controlling the 2 onboard LEDs.*

- [SuperPro Strobe control constants](#)

*Constants for manipulating the six digital strobe outputs.*

- [SuperPro analog output mode constants](#)

*Constants for controlling the 2 analog output modes.*

- [SuperPro digital pin constants](#)

*Constants for controlling the 8 digital pins.*

### Macros

- `#define HTSPRO_REG_CTRL 0x40`
- `#define HTSPRO_REG_A0 0x42`
- `#define HTSPRO_REG_A1 0x44`
- `#define HTSPRO_REG_A2 0x46`
- `#define HTSPRO_REG_A3 0x48`
- `#define HTSPRO_REG_DIN 0x4C`
- `#define HTSPRO_REG_DOUT 0x4D`
- `#define HTSPRO_REG_DCTRL 0x4E`
- `#define HTSPRO_REG_STROBE 0x50`
- `#define HTSPRO_REG_LED 0x51`
- `#define HTSPRO_REG_DAC0_MODE 0x52`
- `#define HTSPRO_REG_DAC0_FREQ 0x53`
- `#define HTSPRO_REG_DAC0_VOLTAGE 0x55`
- `#define HTSPRO_REG_DAC1_MODE 0x57`
- `#define HTSPRO_REG_DAC1_FREQ 0x58`
- `#define HTSPRO_REG_DAC1_VOLTAGE 0x5A`
- `#define HTSPRO_REG_DLADDRESS 0x60`
- `#define HTSPRO_REG_DLDATA 0x62`
- `#define HTSPRO_REG_DLCKSUM 0x6A`
- `#define HTSPRO_REG_DLCONTROL 0x6B`
- `#define HTSPRO_REG_MEMORY_20 0x80`
- `#define HTSPRO_REG_MEMORY_21 0x84`
- `#define HTSPRO_REG_MEMORY_22 0x88`
- `#define HTSPRO_REG_MEMORY_23 0x8C`
- `#define HTSPRO_REG_MEMORY_24 0x90`
- `#define HTSPRO_REG_MEMORY_25 0x94`
- `#define HTSPRO_REG_MEMORY_26 0x98`
- `#define HTSPRO_REG_MEMORY_27 0x9C`

- #define HTSPRO\_REG\_MEMORY\_28 0xA0
- #define HTSPRO\_REG\_MEMORY\_29 0xA4
- #define HTSPRO\_REG\_MEMORY\_2A 0xA8
- #define HTSPRO\_REG\_MEMORY\_2B 0xAC
- #define HTSPRO\_REG\_MEMORY\_2C 0xB0
- #define HTSPRO\_REG\_MEMORY\_2D 0xB4
- #define HTSPRO\_REG\_MEMORY\_2E 0xB8
- #define HTSPRO\_REG\_MEMORY\_2F 0xBC
- #define HTSPRO\_REG\_MEMORY\_30 0xC0
- #define HTSPRO\_REG\_MEMORY\_31 0xC4
- #define HTSPRO\_REG\_MEMORY\_32 0xC8
- #define HTSPRO\_REG\_MEMORY\_33 0xCC
- #define HTSPRO\_REG\_MEMORY\_34 0xD0
- #define HTSPRO\_REG\_MEMORY\_35 0xD4
- #define HTSPRO\_REG\_MEMORY\_36 0xD8
- #define HTSPRO\_REG\_MEMORY\_37 0xDC
- #define HTSPRO\_REG\_MEMORY\_38 0xE0
- #define HTSPRO\_REG\_MEMORY\_39 0xE4
- #define HTSPRO\_REG\_MEMORY\_3A 0xE8
- #define HTSPRO\_REG\_MEMORY\_3B 0xEC
- #define HTSPRO\_REG\_MEMORY\_3C 0xF0
- #define HTSPRO\_REG\_MEMORY\_3D 0xF4
- #define HTSPRO\_REG\_MEMORY\_3E 0xF8
- #define HTSPRO\_REG\_MEMORY\_3F 0xFC

### 5.193.1 Detailed Description

Constants that are for use with the HiTechnic SuperPro board.

### 5.193.2 Macro Definition Documentation

#### 5.193.2.1 #define HTSPRO\_REG\_A0 0x42

SuperPro analog 0 register (10 bits)

#### 5.193.2.2 #define HTSPRO\_REG\_A1 0x44

SuperPro analog 1 register (10 bits)

#### 5.193.2.3 #define HTSPRO\_REG\_A2 0x46

SuperPro analog 2 register (10 bits)

#### 5.193.2.4 #define HTSPRO\_REG\_A3 0x48

SuperPro analog 3 register (10 bits)

#### 5.193.2.5 #define HTSPRO\_REG\_CTRL 0x40

SuperPro program control register

5.193.2.6 #define HTSPRO\_REG\_DAC0\_FREQ 0x53

SuperPro analog output 0 frequency register (2 bytes msb/lsb)

5.193.2.7 #define HTSPRO\_REG\_DAC0\_MODE 0x52

SuperPro analog output 0 mode register

5.193.2.8 #define HTSPRO\_REG\_DAC0\_VOLTAGE 0x55

SuperPro analog output 0 voltage register (10 bits)

5.193.2.9 #define HTSPRO\_REG\_DAC1\_FREQ 0x58

SuperPro analog output 1 frequency register (2 bytes msb/lsb)

5.193.2.10 #define HTSPRO\_REG\_DAC1\_MODE 0x57

SuperPro analog output 1 mode register

5.193.2.11 #define HTSPRO\_REG\_DAC1\_VOLTAGE 0x5A

SuperPro analog output 1 voltage register (10 bits)

5.193.2.12 #define HTSPRO\_REG\_DCTRL 0x4E

SuperPro digital pin control register (8 bits)

5.193.2.13 #define HTSPRO\_REG\_DIN 0x4C

SuperPro digital pin input register (8 bits)

5.193.2.14 #define HTSPRO\_REG\_DLADDRESS 0x60

SuperPro download address register (2 bytes msb/lsb)

5.193.2.15 #define HTSPRO\_REG\_DLCHKSUM 0x6A

SuperPro download checksum register

5.193.2.16 #define HTSPRO\_REG\_DLCONTROL 0x6B

SuperPro download control register

5.193.2.17 #define HTSPRO\_REG\_DLDATA 0x62

SuperPro download data register (8 bytes)

5.193.2.18 #define HTSPRO\_REG\_DOUT 0x4D

SuperPro digital pin output register (8 bits)

5.193.2.19 #define HTSPRO\_REG\_LED 0x51

SuperPro LED control register

5.193.2.20 #define HTSPRO\_REG\_MEMORY\_20 0x80

SuperPro memory address 0x20 register (4 bytes msb/lsb)

5.193.2.21 #define HTSPRO\_REG\_MEMORY\_21 0x84

SuperPro memory address 0x21 register (4 bytes msb/lsb)

5.193.2.22 #define HTSPRO\_REG\_MEMORY\_22 0x88

SuperPro memory address 0x22 register (4 bytes msb/lsb)

5.193.2.23 #define HTSPRO\_REG\_MEMORY\_23 0x8C

SuperPro memory address 0x23 register (4 bytes msb/lsb)

5.193.2.24 #define HTSPRO\_REG\_MEMORY\_24 0x90

SuperPro memory address 0x24 register (4 bytes msb/lsb)

5.193.2.25 #define HTSPRO\_REG\_MEMORY\_25 0x94

SuperPro memory address 0x25 register (4 bytes msb/lsb)

5.193.2.26 #define HTSPRO\_REG\_MEMORY\_26 0x98

SuperPro memory address 0x26 register (4 bytes msb/lsb)

5.193.2.27 #define HTSPRO\_REG\_MEMORY\_27 0x9C

SuperPro memory address 0x27 register (4 bytes msb/lsb)

5.193.2.28 #define HTSPRO\_REG\_MEMORY\_28 0xA0

SuperPro memory address 0x28 register (4 bytes msb/lsb)

5.193.2.29 #define HTSPRO\_REG\_MEMORY\_29 0xA4

SuperPro memory address 0x29 register (4 bytes msb/lsb)

5.193.2.30 #define HTSPRO\_REG\_MEMORY\_2A 0xA8

SuperPro memory address 0x2A register (4 bytes msb/lsb)

5.193.2.31 #define HTSPRO\_REG\_MEMORY\_2B 0xAC

SuperPro memory address 0x2B register (4 bytes msb/lsb)

5.193.2.32 #define HTSPRO\_REG\_MEMORY\_2C 0xB0

SuperPro memory address 0x2C register (4 bytes msb/lsb)

5.193.2.33 #define HTSPRO\_REG\_MEMORY\_2D 0xB4

SuperPro memory address 0x2D register (4 bytes msb/lsb)

5.193.2.34 #define HTSPRO\_REG\_MEMORY\_2E 0xB8

SuperPro memory address 0x2E register (4 bytes msb/lsb)

5.193.2.35 #define HTSPRO\_REG\_MEMORY\_2F 0xBC

SuperPro memory address 0x2F register (4 bytes msb/lsb)

5.193.2.36 #define HTSPRO\_REG\_MEMORY\_30 0xC0

SuperPro memory address 0x30 register (4 bytes msb/lsb)

5.193.2.37 #define HTSPRO\_REG\_MEMORY\_31 0xC4

SuperPro memory address 0x31 register (4 bytes msb/lsb)

5.193.2.38 #define HTSPRO\_REG\_MEMORY\_32 0xC8

SuperPro memory address 0x32 register (4 bytes msb/lsb)

5.193.2.39 #define HTSPRO\_REG\_MEMORY\_33 0xCC

SuperPro memory address 0x33 register (4 bytes msb/lsb)

5.193.2.40 #define HTSPRO\_REG\_MEMORY\_34 0xD0

SuperPro memory address 0x34 register (4 bytes msb/lsb)

5.193.2.41 #define HTSPRO\_REG\_MEMORY\_35 0xD4

SuperPro memory address 0x35 register (4 bytes msb/lsb)

5.193.2.42 #define HTSPRO\_REG\_MEMORY\_36 0xD8

SuperPro memory address 0x36 register (4 bytes msb/lsb)

5.193.2.43 #define HTSPRO\_REG\_MEMORY\_37 0xDC

SuperPro memory address 0x37 register (4 bytes msb/lsb)

5.193.2.44 #define HTSPRO\_REG\_MEMORY\_38 0xE0

SuperPro memory address 0x38 register (4 bytes msb/lsb)

5.193.2.45 #define HTSPRO\_REG\_MEMORY\_39 0xE4

SuperPro memory address 0x39 register (4 bytes msb/lsb)

5.193.2.46 #define HTSPRO\_REG\_MEMORY\_3A 0xE8

SuperPro memory address 0x3A register (4 bytes msb/lsb)

5.193.2.47 #define HTSPRO\_REG\_MEMORY\_3B 0xEC

SuperPro memory address 0x3B register (4 bytes msb/lsb)

5.193.2.48 #define HTSPRO\_REG\_MEMORY\_3C 0xF0

SuperPro memory address 0x3C register (4 bytes msb/lsb)

5.193.2.49 #define HTSPRO\_REG\_MEMORY\_3D 0xF4

SuperPro memory address 0x3D register (4 bytes msb/lsb)

5.193.2.50 #define HTSPRO\_REG\_MEMORY\_3E 0xF8

SuperPro memory address 0x3E register (4 bytes msb/lsb)

5.193.2.51 #define HTSPRO\_REG\_MEMORY\_3F 0xFC

SuperPro memory address 0x3F register (4 bytes msb/lsb)

5.193.2.52 #define HTSPRO\_REG\_STROBE 0x50

SuperPro strobe control register

## 5.194 HiTechnic SuperPro analog input index constants

Constants that are for use with reading the HiTechnic SuperPro analog input values.

### Macros

- #define HTSPRO\_A0 0x42
- #define HTSPRO\_A1 0x44
- #define HTSPRO\_A2 0x46
- #define HTSPRO\_A3 0x48

#### 5.194.1 Detailed Description

Constants that are for use with reading the HiTechnic SuperPro analog input values.

#### 5.194.2 Macro Definition Documentation

##### 5.194.2.1 #define HTSPRO\_A0 0x42

Read SuperPro analog input 0

##### 5.194.2.2 #define HTSPRO\_A1 0x44

Read SuperPro analog input 1

##### 5.194.2.3 #define HTSPRO\_A2 0x46

Read SuperPro analog input 2

##### 5.194.2.4 #define HTSPRO\_A3 0x48

Read SuperPro analog input 3

## 5.195 HiTechnic SuperPro analog output index constants

Constants that are for use with configuraing the HiTechnic SuperPro analog outputs.

### Macros

- #define HTSPRO\_DAC0 0x52
- #define HTSPRO\_DAC1 0x57

#### 5.195.1 Detailed Description

Constants that are for use with configuraing the HiTechnic SuperPro analog outputs.

#### 5.195.2 Macro Definition Documentation

##### 5.195.2.1 #define HTSPRO\_DAC0 0x52

Set SuperPro analog output 0 configuration

##### 5.195.2.2 #define HTSPRO\_DAC1 0x57

Set SuperPro analog output 1 configuration

## 5.196 HiTechnic PIR sensor constants

Constants that are for use with the HiTechnic PIR sensor device.

### Macros

- `#define HTPIR_REG_DEADBAND 0x41`
- `#define HTPIR_REG_READING 0x42`

### 5.196.1 Detailed Description

Constants that are for use with the HiTechnic PIR sensor device.

### 5.196.2 Macro Definition Documentation

#### 5.196.2.1 `#define HTPIR_REG_DEADBAND 0x41`

PIR sensor deadband register

#### 5.196.2.2 `#define HTPIR_REG_READING 0x42`

PIR sensor value register (signed byte)

## 5.197 MindSensors device constants

Constants that are for use with MindSensors devices.

### Modules

- [MindSensors ACCL-Nx constants](#)

*Constants that are for use with the MindSensors ACCL-Nx device.*

- [MindSensors DIST-Nx constants](#)

*Constants that are for use with the MindSensors DIST-Nx device.*

- [MindSensors NXTHID constants](#)

*Constants that are for use with the MindSensors NXTHID device.*

- [MindSensors NXTLineLeader constants](#)

*Constants that are for use with the MindSensors NXTLineLeader device.*

- [MindSensors NXTNumericPad constants](#)

*Constants that are for use with the MindSensors NXTNumericPad device.*

- [MindSensors NXTPowerMeter constants](#)

*Constants that are for use with the MindSensors NXTPowerMeter device.*

- [MindSensors NXTServo constants](#)

*Constants that are for use with the MindSensors NXTServo device.*

- [MindSensors NXTSumoEyes constants](#)

*Constants that are for use with the MindSensors NXTSumoEyes device.*

- [MindSensors NXTTouchPanel constants](#)

*Constants that are for use with the MindSensors NXTTouchPanel device.*

- [MindSensors PFMate constants](#)

*Constants that are for use with the MindSensors PFMate device.*

- [MindSensors PSP-Nx constants](#)

*Constants that are for use with the MindSensors PSP-Nx device.*

- [MindSensors nRLink constants](#)

*Constants that are for use with the MindSensors nRLink device.*

### Macros

- #define MS\_CMD\_ENERGIZED 0x45
- #define MS\_CMD\_DEENERGIZED 0x44
- #define MS\_CMD\_ADPA\_ON 0x4E
- #define MS\_CMD\_ADPA\_OFF 0x4F
- #define MS\_ADDR\_RTCLOCK 0xD0
- #define MS\_ADDR\_DISTNX 0x02
- #define MS\_ADDR\_NRLINK 0x02
- #define MS\_ADDR\_ACCLNX 0x02
- #define MS\_ADDR\_CMPSNX 0x02
- #define MS\_ADDR\_PSPNX 0x02
- #define MS\_ADDR\_LINEldr 0x02
- #define MS\_ADDR\_NXTCAM 0x02
- #define MS\_ADDR\_NXTHID 0x04
- #define MS\_ADDR\_NXTSERVO 0xB0
- #define MS\_ADDR\_NXTSERVO\_EM 0x40

- #define MS\_ADDR\_PFMATE 0x48
- #define MS\_ADDR\_MTRMUX 0xB4
- #define MS\_ADDR\_NXTMMX 0x06
- #define MS\_ADDR\_IVSENS 0x12
- #define MS\_ADDR\_RXMUX 0x7E
- #define MS\_ADDR\_NUMERICPAD 0xB4
- #define MS\_ADDR\_TOUCHPANEL 0x04

### 5.197.1 Detailed Description

Constants that are for use with MindSensors devices.

### 5.197.2 Macro Definition Documentation

#### 5.197.2.1 #define MS\_ADDR\_ACCLNX 0x02

MindSensors ACCL-Nx I2C address

#### 5.197.2.2 #define MS\_ADDR\_CMPSNX 0x02

MindSensors CMPS-Nx I2C address

#### 5.197.2.3 #define MS\_ADDR\_DISTNX 0x02

MindSensors DIST-Nx I2C address

#### 5.197.2.4 #define MS\_ADDR\_IVSENS 0x12

MindSensors IVSens (NXTPowerMeter) I2C address

#### 5.197.2.5 #define MS\_ADDR\_LINELDR 0x02

MindSensors LineLdr I2C address

#### 5.197.2.6 #define MS\_ADDR\_MTRMUX 0xB4

MindSensors MTRMux I2C address

#### 5.197.2.7 #define MS\_ADDR\_NRLINK 0x02

MindSensors NRLink I2C address

#### 5.197.2.8 #define MS\_ADDR\_NUMERICPAD 0xB4

MindSensors NumericPad I2C address

#### 5.197.2.9 #define MS\_ADDR\_NXTCAM 0x02

MindSensors NXTCam I2C address

#### 5.197.2.10 #define MS\_ADDR\_NXTHID 0x04

MindSensors NXTHID I2C address

5.197.2.11 #define MS\_ADDR\_NXTMMX 0x06

MindSensors NXTMMX I2C address

5.197.2.12 #define MS\_ADDR\_NXTSERVO 0xB0

MindSensors NXTServo I2C address

5.197.2.13 #define MS\_ADDR\_NXTSERVO\_EM 0x40

MindSensors NXTServo in edit macro mode I2C address

5.197.2.14 #define MS\_ADDR\_PFMATE 0x48

MindSensors PFMate I2C address

5.197.2.15 #define MS\_ADDR\_PSPNX 0x02

MindSensors PSP-Nx I2C address

5.197.2.16 #define MS\_ADDR\_RTCLOCK 0xD0

MindSensors RTClock I2C address

5.197.2.17 #define MS\_ADDR\_RXMUX 0x7E

MindSensors RXMux I2C address

5.197.2.18 #define MS\_ADDR\_TOUCHPANEL 0x04

MindSensors TouchPanel I2C address

5.197.2.19 #define MS\_CMD\_ADPA\_OFF 0x4F

Turn MindSensors ADPA mode off

5.197.2.20 #define MS\_CMD\_ADPA\_ON 0x4E

Turn MindSensors ADPA mode on

5.197.2.21 #define MS\_CMD\_DEENERGIZED 0x44

De-energize the MindSensors device

5.197.2.22 #define MS\_CMD\_ENERGIZED 0x45

Energize the MindSensors device

## 5.198 MindSensors DIST-Nx constants

Constants that are for use with the MindSensors DIST-Nx device.

### Macros

- #define DIST\_CMD\_GP2D12 0x31
- #define DIST\_CMD\_GP2D120 0x32
- #define DIST\_CMD\_GP2YA21 0x33
- #define DIST\_CMD\_GP2YA02 0x34
- #define DIST\_CMD\_CUSTOM 0x35
- #define DIST\_REG\_DIST 0x42
- #define DIST\_REG\_VOLT 0x44
- #define DIST\_REG\_MODULE\_TYPE 0x50
- #define DIST\_REG\_NUM\_POINTS 0x51
- #define DIST\_REG\_DIST\_MIN 0x52
- #define DIST\_REG\_DIST\_MAX 0x54
- #define DIST\_REG\_VOLT1 0x56
- #define DIST\_REG\_DIST1 0x58

### 5.198.1 Detailed Description

Constants that are for use with the MindSensors DIST-Nx device.

### 5.198.2 Macro Definition Documentation

#### 5.198.2.1 #define DIST\_CMD\_CUSTOM 0x35

Set the DIST-Nx to a custom mode

#### 5.198.2.2 #define DIST\_CMD\_GP2D12 0x31

Set the DIST-Nx to GP2D12 mode

#### 5.198.2.3 #define DIST\_CMD\_GP2D120 0x32

Set the DIST-Nx to GP2D120 mode

#### 5.198.2.4 #define DIST\_CMD\_GP2YA02 0x34

Set the DIST-Nx to GP2YA02 mode

#### 5.198.2.5 #define DIST\_CMD\_GP2YA21 0x33

Set the DIST-Nx to GP2YA21 mode

#### 5.198.2.6 #define DIST\_REG\_DIST 0x42

The DIST-Nx distance register

#### 5.198.2.7 #define DIST\_REG\_DIST1 0x58

The DIST-Nx distance 1 register

5.198.2.8 #define DIST\_REG\_DIST\_MAX 0x54

The DIST-Nx maximum distance register

5.198.2.9 #define DIST\_REG\_DIST\_MIN 0x52

The DIST-Nx minimum distance register

5.198.2.10 #define DIST\_REG\_MODULE\_TYPE 0x50

The DIST-Nx module type register

5.198.2.11 #define DIST\_REG\_NUM\_POINTS 0x51

The DIST-Nx number of data points in Custom curve register

5.198.2.12 #define DIST\_REG\_VOLT 0x44

The DIST-Nx voltage register

5.198.2.13 #define DIST\_REG\_VOLT1 0x56

The DIST-Nx voltage 1 register

## 5.199 MindSensors PSP-Nx constants

Constants that are for use with the MindSensors PSP-Nx device.

### Modules

- [MindSensors PSP-Nx button set 1 constants](#)

*Constants that are for interpreting MindSensors PSP-Nx button set 1 values.*

- [MindSensors PSP-Nx button set 2 constants](#)

*Constants that are for interpreting MindSensors PSP-Nx button set 2 values.*

### Macros

- `#define PSP_CMD_DIGITAL 0x41`
- `#define PSP_CMD_ANALOG 0x73`
- `#define PSP_REG_BTNSET1 0x42`
- `#define PSP_REG_BTNSET2 0x43`
- `#define PSP_REG_XLEFT 0x44`
- `#define PSP_REG_YLEFT 0x45`
- `#define PSP_REG_XRIGHT 0x46`
- `#define PSP_REG_YRIGHT 0x47`

### 5.199.1 Detailed Description

Constants that are for use with the MindSensors PSP-Nx device.

### 5.199.2 Macro Definition Documentation

#### 5.199.2.1 `#define PSP_CMD_ANALOG 0x73`

Set the PSP-Nx to analog mode

#### 5.199.2.2 `#define PSP_CMD_DIGITAL 0x41`

Set the PSP-Nx to digital mode

#### 5.199.2.3 `#define PSP_REG_BTNSET1 0x42`

The PSP-Nx button set 1 register

#### 5.199.2.4 `#define PSP_REG_BTNSET2 0x43`

The PSP-Nx button set 2 register

#### 5.199.2.5 `#define PSP_REG_XLEFT 0x44`

The PSP-Nx X left register

#### 5.199.2.6 `#define PSP_REG_XRIGHT 0x46`

The PSP-Nx X right register

5.199.2.7 #define PSP\_REG\_YLEFT 0x45

The PSP-Nx Y left register

5.199.2.8 #define PSP\_REG\_YRIGHT 0x47

The PSP-Nx Y right register

## 5.200 MindSensors PSP-Nx button set 1 constants

Constants that are for interpreting MindSensors PSP-Nx button set 1 values.

### Macros

- #define PSP\_BTNSET1\_LEFT 0x80
- #define PSP\_BTNSET1\_DOWN 0x40
- #define PSP\_BTNSET1\_RIGHT 0x20
- #define PSP\_BTNSET1\_UP 0x10
- #define PSP\_BTNSET1\_START 0x08
- #define PSP\_BTNSET1\_R3 0x04
- #define PSP\_BTNSET1\_L3 0x02
- #define PSP\_BTNSET1\_SELECT 0x01

### 5.200.1 Detailed Description

Constants that are for interpreting MindSensors PSP-Nx button set 1 values.

### 5.200.2 Macro Definition Documentation

#### 5.200.2.1 #define PSP\_BTNSET1\_DOWN 0x40

The PSP-Nx button set 1 down arrow

#### 5.200.2.2 #define PSP\_BTNSET1\_L3 0x02

The PSP-Nx button set 1 L3

#### 5.200.2.3 #define PSP\_BTNSET1\_LEFT 0x80

The PSP-Nx button set 1 left arrow

#### 5.200.2.4 #define PSP\_BTNSET1\_R3 0x04

The PSP-Nx button set 1 R3

#### 5.200.2.5 #define PSP\_BTNSET1\_RIGHT 0x20

The PSP-Nx button set 1 right arrow

#### 5.200.2.6 #define PSP\_BTNSET1\_SELECT 0x01

The PSP-Nx button set 1 select

#### 5.200.2.7 #define PSP\_BTNSET1\_START 0x08

The PSP-Nx button set 1 start

#### 5.200.2.8 #define PSP\_BTNSET1\_UP 0x10

The PSP-Nx button set 1 up arrow

## 5.201 MindSensors PSP-Nx button set 2 constants

Constants that are for interpreting MindSensors PSP-Nx button set 2 values.

### Macros

- `#define PSP_BTNSET2_SQUARE 0x80`
- `#define PSP_BTNSET2_CROSS 0x40`
- `#define PSP_BTNSET2_CIRCLE 0x20`
- `#define PSP_BTNSET2_TRIANGLE 0x10`
- `#define PSP_BTNSET2_R1 0x08`
- `#define PSP_BTNSET2_L1 0x04`
- `#define PSP_BTNSET2_R2 0x02`
- `#define PSP_BTNSET2_L2 0x01`

### 5.201.1 Detailed Description

Constants that are for interpreting MindSensors PSP-Nx button set 2 values.

### 5.201.2 Macro Definition Documentation

#### 5.201.2.1 `#define PSP_BTNSET2_CIRCLE 0x20`

The PSP-Nx button set 2 circle

#### 5.201.2.2 `#define PSP_BTNSET2_CROSS 0x40`

The PSP-Nx button set 2 cross

#### 5.201.2.3 `#define PSP_BTNSET2_L1 0x04`

The PSP-Nx button set 2 L1

#### 5.201.2.4 `#define PSP_BTNSET2_L2 0x01`

The PSP-Nx button set 2 L2

#### 5.201.2.5 `#define PSP_BTNSET2_R1 0x08`

The PSP-Nx button set 2 R1

#### 5.201.2.6 `#define PSP_BTNSET2_R2 0x02`

The PSP-Nx button set 2 R2

#### 5.201.2.7 `#define PSP_BTNSET2_SQUARE 0x80`

The PSP-Nx button set 2 square

#### 5.201.2.8 `#define PSP_BTNSET2_TRIANGLE 0x10`

The PSP-Nx button set 2 triangle

## 5.202 MindSensors nRLink constants

Constants that are for use with the MindSensors nRLink device.

### Macros

- `#define NRLINK_CMD_2400 0x44`
- `#define NRLINK_CMD_FLUSH 0x46`
- `#define NRLINK_CMD_4800 0x48`
- `#define NRLINK_CMD_IR_LONG 0x4C`
- `#define NRLINK_CMD_IR_SHORT 0x53`
- `#define NRLINK_CMD_RUN_MACRO 0x52`
- `#define NRLINK_CMD_TX_RAW 0x55`
- `#define NRLINK_CMD_SET_RCX 0x58`
- `#define NRLINK_CMD_SET_TRAIN 0x54`
- `#define NRLINK_CMD_SET_PF 0x50`
- `#define NRLINK_REG_BYTES 0x40`
- `#define NRLINK_REG_DATA 0x42`
- `#define NRLINK_REG_EEPROM 0x50`

### 5.202.1 Detailed Description

Constants that are for use with the MindSensors nRLink device.

### 5.202.2 Macro Definition Documentation

#### 5.202.2.1 `#define NRLINK_CMD_2400 0x44`

Set NRLink to 2400 baud

#### 5.202.2.2 `#define NRLINK_CMD_4800 0x48`

Set NRLink to 4800 baud

#### 5.202.2.3 `#define NRLINK_CMD_FLUSH 0x46`

Flush the NRLink

#### 5.202.2.4 `#define NRLINK_CMD_IR_LONG 0x4C`

Set the NRLink to long range IR

#### 5.202.2.5 `#define NRLINK_CMD_IR_SHORT 0x53`

Set the NRLink to short range IR

#### 5.202.2.6 `#define NRLINK_CMD_RUN_MACRO 0x52`

Run an NRLink macro

#### 5.202.2.7 `#define NRLINK_CMD_SET_PF 0x50`

Set the NRLink to Power Function mode

5.202.2.8 #define NRLINK\_CMD\_SET\_RCX 0x58

Set the NRLink to RCX mode

5.202.2.9 #define NRLINK\_CMD\_SET\_TRAIN 0x54

Set the NRLink to IR Train mode

5.202.2.10 #define NRLINK\_CMD\_TX\_RAW 0x55

Set the NRLink to transmit raw bytes

5.202.2.11 #define NRLINK\_REG\_BYTES 0x40

The NRLink bytes register

5.202.2.12 #define NRLINK\_REG\_DATA 0x42

The NRLink data register

5.202.2.13 #define NRLINK\_REG\_EEPROM 0x50

The NRLink eeprom register

## 5.203 MindSensors ACCL-Nx constants

Constants that are for use with the MindSensors ACCL-Nx device.

### Modules

- [MindSensors ACCL-Nx sensitivity level constants](#)

*Constants that are for setting the MindSensors ACCL-Nx sensitivity level.*

### Macros

- `#define ACCL_CMD_X_CAL 0x58`
- `#define ACCL_CMD_Y_CAL 0x59`
- `#define ACCL_CMD_Z_CAL 0x5a`
- `#define ACCL_CMD_X_CAL_END 0x78`
- `#define ACCL_CMD_Y_CAL_END 0x79`
- `#define ACCL_CMD_Z_CAL_END 0x7a`
- `#define ACCL_CMD_RESET_CAL 0x52`
- `#define ACCL_REG_SENS_LVL 0x19`
- `#define ACCL_REG_X_TILT 0x42`
- `#define ACCL_REG_Y_TILT 0x43`
- `#define ACCL_REG_Z_TILT 0x44`
- `#define ACCL_REG_X_ACCEL 0x45`
- `#define ACCL_REG_Y_ACCEL 0x47`
- `#define ACCL_REG_Z_ACCEL 0x49`
- `#define ACCL_REG_X_OFFSET 0x4b`
- `#define ACCL_REG_X_RANGE 0x4d`
- `#define ACCL_REG_Y_OFFSET 0x4f`
- `#define ACCL_REG_Y_RANGE 0x51`
- `#define ACCL_REG_Z_OFFSET 0x53`
- `#define ACCL_REG_Z_RANGE 0x55`

### 5.203.1 Detailed Description

Constants that are for use with the MindSensors ACCL-Nx device.

### 5.203.2 Macro Definition Documentation

#### 5.203.2.1 `#define ACCL_CMD_RESET_CAL 0x52`

Reset to factory calibration

#### 5.203.2.2 `#define ACCL_CMD_X_CAL 0x58`

Acquire X-axis calibration point

#### 5.203.2.3 `#define ACCL_CMD_X_CAL_END 0x78`

Acquire X-axis calibration point and end calibration

5.203.2.4 #define ACCL\_CMD\_Y\_CAL 0x59

Acquire Y-axis calibration point

5.203.2.5 #define ACCL\_CMD\_Y\_CAL\_END 0x79

Acquire Y-axis calibration point and end calibration

5.203.2.6 #define ACCL\_CMD\_Z\_CAL 0x5a

Acquire Z-axis calibration point

5.203.2.7 #define ACCL\_CMD\_Z\_CAL\_END 0x7a

Acquire Z-axis calibration point and end calibration

5.203.2.8 #define ACCL\_REG\_SENS\_LVL 0x19

The current sensitivity

5.203.2.9 #define ACCL\_REG\_X\_ACCEL 0x45

The X-axis acceleration data

5.203.2.10 #define ACCL\_REG\_X\_OFFSET 0x4b

The X-axis offset

5.203.2.11 #define ACCL\_REG\_X\_RANGE 0x4d

The X-axis range

5.203.2.12 #define ACCL\_REG\_X\_TILT 0x42

The X-axis tilt data

5.203.2.13 #define ACCL\_REG\_Y\_ACCEL 0x47

The Y-axis acceleration data

5.203.2.14 #define ACCL\_REG\_Y\_OFFSET 0x4f

The Y-axis offset

5.203.2.15 #define ACCL\_REG\_Y\_RANGE 0x51

The Y-axis range

5.203.2.16 #define ACCL\_REG\_Y\_TILT 0x43

The Y-axis tilt data

5.203.2.17 #define ACCL\_REG\_Z\_ACCEL 0x49

The Z-axis acceleration data

5.203.2.18 #define ACCL\_REG\_Z\_OFFSET 0x53

The Z-axis offset

5.203.2.19 #define ACCL\_REG\_Z\_RANGE 0x55

The Z-axis range

5.203.2.20 #define ACCL\_REG\_Z\_TILT 0x44

The Z-axis tilt data

## 5.204 MindSensors ACCL-Nx sensitivity level constants

Constants that are for setting the MindSensors ACCL-Nx sensitivity level.

### Macros

- #define ACCL\_SENSITIVITY\_LEVEL\_1 0x31
- #define ACCL\_SENSITIVITY\_LEVEL\_2 0x32
- #define ACCL\_SENSITIVITY\_LEVEL\_3 0x33
- #define ACCL\_SENSITIVITY\_LEVEL\_4 0x34

### 5.204.1 Detailed Description

Constants that are for setting the MindSensors ACCL-Nx sensitivity level.

### 5.204.2 Macro Definition Documentation

#### 5.204.2.1 #define ACCL\_SENSITIVITY\_LEVEL\_1 0x31

The ACCL-Nx sensitivity level 1

#### 5.204.2.2 #define ACCL\_SENSITIVITY\_LEVEL\_2 0x32

The ACCL-Nx sensitivity level 2

#### 5.204.2.3 #define ACCL\_SENSITIVITY\_LEVEL\_3 0x33

The ACCL-Nx sensitivity level 3

#### 5.204.2.4 #define ACCL\_SENSITIVITY\_LEVEL\_4 0x34

The ACCL-Nx sensitivity level 4

## 5.205 MindSensors PFMate constants

Constants that are for use with the MindSensors PFMate device.

### Modules

- [PFMate channel constants](#)  
*Constants that are for specifying PFMate channels.*
- [PFMate motor constants](#)  
*Constants that are for specifying PFMate motors.*

### Macros

- `#define PFMATE_REG_CMD 0x41`
- `#define PFMATE_REG_CHANNEL 0x42`
- `#define PFMATE_REG_MOTORS 0x43`
- `#define PFMATE_REG_A_CMD 0x44`
- `#define PFMATE_REG_A_SPEED 0x45`
- `#define PFMATE_REG_B_CMD 0x46`
- `#define PFMATE_REG_B_SPEED 0x47`
- `#define PFMATE_CMD_GO 0x47`
- `#define PFMATE_CMD_RAW 0x52`

### 5.205.1 Detailed Description

Constants that are for use with the MindSensors PFMate device.

### 5.205.2 Macro Definition Documentation

#### 5.205.2.1 `#define PFMATE_CMD_GO 0x47`

Send IR signal to IR receiver

#### 5.205.2.2 `#define PFMATE_CMD_RAW 0x52`

Send raw IR signal to IR receiver

#### 5.205.2.3 `#define PFMATE_REG_A_CMD 0x44`

PF command for motor A? (PF\_CMD\_FLOAT, PF\_CMD\_FWD, PF\_CMD\_REV, PF\_CMD\_BRAKE)

#### 5.205.2.4 `#define PFMATE_REG_A_SPEED 0x45`

PF speed for motor A? (0-7)

#### 5.205.2.5 `#define PFMATE_REG_B_CMD 0x46`

PF command for motor B? (PF\_CMD\_FLOAT, PF\_CMD\_FWD, PF\_CMD\_REV, PF\_CMD\_BRAKE)

#### 5.205.2.6 `#define PFMATE_REG_B_SPEED 0x47`

PF speed for motor B? (0-7)

5.205.2.7 #define PFMATE\_REG\_CHANNEL 0x42

PF channel? 1, 2, 3, or 4

5.205.2.8 #define PFMATE\_REG\_CMD 0x41

PFMate command

5.205.2.9 #define PFMATE\_REG\_MOTORS 0x43

PF motors? (0 = both, 1 = A, 2 = B)

## 5.206 PFMate motor constants

Constants that are for specifying PFMate motors.

### Macros

- `#define PFMATE_MOTORS_BOTH 0x00`
- `#define PFMATE_MOTORS_A 0x01`
- `#define PFMATE_MOTORS_B 0x02`

### 5.206.1 Detailed Description

Constants that are for specifying PFMate motors.

### 5.206.2 Macro Definition Documentation

#### 5.206.2.1 `#define PFMATE_MOTORS_A 0x01`

Control only motor A

#### 5.206.2.2 `#define PFMATE_MOTORS_B 0x02`

Control only motor B

#### 5.206.2.3 `#define PFMATE_MOTORS_BOTH 0x00`

Control both motors

## 5.207 PFMate channel constants

Constants that are for specifying PFMate channels.

### Macros

- #define PFMATE\_CHANNEL\_1 1
- #define PFMATE\_CHANNEL\_2 2
- #define PFMATE\_CHANNEL\_3 3
- #define PFMATE\_CHANNEL\_4 4

### 5.207.1 Detailed Description

Constants that are for specifying PFMate channels.

### 5.207.2 Macro Definition Documentation

#### 5.207.2.1 #define PFMATE\_CHANNEL\_1 1

Power function channel 1

#### 5.207.2.2 #define PFMATE\_CHANNEL\_2 2

Power function channel 2

#### 5.207.2.3 #define PFMATE\_CHANNEL\_3 3

Power function channel 3

#### 5.207.2.4 #define PFMATE\_CHANNEL\_4 4

Power function channel 4

## 5.208 MindSensors NXTServo constants

Constants that are for use with the MindSensors NXTServo device.

### Modules

- [MindSensors NXTServo commands](#)  
*NXTServo device command constants.*
- [MindSensors NXTServo position constants](#)  
*NXTServo device position constants.*
- [MindSensors NXTServo quick position constants](#)  
*NXTServo device quick position constants.*
- [MindSensors NXTServo registers](#)  
*NXTServo device register constants.*
- [MindSensors NXTServo servo numbers](#)  
*NXTServo device servo number constants.*

### 5.208.1 Detailed Description

Constants that are for use with the MindSensors NXTServo device.

## 5.209 MindSensors NXTServo registers

NXTServo device register constants.

### Macros

- #define NXTSERVO\_REG\_VOLTAGE 0x41
- #define NXTSERVO\_REG\_CMD 0x41
- #define NXTSERVO\_REG\_S1\_POS 0x42
- #define NXTSERVO\_REG\_S2\_POS 0x44
- #define NXTSERVO\_REG\_S3\_POS 0x46
- #define NXTSERVO\_REG\_S4\_POS 0x48
- #define NXTSERVO\_REG\_S5\_POS 0x4A
- #define NXTSERVO\_REG\_S6\_POS 0x4C
- #define NXTSERVO\_REG\_S7\_POS 0x4E
- #define NXTSERVO\_REG\_S8\_POS 0x50
- #define NXTSERVO\_REG\_S1\_SPEED 0x52
- #define NXTSERVO\_REG\_S2\_SPEED 0x53
- #define NXTSERVO\_REG\_S3\_SPEED 0x54
- #define NXTSERVO\_REG\_S4\_SPEED 0x55
- #define NXTSERVO\_REG\_S5\_SPEED 0x56
- #define NXTSERVO\_REG\_S6\_SPEED 0x57
- #define NXTSERVO\_REG\_S7\_SPEED 0x58
- #define NXTSERVO\_REG\_S8\_SPEED 0x59
- #define NXTSERVO\_REG\_S1\_QPOS 0x5A
- #define NXTSERVO\_REG\_S2\_QPOS 0x5B
- #define NXTSERVO\_REG\_S3\_QPOS 0x5C
- #define NXTSERVO\_REG\_S4\_QPOS 0x5D
- #define NXTSERVO\_REG\_S5\_QPOS 0x5E
- #define NXTSERVO\_REG\_S6\_QPOS 0x5F
- #define NXTSERVO\_REG\_S7\_QPOS 0x60
- #define NXTSERVO\_REG\_S8\_QPOS 0x61
- #define NXTSERVO\_EM\_REG\_CMD 0x00
- #define NXTSERVO\_EM\_REG\_EEPROM\_START 0x21
- #define NXTSERVO\_EM\_REG\_EEPROM\_END 0xFF

### 5.209.1 Detailed Description

NXTServo device register constants.

### 5.209.2 Macro Definition Documentation

#### 5.209.2.1 #define NXTSERVO\_EM\_REG\_CMD 0x00

NXTServo in macro edit mode command register.

#### 5.209.2.2 #define NXTSERVO\_EM\_REG\_EEPROM\_END 0xFF

NXTServo in macro edit mode EEPROM end register.

5.209.2.3 `#define NXTSERVO_EM_REG_EEPROM_START 0x21`

NXTServo in macro edit mode EEPROM start register.

5.209.2.4 `#define NXTSERVO_REG_CMD 0x41`

NXTServo command register. See [MindSensors NXTServo commands](#) group. (write only)

5.209.2.5 `#define NXTSERVO_REG_S1_POS 0x42`

NXTServo servo 1 position register.

5.209.2.6 `#define NXTSERVO_REG_S1_QPOS 0x5A`

NXTServo servo 1 quick position register. (write only)

5.209.2.7 `#define NXTSERVO_REG_S1_SPEED 0x52`

NXTServo servo 1 speed register.

5.209.2.8 `#define NXTSERVO_REG_S2_POS 0x44`

NXTServo servo 2 position register.

5.209.2.9 `#define NXTSERVO_REG_S2_QPOS 0x5B`

NXTServo servo 2 quick position register. (write only)

5.209.2.10 `#define NXTSERVO_REG_S2_SPEED 0x53`

NXTServo servo 2 speed register.

5.209.2.11 `#define NXTSERVO_REG_S3_POS 0x46`

NXTServo servo 3 position register.

5.209.2.12 `#define NXTSERVO_REG_S3_QPOS 0x5C`

NXTServo servo 3 quick position register. (write only)

5.209.2.13 `#define NXTSERVO_REG_S3_SPEED 0x54`

NXTServo servo 3 speed register.

5.209.2.14 `#define NXTSERVO_REG_S4_POS 0x48`

NXTServo servo 4 position register.

5.209.2.15 `#define NXTSERVO_REG_S4_QPOS 0x5D`

NXTServo servo 4 quick position register. (write only)

5.209.2.16 `#define NXTSERVO_REG_S4_SPEED 0x55`

NXTServo servo 4 speed register.

5.209.2.17 #define NXTSERVO\_REG\_S5\_POS 0x4A

NXTServo servo 5 position register.

5.209.2.18 #define NXTSERVO\_REG\_S5\_QPOS 0x5E

NXTServo servo 5 quick position register. (write only)

5.209.2.19 #define NXTSERVO\_REG\_S5\_SPEED 0x56

NXTServo servo 5 speed register.

5.209.2.20 #define NXTSERVO\_REG\_S6\_POS 0x4C

NXTServo servo 6 position register.

5.209.2.21 #define NXTSERVO\_REG\_S6\_QPOS 0x5F

NXTServo servo 6 quick position register. (write only)

5.209.2.22 #define NXTSERVO\_REG\_S6\_SPEED 0x57

NXTServo servo 6 speed register.

5.209.2.23 #define NXTSERVO\_REG\_S7\_POS 0x4E

NXTServo servo 7 position register.

5.209.2.24 #define NXTSERVO\_REG\_S7\_QPOS 0x60

NXTServo servo 7 quick position register. (write only)

5.209.2.25 #define NXTSERVO\_REG\_S7\_SPEED 0x58

NXTServo servo 7 speed register.

5.209.2.26 #define NXTSERVO\_REG\_S8\_POS 0x50

NXTServo servo 8 position register.

5.209.2.27 #define NXTSERVO\_REG\_S8\_QPOS 0x61

NXTServo servo 8 quick position register. (write only)

5.209.2.28 #define NXTSERVO\_REG\_S8\_SPEED 0x59

NXTServo servo 8 speed register.

5.209.2.29 #define NXTSERVO\_REG\_VOLTAGE 0x41

Battery voltage register. (read only)

## 5.210 MindSensors NXTServo position constants

NXTServo device position constants.

### Macros

- `#define NXTSERVO_POS_CENTER 1500`
- `#define NXTSERVO_POS_MIN 500`
- `#define NXTSERVO_POS_MAX 2500`

#### 5.210.1 Detailed Description

NXTServo device position constants.

#### 5.210.2 Macro Definition Documentation

##### 5.210.2.1 `#define NXTSERVO_POS_CENTER 1500`

Center position for 1500us servos.

##### 5.210.2.2 `#define NXTSERVO_POS_MAX 2500`

Maximum position for 1500us servos.

##### 5.210.2.3 `#define NXTSERVO_POS_MIN 500`

Minimum position for 1500us servos.

## 5.211 MindSensors NXTServo quick position constants

NXTServo device quick position constants.

### Macros

- `#define NXTSERVO_QPOS_CENTER 150`
- `#define NXTSERVO_QPOS_MIN 50`
- `#define NXTSERVO_QPOS_MAX 250`

#### 5.211.1 Detailed Description

NXTServo device quick position constants.

#### 5.211.2 Macro Definition Documentation

##### 5.211.2.1 `#define NXTSERVO_QPOS_CENTER 150`

Center quick position for 1500us servos.

##### 5.211.2.2 `#define NXTSERVO_QPOS_MAX 250`

Maximum quick position for 1500us servos.

##### 5.211.2.3 `#define NXTSERVO_QPOS_MIN 50`

Minimum quick position for 1500us servos.

## 5.212 MindSensors NXTServo servo numbers

NXTServo device servo number constants.

### Macros

- `#define NXTSERVO_SERVO_1 0`
- `#define NXTSERVO_SERVO_2 1`
- `#define NXTSERVO_SERVO_3 2`
- `#define NXTSERVO_SERVO_4 3`
- `#define NXTSERVO_SERVO_5 4`
- `#define NXTSERVO_SERVO_6 5`
- `#define NXTSERVO_SERVO_7 6`
- `#define NXTSERVO_SERVO_8 7`

### 5.212.1 Detailed Description

NXTServo device servo number constants.

### 5.212.2 Macro Definition Documentation

#### 5.212.2.1 `#define NXTSERVO_SERVO_1 0`

NXTServo server number 1.

#### 5.212.2.2 `#define NXTSERVO_SERVO_2 1`

NXTServo server number 2.

#### 5.212.2.3 `#define NXTSERVO_SERVO_3 2`

NXTServo server number 3.

#### 5.212.2.4 `#define NXTSERVO_SERVO_4 3`

NXTServo server number 4.

#### 5.212.2.5 `#define NXTSERVO_SERVO_5 4`

NXTServo server number 5.

#### 5.212.2.6 `#define NXTSERVO_SERVO_6 5`

NXTServo server number 6.

#### 5.212.2.7 `#define NXTSERVO_SERVO_7 6`

NXTServo server number 7.

#### 5.212.2.8 `#define NXTSERVO_SERVO_8 7`

NXTServo server number 8.

## 5.213 MindSensors NXTServo commands

NXTServo device command constants.

### Macros

- #define NXTSERVO\_CMD\_INIT 0x49
- #define NXTSERVO\_CMD\_RESET 0x53
- #define NXTSERVO\_CMD\_HALT 0x48
- #define NXTSERVO\_CMD\_RESUME 0x52
- #define NXTSERVO\_CMD\_GOTO 0x47
- #define NXTSERVO\_CMD\_PAUSE 0x50
- #define NXTSERVO\_CMD\_EDIT1 0x45
- #define NXTSERVO\_CMD\_EDIT2 0x4D
- #define NXTSERVO\_EM\_CMD\_QUIT 0x51

### 5.213.1 Detailed Description

NXTServo device command constants. These are written to the command register to control the device.

### 5.213.2 Macro Definition Documentation

#### 5.213.2.1 #define NXTSERVO\_CMD\_EDIT1 0x45

Edit Macro (part 1 of 2 character command sequence)

#### 5.213.2.2 #define NXTSERVO\_CMD\_EDIT2 0x4D

Edit Macro (part 2 of 2 character command sequence)

#### 5.213.2.3 #define NXTSERVO\_CMD\_GOTO 0x47

Goto EEPROM position x. This command re-initializes the macro environment.

#### 5.213.2.4 #define NXTSERVO\_CMD\_HALT 0x48

Halt Macro. This command re-initializes the macro environment.

#### 5.213.2.5 #define NXTSERVO\_CMD\_INIT 0x49

Store the initial speed and position properties of the servo motor 'n'. Current speed and position values of the nth servo is read from the servo speed register and servo position register and written to permanent memory.

#### 5.213.2.6 #define NXTSERVO\_CMD\_PAUSE 0x50

Pause Macro. This command will pause the macro, and save the environment for subsequent resumption.

#### 5.213.2.7 #define NXTSERVO\_CMD\_RESET 0x53

Reset servo properties to factory default. Initial Position of servos to 1500, and speed to 0.

#### 5.213.2.8 #define NXTSERVO\_CMD\_RESUME 0x52

Resume macro Execution. This command resumes macro where it was paused last, using the same environment.

5.213.2.9 #define NXTSERVO\_EM\_CMD\_QUIT 0x51

Exit edit macro mode

## 5.214 MindSensors NXTHID constants

Constants that are for use with the MindSensors NXTHID device.

### Modules

- [MindSensors NXTHID commands](#)  
*NXTHID device command constants.*
- [MindSensors NXTHID modifier keys](#)  
*NXTHID device modifier key constants.*
- [MindSensors NXTHID registers](#)  
*NXTHID device register constants.*

### 5.214.1 Detailed Description

Constants that are for use with the MindSensors NXTHID device.

## 5.215 MindSensors NXT HID registers

NXT HID device register constants.

### Macros

- `#define NXT HID_REG_CMD 0x41`
- `#define NXT HID_REG_MODIFIER 0x42`
- `#define NXT HID_REG_DATA 0x43`

### 5.215.1 Detailed Description

NXT HID device register constants.

### 5.215.2 Macro Definition Documentation

#### 5.215.2.1 `#define NXT HID_REG_CMD 0x41`

NXT HID command register. See [MindSensors NXT HID commands group](#).

#### 5.215.2.2 `#define NXT HID_REG_DATA 0x43`

NXT HID data register.

#### 5.215.2.3 `#define NXT HID_REG_MODIFIER 0x42`

NXT HID modifier register. See [MindSensors NXT HID modifier keys group](#).

## 5.216 MindSensors NXT HID modifier keys

NXT HID device modifier key constants.

### Macros

- `#define NXT HID_MOD_NONE 0x00`
- `#define NXT HID_MOD_LEFT_CTRL 0x01`
- `#define NXT HID_MOD_LEFT_SHIFT 0x02`
- `#define NXT HID_MOD_LEFT_ALT 0x04`
- `#define NXT HID_MOD_LEFT_GUI 0x08`
- `#define NXT HID_MOD_RIGHT_CTRL 0x10`
- `#define NXT HID_MOD_RIGHT_SHIFT 0x20`
- `#define NXT HID_MOD_RIGHT_ALT 0x40`
- `#define NXT HID_MOD_RIGHT_GUI 0x80`

### 5.216.1 Detailed Description

NXT HID device modifier key constants.

### 5.216.2 Macro Definition Documentation

#### 5.216.2.1 `#define NXT HID_MOD_LEFT_ALT 0x04`

NXT HID left alt modifier.

#### 5.216.2.2 `#define NXT HID_MOD_LEFT_CTRL 0x01`

NXT HID left control modifier.

#### 5.216.2.3 `#define NXT HID_MOD_LEFT_GUI 0x08`

NXT HID left gui modifier.

#### 5.216.2.4 `#define NXT HID_MOD_LEFT_SHIFT 0x02`

NXT HID left shift modifier.

#### 5.216.2.5 `#define NXT HID_MOD_NONE 0x00`

NXT HID no modifier.

#### 5.216.2.6 `#define NXT HID_MOD_RIGHT_ALT 0x40`

NXT HID right alt modifier.

#### 5.216.2.7 `#define NXT HID_MOD_RIGHT_CTRL 0x10`

NXT HID right control modifier.

#### 5.216.2.8 `#define NXT HID_MOD_RIGHT_GUI 0x80`

NXT HID right gui modifier.

5.216.2.9 #define NXT HID\_MOD\_RIGHT\_SHIFT 0x20

NXT HID right shift modifier.

## 5.217 MindSensors NXT HID commands

NXT HID device command constants.

### Macros

- `#define NXT HID_CMD ASCII 0x41`
- `#define NXT HID_CMD DIRECT 0x44`
- `#define NXT HID_CMD TRANSMIT 0x54`

### 5.217.1 Detailed Description

NXT HID device command constants. These are written to the command register to control the device.

### 5.217.2 Macro Definition Documentation

#### 5.217.2.1 `#define NXT HID_CMD ASCII 0x41`

Use ASCII data mode. In ASCII mode no non-printable characters can be sent.

#### 5.217.2.2 `#define NXT HID_CMD DIRECT 0x44`

Use direct data mode. In direct mode any character can be sent.

#### 5.217.2.3 `#define NXT HID_CMD TRANSMIT 0x54`

Transmit data to the host computer.

## 5.218 MindSensors NXTPowerMeter constants

Constants that are for use with the MindSensors NXTPowerMeter device.

### Modules

- [MindSensors NXTPowerMeter commands](#)  
*NXTPowerMeter device command constants.*
- [MindSensors NXTPowerMeter registers](#)  
*NXTPowerMeter device register constants.*

#### 5.218.1 Detailed Description

Constants that are for use with the MindSensors NXTPowerMeter device.

## 5.219 MindSensors NXTPowerMeter registers

NXTPowerMeter device register constants.

### Macros

- `#define NXTPM_REG_CMD 0x41`
- `#define NXTPM_REG_CURRENT 0x42`
- `#define NXTPM_REG_VOLTAGE 0x44`
- `#define NXTPM_REG_CAPACITY 0x46`
- `#define NXTPM_REG_POWER 0x48`
- `#define NXTPM_REG_TOTALPOWER 0x4A`
- `#define NXTPM_REG_MAXCURRENT 0x4E`
- `#define NXTPM_REG_MINCURRENT 0x50`
- `#define NXTPM_REG_MAXVOLTAGE 0x52`
- `#define NXTPM_REG_MINVOLTAGE 0x54`
- `#define NXTPM_REG_TIME 0x56`
- `#define NXTPM_REG_USERGAIN 0x5A`
- `#define NXTPM_REG_GAIN 0x5E`
- `#define NXTPM_REG_ERRORCOUNT 0x5F`

### 5.219.1 Detailed Description

NXTPowerMeter device register constants.

### 5.219.2 Macro Definition Documentation

#### 5.219.2.1 `#define NXTPM_REG_CAPACITY 0x46`

NXTPowerMeter capacity used since last reset register. (2 bytes)

#### 5.219.2.2 `#define NXTPM_REG_CMD 0x41`

NXTPowerMeter command register. See the [MindSensors NXTPowerMeter commands](#) group.

#### 5.219.2.3 `#define NXTPM_REG_CURRENT 0x42`

NXTPowerMeter present current in mA register. (2 bytes)

#### 5.219.2.4 `#define NXTPM_REG_ERRORCOUNT 0x5F`

NXTPowerMeter error count register. (2 bytes)

#### 5.219.2.5 `#define NXTPM_REG_GAIN 0x5E`

NXTPowerMeter gain register. (1 byte)

#### 5.219.2.6 `#define NXTPM_REG_MAXCURRENT 0x4E`

NXTPowerMeter max current register. (2 bytes)

5.219.2.7 #define NXTPM\_REG\_MAXVOLTAGE 0x52

NXTPowerMeter max voltage register. (2 bytes)

5.219.2.8 #define NXTPM\_REG\_MINCURRENT 0x50

NXTPowerMeter min current register. (2 bytes)

5.219.2.9 #define NXTPM\_REG\_MINVOLTAGE 0x54

NXTPowerMeter min voltage register. (2 bytes)

5.219.2.10 #define NXTPM\_REG\_POWER 0x48

NXTPowerMeter present power register. (2 bytes)

5.219.2.11 #define NXTPM\_REG\_TIME 0x56

NXTPowerMeter time register. (4 bytes)

5.219.2.12 #define NXTPM\_REG\_TOTALPOWER 0x4A

NXTPowerMeter total power consumed since last reset register. (4 bytes)

5.219.2.13 #define NXTPM\_REG\_USERGAIN 0x5A

NXTPowerMeter user gain register. Not yet implemented. (4 bytes)

5.219.2.14 #define NXTPM\_REG\_VOLTAGE 0x44

NXTPowerMeter present voltage in mV register. (2 bytes)

## 5.220 MindSensors NXTPowerMeter commands

NXTPowerMeter device command constants.

### Macros

- `#define NXTPM_CMD_RESET 0x52`

#### 5.220.1 Detailed Description

NXTPowerMeter device command constants. These are written to the command register to control the device.

#### 5.220.2 Macro Definition Documentation

##### 5.220.2.1 `#define NXTPM_CMD_RESET 0x52`

Reset counters.

## 5.221 MindSensors NXTSumoEyes constants

Constants that are for use with the MindSensors NXTSumoEyes device.

### Macros

- `#define NXTSE_ZONE_NONE 0`
- `#define NXTSE_ZONE_FRONT 1`
- `#define NXTSE_ZONE_LEFT 2`
- `#define NXTSE_ZONE_RIGHT 3`

### 5.221.1 Detailed Description

Constants that are for use with the MindSensors NXTSumoEyes device.

### 5.221.2 Macro Definition Documentation

#### 5.221.2.1 `#define NXTSE_ZONE_FRONT 1`

Obstacle zone front.

#### 5.221.2.2 `#define NXTSE_ZONE_LEFT 2`

Obstacle zone left.

#### 5.221.2.3 `#define NXTSE_ZONE_NONE 0`

Obstacle zone none.

#### 5.221.2.4 `#define NXTSE_ZONE_RIGHT 3`

Obstacle zone right.

## 5.222 MindSensors NXTLineLeader constants

Constants that are for use with the MindSensors NXTLineLeader device.

### Modules

- [MindSensors NXTLineLeader commands](#)  
*NXTLineLeader device command constants.*
- [MindSensors NXTLineLeader registers](#)  
*NXTLineLeader device register constants.*

#### 5.222.1 Detailed Description

Constants that are for use with the MindSensors NXTLineLeader device.

## 5.223 MindSensors NXTLineLeader registers

NXTLineLeader device register constants.

### Macros

- `#define NXTLL_REG_CMD 0x41`
- `#define NXTLL_REG_STEERING 0x42`
- `#define NXTLL_REG_AVERAGE 0x43`
- `#define NXTLL_REG_RESULT 0x44`
- `#define NXTLL_REG_SETPOINT 0x45`
- `#define NXTLL_REG_KP_VALUE 0x46`
- `#define NXTLL_REG_KI_VALUE 0x47`
- `#define NXTLL_REG_KD_VALUE 0x48`
- `#define NXTLL_REG_CALIBRATED 0x49`
- `#define NXTLL_REG_WHITELIMITS 0x51`
- `#define NXTLL_REG_BLACKLIMITS 0x59`
- `#define NXTLL_REG_KP_FACTOR 0x61`
- `#define NXTLL_REG_KI_FACTOR 0x62`
- `#define NXTLL_REG_KD_FACTOR 0x63`
- `#define NXTLL_REG_WHITEDATA 0x64`
- `#define NXTLL_REG_BLACKDATA 0x6C`
- `#define NXTLL_REG_RAWVOLTAGE 0x74`

### 5.223.1 Detailed Description

NXTLineLeader device register constants.

### 5.223.2 Macro Definition Documentation

#### 5.223.2.1 `#define NXTLL_REG_AVERAGE 0x43`

NXTLineLeader average result register.

#### 5.223.2.2 `#define NXTLL_REG_BLACKDATA 0x6C`

NXTLineLeader black calibration data registers. 8 bytes.

#### 5.223.2.3 `#define NXTLL_REG_BLACKLIMITS 0x59`

NXTLineLeader black limit registers. 8 bytes.

#### 5.223.2.4 `#define NXTLL_REG_CALIBRATED 0x49`

NXTLineLeader calibrated sensor reading registers. 8 bytes.

#### 5.223.2.5 `#define NXTLL_REG_CMD 0x41`

NXTLineLeader command register. See the [MindSensors NXTLineLeader commands](#) group.

#### 5.223.2.6 `#define NXTLL_REG_KD_FACTOR 0x63`

NXTLineLeader Kd factor register. Default = 32.

5.223.2.7 #define NXTLL\_REG\_KD\_VALUE 0x48

NXTLineLeader Kd value register. Default = 8.

5.223.2.8 #define NXTLL\_REG\_KI\_FACTOR 0x62

NXTLineLeader Ki factor register. Default = 32.

5.223.2.9 #define NXTLL\_REG\_KI\_VALUE 0x47

NXTLineLeader Ki value register. Default = 0.

5.223.2.10 #define NXTLL\_REG\_KP\_FACTOR 0x61

NXTLineLeader Kp factor register. Default = 32.

5.223.2.11 #define NXTLL\_REG\_KP\_VALUE 0x46

NXTLineLeader Kp value register. Default = 25.

5.223.2.12 #define NXTLL\_REG\_RAWVOLTAGE 0x74

NXTLineLeader uncalibrated sensor voltage registers. 16 bytes.

5.223.2.13 #define NXTLL\_REG\_RESULT 0x44

NXTLineLeader result register (sensor bit values).

5.223.2.14 #define NXTLL\_REG\_SETPOINT 0x45

NXTLineLeader user settable average (setpoint) register. Default = 45.

5.223.2.15 #define NXTLL\_REG\_STEERING 0x42

NXTLineLeader steering register.

5.223.2.16 #define NXTLL\_REG\_WHITEDATA 0x64

NXTLineLeader white calibration data registers. 8 bytes.

5.223.2.17 #define NXTLL\_REG\_WHITELIMITS 0x51

NXTLineLeader white limit registers. 8 bytes.

## 5.224 MindSensors NXTLineLeader commands

NXTLineLeader device command constants.

### Macros

- #define NXTLL\_CMD\_USA 0x41
- #define NXTLL\_CMD\_BLACK 0x42
- #define NXTLL\_CMD\_POWERDOWN 0x44
- #define NXTLL\_CMD\_EUROPEAN 0x45
- #define NXTLL\_CMD\_INVERT 0x49
- #define NXTLL\_CMD\_POWERUP 0x50
- #define NXTLL\_CMD\_RESET 0x52
- #define NXTLL\_CMD\_SNAPSHOT 0x53
- #define NXTLL\_CMD\_UNIVERSAL 0x55
- #define NXTLL\_CMD\_WHITE 0x57

### 5.224.1 Detailed Description

NXTLineLeader device command constants. These are written to the command register to control the device.

### 5.224.2 Macro Definition Documentation

#### 5.224.2.1 #define NXTLL\_CMD\_BLACK 0x42

Black calibration.

#### 5.224.2.2 #define NXTLL\_CMD\_EUROPEAN 0x45

European power frequency. (50hz)

#### 5.224.2.3 #define NXTLL\_CMD\_INVERT 0x49

Invert color.

#### 5.224.2.4 #define NXTLL\_CMD\_POWERDOWN 0x44

Power down the device.

#### 5.224.2.5 #define NXTLL\_CMD\_POWERUP 0x50

Power up the device.

#### 5.224.2.6 #define NXTLL\_CMD\_RESET 0x52

Reset inversion.

#### 5.224.2.7 #define NXTLL\_CMD\_SNAPSHOT 0x53

Setpoint based on snapshot (automatically sets invert if needed).

#### 5.224.2.8 #define NXTLL\_CMD\_UNIVERSAL 0x55

Universal power frequency. The sensor auto adjusts for any frequency. This is the default mode.

5.224.2.9 #define NXTLL\_CMD\_USA 0x41

USA power frequency. (60hz)

5.224.2.10 #define NXTLL\_CMD\_WHITE 0x57

White balance calibration.

## 5.225 MindSensors NXTNumericPad constants

Constants that are for use with the MindSensors NXTNumericPad device.

### Modules

- [MindSensors NXTNumericPad registers](#)  
*NXTNumericPad device register constants.*

#### 5.225.1 Detailed Description

Constants that are for use with the MindSensors NXTNumericPad device.

## 5.226 MindSensors NXTNumericPad registers

NXTNumericPad device register constants.

### Macros

- `#define NXTNP_REG_BUTTONS 0x00`

#### 5.226.1 Detailed Description

NXTNumericPad device register constants.

#### 5.226.2 Macro Definition Documentation

##### 5.226.2.1 `#define NXTNP_REG_BUTTONS 0x00`

NXTNumericPad buttons register.

## 5.227 MindSensors NXTTouchPanel constants

Constants that are for use with the MindSensors NXTTouchPanel device.

### Modules

- [MindSensors NXTTouchPanel commands](#)  
*NXTTouchPanel device command constants.*
- [MindSensors NXTTouchPanel registers](#)  
*NXTTouchPanel device register constants.*

#### 5.227.1 Detailed Description

Constants that are for use with the MindSensors NXTTouchPanel device.

## 5.228 MindSensors NXTTouchPanel registers

NXTTouchPanel device register constants.

### Macros

- `#define NXTTP_REG_CMD 0x41`

#### 5.228.1 Detailed Description

NXTTouchPanel device register constants.

#### 5.228.2 Macro Definition Documentation

##### 5.228.2.1 `#define NXTTP_REG_CMD 0x41`

NXTTouchPanel command register. See the [MindSensors NXTTouchPanel commands](#) group.

## 5.229 MindSensors NXTTouchPanel commands

NXTTouchPanel device command constants.

### Macros

- `#define NXTTP_CMD_USA 0x41`

#### 5.229.1 Detailed Description

NXTTouchPanel device command constants. These are written to the command register to control the device.

#### 5.229.2 Macro Definition Documentation

##### 5.229.2.1 `#define NXTTP_CMD_USA 0x41`

USA power frequency. (60hz)

## 5.230 Codatex device constants

Constants that are for use with Codatex devices.

### Modules

- [Codatex RFID sensor constants](#)

*Constants that are for use with the Codatex RFID sensor device.*

### 5.230.1 Detailed Description

Constants that are for use with Codatex devices.

## 5.231 Codatex RFID sensor constants

Constants that are for use with the Codatex RFID sensor device.

### Modules

- [Codatex RFID sensor modes](#)

*Constants that are for configuring the Codatex RFID sensor mode.*

### Macros

- `#define CT_ADDR_RFID 0x04`
- `#define CT_REG_STATUS 0x32`
- `#define CT_REG_MODE 0x41`
- `#define CT_REG_DATA 0x42`

#### 5.231.1 Detailed Description

Constants that are for use with the Codatex RFID sensor device.

#### 5.231.2 Macro Definition Documentation

##### 5.231.2.1 `#define CT_ADDR_RFID 0x04`

RFID I2C address

##### 5.231.2.2 `#define CT_REG_DATA 0x42`

RFID data register

##### 5.231.2.3 `#define CT_REG_MODE 0x41`

RFID mode register

##### 5.231.2.4 `#define CT_REG_STATUS 0x32`

RFID status register

## 5.232 Codatex RFID sensor modes

Constants that are for configuring the Codatex RFID sensor mode.

### Macros

- #define `RFID_MODE_STOP` 0
- #define `RFID_MODE_SINGLE` 1
- #define `RFID_MODE_CONTINUOUS` 2

### 5.232.1 Detailed Description

Constants that are for configuring the Codatex RFID sensor mode.

### 5.232.2 Macro Definition Documentation

#### 5.232.2.1 #define `RFID_MODE_CONTINUOUS` 2

Configure the RFID device for continuous reading

#### 5.232.2.2 #define `RFID_MODE_SINGLE` 1

Configure the RFID device for a single reading

#### 5.232.2.3 #define `RFID_MODE_STOP` 0

Stop the RFID device

## 5.233 Dexter Industries device constants

Constants that are for use with Dexter Industries devices.

### Modules

- [Dexter Industries GPS sensor constants](#)

*Constants that are for use with the Dexter Industries GPS sensor.*

- [Dexter Industries IMU sensor constants](#)

*Constants that are for use with the Dexter Industries IMU sensor.*

### 5.233.1 Detailed Description

Constants that are for use with Dexter Industries devices.

## 5.234 Dexter Industries GPS sensor constants

Constants that are for use with the Dexter Industries GPS sensor.

### Macros

- #define DI\_ADDR\_DGPS 0x06
- #define DGPS\_REG\_TIME 0x00
- #define DGPS\_REG\_STATUS 0x01
- #define DGPS\_REG\_LATITUDE 0x02
- #define DGPS\_REG\_LONGITUDE 0x04
- #define DGPS\_REG\_VELOCITY 0x06
- #define DGPS\_REG\_HEADING 0x07
- #define DGPS\_REG\_DISTANCE 0x08
- #define DGPS\_REG\_WAYANGLE 0x09
- #define DGPS\_REG\_LASTANGLE 0x0A
- #define DGPS\_REG\_SETLATITUDE 0x0B
- #define DGPS\_REG\_SETLONGITUDE 0x0C

### 5.234.1 Detailed Description

Constants that are for use with the Dexter Industries GPS sensor.

### 5.234.2 Macro Definition Documentation

#### 5.234.2.1 #define DGPS\_REG\_DISTANCE 0x08

Read distance to current waypoint in meters.

#### 5.234.2.2 #define DGPS\_REG\_HEADING 0x07

Read heading in degrees.

#### 5.234.2.3 #define DGPS\_REG\_LASTANGLE 0x0A

Read angle travelled since last request, resets the request coordinates on the GPS sensor, sends the angle of travel since last reset.

#### 5.234.2.4 #define DGPS\_REG\_LATITUDE 0x02

Read integer latitude.(dddddd; Positive = North; Negative = South).

#### 5.234.2.5 #define DGPS\_REG\_LONGITUDE 0x04

Read integer longitude (dddddd; Positive = East; Negative = West).

#### 5.234.2.6 #define DGPS\_REG\_SETLATITUDE 0x0B

Set waypoint latitude as a 4 byte integer.

#### 5.234.2.7 #define DGPS\_REG\_SETLONGITUDE 0x0C

Set waypoint longitude as a 4 byte integer.

**5.234.2.8 #define DGPS\_REG\_STATUS 0x01**

Read status of the GPS (0 - invalid signal, 1 - valid signal).

**5.234.2.9 #define DGPS\_REG\_TIME 0x00**

Read time in UTC (hhmmss).

**5.234.2.10 #define DGPS\_REG\_VELOCITY 0x06**

Read velocity in cm/s.

**5.234.2.11 #define DGPS\_REG\_WAYANGLE 0x09**

Read angle to current waypoint in degrees.

**5.234.2.12 #define DI\_ADDR\_DGPS 0x06**

Dexter Industries DGPS I2C address

## 5.235 Dexter Industries IMU sensor constants

Constants that are for use with the Dexter Industries IMU sensor.

### Modules

- [Dexter Industries IMU Accelerometer control register 1 constants](#)  
*Constants that are for use with the Dexter Industries IMU Accelerometer sensor's control register 1.*
- [Dexter Industries IMU Accelerometer control register 2 constants](#)  
*Constants that are for use with the Dexter Industries IMU Accelerometer sensor's control register 2.*
- [Dexter Industries IMU Accelerometer interrupt latch reset register constants](#)  
*Constants that are for use with the Dexter Industries IMU Accelerometer sensor's interrupt latch reset register.*
- [Dexter Industries IMU Accelerometer mode control register constants](#)  
*Constants that are for use with the Dexter Industries IMU Accelerometer sensor's mode control register.*
- [Dexter Industries IMU Accelerometer register constants](#)  
*Constants that define the Dexter Industries IMU Accelerometer registers.*
- [Dexter Industries IMU Accelerometer status register constants](#)  
*Constants that are for use with the Dexter Industries IMU Accelerometer sensor's status register.*
- [Dexter Industries IMU Gyro FIFO control register constants](#)  
*Constants that are for use with the Dexter Industries IMU Gyro sensor's FIFO control register.*
- [Dexter Industries IMU Gyro control register 1 constants](#)  
*Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 1.*
- [Dexter Industries IMU Gyro control register 2 constants](#)  
*Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 2.*
- [Dexter Industries IMU Gyro control register 3 constants](#)  
*Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 3.*
- [Dexter Industries IMU Gyro control register 4 constants](#)  
*Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 4.*
- [Dexter Industries IMU Gyro control register 5 constants](#)  
*Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 5.*
- [Dexter Industries IMU Gyro register constants](#)  
*Constants that define the Dexter Industries IMU Gyro registers.*
- [Dexter Industries IMU Gyro status register constants](#)  
*Constants that are for use with the Dexter Industries IMU Gyro sensor's status register.*

### Macros

- `#define DI_ADDR_GYRO 0xD2`
- `#define DI_ADDR_ACCL 0x3A`

#### 5.235.1 Detailed Description

Constants that are for use with the Dexter Industries IMU sensor.

#### 5.235.2 Macro Definition Documentation

##### 5.235.2.1 `#define DI_ADDR_ACCL 0x3A`

Dexter Industries DIMU Accelerometer I2C address

### 5.235.2.2 #define DI\_ADDR\_GYRO 0xD2

Dexter Industries DIMU Gyro I2C address

## 5.236 Dexter Industries IMU Gyro register constants

Constants that define the Dexter Industries IMU Gyro registers.

### Macros

- #define DIGYRO\_REG\_WHOAMI 0x0F
- #define DIGYRO\_REG\_CTRL1 0x20
- #define DIGYRO\_REG\_CTRL2 0x21
- #define DIGYRO\_REG\_CTRL3 0x22
- #define DIGYRO\_REG\_CTRL4 0x23
- #define DIGYRO\_REG\_CTRL5 0x24
- #define DIGYRO\_REG\_REFERENCE 0x25
- #define DIGYRO\_REG\_OUTTEMP 0x26
- #define DIGYRO\_REG\_STATUS 0x27
- #define DIGYRO\_REG\_XLOW 0x28
- #define DIGYRO\_REG\_XHIGH 0x29
- #define DIGYRO\_REG\_YLOW 0x2A
- #define DIGYRO\_REG\_YHIGH 0x2B
- #define DIGYRO\_REG\_ZLOW 0x2C
- #define DIGYRO\_REG\_ZHIGH 0x2D
- #define DIGYRO\_REG\_FIFOCTRL 0x2E
- #define DIGYRO\_REG\_FIFOSRC 0x2F
- #define DIGYRO\_REG\_INT1\_CFG 0x30
- #define DIGYRO\_REG\_INT1\_SRC 0x31
- #define DIGYRO\_REG\_INT1\_XHI 0x32
- #define DIGYRO\_REG\_INT1\_XLO 0x33
- #define DIGYRO\_REG\_INT1\_YHI 0x34
- #define DIGYRO\_REG\_INT1\_YLO 0x35
- #define DIGYRO\_REG\_INT1\_ZHI 0x36
- #define DIGYRO\_REG\_INT1\_ZLO 0x37
- #define DIGYRO\_REG\_INT1\_DUR 0x38
- #define DIGYRO\_REG\_CTRL1AUTO 0xA0
- #define DIGYRO\_REG\_TEMP AUTO 0xA6
- #define DIGYRO\_REG\_XLOWBURST 0xA8
- #define DIGYRO\_REG\_YLOWBURST 0xAA
- #define DIGYRO\_REG\_ZLOWBURST 0xAC

### 5.236.1 Detailed Description

Constants that define the Dexter Industries IMU Gyro registers.

### 5.236.2 Macro Definition Documentation

#### 5.236.2.1 #define DIGYRO\_REG\_CTRL1 0x20

Gyro control register 1

#### 5.236.2.2 #define DIGYRO\_REG\_CTRL1AUTO 0xA0

Gyro control register 1 - auto increment write

5.236.2.3 #define DIGYRO\_REG\_CTRL2 0x21

Gyro control register 2

5.236.2.4 #define DIGYRO\_REG\_CTRL3 0x22

Gyro control register 3

5.236.2.5 #define DIGYRO\_REG\_CTRL4 0x23

Gyro control register 4

5.236.2.6 #define DIGYRO\_REG\_CTRL5 0x24

Gyro control register 5

5.236.2.7 #define DIGYRO\_REG\_FIFOCTRL 0x2E

Gyro FIFO control register

5.236.2.8 #define DIGYRO\_REG\_FIFOSRC 0x2F

Gyro FIFO source register (read only)

5.236.2.9 #define DIGYRO\_REG\_INT1\_CFG 0x30

Gyro interrupt 1 config register

5.236.2.10 #define DIGYRO\_REG\_INT1\_DUR 0x38

Gyro interrupt 1 duration register

5.236.2.11 #define DIGYRO\_REG\_INT1\_SRC 0x31

Gyro interrupt 1 source register

5.236.2.12 #define DIGYRO\_REG\_INT1\_XHI 0x32

Gyro interrupt 1 x-axis high threshold register

5.236.2.13 #define DIGYRO\_REG\_INT1\_XLO 0x33

Gyro interrupt 1 x-axis low threshold register

5.236.2.14 #define DIGYRO\_REG\_INT1\_YHI 0x34

Gyro interrupt 1 y-axis high threshold register

5.236.2.15 #define DIGYRO\_REG\_INT1\_YLO 0x35

Gyro interrupt 1 y-axis low threshold register

5.236.2.16 #define DIGYRO\_REG\_INT1\_ZHI 0x36

Gyro interrupt 1 z-axis high threshold register

5.236.2.17 #define DIGYRO\_REG\_INT1\_ZLO 0x37

Gyro interrupt 1 z-axis low threshold register

5.236.2.18 #define DIGYRO\_REG\_OUTTEMP 0x26

Gyro temperature register (read only) - stores temperature data

5.236.2.19 #define DIGYRO\_REG\_REFERENCE 0x25

Gyro reference register - stores the reference value used for interrupt generation

5.236.2.20 #define DIGYRO\_REG\_STATUS 0x27

Gyro status register (read only)

5.236.2.21 #define DIGYRO\_REG\_TEMPAUTO 0xA6

Gyro temperature register - read burst mode (read only)

5.236.2.22 #define DIGYRO\_REG\_WHOAMI 0x0F

Gyro device identification register (read only)

5.236.2.23 #define DIGYRO\_REG\_XHIGH 0x29

Gyro x-axis high byte register (read only)

5.236.2.24 #define DIGYRO\_REG\_XLOW 0x28

Gyro x-axis low byte register (read only)

5.236.2.25 #define DIGYRO\_REG\_XLOWBURST 0xA8

Gyro x-axis low byte register - read burst mode (read only)

5.236.2.26 #define DIGYRO\_REG\_YHIGH 0x2B

Gyro y-axis high byte register (read only)

5.236.2.27 #define DIGYRO\_REG\_YLOW 0x2A

Gyro y-axis low byte register (read only)

5.236.2.28 #define DIGYRO\_REG\_YLOWBURST 0xAA

Gyro y-axis low byte register - read burst mode (read only)

5.236.2.29 #define DIGYRO\_REG\_ZHIGH 0x2D

Gyro z-axis high byte register (read only)

5.236.2.30 #define DIGYRO\_REG\_ZLOW 0x2C

Gyro z-axis low byte register (read only)

## 5.236.2.31 #define DIGYRO\_REG\_ZLOWBURST 0xAC

Gyro y-axis low byte register - read burst mode (read only)

## 5.237 Dexter Industries IMU Gyro control register 1 constants

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 1.

### Macros

- #define DIGYRO\_CTRL1\_XENABLE 0x01
- #define DIGYRO\_CTRL1\_YENABLE 0x02
- #define DIGYRO\_CTRL1\_ZENABLE 0x04
- #define DIGYRO\_CTRL1\_POWERDOWN 0x00
- #define DIGYRO\_CTRL1\_NORMAL 0x08
- #define DIGYRO\_CTRL1\_BANDWIDTH\_1 0x00
- #define DIGYRO\_CTRL1\_BANDWIDTH\_2 0x10
- #define DIGYRO\_CTRL1\_BANDWIDTH\_3 0x20
- #define DIGYRO\_CTRL1\_BANDWIDTH\_4 0x30
- #define DIGYRO\_CTRL1\_DATARATE\_100 0x00
- #define DIGYRO\_CTRL1\_DATARATE\_200 0x40
- #define DIGYRO\_CTRL1\_DATARATE\_400 0x80
- #define DIGYRO\_CTRL1\_DATARATE\_800 0xC0

### 5.237.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 1.

### 5.237.2 Macro Definition Documentation

#### 5.237.2.1 #define DIGYRO\_CTRL1\_BANDWIDTH\_1 0x00

Gyro LPF2 cut-off frequency bandwidth level 1 (12.5hz, 12.5hz, 20hz, 30hz)

#### 5.237.2.2 #define DIGYRO\_CTRL1\_BANDWIDTH\_2 0x10

Gyro LPF2 cut-off frequency bandwidth level 2 (12.5hz, 25hz, 50hz, 70hz)

#### 5.237.2.3 #define DIGYRO\_CTRL1\_BANDWIDTH\_3 0x20

Gyro LPF2 cut-off frequency bandwidth level 3 (20hz, 25hz, 50hz, 110hz)

#### 5.237.2.4 #define DIGYRO\_CTRL1\_BANDWIDTH\_4 0x30

Gyro LPF2 cut-off frequency bandwidth level 4 (30hz, 35hz, 50hz, 110hz)

#### 5.237.2.5 #define DIGYRO\_CTRL1\_DATARATE\_100 0x00

Gyro output data rate 100 hz

#### 5.237.2.6 #define DIGYRO\_CTRL1\_DATARATE\_200 0x40

Gyro output data rate 200 hz

#### 5.237.2.7 #define DIGYRO\_CTRL1\_DATARATE\_400 0x80

Gyro output data rate 400 hz

5.237.2.8 #define DIGYRO\_CTRL1\_DATARATE\_800 0xC0

Gyro output data rate 800 hz

5.237.2.9 #define DIGYRO\_CTRL1\_NORMAL 0x08

Gyro disable power down mode

5.237.2.10 #define DIGYRO\_CTRL1\_POWERDOWN 0x00

Gyro enable power down mode

5.237.2.11 #define DIGYRO\_CTRL1\_XENABLE 0x01

Gyro enable X axis

5.237.2.12 #define DIGYRO\_CTRL1\_YENABLE 0x02

Gyro enable Y axis

5.237.2.13 #define DIGYRO\_CTRL1\_ZENABLE 0x04

Gyro enable Z axis

## 5.238 Dexter Industries IMU Gyro control register 2 constants

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 2.

### Macros

- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_8 0x00
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_4 0x01
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_2 0x02
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_1 0x03
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_05 0x04
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_02 0x05
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_01 0x06
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_005 0x07
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_002 0x08
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_001 0x09
- #define DIGYRO\_CTRL2\_HPMODE\_RESET 0x00
- #define DIGYRO\_CTRL2\_HPMODE\_REFSIG 0x10
- #define DIGYRO\_CTRL2\_HPMODE\_NORMAL 0x20
- #define DIGYRO\_CTRL2\_HPMODE\_AUTOINT 0x30

### 5.238.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 2.

### 5.238.2 Macro Definition Documentation

#### 5.238.2.1 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_001 0x09

Gyro high pass filter cutoff frequency 0.01 hz

#### 5.238.2.2 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_002 0x08

Gyro high pass filter cutoff frequency 0.02 hz

#### 5.238.2.3 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_005 0x07

Gyro high pass filter cutoff frequency 0.05 hz

#### 5.238.2.4 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_01 0x06

Gyro high pass filter cutoff frequency 0.1 hz

#### 5.238.2.5 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_02 0x05

Gyro high pass filter cutoff frequency 0.2 hz

#### 5.238.2.6 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_05 0x04

Gyro high pass filter cutoff frequency 0.5 hz

5.238.2.7 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_1 0x03

Gyro high pass filter cutoff frequency 1 hz

5.238.2.8 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_2 0x02

Gyro high pass filter cutoff frequency 2 hz

5.238.2.9 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_4 0x01

Gyro high pass filter cutoff frequency 4 hz

5.238.2.10 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_8 0x00

Gyro high pass filter cutoff frequency 8 hz

5.238.2.11 #define DIGYRO\_CTRL2\_HPMODE\_AUTOINT 0x30

Gyro high pass filter autoreset on interrupt event mode

5.238.2.12 #define DIGYRO\_CTRL2\_HPMODE\_NORMAL 0x20

Gyro high pass filter normal mode

5.238.2.13 #define DIGYRO\_CTRL2\_HPMODE\_REFSIG 0x10

Gyro high pass filter reference signal mode

5.238.2.14 #define DIGYRO\_CTRL2\_HPMODE\_RESET 0x00

Gyro high pass filter reset mode

## 5.239 Dexter Industries IMU Gyro control register 3 constants

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 3.

### Macros

- #define DIGYRO\_CTRL3\_INT1\_ENABLE 0x80
- #define DIGYRO\_CTRL3\_INT1\_BOOT 0x40
- #define DIGYRO\_CTRL3\_INT1\_LOWACTIVE 0x20
- #define DIGYRO\_CTRL3\_OPENDRAIN 0x10
- #define DIGYRO\_CTRL3\_INT2\_DATAREADY 0x08
- #define DIGYRO\_CTRL3\_INT2\_WATERMARK 0x04
- #define DIGYRO\_CTRL3\_INT2\_OVERRUN 0x02
- #define DIGYRO\_CTRL3\_INT2\_EMPTY 0x01

### 5.239.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 3.

### 5.239.2 Macro Definition Documentation

#### 5.239.2.1 #define DIGYRO\_CTRL3\_INT1\_BOOT 0x40

Gyro boot status available on INT1

#### 5.239.2.2 #define DIGYRO\_CTRL3\_INT1\_ENABLE 0x80

Gyro interrupt enable on INT1 pin

#### 5.239.2.3 #define DIGYRO\_CTRL3\_INT1\_LOWACTIVE 0x20

Gyro interrupt active low on INT1

#### 5.239.2.4 #define DIGYRO\_CTRL3\_INT2\_DATAREADY 0x08

Gyro data ready on DRDY/INT2

#### 5.239.2.5 #define DIGYRO\_CTRL3\_INT2\_EMPTY 0x01

Gyro FIFO empty interrupt on DRDY/INT2

#### 5.239.2.6 #define DIGYRO\_CTRL3\_INT2\_OVERRUN 0x02

Gyro FIFO overrun interrupt on DRDY/INT2

#### 5.239.2.7 #define DIGYRO\_CTRL3\_INT2\_WATERMARK 0x04

Gyro FIFO watermark interrupt on DRDY/INT2

#### 5.239.2.8 #define DIGYRO\_CTRL3\_OPENDRAIN 0x10

Gyro use open drain rather than push-pull

## 5.240 Dexter Industries IMU Gyro control register 4 constants

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 4.

### Macros

- #define DIGYRO\_CTRL4\_BLOCKDATA 0x80
- #define DIGYRO\_CTRL4\_BIGENDIAN 0x40
- #define DIGYRO\_CTRL4\_SCALE\_250 0x00
- #define DIGYRO\_CTRL4\_SCALE\_500 0x10
- #define DIGYRO\_CTRL4\_SCALE\_2000 0x30

### 5.240.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 4.

### 5.240.2 Macro Definition Documentation

#### 5.240.2.1 #define DIGYRO\_CTRL4\_BIGENDIAN 0x40

Gyro use big endian - MSB/LSB rather than LSB/MSB in output registers

#### 5.240.2.2 #define DIGYRO\_CTRL4\_BLOCKDATA 0x80

Gyro block data update - output registers are not updated until MSB and LSB reading

#### 5.240.2.3 #define DIGYRO\_CTRL4\_SCALE\_2000 0x30

Gyro 2000 degrees per second scale

#### 5.240.2.4 #define DIGYRO\_CTRL4\_SCALE\_250 0x00

Gyro 250 degrees per second scale

#### 5.240.2.5 #define DIGYRO\_CTRL4\_SCALE\_500 0x10

Gyro 500 degrees per second scale

## 5.241 Dexter Industries IMU Gyro control register 5 constants

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 5.

### Macros

- #define DIGYRO\_CTRL5\_REBOOTMEM 0x80
- #define DIGYRO\_CTRL5\_FIFOENABLE 0x40
- #define DIGYRO\_CTRL5\_HPENABLE 0x10
- #define DIGYRO\_CTRL5\_OUT\_SEL\_1 0x00
- #define DIGYRO\_CTRL5\_OUT\_SEL\_2 0x01
- #define DIGYRO\_CTRL5\_OUT\_SEL\_3 0x02
- #define DIGYRO\_CTRL5\_INT1\_SEL\_1 0x00
- #define DIGYRO\_CTRL5\_INT1\_SEL\_2 0x04
- #define DIGYRO\_CTRL5\_INT1\_SEL\_3 0x08

### 5.241.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 5.

### 5.241.2 Macro Definition Documentation

#### 5.241.2.1 #define DIGYRO\_CTRL5\_FIFOENABLE 0x40

Gyro enable FIFO

#### 5.241.2.2 #define DIGYRO\_CTRL5\_HPENABLE 0x10

Gyro enable high pass filter

#### 5.241.2.3 #define DIGYRO\_CTRL5\_INT1\_SEL\_1 0x00

Gyro non-high-pass-filtered data are used for interrupt generation

#### 5.241.2.4 #define DIGYRO\_CTRL5\_INT1\_SEL\_2 0x04

Gyro high-pass-filtered data are used for interrupt generation

#### 5.241.2.5 #define DIGYRO\_CTRL5\_INT1\_SEL\_3 0x08

Gyro low-pass-filtered data are used for interrupt generation

#### 5.241.2.6 #define DIGYRO\_CTRL5\_OUT\_SEL\_1 0x00

Gyro data in data registers and FIFO are not high-pass filtered

#### 5.241.2.7 #define DIGYRO\_CTRL5\_OUT\_SEL\_2 0x01

Gyro data in data registers and FIFO are high-pass filtered

#### 5.241.2.8 #define DIGYRO\_CTRL5\_OUT\_SEL\_3 0x02

Gyro data in data registers and FIFO are low-pass filtered by LPF2

5.241.2.9 #define DIGYRO\_CTRL5\_REBOOTMEM 0x80

Gyro reboot memory content

## 5.242 Dexter Industries IMU Gyro FIFO control register constants

Constants that are for use with the Dexter Industries IMU Gyro sensor's FIFO control register.

### Macros

- #define DIGYRO\_FIFOCTRL\_BYPASS 0x00
- #define DIGYRO\_FIFOCTRL\_FIFO 0x20
- #define DIGYRO\_FIFOCTRL\_STREAM 0x40
- #define DIGYRO\_FIFOCTRL\_STREAM2FIFO 0x60
- #define DIGYRO\_FIFOCTRL\_BYPASS2STREAM 0x80
- #define DIGYRO\_FIFOCTRL\_WATERMARK\_MASK 0x1F

### 5.242.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's FIFO control register.

### 5.242.2 Macro Definition Documentation

#### 5.242.2.1 #define DIGYRO\_FIFOCTRL\_BYPASS 0x00

Gyro FIFO bypass mode

#### 5.242.2.2 #define DIGYRO\_FIFOCTRL\_BYPASS2STREAM 0x80

Gyro FIFO bypass-to-stream mode

#### 5.242.2.3 #define DIGYRO\_FIFOCTRL\_FIFO 0x20

Gyro FIFO mode

#### 5.242.2.4 #define DIGYRO\_FIFOCTRL\_STREAM 0x40

Gyro FIFO stream mode

#### 5.242.2.5 #define DIGYRO\_FIFOCTRL\_STREAM2FIFO 0x60

Gyro FIFO stream-to-FIFO mode

#### 5.242.2.6 #define DIGYRO\_FIFOCTRL\_WATERMARK\_MASK 0x1F

Gyro FIFO threshold. Watermark level setting mask (values from 0x00 to 0x1F)

## 5.243 Dexter Industries IMU Gyro status register constants

Constants that are for use with the Dexter Industries IMU Gyro sensor's status register.

### Macros

- #define DIGYRO\_STATUS\_XDATA 0x01
- #define DIGYRO\_STATUS\_YDATA 0x02
- #define DIGYRO\_STATUS\_ZDATA 0x04
- #define DIGYRO\_STATUS\_XYZDATA 0x08
- #define DIGYRO\_STATUS\_XOVER 0x10
- #define DIGYRO\_STATUS\_YOVER 0x20
- #define DIGYRO\_STATUS\_ZOVER 0x40
- #define DIGYRO\_STATUS\_XYZOVER 0x80

### 5.243.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's status register.

### 5.243.2 Macro Definition Documentation

#### 5.243.2.1 #define DIGYRO\_STATUS\_XDATA 0x01

Gyro X-axis new data available

#### 5.243.2.2 #define DIGYRO\_STATUS\_XOVER 0x10

Gyro X-axis data overrun - new data for the X-axis has overwritten the previous one

#### 5.243.2.3 #define DIGYRO\_STATUS\_XYZDATA 0x08

Gyro X, Y, or Z-axis new data available - a new set of data is available

#### 5.243.2.4 #define DIGYRO\_STATUS\_XYZOVER 0x80

Gyro X, Y, or Z-axis data overrun - new data has overwritten the previous one before it was read

#### 5.243.2.5 #define DIGYRO\_STATUS\_YDATA 0x02

Gyro Y-axis new data available

#### 5.243.2.6 #define DIGYRO\_STATUS\_YOVER 0x20

Gyro Y-axis data overrun - new data for the Y-axis has overwritten the previous one

#### 5.243.2.7 #define DIGYRO\_STATUS\_ZDATA 0x04

Gyro Z-axis new data available

#### 5.243.2.8 #define DIGYRO\_STATUS\_ZOVER 0x40

Gyro Z-axis data overrun - new data for the Z-axis has overwritten the previous one

## 5.244 Dexter Industries IMU Accelerometer register constants

Constants that define the Dexter Industries IMU Accelerometer registers.

### Macros

- #define DIACCL\_REG\_XLOW 0x00
- #define DIACCL\_REG\_XHIGH 0x01
- #define DIACCL\_REG\_YLOW 0x02
- #define DIACCL\_REG\_YHIGH 0x03
- #define DIACCL\_REG\_ZLOW 0x04
- #define DIACCL\_REG\_ZHIGH 0x05
- #define DIACCL\_REG\_X8 0x06
- #define DIACCL\_REG\_Y8 0x07
- #define DIACCL\_REG\_Z8 0x08
- #define DIACCL\_REG\_STATUS 0x09
- #define DIACCL\_REG\_DETECTSRC 0xA
- #define DIACCL\_REG\_OUTTEMP 0xB
- #define DIACCL\_REG\_I2CADDR 0xD
- #define DIACCL\_REG\_USERINFO 0xE
- #define DIACCL\_REG\_WHOAMI 0xF
- #define DIACCL\_REG\_XLOWDRIFT 0x10
- #define DIACCL\_REG\_XHIGHDRIFT 0x11
- #define DIACCL\_REG\_YLOWDRIFT 0x12
- #define DIACCL\_REG\_YHIGHDRIFT 0x13
- #define DIACCL\_REG\_ZLOWDRIFT 0x14
- #define DIACCL\_REG\_ZHIGHDRIFT 0x15
- #define DIACCL\_REG\_MODECTRL 0x16
- #define DIACCL\_REG\_INTLATCH 0x17
- #define DIACCL\_REG\_CTRL1 0x18
- #define DIACCL\_REG\_CTRL2 0x19
- #define DIACCL\_REG\_LVLDETTHR 0x1A
- #define DIACCL\_REG\_PLSDETTHR 0x1B
- #define DIACCL\_REG\_PLSDURVAL 0x1C
- #define DIACCL\_REG\_LATENCYTM 0x1D
- #define DIACCL\_REG\_TIMEWINDOW 0x1E

### 5.244.1 Detailed Description

Constants that define the Dexter Industries IMU Accelerometer registers.

### 5.244.2 Macro Definition Documentation

#### 5.244.2.1 #define DIACCL\_REG\_CTRL1 0x18

Accelerometer control register 1 (read/write)

#### 5.244.2.2 #define DIACCL\_REG\_CTRL2 0x19

Accelerometer control register 1 (read/write)

5.244.2.3 #define DIACCL\_REG\_DETECTSRC 0x0A

Accelerometer detection source register (read only)

5.244.2.4 #define DIACCL\_REG\_I2CADDR 0x0D

Accelerometer I2C address register (read only)

5.244.2.5 #define DIACCL\_REG\_INTLATCH 0x17

Accelerometer interrupt latch reset register (read/write)

5.244.2.6 #define DIACCL\_REG\_LATENCYTM 0x1D

Accelerometer latency time value register (read/write)

5.244.2.7 #define DIACCL\_REG\_LVLDETTTHR 0x1A

Accelerometer level detection threshold limit value register (read/write)

5.244.2.8 #define DIACCL\_REG\_MODECTRL 0x16

Accelerometer mode control register (read/write)

5.244.2.9 #define DIACCL\_REG\_OUTTEMP 0x0B

Accelerometer temperature output register (read only)

5.244.2.10 #define DIACCL\_REG\_PLSDETTTHR 0x1B

Accelerometer pulse detection threshold limit value register (read/write)

5.244.2.11 #define DIACCL\_REG\_PLSDURVAL 0x1C

Accelerometer pulse duration value register (read/write)

5.244.2.12 #define DIACCL\_REG\_STATUS 0x09

Accelerometer status register (read only)

5.244.2.13 #define DIACCL\_REG\_TIMEWINDOW 0x1E

Accelerometer time window for 2nd pulse value register (read/write)

5.244.2.14 #define DIACCL\_REG\_USERINFO 0x0E

Accelerometer user information register (read only)

5.244.2.15 #define DIACCL\_REG\_WHOAMI 0x0F

Accelerometer device identification register (read only)

5.244.2.16 #define DIACCL\_REG\_X8 0x06

Accelerometer x-axis 8-bit register (read only)

5.244.2.17 #define DIACCL\_REG\_XHIGH 0x01

Accelerometer x-axis high byte register (read only)

5.244.2.18 #define DIACCL\_REG\_XHIGHDRIFT 0x11

Accelerometer x-axis offset drift high byte register (read/write)

5.244.2.19 #define DIACCL\_REG\_XLOW 0x00

Accelerometer x-axis low byte register (read only)

5.244.2.20 #define DIACCL\_REG\_XLOWDRIFT 0x10

Accelerometer x-axis offset drift low byte register (read/write)

5.244.2.21 #define DIACCL\_REG\_Y8 0x07

Accelerometer x-axis 8-bit register (read only)

5.244.2.22 #define DIACCL\_REG\_YHIGH 0x03

Accelerometer y-axis high byte register (read only)

5.244.2.23 #define DIACCL\_REG\_YHIGHDRIFT 0x13

Accelerometer y-axis offset drift high byte register (read/write)

5.244.2.24 #define DIACCL\_REG\_YLOW 0x02

Accelerometer y-axis low byte register (read only)

5.244.2.25 #define DIACCL\_REG\_YLOWDRIFT 0x12

Accelerometer y-axis offset drift low byte register (read/write)

5.244.2.26 #define DIACCL\_REG\_Z8 0x08

Accelerometer x-axis 8-bit register (read only)

5.244.2.27 #define DIACCL\_REG\_ZHIGH 0x05

Accelerometer z-axis high byte register (read only)

5.244.2.28 #define DIACCL\_REG\_ZHIGHDRIFT 0x15

Accelerometer z-axis offset drift high byte register (read/write)

5.244.2.29 #define DIACCL\_REG\_ZLOW 0x04

Accelerometer z-axis low byte register (read only)

5.244.2.30 #define DIACCL\_REG\_ZLOWDRIFT 0x14

Accelerometer z-axis offset drift low byte register (read/write)

## 5.245 Dexter Industries IMU Accelerometer status register constants

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's status register.

### Macros

- #define DIACCL\_STATUS\_DATAREADY 0x01
- #define DIACCL\_STATUS\_DATAOVER 0x02
- #define DIACCL\_STATUS\_PARITYERR 0x04

#### 5.245.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's status register.

#### 5.245.2 Macro Definition Documentation

##### 5.245.2.1 #define DIACCL\_STATUS\_DATAOVER 0x02

Accelerometer data is overwritten

##### 5.245.2.2 #define DIACCL\_STATUS\_DATAREADY 0x01

Accelerometer data is ready

##### 5.245.2.3 #define DIACCL\_STATUS\_PARITYERR 0x04

Accelerometer parity error is detected in trim data

## 5.246 Dexter Industries IMU Accelerometer mode control register constants

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's mode control register.

### Macros

- #define DIACCL\_MODE\_STANDBY 0x00
- #define DIACCL\_MODE\_MEASURE 0x01
- #define DIACCL\_MODE\_LVLDETECT 0x02
- #define DIACCL\_MODE\_PLSDETECT 0x03
- #define DIACCL\_MODE\_GLVL8 0x00
- #define DIACCL\_MODE\_GLVL2 0x04
- #define DIACCL\_MODE\_GLVL4 0x08

### 5.246.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's mode control register.

### 5.246.2 Macro Definition Documentation

#### 5.246.2.1 #define DIACCL\_MODE\_GLVL2 0x04

Accelerometer 2G measurement range

#### 5.246.2.2 #define DIACCL\_MODE\_GLVL4 0x08

Accelerometer 4G measurement range

#### 5.246.2.3 #define DIACCL\_MODE\_GLVL8 0x00

Accelerometer 8G measurement range

#### 5.246.2.4 #define DIACCL\_MODE\_LVLDETECT 0x02

Accelerometer level detect mode

#### 5.246.2.5 #define DIACCL\_MODE\_MEASURE 0x01

Accelerometer measurement mode

#### 5.246.2.6 #define DIACCL\_MODE\_PLSDETECT 0x03

Accelerometer pulse detect mode

#### 5.246.2.7 #define DIACCL\_MODE\_STANDBY 0x00

Accelerometer standby mode

## 5.247 Dexter Industries IMU Accelerometer interrupt latch reset register constants

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's interrupt latch reset register.

### Macros

- #define DIACCL\_INTERRUPT\_LATCH\_CLEAR1 0x01
- #define DIACCL\_INTERRUPT\_LATCH\_CLEAR2 0x02

#### 5.247.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's interrupt latch reset register.

#### 5.247.2 Macro Definition Documentation

##### 5.247.2.1 #define DIACCL\_INTERRUPT\_LATCH\_CLEAR1 0x01

Accelerometer clear interrupt 1

##### 5.247.2.2 #define DIACCL\_INTERRUPT\_LATCH\_CLEAR2 0x02

Accelerometer clear interrupt 2

## 5.248 Dexter Industries IMU Accelerometer control register 1 constants

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's control register 1.

### Macros

- #define DIACCL\_CTRL1\_INT2TOINT1 0x01
- #define DIACCL\_CTRL1\_LEVELPULSE 0x00
- #define DIACCL\_CTRL1\_PULSELEVEL 0x02
- #define DIACCL\_CTRL1\_PULSEPULSE 0x04
- #define DIACCL\_CTRL1\_NO\_XDETECT 0x08
- #define DIACCL\_CTRL1\_NO\_YDETECT 0x10
- #define DIACCL\_CTRL1\_NO\_ZDETECT 0x20
- #define DIACCL\_CTRL1\_THRESH\_INT 0x40
- #define DIACCL\_CTRL1\_FILT\_BW125 0x80

### 5.248.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's control register 1.

### 5.248.2 Macro Definition Documentation

#### 5.248.2.1 #define DIACCL\_CTRL1\_FILT\_BW125 0x80

Accelerometer digital filter band width is 125 Hz.

#### 5.248.2.2 #define DIACCL\_CTRL1\_INT2TOINT1 0x01

Accelerometer INT2 pin is routed to INT1 bit in Detection Source Register (\$0A) and INT1 pin is routed to INT2 bit in Detection Source Register (\$0A)

#### 5.248.2.3 #define DIACCL\_CTRL1\_LEVELPULSE 0x00

Accelerometer INT1 register is detecting Level while INT2 is detecting pulse

#### 5.248.2.4 #define DIACCL\_CTRL1\_NO\_XDETECT 0x08

Accelerometer disable x-axis detection.

#### 5.248.2.5 #define DIACCL\_CTRL1\_NO\_YDETECT 0x10

Accelerometer disable y-axis detection.

#### 5.248.2.6 #define DIACCL\_CTRL1\_NO\_ZDETECT 0x20

Accelerometer disable z-axis detection.

#### 5.248.2.7 #define DIACCL\_CTRL1\_PULSELEVEL 0x02

Accelerometer INT1 Register is detecting Pulse while INT2 is detecting Level

**5.248.2.8 #define DIACCL\_CTRL1\_PULSEPULSE 0x04**

Accelerometer INT1 Register is detecting a Single Pulse and INT2 is detecting Single Pulse (if 2nd Time Window = 0) or if there is a latency time window and second time window > 0 then INT2 will detect the double pulse only.

**5.248.2.9 #define DIACCL\_CTRL1\_THRESH\_INT 0x40**

Accelerometer threshold value can be an integer.

## 5.249 Dexter Industries IMU Accelerometer control register 2 constants

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's control register 2.

### Macros

- #define DIACCL\_CTRL2\_LVLPO\_LNEGAND 0x01
- #define DIACCL\_CTRL2\_DETPOL\_NEGAND 0x02
- #define DIACCL\_CTRL2\_DRIVE\_STRONG 0x04

#### 5.249.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's control register 2.

#### 5.249.2 Macro Definition Documentation

##### 5.249.2.1 #define DIACCL\_CTRL2\_DETPOL\_NEGAND 0x02

Accelerometer pulse detection polarity is negative and detecting condition is AND all 3 axes

##### 5.249.2.2 #define DIACCL\_CTRL2\_DRIVE\_STRONG 0x04

Accelerometer strong drive strength on SDA/SDO pin

##### 5.249.2.3 #define DIACCL\_CTRL2\_LVLPO\_LNEGAND 0x01

Accelerometer level detection polarity is negative and detecting condition is AND all 3 axes

## 5.250 Microinfinity device constants

Constants that are for use with Microinfinity devices.

### Modules

- [Microinfinity CruizCore XG1300L sensor constants](#)

*Constants that are for use with the CruizCore XG1300L sensor.*

### 5.250.1 Detailed Description

Constants that are for use with Microinfinity devices.

## 5.251 Microinfinity CruizCore XG1300L sensor constants

Constants that are for use with the CruizCore XG1300L sensor.

### Modules

- [Microinfinity CruizCore XG1300L](#)

*sensor scale factor constants Constants for setting the scale factor of the CruizCore XG1300L sensor.*

### Macros

- `#define MI_ADDR_XG1300L 0x02`
- `#define XG1300L_REG_ANGLE 0x42`
- `#define XG1300L_REG_TURNRATE 0x44`
- `#define XG1300L_REG_XAXIS 0x46`
- `#define XG1300L_REG_YAXIS 0x48`
- `#define XG1300L_REG_ZAXIS 0x4A`
- `#define XG1300L_REG_RESET 0x60`
- `#define XG1300L_REG_2G 0x61`
- `#define XG1300L_REG_4G 0x62`
- `#define XG1300L_REG_8G 0x63`

### 5.251.1 Detailed Description

Constants that are for use with the CruizCore XG1300L sensor.

### 5.251.2 Macro Definition Documentation

#### 5.251.2.1 `#define MI_ADDR_XG1300L 0x02`

XG1300L I2C address

#### 5.251.2.2 `#define XG1300L_REG_2G 0x61`

Select +/- 2G accelerometer range.

#### 5.251.2.3 `#define XG1300L_REG_4G 0x62`

Select +/- 4G accelerometer range.

#### 5.251.2.4 `#define XG1300L_REG_8G 0x63`

Select +/- 8G accelerometer range.

#### 5.251.2.5 `#define XG1300L_REG_ANGLE 0x42`

Read accumulated angle (2 bytes little endian) in 1/100s of degrees.

#### 5.251.2.6 `#define XG1300L_REG_RESET 0x60`

Reset the XG1300L device.

**5.251.2.7 #define XG1300L\_REG\_TURNRATE 0x44**

Read rate of turn (2 bytes little endian) in 1/100s of degrees/second.

**5.251.2.8 #define XG1300L\_REG\_XAXIS 0x46**

Read x-axis acceleration (2 bytes little endian) in m/s<sup>2</sup> scaled by 100/ACC\_RANGE\*2, where ACC\_RANGE is 2, 4, or 8.

**5.251.2.9 #define XG1300L\_REG\_YAXIS 0x48**

Read y-axis acceleration (2 bytes little endian) in m/s<sup>2</sup> scaled by 100/ACC\_RANGE\*2, where ACC\_RANGE is 2, 4, or 8.

**5.251.2.10 #define XG1300L\_REG\_ZAXIS 0x4A**

Read z-axis acceleration (2 bytes little endian) in m/s<sup>2</sup> scaled by 100/ACC\_RANGE\*2, where ACC\_RANGE is 2, 4, or 8.

## 5.252 Microinfinity CruizCore XG1300L

sensor scale factor constants Constants for setting the scale factor of the CruizCore XG1300L sensor.

### Macros

- #define XG1300L\_SCALE\_2G 0x01
- #define XG1300L\_SCALE\_4G 0x02
- #define XG1300L\_SCALE\_8G 0x04

### 5.252.1 Detailed Description

sensor scale factor constants Constants for setting the scale factor of the CruizCore XG1300L sensor.

### 5.252.2 Macro Definition Documentation

#### 5.252.2.1 #define XG1300L\_SCALE\_2G 0x01

Select +/- 2G accelerometer range.

#### 5.252.2.2 #define XG1300L\_SCALE\_4G 0x02

Select +/- 4G accelerometer range.

#### 5.252.2.3 #define XG1300L\_SCALE\_8G 0x04

Select +/- 8G accelerometer range.

## 5.253 Data type limits

Constants that define various data type limits.

### Macros

- `#define CHAR_BIT 8`
- `#define SCHAR_MIN -128`
- `#define SCHAR_MAX 127`
- `#define UCHAR_MAX 255`
- `#define CHAR_MIN -128`
- `#define CHAR_MAX 127`
- `#define SHRT_MIN -32768`
- `#define SHRT_MAX 32767`
- `#define USHRT_MAX 65535`
- `#define INT_MIN -32768`
- `#define INT_MAX 32767`
- `#define UINT_MAX 65535`
- `#define LONG_MIN -2147483648`
- `#define LONG_MAX 2147483647`
- `#define ULONG_MAX 4294967295`
- `#define RAND_MAX 2147483646`

### 5.253.1 Detailed Description

Constants that define various data type limits.

### 5.253.2 Macro Definition Documentation

#### 5.253.2.1 `#define CHAR_BIT 8`

The number of bits in the char type

#### 5.253.2.2 `#define CHAR_MAX 127`

The maximum value of the char type

#### 5.253.2.3 `#define CHAR_MIN -128`

The minimum value of the char type

#### 5.253.2.4 `#define INT_MAX 32767`

The maximum value of the int type

#### 5.253.2.5 `#define INT_MIN -32768`

The minimum value of the int type

#### 5.253.2.6 `#define LONG_MAX 2147483647`

The maximum value of the long type

5.253.2.7 #define LONG\_MIN -2147483648

The minimum value of the long type

5.253.2.8 #define RAND\_MAX 2147483646

The maximum long random number returned by rand

5.253.2.9 #define SCHAR\_MAX 127

The maximum value of the signed char type

5.253.2.10 #define SCHAR\_MIN -128

The minimum value of the signed char type

5.253.2.11 #define SHRT\_MAX 32767

The maximum value of the short type

5.253.2.12 #define SHRT\_MIN -32768

The minimum value of the short type

5.253.2.13 #define UCHAR\_MAX 255

The maximum value of the unsigned char type

5.253.2.14 #define UINT\_MAX 65535

The maximum value of the unsigned int type

5.253.2.15 #define ULONG\_MAX 4294967295

The maximum value of the unsigned long type

5.253.2.16 #define USHRT\_MAX 65535

The maximum value of the unsigned short type

## 5.254 Graphics library begin modes

Constants that are used to specify the polygon surface begin mode.

### Macros

- #define GL\_POLYGON 1
- #define GL\_LINE 2
- #define GL\_POINT 3
- #define GL\_CIRCLE 4

### 5.254.1 Detailed Description

Constants that are used to specify the polygon surface begin mode.

### 5.254.2 Macro Definition Documentation

#### 5.254.2.1 #define GL\_CIRCLE 4

Use circle mode.

#### 5.254.2.2 #define GL\_LINE 2

Use line mode.

#### 5.254.2.3 #define GL\_POINT 3

Use point mode.

#### 5.254.2.4 #define GL\_POLYGON 1

Use polygon mode.

## 5.255 Graphics library actions

Constants that are used to specify a graphics library action.

### Macros

- #define GL\_TRANSLATE\_X 1
- #define GL\_TRANSLATE\_Y 2
- #define GL\_TRANSLATE\_Z 3
- #define GL\_ROTATE\_X 4
- #define GL\_ROTATE\_Y 5
- #define GL\_ROTATE\_Z 6
- #define GL\_SCALE\_X 7
- #define GL\_SCALE\_Y 8
- #define GL\_SCALE\_Z 9

### 5.255.1 Detailed Description

Constants that are used to specify a graphics library action.

### 5.255.2 Macro Definition Documentation

#### 5.255.2.1 #define GL\_ROTATE\_X 4

Rotate around the X axis.

#### 5.255.2.2 #define GL\_ROTATE\_Y 5

Rotate around the Y axis.

#### 5.255.2.3 #define GL\_ROTATE\_Z 6

Rotate around the Z axis.

#### 5.255.2.4 #define GL\_SCALE\_X 7

Scale along the X axis.

#### 5.255.2.5 #define GL\_SCALE\_Y 8

Scale along the Y axis.

#### 5.255.2.6 #define GL\_SCALE\_Z 9

Scale along the Z axis.

#### 5.255.2.7 #define GL\_TRANSLATE\_X 1

Translate along the X axis.

#### 5.255.2.8 #define GL\_TRANSLATE\_Y 2

Translate along the Y axis.

**5.255.2.9 #define GL\_TRANSLATE\_Z 3**

Translate along the Z axis.

## 5.256 Graphics library settings

Constants that are used to configure the graphics library settings.

### Macros

- #define GL\_CIRCLE\_SIZE 1
- #define GL\_CULL\_MODE 2
- #define GL\_CAMERA\_DEPTH 3
- #define GL\_ZOOM\_FACTOR 4

### 5.256.1 Detailed Description

Constants that are used to configure the graphics library settings.

### 5.256.2 Macro Definition Documentation

#### 5.256.2.1 #define GL\_CAMERA\_DEPTH 3

Set the camera depth.

#### 5.256.2.2 #define GL\_CIRCLE\_SIZE 1

Set the circle size.

#### 5.256.2.3 #define GL\_CULL\_MODE 2

Set the cull mode.

#### 5.256.2.4 #define GL\_ZOOM\_FACTOR 4

Set the zoom factor.

## 5.257 Graphics library cull mode

Constants to use when setting the graphics library cull mode.

### Macros

- #define GL\_CULL\_BACK 2
- #define GL\_CULL\_FRONT 3
- #define GL\_CULL\_NONE 4

#### 5.257.1 Detailed Description

Constants to use when setting the graphics library cull mode.

#### 5.257.2 Macro Definition Documentation

##### 5.257.2.1 #define GL\_CULL\_BACK 2

Cull lines in back.

##### 5.257.2.2 #define GL\_CULL\_FRONT 3

Cull lines in front.

##### 5.257.2.3 #define GL\_CULL\_NONE 4

Do not cull any lines.

## 6 File Documentation

### 6.1 NBCAPIDocs.h File Reference

Additional documentation for the NBC API.

```
#include "NXTDefs.h"
```

#### 6.1.1 Detailed Description

Additional documentation for the NBC API. [NBCAPIDocs.h](#) contains additional documentation for the NBC API

License:

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Initial Developer of this code is John Hansen. Portions created by John Hansen are Copyright (C) 2009-2012 John Hansen. All Rights Reserved.

Author

John Hansen ([bricxcc\\_at\\_comcast.net](mailto:bricxcc_at_comcast.net))

Date

2013-02-17

Version

11

### 6.2 NBCCommon.h File Reference

Constants and macros common to both NBC and NXC.

Macros

- #define **TRUE** 1
- #define **FALSE** 0
- #define **NA** 0xFFFF
- #define **RC\_PROP\_BTNOFF** 0x0
- #define **RC\_PROP\_SOUND\_LEVEL** 0x1
- #define **RC\_PROP\_SLEEP\_TIMEOUT** 0x2
- #define **RC\_PROP\_DEBUGGING** 0xF
- #define **VT\_UBYTE** 0x01
- #define **VT\_SBYTE** 0x02
- #define **VT\_UWORD** 0x03
- #define **VT\_SWORD** 0x04
- #define **VT ULONG** 0x05

- #define VT\_SLONG 0x06
- #define VT\_STRUCT 0x08
- #define VT\_MUTEX 0x09
- #define VT\_FLOAT 0x0A
- #define VT\_A1\_UBYTE 0x11
- #define VT\_A1\_SBYTE 0x12
- #define VT\_A1\_UWORD 0x13
- #define VT\_A1\_SWORD 0x14
- #define VT\_A1 ULONG 0x15
- #define VT\_A1\_SLONG 0x16
- #define VT\_A1\_STRUCT 0x17
- #define VT\_A1\_FLOAT 0x1A
- #define VT\_A2\_UBYTE 0x21
- #define VT\_A2\_SBYTE 0x22
- #define VT\_A2\_UWORD 0x23
- #define VT\_A2\_SWORD 0x24
- #define VT\_A2 ULONG 0x25
- #define VT\_A2\_SLONG 0x26
- #define VT\_A2\_STRUCT 0x27
- #define VT\_A2\_FLOAT 0x2A
- #define VT\_ARRAY\_MASK 0xF0
- #define OPARR\_SUM 0x00
- #define OPARR\_MEAN 0x01
- #define OPARR\_SUMSQR 0x02
- #define OPARR\_STD 0x03
- #define OPARR\_MIN 0x04
- #define OPARR\_MAX 0x05
- #define OPARR\_SORT 0x06
- #define OPARR\_TOUPPER 0x07
- #define OPARR\_TOLOWER 0x08
- #define PI 3.141593
- #define RADIANS\_PER\_DEGREE PI/180
- #define DEGREES\_PER\_RADIAN 180/PI
- #define FileOpenRead 0
- #define FileOpenWrite 1
- #define FileOpenAppend 2
- #define FileRead 3
- #define FileWrite 4
- #define FileClose 5
- #define FileResolveHandle 6
- #define FileRename 7
- #define FileDelete 8
- #define SoundPlayFile 9
- #define SoundPlayTone 10
- #define SoundGetState 11
- #define SoundSetState 12
- #define DrawText 13
- #define DrawPoint 14
- #define DrawLine 15
- #define DrawCircle 16
- #define DrawRect 17

- #define DrawGraphic 18
- #define SetScreenMode 19
- #define ReadButton 20
- #define CommLSWrite 21
- #define CommLSRead 22
- #define CommLSCheckStatus 23
- #define RandomNumber 24
- #define GetStartTick 25
- #define MessageWrite 26
- #define MessageRead 27
- #define CommBTCheckStatus 28
- #define CommBTWrite 29
- #define CommBTRead 30
- #define KeepAlive 31
- #define IOMapRead 32
- #define IOMapWrite 33
- #define ColorSensorRead 34
- #define CommBTOnOff 35
- #define CommBTConnection 36
- #define CommHSWrite 37
- #define CommHSRead 38
- #define CommHSCheckStatus 39
- #define ReadSemData 40
- #define WriteSemData 41
- #define ComputeCalibValue 42
- #define UpdateCalibCacheInfo 43
- #define DatalogWrite 44
- #define DatalogGetTimes 45
- #define SetSleepTimeoutVal 46
- #define ListFiles 47
- #define InputPinFunction 77
- #define IOMapReadByID 78
- #define IOMapWriteByID 79
- #define DisplayExecuteFunction 80
- #define CommExecuteFunction 81
- #define LoaderExecuteFunction 82
- #define FileFindFirst 83
- #define FileFindNext 84
- #define FileOpenWriteLinear 85
- #define FileOpenWriteNonLinear 86
- #define FileOpenReadLinear 87
- #define CommHSCControl 88
- #define CommLSWriteEx 89
- #define FileSeek 90
- #define FileResize 91
- #define DrawGraphicArray 92
- #define DrawPolygon 93
- #define DrawEllipse 94
- #define DrawFont 95
- #define MemoryManager 96
- #define ReadLastResponse 97

- #define FileTell 98
- #define RandomEx 99
- #define LCD\_LINE8 0
- #define LCD\_LINE7 8
- #define LCD\_LINE6 16
- #define LCD\_LINE5 24
- #define LCD\_LINE4 32
- #define LCD\_LINE3 40
- #define LCD\_LINE2 48
- #define LCD\_LINE1 56
- #define MS\_1 1
- #define MS\_2 2
- #define MS\_3 3
- #define MS\_4 4
- #define MS\_5 5
- #define MS\_6 6
- #define MS\_7 7
- #define MS\_8 8
- #define MS\_9 9
- #define MS\_10 10
- #define MS\_20 20
- #define MS\_30 30
- #define MS\_40 40
- #define MS\_50 50
- #define MS\_60 60
- #define MS\_70 70
- #define MS\_80 80
- #define MS\_90 90
- #define MS\_100 100
- #define MS\_150 150
- #define MS\_200 200
- #define MS\_250 250
- #define MS\_300 300
- #define MS\_350 350
- #define MS\_400 400
- #define MS\_450 450
- #define MS\_500 500
- #define MS\_600 600
- #define MS\_700 700
- #define MS\_800 800
- #define MS\_900 900
- #define SEC\_1 1000
- #define SEC\_2 2000
- #define SEC\_3 3000
- #define SEC\_4 4000
- #define SEC\_5 5000
- #define SEC\_6 6000
- #define SEC\_7 7000
- #define SEC\_8 8000
- #define SEC\_9 9000
- #define SEC\_10 10000

```
• #define SEC_15 15000
• #define SEC_20 20000
• #define SEC_30 30000
• #define MIN_1 60000
• #define NOTE_WHOLE 1000
• #define NOTE_HALF (NOTE_WHOLE/2)
• #define NOTE_QUARTER (NOTE_WHOLE/4)
• #define NOTE_EIGHT (NOTE_WHOLE/8)
• #define NOTE_SIXTEEN (NOTE_WHOLE/16)
• #define MAILBOX1 0
• #define MAILBOX2 1
• #define MAILBOX3 2
• #define MAILBOX4 3
• #define MAILBOX5 4
• #define MAILBOX6 5
• #define MAILBOX7 6
• #define MAILBOX8 7
• #define MAILBOX9 8
• #define MAILBOX10 9
• #define CommandModuleName "Command.mod"
• #define IOCtrlModuleName "IOCtrl.mod"
• #define LoaderModuleName "Loader.mod"
• #define SoundModuleName "Sound.mod"
• #define ButtonModuleName "Button.mod"
• #define UIModuleName "Ui.mod"
• #define InputModuleName "Input.mod"
• #define OutputModuleName "Output.mod"
• #define LowSpeedModuleName "Low Speed.mod"
• #define DisplayModuleName "Display.mod"
• #define CommModuleName "Comm.mod"
• #define CommandModuleID 0x00010001
• #define IOCtrlModuleID 0x00060001
• #define LoaderModuleID 0x00090001
• #define SoundModuleID 0x00080001
• #define ButtonModuleID 0x00040001
• #define UIModuleID 0x000C0001
• #define InputModuleID 0x00030001
• #define OutputModuleID 0x00020001
• #define LowSpeedModuleID 0x000B0001
• #define DisplayModuleID 0x000A0001
• #define CommModuleID 0x00050001
• #define STAT_MSG_EMPTY_MAILBOX 64
• #define STAT_COMM_PENDING 32
• #define POOL_MAX_SIZE 32768
• #define TIMES_UP 6
• #define ROTATE_QUEUE 5
• #define STOP_REQ 4
• #define BREAKOUT_REQ 3
• #define CLUMP_SUSPEND 2
• #define CLUMP_DONE 1
• #define NO_ERR 0
```

- #define ERR\_ARG -1
- #define ERR\_INSTR -2
- #define ERR\_FILE -3
- #define ERR\_VER -4
- #define ERR\_MEM -5
- #define ERR\_BAD\_PTR -6
- #define ERR\_CLUMP\_COUNT -7
- #define ERR\_NO\_CODE -8
- #define ERR\_INSANE\_OFFSET -9
- #define ERR\_BAD\_POOL\_SIZE -10
- #define ERR\_LOADER\_ERR -11
- #define ERR\_SPOTCHECK\_FAIL -12
- #define ERR\_NO\_ACTIVE\_CLUMP -13
- #define ERR\_DEFAULT\_OFFSETS -14
- #define ERR\_MEMMGR\_FAIL -15
- #define ERR\_NON\_FATAL -16
- #define ERR\_INVALID\_PORT -16
- #define ERR\_INVALID\_FIELD -17
- #define ERR\_INVALID\_QUEUE -18
- #define ERR\_INVALID\_SIZE -19
- #define ERR\_NO\_PROG -20
- #define ERR\_COMM\_CHAN\_NOT\_READY -32
- #define ERR\_COMM\_CHAN\_INVALID -33
- #define ERR\_COMM\_BUFFER\_FULL -34
- #define ERR\_COMM\_BUS\_ERR -35
- #define ERR\_RC\_ILLEGAL\_VAL -64
- #define ERR\_RC\_BAD\_PACKET -65
- #define ERR\_RC\_UNKNOWN\_CMD -66
- #define ERR\_RC\_FAILED -67
- #define PROG\_IDLE 0
- #define PROG\_OK 1
- #define PROG\_RUNNING 2
- #define PROG\_ERROR 3
- #define PROG\_ABORT 4
- #define PROG\_RESET 5
- #define CommandOffsetFormatString 0
- #define CommandOffsetPRCHandler 16
- #define CommandOffsetTick 20
- #define CommandOffsetOffsetDS 24
- #define CommandOffsetOffsetDVA 26
- #define CommandOffsetProgStatus 28
- #define CommandOffsetAwake 29
- #define CommandOffsetActivateFlag 30
- #define CommandOffsetDeactivateFlag 31
- #define CommandOffsetFileName 32
- #define CommandOffsetMemoryPool 52
- #define CommandOffsetSyncTime 32820
- #define CommandOffsetSyncTick 32824
- #define IOCTL\_POWERDOWN 0x5A00
- #define IOCTL\_BOOT 0xA55A
- #define IOCtrlOffsetPowerOn 0

- #define LoaderOffsetPFunc 0
- #define LoaderOffsetFreeUserFlash 4
- #define EOF -1
- #define NULL 0
- #define LDR\_SUCCESS 0x0000
- #define LDR\_INPROGRESS 0x0001
- #define LDR\_REQPIN 0x0002
- #define LDR\_NOMOREHANDLES 0x8100
- #define LDR\_NOSPACE 0x8200
- #define LDR\_NOMOREFILES 0x8300
- #define LDR\_EOFEXPECTED 0x8400
- #define LDR\_ENDOFFILE 0x8500
- #define LDR\_NOTLINEARFILE 0x8600
- #define LDR\_FILENOFOUND 0x8700
- #define LDR\_HANDLEALREADYCLOSED 0x8800
- #define LDR\_NOLINEARSPACE 0x8900
- #define LDR\_UNDEFINEDERROR 0x8A00
- #define LDR\_FILEISBUSY 0x8B00
- #define LDR\_NOWRITEBUFFERS 0x8C00
- #define LDR\_APPENDNOTPOSSIBLE 0x8D00
- #define LDR\_FILEISFULL 0x8E00
- #define LDR\_FILEEXISTS 0x8F00
- #define LDR\_MODULENOTFOUND 0x9000
- #define LDR\_OUTOFBOUNDARY 0x9100
- #define LDR\_ILLEGALFILENAME 0x9200
- #define LDR\_ILLEGALHANDLE 0x9300
- #define LDR\_BTBUSY 0x9400
- #define LDR\_BTCONNECTFAIL 0x9500
- #define LDR\_BTTIMEOUT 0x9600
- #define LDR\_FILETX\_TIMEOUT 0x9700
- #define LDR\_FILETX\_DSTEXISTS 0x9800
- #define LDR\_FILETX\_SRCMISSING 0x9900
- #define LDR\_FILETX\_STREAMERROR 0x9A00
- #define LDR\_FILETX\_CLOSEERROR 0x9B00
- #define LDR\_INVALIDSEEK 0x9C00
- #define LDR\_CMD\_OPENREAD 0x80
- #define LDR\_CMD\_OPENWRITE 0x81
- #define LDR\_CMD\_READ 0x82
- #define LDR\_CMD\_WRITE 0x83
- #define LDR\_CMD\_CLOSE 0x84
- #define LDR\_CMD\_DELETE 0x85
- #define LDR\_CMD\_FINDFIRST 0x86
- #define LDR\_CMD\_FINDNEXT 0x87
- #define LDR\_CMD\_VERSIONS 0x88
- #define LDR\_CMD\_OPENWITELINEAR 0x89
- #define LDR\_CMD\_OPENREADLINEAR 0x8A
- #define LDR\_CMD\_OPENWRITEDATA 0x8B
- #define LDR\_CMD\_OPENAPPENDDATA 0x8C
- #define LDR\_CMD\_CROPDATAFILE 0x8D
- #define LDR\_CMD\_FINDFIRSTMODULE 0x90
- #define LDR\_CMD\_FINDNEXTMODULE 0x91

- #define LDR\_CMD\_CLOSEMODHANDLE 0x92
- #define LDR\_CMD\_IOMAPREAD 0x94
- #define LDR\_CMD\_IOMAPWRITE 0x95
- #define LDR\_CMD\_BOOTCMD 0x97
- #define LDR\_CMD\_SETBRICKNAME 0x98
- #define LDR\_CMD\_BTGETADR 0x9A
- #define LDR\_CMD\_DEVICEINFO 0x9B
- #define LDR\_CMD\_DELETEUSERFLASH 0xA0
- #define LDR\_CMD\_POLLCMDLEN 0xA1
- #define LDR\_CMD\_POLLCMD 0xA2
- #define LDR\_CMD\_RENAMEFILE 0xA3
- #define LDR\_CMD\_BTFACTORYRESET 0xA4
- #define LDR\_CMD\_RESIZEDATAFILE 0xD0
- #define LDR\_CMD\_SEEKFROMSTART 0xD1
- #define LDR\_CMD\_SEEKFROMCURRENT 0xD2
- #define LDR\_CMD\_SEEKFROMEND 0xD3
- #define SOUND\_FLAGS\_IDLE 0x00
- #define SOUND\_FLAGS\_UPDATE 0x01
- #define SOUND\_FLAGS\_RUNNING 0x02
- #define SOUND\_STATE\_IDLE 0x00
- #define SOUND\_STATE\_FILE 0x02
- #define SOUND\_STATE\_TONE 0x03
- #define SOUND\_STATE\_STOP 0x04
- #define SOUND\_MODE\_ONCE 0x00
- #define SOUND\_MODE\_LOOP 0x01
- #define SOUND\_MODE\_TONE 0x02
- #define SoundOffsetFreq 0
- #define SoundOffsetDuration 2
- #define SoundOffsetSampleRate 4
- #define SoundOffsetSoundFilename 6
- #define SoundOffsetFlags 26
- #define SoundOffsetState 27
- #define SoundOffsetMode 28
- #define SoundOffsetVolume 29
- #define FREQUENCY\_MIN 220
- #define FREQUENCY\_MAX 14080
- #define SAMPLERATE\_MIN 2000
- #define SAMPLERATE\_DEFAULT 8000
- #define SAMPLERATE\_MAX 16000
- #define TONE\_C3 131
- #define TONE\_CS3 139
- #define TONE\_D3 147
- #define TONE\_DS3 156
- #define TONE\_E3 165
- #define TONE\_F3 175
- #define TONE\_FS3 185
- #define TONE\_G3 196
- #define TONE\_GS3 208
- #define TONE\_A3 220
- #define TONE\_AS3 233
- #define TONE\_B3 247

- #define TONE\_C4 262
- #define TONE\_CS4 277
- #define TONE\_D4 294
- #define TONE\_DS4 311
- #define TONE\_E4 330
- #define TONE\_F4 349
- #define TONE\_FS4 370
- #define TONE\_G4 392
- #define TONE\_GS4 415
- #define TONE\_A4 440
- #define TONE\_AS4 466
- #define TONE\_B4 494
- #define TONE\_C5 523
- #define TONE\_CS5 554
- #define TONE\_D5 587
- #define TONE\_DS5 622
- #define TONE\_E5 659
- #define TONE\_F5 698
- #define TONE\_FS5 740
- #define TONE\_G5 784
- #define TONE\_GS5 831
- #define TONE\_A5 880
- #define TONE\_AS5 932
- #define TONE\_B5 988
- #define TONE\_C6 1047
- #define TONE\_CS6 1109
- #define TONE\_D6 1175
- #define TONE\_DS6 1245
- #define TONE\_E6 1319
- #define TONE\_F6 1397
- #define TONE\_FS6 1480
- #define TONE\_G6 1568
- #define TONE\_GS6 1661
- #define TONE\_A6 1760
- #define TONE\_AS6 1865
- #define TONE\_B6 1976
- #define TONE\_C7 2093
- #define TONE\_CS7 2217
- #define TONE\_D7 2349
- #define TONE\_DS7 2489
- #define TONE\_E7 2637
- #define TONE\_F7 2794
- #define TONE\_FS7 2960
- #define TONE\_G7 3136
- #define TONE\_GS7 3322
- #define TONE\_A7 3520
- #define TONE\_AS7 3729
- #define TONE\_B7 3951
- #define BTN1 0
- #define BTN2 1
- #define BTN3 2

- #define **BTN4** 3
- #define **BTNEXTIT** **BTN1**
- #define **BTNRIGHT** **BTN2**
- #define **BTNLEFT** **BTN3**
- #define **BTNCENTER** **BTN4**
- #define **NO\_OF\_BTNS** 4
- #define **BTNSTATE\_PRESSED\_EV** 0x01
- #define **BTNSTATE\_SHORT\_RELEASED\_EV** 0x02
- #define **BTNSTATE\_LONG\_PRESSED\_EV** 0x04
- #define **BTNSTATE\_LONG\_RELEASED\_EV** 0x08
- #define **BTNSTATE\_PRESSED\_STATE** 0x80
- #define **BTNSTATE\_NONE** 0x10
- #define **ButtonOffsetPressedCnt**(b) (((b)\*8)+0)
- #define **ButtonOffsetLongPressCnt**(b) (((b)\*8)+1)
- #define **ButtonOffsetShortRelCnt**(b) (((b)\*8)+2)
- #define **ButtonOffsetLongRelCnt**(b) (((b)\*8)+3)
- #define **ButtonOffsetRelCnt**(b) (((b)\*8)+4)
- #define **ButtonOffsetState**(b) ((b)+32)
- #define **UI\_FLAGS\_UPDATE** 0x01
- #define **UI\_FLAGS\_DISABLE\_LEFT\_RIGHT\_ENTER** 0x02
- #define **UI\_FLAGS\_DISABLE\_EXIT** 0x04
- #define **UI\_FLAGS\_REDRAW\_STATUS** 0x08
- #define **UI\_FLAGS\_RESET\_SLEEP\_TIMER** 0x10
- #define **UI\_FLAGS\_EXECUTE\_LMS\_FILE** 0x20
- #define **UI\_FLAGS\_BUSY** 0x40
- #define **UI\_FLAGS\_ENABLE\_STATUS\_UPDATE** 0x80
- #define **UI\_STATE\_INIT\_DISPLAY** 0
- #define **UI\_STATE\_INIT\_LOW\_BATTERY** 1
- #define **UI\_STATE\_INIT\_INTRO** 2
- #define **UI\_STATE\_INIT\_WAIT** 3
- #define **UI\_STATE\_INIT\_MENU** 4
- #define **UI\_STATE\_NEXT\_MENU** 5
- #define **UI\_STATE\_DRAW\_MENU** 6
- #define **UI\_STATE\_TEST\_BUTTONS** 7
- #define **UI\_STATE\_LEFT\_PRESSED** 8
- #define **UI\_STATE\_RIGHT\_PRESSED** 9
- #define **UI\_STATE\_ENTER\_PRESSED** 10
- #define **UI\_STATE\_EXIT\_PRESSED** 11
- #define **UI\_STATE\_CONNECT\_REQUEST** 12
- #define **UI\_STATE\_EXECUTE\_FILE** 13
- #define **UI\_STATE\_EXECUTING\_FILE** 14
- #define **UI\_STATE\_LOW\_BATTERY** 15
- #define **UI\_STATE\_BT\_ERROR** 16
- #define **UI\_BUTTON\_NONE** 0
- #define **UI\_BUTTON\_LEFT** 1
- #define **UI\_BUTTON\_ENTER** 2
- #define **UI\_BUTTON\_RIGHT** 3
- #define **UI\_BUTTON\_EXIT** 4
- #define **UI\_BT\_STATE\_VISIBLE** 0x01
- #define **UI\_BT\_STATE\_CONNECTED** 0x02
- #define **UI\_BT\_STATE\_OFF** 0x04

- #define UI\_BT\_ERROR\_ATTENTION 0x08
- #define UI\_BT\_CONNECT\_REQUEST 0x40
- #define UI\_BT\_PIN\_REQUEST 0x80
- #define UI\_VM\_IDLE 0
- #define UI\_VM\_RUN\_FREE 1
- #define UI\_VM\_RUN\_SINGLE 2
- #define UI\_VM\_RUN\_PAUSE 3
- #define UI\_VM\_RESET1 4
- #define UI\_VM\_RESET2 5
- #define UIOffsetPMenu 0
- #define UIOffsetBatteryVoltage 4
- #define UIOffsetLMSfilename 6
- #define UIOffsetFlags 26
- #define UIOffsetState 27
- #define UIOffsetButton 28
- #define UIOffsetRunState 29
- #define UIOffsetBatteryState 30
- #define UIOffsetBluetoothState 31
- #define UIOffsetUsbState 32
- #define UIOffsetSleepTimeout 33
- #define UIOffsetSleepTimer 34
- #define UIOffsetRechargeable 35
- #define UIOffsetVolume 36
- #define UIOffsetError 37
- #define UIOffsetOBPPointer 38
- #define UIOffsetForceOff 39
- #define UIOffsetAbortFlag 40
- #define IN\_1 0x00
- #define IN\_2 0x01
- #define IN\_3 0x02
- #define IN\_4 0x03
- #define IN\_TYPE\_NO\_SENSOR 0x00
- #define IN\_TYPE\_SWITCH 0x01
- #define IN\_TYPE\_TEMPERATURE 0x02
- #define IN\_TYPE\_REFLECTION 0x03
- #define IN\_TYPE\_ANGLE 0x04
- #define IN\_TYPE\_LIGHT\_ACTIVE 0x05
- #define IN\_TYPE\_LIGHT\_INACTIVE 0x06
- #define IN\_TYPE\_SOUND\_DB 0x07
- #define IN\_TYPE\_SOUND\_DBA 0x08
- #define IN\_TYPE\_CUSTOM 0x09
- #define IN\_TYPE\_LOWSPEED 0x0A
- #define IN\_TYPE\_LOWSPEED\_9V 0x0B
- #define IN\_TYPE\_HISPEED 0x0C
- #define IN\_TYPE\_COLORFULL 0x0D
- #define IN\_TYPE\_COLORRED 0x0E
- #define IN\_TYPE\_COLORGREEN 0x0F
- #define IN\_TYPE\_COLORBLUE 0x10
- #define IN\_TYPE\_COLORNONE 0x11
- #define IN\_TYPE\_COLOREXIT 0x12
- #define IN\_MODE\_RAW 0x00

- #define IN\_MODE\_BOOLEAN 0x20
- #define IN\_MODE\_TRANSITIONCNT 0x40
- #define IN\_MODE\_PERIODCOUNTER 0x60
- #define IN\_MODE\_PCTFULLSCALE 0x80
- #define IN\_MODE\_CELSIUS 0xA0
- #define IN\_MODE\_FAHRENHEIT 0xC0
- #define IN\_MODE\_ANGLESTEP 0xE0
- #define IN\_MODE\_SLOPEMASK 0x1F
- #define IN\_MODE\_MODEMASK 0xE0
- #define TypeField 0
- #define InputModeField 1
- #define RawValueField 2
- #define NormalizedValueField 3
- #define ScaledValueField 4
- #define InvalidDataField 5
- #define INPUT\_DIGIO 0x01
- #define INPUT\_DIGI1 0x02
- #define INPUT\_CUSTOMINACTIVE 0x00
- #define INPUT\_CUSTOM9V 0x01
- #define INPUT\_CUSTOMACTIVE 0x02
- #define INPUT\_INVALID\_DATA 0x01
- #define INPUT\_RED 0
- #define INPUT\_GREEN 1
- #define INPUT\_BLUE 2
- #define INPUT\_BLANK 3
- #define INPUT\_NO\_OF\_COLORS 4
- #define INPUT\_BLACKCOLOR 1
- #define INPUT\_BLUECOLOR 2
- #define INPUT\_GREENCOLOR 3
- #define INPUT\_YELLOWCOLOR 4
- #define INPUT\_REDCOLOR 5
- #define INPUT\_WHITECOLOR 6
- #define INPUT\_SENSORCAL 0x01
- #define INPUT\_SENSOROFF 0x02
- #define INPUT\_RUNNINGCAL 0x20
- #define INPUT\_STARTCAL 0x40
- #define INPUT\_RESETCAL 0x80
- #define INPUT\_CAL\_POINT\_0 0
- #define INPUT\_CAL\_POINT\_1 1
- #define INPUT\_CAL\_POINT\_2 2
- #define INPUT\_NO\_OF\_POINTS 3
- #define InputOffsetCustomZeroOffset(p) (((p)\*20)+0)
- #define InputOffsetADRaw(p) (((p)\*20)+2)
- #define InputOffsetSensorRaw(p) (((p)\*20)+4)
- #define InputOffsetSensorValue(p) (((p)\*20)+6)
- #define InputOffsetSensorType(p) (((p)\*20)+8)
- #define InputOffsetSensorMode(p) (((p)\*20)+9)
- #define InputOffsetSensorBoolean(p) (((p)\*20)+10)
- #define InputOffsetDigiPinsDir(p) (((p)\*20)+11)
- #define InputOffsetDigiPinsIn(p) (((p)\*20)+12)
- #define InputOffsetDigiPinsOut(p) (((p)\*20)+13)

- #define InputOffsetCustomPctFullScale(p) (((p)\*20)+14)
- #define InputOffsetCustomActiveStatus(p) (((p)\*20)+15)
- #define InputOffsetInvalidData(p) (((p)\*20)+16)
- #define InputOffsetColorCalibration(p, np, nc) (80+((p)\*84)+0+((np)\*16)+((nc)\*4))
- #define InputOffsetColorCalLimits(p, np) (80+((p)\*84)+48+((np)\*2))
- #define InputOffsetColorADRaw(p, nc) (80+((p)\*84)+52+((nc)\*2))
- #define InputOffsetColorSensorRaw(p, nc) (80+((p)\*84)+60+((nc)\*2))
- #define InputOffsetColorSensorValue(p, nc) (80+((p)\*84)+68+((nc)\*2))
- #define InputOffsetColorBoolean(p, nc) (80+((p)\*84)+76+((nc)\*2))
- #define InputOffsetColorCalibrationState(p) (80+((p)\*84)+80)
- #define INPUT\_PINCMD\_DIR 0x00
- #define INPUT\_PINCMD\_SET 0x01
- #define INPUT\_PINCMD\_CLEAR 0x02
- #define INPUT\_PINCMD\_READ 0x03
- #define INPUT\_PINCMD\_MASK 0x03
- #define INPUT\_PINCMD\_WAIT(\_usec) (( \_usec)<<2)
- #define INPUT\_PINDIR\_OUTPUT 0x00
- #define INPUT\_PINDIR\_INPUT 0x04
- #define OUT\_A 0x00
- #define OUT\_B 0x01
- #define OUT\_C 0x02
- #define OUT\_AB 0x03
- #define OUT\_AC 0x04
- #define OUT\_BC 0x05
- #define OUT\_ABC 0x06
- #define PID\_0 0
- #define PID\_1 32
- #define PID\_2 64
- #define PID\_3 96
- #define PID\_4 128
- #define PID\_5 160
- #define PID\_6 192
- #define PID\_7 224
- #define UF\_UPDATE\_MODE 0x01
- #define UF\_UPDATE\_SPEED 0x02
- #define UF\_UPDATE\_TACHO\_LIMIT 0x04
- #define UF\_UPDATE\_RESET\_COUNT 0x08
- #define UF\_UPDATE\_PID\_VALUES 0x10
- #define UF\_UPDATE\_RESET\_BLOCK\_COUNT 0x20
- #define UF\_UPDATE\_RESET\_ROTATION\_COUNT 0x40
- #define UF\_PENDING\_UPDATES 0x80
- #define RESET\_NONE 0x00
- #define RESET\_COUNT 0x08
- #define RESET\_BLOCK\_COUNT 0x20
- #define RESET\_ROTATION\_COUNT 0x40
- #define RESET\_BLOCKANDTACHO 0x28
- #define RESET\_ALL 0x68
- #define OUT\_MODE\_COAST 0x00
- #define OUT\_MODE\_MOTORON 0x01
- #define OUT\_MODE\_BRAKE 0x02
- #define OUT\_MODE\_REGULATED 0x04

- #define OUT\_MODE\_REGMETHOD 0xF0
- #define OUT\_OPTION\_HOLDATLIMIT 0x10
- #define OUT\_OPTION\_RAMPDOWNTOLIMIT 0x20
- #define OUT\_REGOPTION\_NO\_SATURATION 0x01
- #define OUT\_RUNSTATE\_IDLE 0x00
- #define OUT\_RUNSTATE\_RAMPUP 0x10
- #define OUT\_RUNSTATE\_RUNNING 0x20
- #define OUT\_RUNSTATE\_RAMPDOWN 0x40
- #define OUT\_RUNSTATE\_HOLD 0x60
- #define OUT\_REGMODE\_IDLE 0
- #define OUT\_REGMODE\_SPEED 1
- #define OUT\_REGMODE\_SYNC 2
- #define OUT\_REGMODE\_POS 4
- #define UpdateFlagsField 0
  - Update flags field.*
- #define OutputModeField 1
  - Mode field.*
- #define PowerField 2
  - Power field.*
- #define ActualSpeedField 3
  - Actual speed field.*
- #define TachoCountField 4
  - Internal tachometer count field.*
- #define TachoLimitField 5
  - Tachometer limit field.*
- #define RunStateField 6
  - Run state field.*
- #define TurnRatioField 7
  - Turn ratio field.*
- #define RegModeField 8
  - Regulation mode field.*
- #define OverloadField 9
  - Overload field.*
- #define RegPValueField 10
  - Proportional field.*
- #define RegIValueField 11
  - Integral field.*
- #define RegDValueField 12
  - Derivative field.*
- #define BlockTachoCountField 13
  - NXT-G block tachometer count field.*
- #define RotationCountField 14
  - Rotation counter field.*
- #define OutputOptionsField 15
  - Options field.*
- #define MaxSpeedField 16
  - MaxSpeed field.*
- #define MaxAccelerationField 17

- MaxAcceleration field.*
- #define OutputOffsetTachoCount(p) (((p)\*32)+0)
  - #define OutputOffsetBlockTachoCount(p) (((p)\*32)+4)
  - #define OutputOffsetRotationCount(p) (((p)\*32)+8)
  - #define OutputOffsetTachoLimit(p) (((p)\*32)+12)
  - #define OutputOffsetMotorRPM(p) (((p)\*32)+16)
  - #define OutputOffsetFlags(p) (((p)\*32)+18)
  - #define OutputOffsetMode(p) (((p)\*32)+19)
  - #define OutputOffsetSpeed(p) (((p)\*32)+20)
  - #define OutputOffsetActualSpeed(p) (((p)\*32)+21)
  - #define OutputOffsetRegPParameter(p) (((p)\*32)+22)
  - #define OutputOffsetRegIParameter(p) (((p)\*32)+23)
  - #define OutputOffsetRegDParameter(p) (((p)\*32)+24)
  - #define OutputOffsetRunState(p) (((p)\*32)+25)
  - #define OutputOffsetRegMode(p) (((p)\*32)+26)
  - #define OutputOffsetOverloaded(p) (((p)\*32)+27)
  - #define OutputOffsetSyncTurnParameter(p) (((p)\*32)+28)
  - #define OutputOffsetOptions(p) (((p)\*32)+29)
  - #define OutputOffsetMaxSpeed(p) (((p)\*32)+30)
  - #define OutputOffsetMaxAccel(p) (((p)\*32)+31)
  - #define OutputOffsetRegulationTime 96
  - #define OutputOffsetRegulationOptions 97
  - #define COM\_CHANNEL\_NONE\_ACTIVE 0x00
  - #define COM\_CHANNEL\_ONE\_ACTIVE 0x01
  - #define COM\_CHANNEL\_TWO\_ACTIVE 0x02
  - #define COM\_CHANNEL\_THREE\_ACTIVE 0x04
  - #define COM\_CHANNEL\_FOUR\_ACTIVE 0x08
  - #define LOWSPEED\_IDLE 0
  - #define LOWSPEED\_INIT 1
  - #define LOWSPEED\_LOAD\_BUFFER 2
  - #define LOWSPEED\_COMMUNICATING 3
  - #define LOWSPEED\_ERROR 4
  - #define LOWSPEED\_DONE 5
  - #define LOWSPEED\_TRANSMITTING 1
  - #define LOWSPEED RECEIVING 2
  - #define LOWSPEED\_DATA\_RECEIVED 3
  - #define LOWSPEED\_NO\_ERROR 0
  - #define LOWSPEED\_CH\_NOT\_READY 1
  - #define LOWSPEED\_TX\_ERROR 2
  - #define LOWSPEED\_RX\_ERROR 3
  - #define LowSpeedOffsetInBufBuf(p) (((p)\*19)+0)
  - #define LowSpeedOffsetInBufInPtr(p) (((p)\*19)+16)
  - #define LowSpeedOffsetInBufOutPtr(p) (((p)\*19)+17)
  - #define LowSpeedOffsetInBufBytesToRx(p) (((p)\*19)+18)
  - #define LowSpeedOffsetOutBufBuf(p) (((p)\*19)+76)
  - #define LowSpeedOffsetOutBufInPtr(p) (((p)\*19)+92)
  - #define LowSpeedOffsetOutBufOutPtr(p) (((p)\*19)+93)
  - #define LowSpeedOffsetOutBufBytesToRx(p) (((p)\*19)+94)
  - #define LowSpeedOffsetMode(p) ((p)+152)
  - #define LowSpeedOffsetChannelState(p) ((p)+156)
  - #define LowSpeedOffsetErrorType(p) ((p)+160)

- #define LowSpeedOffsetState 164
- #define LowSpeedOffsetSpeed 165
- #define LowSpeedOffsetNoRestartOnRead 166
- #define LSREAD\_RESTART\_ALL 0x00
- #define LSREAD\_NO\_RESTART\_1 0x01
- #define LSREAD\_NO\_RESTART\_2 0x02
- #define LSREAD\_NO\_RESTART\_3 0x04
- #define LSREAD\_NO\_RESTART\_4 0x08
- #define LSREAD\_RESTART\_NONE 0x0F
- #define LSREAD\_NO\_RESTART\_MASK 0x10
- #define I2C\_ADDR\_DEFAULT 0x02
- #define I2C\_REG\_VERSION 0x00
- #define I2C\_REG\_VENDOR\_ID 0x08
- #define I2C\_REG\_DEVICE\_ID 0x10
- #define I2C\_REG\_CMD 0x41
- #define LEGO\_ADDR\_US 0x02
- #define LEGO\_ADDR\_TEMP 0x98
- #define LEGO\_ADDR\_EMETER 0x04
- #define US\_CMD\_OFF 0x00
- #define US\_CMD\_SINGLESHOT 0x01
- #define US\_CMD\_CONTINUOUS 0x02
- #define US\_CMD\_EVENTCAPTURE 0x03
- #define US\_CMD\_WARMRESET 0x04
- #define US\_REG\_CM\_INTERVAL 0x40
- #define US\_REG\_ACTUAL\_ZERO 0x50
- #define US\_REG\_SCALE\_FACTOR 0x51
- #define US\_REG\_SCALE\_DIVISOR 0x52
- #define US\_REG\_FACTORY\_ACTUAL\_ZERO 0x11
- #define US\_REG\_FACTORY\_SCALE\_FACTOR 0x12
- #define US\_REG\_FACTORY\_SCALE\_DIVISOR 0x13
- #define US\_REG\_MEASUREMENT\_UNITS 0x14
- #define TEMP\_RES\_9BIT 0x00
- #define TEMP\_RES\_10BIT 0x20
- #define TEMP\_RES\_11BIT 0x40
- #define TEMP\_RES\_12BIT 0x60
- #define TEMP\_SD\_CONTINUOUS 0x00
- #define TEMP\_SD\_SHUTDOWN 0x01
- #define TEMP\_TM\_COMPARATOR 0x00
- #define TEMP\_TM\_INTERRUPT 0x02
- #define TEMP\_OS\_ONESHOT 0x80
- #define TEMP\_FQ\_1 0x00
- #define TEMP\_FQ\_2 0x08
- #define TEMP\_FQ\_4 0x10
- #define TEMP\_FQ\_6 0x18
- #define TEMP\_POL\_LOW 0x00
- #define TEMP\_POL\_HIGH 0x04
- #define TEMP\_REG\_TEMP 0x00
- #define TEMP\_REG\_CONFIG 0x01
- #define TEMP\_REG\_TLOW 0x02
- #define TEMP\_REG\_THIGH 0x03
- #define EMETER\_REG\_VIN 0x0a

- #define EMETER\_REG\_AIN 0x0c
- #define EMETER\_REG\_VOUT 0x0e
- #define EMETER\_REG\_AOUT 0x10
- #define EMETER\_REG\_JOULES 0x12
- #define EMETER\_REG\_WIN 0x14
- #define EMETER\_REG\_WOUT 0x16
- #define I2C\_OPTION\_STANDARD 0x00
- #define I2C\_OPTION\_NORESTART 0x04
- #define I2C\_OPTION\_FAST 0x08
- #define DISPLAY\_ERASE\_ALL 0x00
- #define DISPLAY\_PIXEL 0x01
- #define DISPLAY\_HORIZONTAL\_LINE 0x02
- #define DISPLAY\_VERTICAL\_LINE 0x03
- #define DISPLAY\_CHAR 0x04
- #define DISPLAY\_ERASE\_LINE 0x05
- #define DISPLAY\_FILL\_REGION 0x06
- #define DISPLAY\_FRAME 0x07
- #define DRAW\_OPT\_NORMAL (0x0000)
- #define DRAW\_OPT\_CLEAR\_WHOLE\_SCREEN (0x0001)
- #define DRAW\_OPT\_CLEAR\_EXCEPT\_STATUS\_SCREEN (0x0002)
- #define DRAW\_OPT\_CLEAR\_PIXELS (0x0004)
- #define DRAW\_OPT\_CLEAR (0x0004)
- #define DRAW\_OPT\_INVERT (0x0004)
- #define DRAW\_OPT\_LOGICAL\_COPY (0x0000)
- #define DRAW\_OPT\_LOGICAL\_AND (0x0008)
- #define DRAW\_OPT\_LOGICAL\_OR (0x0010)
- #define DRAW\_OPT\_LOGICAL\_XOR (0x0018)
- #define DRAW\_OPT\_FILL\_SHAPE (0x0020)
- #define DRAW\_OPT\_CLEAR\_SCREEN\_MODES (0x0003)
- #define DRAW\_OPT\_LOGICAL\_OPERATIONS (0x0018)
- #define DRAW\_OPT\_POLYGON\_POLYLINE (0x0400)
- #define DRAW\_OPT\_CLEAR\_LINE (0x0800)
- #define DRAW\_OPT\_CLEAR\_EOL (0x1000)
- #define DRAW\_OPT\_FONT\_DIRECTIONS (0x01C0)
- #define DRAW\_OPT\_FONT\_WRAP (0x0200)
- #define DRAW\_OPT\_FONT\_DIR\_L2RB (0x0000)
- #define DRAW\_OPT\_FONT\_DIR\_L2RT (0x0040)
- #define DRAW\_OPT\_FONT\_DIR\_R2LB (0x0080)
- #define DRAW\_OPT\_FONT\_DIR\_R2LT (0x00C0)
- #define DRAW\_OPT\_FONT\_DIR\_B2TL (0x0100)
- #define DRAW\_OPT\_FONT\_DIR\_B2TR (0x0140)
- #define DRAW\_OPT\_FONT\_DIR\_T2BL (0x0180)
- #define DRAW\_OPT\_FONT\_DIR\_T2BR (0x01C0)
- #define DISPLAY\_ON 0x01
- #define DISPLAY\_REFRESH 0x02
- #define DISPLAY\_POPUP 0x08
- #define DISPLAY\_REFRESH\_DISABLED 0x40
- #define DISPLAY\_BUSY 0x80
- #define DISPLAY\_CONTRAST\_DEFAULT 0x5A
- #define DISPLAY\_CONTRAST\_MAX 0x7F
- #define SCREEN\_MODE\_RESTORE 0x00

- #define SCREEN\_MODE\_CLEAR 0x01
- #define DISPLAY\_HEIGHT 64
- #define DISPLAY\_WIDTH 100
- #define DISPLAY\_MENUICONS\_Y 40
- #define DISPLAY\_MENUICONS\_X\_OFFSET 7
- #define DISPLAY\_MENUICONS\_X\_DIFF 31
- #define TEXTLINE\_1 0
- #define TEXTLINE\_2 1
- #define TEXTLINE\_3 2
- #define TEXTLINE\_4 3
- #define TEXTLINE\_5 4
- #define TEXTLINE\_6 5
- #define TEXTLINE\_7 6
- #define TEXTLINE\_8 7
- #define TEXTLINES 8
- #define MENUICON\_LEFT 0
- #define MENUICON\_CENTER 1
- #define MENUICON\_RIGHT 2
- #define MENUICONS 3
- #define FRAME\_SELECT 0
- #define STATUSTEXT 1
- #define MENUTEXT 2
- #define STEPLINE 3
- #define TOPLINE 4
- #define SPECIALS 5
- #define STATUSICON\_BLUETOOTH 0
- #define STATUSICON\_USB 1
- #define STATUSICON\_VM 2
- #define STATUSICON\_BATTERY 3
- #define STATUSICONS 4
- #define SCREEN\_BACKGROUND 0
- #define SCREEN\_LARGE 1
- #define SCREEN\_SMALL 2
- #define SCREENS 3
- #define BITMAP\_1 0
- #define BITMAP\_2 1
- #define BITMAP\_3 2
- #define BITMAP\_4 3
- #define BITMAPS 4
- #define STEPICON\_1 0
- #define STEPICON\_2 1
- #define STEPICON\_3 2
- #define STEPICON\_4 3
- #define STEPICON\_5 4
- #define STEPICONS 5
- #define DisplayOffsetPFunc 0
- #define DisplayOffsetEraseMask 4
- #define DisplayOffsetUpdateMask 8
- #define DisplayOffsetPFont 12
- #define DisplayOffsetPTextLines(p) (((p)\*4)+16)
- #define DisplayOffsetPStatusText 48

- #define DisplayOffsetPStatusIcons 52
- #define DisplayOffsetPScreens(p) (((p)\*4)+56)
- #define DisplayOffsetPBitmaps(p) (((p)\*4)+68)
- #define DisplayOffsetPMenuText 84
- #define DisplayOffsetPMenuIcons(p) (((p)\*4)+88)
- #define DisplayOffsetPStepIcons 100
- #define DisplayOffsetDisplay 104
- #define DisplayOffsetStatusIcons(p) ((p)+108)
- #define DisplayOffsetStepIcons(p) ((p)+112)
- #define DisplayOffsetFlags 117
- #define DisplayOffsetTextLinesCenterFlags 118
- #define DisplayOffsetNormal(l, w) (((l)\*100)+(w)+119)
- #define DisplayOffsetPopup(l, w) (((l)\*100)+(w)+919)
- #define DisplayOffsetContrast 1719
- #define SIZE\_OF\_USBBUF 64
- #define USB\_PROTOCOL\_OVERHEAD 2
- #define SIZE\_OF\_USBDATA 62
- #define SIZE\_OF\_HSBUF 128
- #define SIZE\_OF\_BTBUF 128
- #define BT\_CMD\_BYTE 1
- #define SIZE\_OF\_BT\_DEVICE\_TABLE 30
- #define SIZE\_OF\_BT\_CONNECT\_TABLE 4
- #define SIZE\_OF\_BT\_NAME 16
- #define SIZE\_OF\_BRICK\_NAME 8
- #define SIZE\_OF\_CLASS\_OF\_DEVICE 4
- #define SIZE\_OF\_BT\_PINCODE 16
- #define SIZE\_OF\_BDADDR 7
- #define MAX\_BT\_MSG\_SIZE 60000
- #define BT\_DEFAULT\_INQUIRY\_MAX 0
- #define BT\_DEFAULT\_INQUIRY\_TIMEOUT\_LO 15
- #define JOY\_BTN\_01 0x00000001
- #define JOY\_BTN\_02 0x00000002
- #define JOY\_BTN\_03 0x00000004
- #define JOY\_BTN\_04 0x00000008
- #define JOY\_BTN\_05 0x00000010
- #define JOY\_BTN\_06 0x00000020
- #define JOY\_BTN\_07 0x00000040
- #define JOY\_BTN\_08 0x00000080
- #define JOY\_BTN\_09 0x00000100
- #define JOY\_BTN\_10 0x00000200
- #define JOY\_BTN\_11 0x00000400
- #define JOY\_BTN\_12 0x00000800
- #define JOY\_BTN\_13 0x00001000
- #define JOY\_BTN\_14 0x00002000
- #define JOY\_BTN\_15 0x00004000
- #define JOY\_BTN\_16 0x00008000
- #define JOY\_BTN\_17 0x00010000
- #define JOY\_BTN\_18 0x00020000
- #define JOY\_BTN\_19 0x00040000
- #define JOY\_BTN\_20 0x00080000
- #define JOY\_BTN\_21 0x00100000

- #define JOY\_BTN\_22 0x00200000
- #define JOY\_BTN\_23 0x00400000
- #define JOY\_BTN\_24 0x00800000
- #define JOY\_BTN\_25 0x01000000
- #define JOY\_BTN\_26 0x02000000
- #define JOY\_BTN\_27 0x04000000
- #define JOY\_BTN\_28 0x08000000
- #define JOY\_BTN\_29 0x10000000
- #define JOY\_BTN\_30 0x20000000
- #define JOY\_BTN\_31 0x40000000
- #define JOY\_BTN\_32 0x80000000
- #define JOY\_POV\_FORWARD 0
- #define JOY\_POV\_TOPRIGHT 4500
- #define JOY\_POV\_RIGHT 9000
- #define JOY\_POV\_BOTRIGHT 13500
- #define JOY\_POV\_BACKWARD 18000
- #define JOY\_POV\_BOTLEFT 22500
- #define JOY\_POV\_LEFT 27000
- #define JOY\_POV\_TOPLEFT 31500
- #define JOY\_POV\_CENTERED 65535
- #define BT\_ARM\_OFF 0
- #define BT\_ARM\_CMD\_MODE 1
- #define BT\_ARM\_DATA\_MODE 2
- #define DATA\_MODE\_NXT 0x0
- #define DATA\_MODE\_GPS 0x01
- #define DATA\_MODE\_RAW 0x02
- #define DATA\_MODE\_MASK 0x07
- #define DATA\_MODE\_UPDATE 0x08
- #define BT\_BRICK\_VISIBILITY 0x01
- #define BT\_BRICK\_PORT\_OPEN 0x02
- #define BT\_CONNECTION\_0\_ENABLE 0x10
- #define BT\_CONNECTION\_1\_ENABLE 0x20
- #define BT\_CONNECTION\_2\_ENABLE 0x40
- #define BT\_CONNECTION\_3\_ENABLE 0x80
- #define CONN\_BT0 0x0
- #define CONN\_BT1 0x1
- #define CONN\_BT2 0x2
- #define CONN\_BT3 0x3
- #define CONN\_HS4 0x4
- #define CONN\_HS\_ALL 0x4
- #define CONN\_HS\_1 0x5
- #define CONN\_HS\_2 0x6
- #define CONN\_HS\_3 0x7
- #define CONN\_HS\_4 0x8
- #define CONN\_HS\_5 0x9
- #define CONN\_HS\_6 0xa
- #define CONN\_HS\_7 0xb
- #define CONN\_HS\_8 0xc
- #define BT\_ENABLE 0x00
- #define BT\_DISABLE 0x01
- #define HS\_UPDATE 1

- #define HS\_INITIALISE 1
- #define HS\_INIT\_RECEIVER 2
- #define HS\_SEND\_DATA 3
- #define HS\_DISABLE 4
- #define HS\_ENABLE 5
- #define HS\_DEFAULT 6
- #define HS\_BYTES\_REMAINING 16
- #define HS\_CTRL\_INIT 0
- #define HS\_CTRL\_UART 1
- #define HS\_CTRL\_EXIT 2
- #define HS\_BAUD\_1200 0
- #define HS\_BAUD\_2400 1
- #define HS\_BAUD\_3600 2
- #define HS\_BAUD\_4800 3
- #define HS\_BAUD\_7200 4
- #define HS\_BAUD\_9600 5
- #define HS\_BAUD\_14400 6
- #define HS\_BAUD\_19200 7
- #define HS\_BAUD\_28800 8
- #define HS\_BAUD\_38400 9
- #define HS\_BAUD\_57600 10
- #define HS\_BAUD\_76800 11
- #define HS\_BAUD\_115200 12
- #define HS\_BAUD\_230400 13
- #define HS\_BAUD\_460800 14
- #define HS\_BAUD\_921600 15
- #define HS\_BAUD\_DEFAULT 15
- #define HS\_MODE\_UART\_RS485 0x0
- #define HS\_MODE\_UART\_RS232 0x1
- #define HS\_MODE\_MASK 0x3EC0
- #define HS\_UART\_MASK 0x000F
- #define HS\_MODE\_DEFAULT HS\_MODE\_8N1
- #define HS\_MODE\_5\_DATA 0x0000
- #define HS\_MODE\_6\_DATA 0x0040
- #define HS\_MODE\_7\_DATA 0x0080
- #define HS\_MODE\_8\_DATA 0x00C0
- #define HS\_MODE\_10\_STOP 0x0000
- #define HS\_MODE\_15\_STOP 0x1000
- #define HS\_MODE\_20\_STOP 0x2000
- #define HS\_MODE\_E\_PARITY 0x0000
- #define HS\_MODE\_O\_PARITY 0x0200
- #define HS\_MODE\_S\_PARITY 0x0400
- #define HS\_MODE\_M\_PARITY 0x0600
- #define HS\_MODE\_N\_PARITY 0x0800
- #define HS\_MODE\_8N1 (HS\_MODE\_8\_DATA|HS\_MODE\_N\_PARITY|HS\_MODE\_10\_STOP)
- #define HS\_MODE\_7E1 (HS\_MODE\_7\_DATA|HS\_MODE\_E\_PARITY|HS\_MODE\_10\_STOP)
- #define HS\_ADDRESS\_ALL 0
- #define HS\_ADDRESS\_1 1
- #define HS\_ADDRESS\_2 2
- #define HS\_ADDRESS\_3 3
- #define HS\_ADDRESS\_4 4

- #define HS\_ADDRESS\_5 5
- #define HS\_ADDRESS\_6 6
- #define HS\_ADDRESS\_7 7
- #define HS\_ADDRESS\_8 8
- #define BT\_DEVICE\_EMPTY 0x00
- #define BT\_DEVICE\_UNKNOWN 0x01
- #define BT\_DEVICE\_KNOWN 0x02
- #define BT\_DEVICE\_NAME 0x40
- #define BT\_DEVICE\_AWAY 0x80
- #define INTF\_SENDFILE 0
- #define INTF\_SEARCH 1
- #define INTF\_STOPSEARCH 2
- #define INTF\_CONNECT 3
- #define INTF\_DISCONNECT 4
- #define INTF\_DISCONNECTALL 5
- #define INTF\_REMOVEDevice 6
- #define INTF\_VISIBILITY 7
- #define INTF\_SETCMDMODE 8
- #define INTF\_OPENSTREAM 9
- #define INTF\_SENDDATA 10
- #define INTF\_FACTORYRESET 11
- #define INTF\_BTON 12
- #define INTF\_BTOFF 13
- #define INTF\_SETBTNAME 14
- #define INTF\_EXTREAD 15
- #define INTF\_PINREQ 16
- #define INTF\_CONNECTREQ 17
- #define INTF\_CONNECTBYNAME 18
- #define LR\_SUCCESS 0x50
- #define LR\_COULD\_NOT\_SAVE 0x51
- #define LR\_STORE\_IS\_FULL 0x52
- #define LR\_ENTRY\_REMOVED 0x53
- #define LR\_UNKNOWN\_ADDR 0x54
- #define USB\_CMD\_READY 0x01
- #define BT\_CMD\_READY 0x02
- #define HS\_CMD\_READY 0x04
- #define CommOffsetPFunc 0
- #define CommOffsetPFuncTwo 4
- #define CommOffsetBtDeviceTableName(p) (((p)\*31)+8)
- #define CommOffsetBtDeviceTableClassOfDevice(p) (((p)\*31)+24)
- #define CommOffsetBtDeviceTableBdAddr(p) (((p)\*31)+28)
- #define CommOffsetBtDeviceTableDeviceStatus(p) (((p)\*31)+35)
- #define CommOffsetBtConnectTableName(p) (((p)\*47)+938)
- #define CommOffsetBtConnectTableClassOfDevice(p) (((p)\*47)+954)
- #define CommOffsetBtConnectTablePinCode(p) (((p)\*47)+958)
- #define CommOffsetBtConnectTableBdAddr(p) (((p)\*47)+974)
- #define CommOffsetBtConnectTableHandleNr(p) (((p)\*47)+981)
- #define CommOffsetBtConnectTableStreamStatus(p) (((p)\*47)+982)
- #define CommOffsetBtConnectTableLinkQuality(p) (((p)\*47)+983)
- #define CommOffsetBrickDataName 1126
- #define CommOffsetBrickDataBluecoreVersion 1142

- #define CommOffsetBrickDataBdAddr 1144
- #define CommOffsetBrickDataBtStateStatus 1151
- #define CommOffsetBrickDataBtHwStatus 1152
- #define CommOffsetBrickDataTimeOutValue 1153
- #define CommOffsetBtInBufBuf 1157
- #define CommOffsetBtInBufInPtr 1285
- #define CommOffsetBtInBufOutPtr 1286
- #define CommOffsetBtOutBufBuf 1289
- #define CommOffsetBtOutBufInPtr 1417
- #define CommOffsetBtOutBufOutPtr 1418
- #define CommOffsetHsInBufBuf 1421
- #define CommOffsetHsInBufInPtr 1549
- #define CommOffsetHsInBufOutPtr 1550
- #define CommOffsetHsOutBufBuf 1553
- #define CommOffsetHsOutBufInPtr 1681
- #define CommOffsetHsOutBufOutPtr 1682
- #define CommOffsetUsbInBufBuf 1685
- #define CommOffsetUsbInBufInPtr 1749
- #define CommOffsetUsbInBufOutPtr 1750
- #define CommOffsetUsbOutBufBuf 1753
- #define CommOffsetUsbOutBufInPtr 1817
- #define CommOffsetUsbOutBufOutPtr 1818
- #define CommOffsetUsbPollBufBuf 1821
- #define CommOffsetUsbPollBufInPtr 1885
- #define CommOffsetUsbPollBufOutPtr 1886
- #define CommOffsetBtDeviceCnt 1889
- #define CommOffsetBtDeviceNameCnt 1890
- #define CommOffsetHsFlags 1891
- #define CommOffsetHsSpeed 1892
- #define CommOffsetHsState 1893
- #define CommOffsetUsbState 1894
- #define CommOffsetHsAddress 1895
- #define CommOffsetHsMode 1896
- #define CommOffsetBtDataMode 1898
- #define CommOffsetHsDataMode 1899
- #define RCX\_OUT\_A 0x01
- #define RCX\_OUT\_B 0x02
- #define RCX\_OUT\_C 0x04
- #define RCX\_OUT\_AB 0x03
- #define RCX\_OUT\_AC 0x05
- #define RCX\_OUT\_BC 0x06
- #define RCX\_OUT\_ABC 0x07
- #define RCX\_OUT\_FLOAT 0
- #define RCX\_OUT\_OFF 0x40
- #define RCX\_OUT\_ON 0x80
- #define RCX\_OUT\_REV 0
- #define RCX\_OUT\_TOGGLE 0x40
- #define RCX\_OUT\_FWD 0x80
- #define RCX\_OUT\_LOW 0
- #define RCX\_OUT\_HALF 3
- #define RCX\_OUT\_FULL 7

- #define RCX\_RemoteKeysReleased 0x0000
- #define RCX\_RemotePBMessag1 0x0100
- #define RCX\_RemotePBMessag2 0x0200
- #define RCX\_RemotePBMessag3 0x0400
- #define RCX\_RemoteOutAForward 0x0800
- #define RCX\_RemoteOutBForward 0x1000
- #define RCX\_RemoteOutCForward 0x2000
- #define RCX\_RemoteOutABackward 0x4000
- #define RCX\_RemoteOutBBackward 0x8000
- #define RCX\_RemoteOutCBackward 0x0001
- #define RCX\_RemoteSelProgram1 0x0002
- #define RCX\_RemoteSelProgram2 0x0004
- #define RCX\_RemoteSelProgram3 0x0008
- #define RCX\_RemoteSelProgram4 0x0010
- #define RCX\_RemoteSelProgram5 0x0020
- #define RCX\_RemoteStopOutOff 0x0040
- #define RCX\_RemotePlayASound 0x0080
- #define SOUND\_CLICK 0
- #define SOUND\_DOUBLE\_BEEP 1
- #define SOUND\_DOWN 2
- #define SOUND\_UP 3
- #define SOUND\_LOW\_BEEP 4
- #define SOUND\_FAST\_UP 5
- #define SCOUT\_LIGHT\_ON 0x80
- #define SCOUT\_LIGHT\_OFF 0
- #define SCOUT\_SOUND\_REMOTE 6
- #define SCOUT\_SOUND\_ENTERSA 7
- #define SCOUT\_SOUND\_KEYERROR 8
- #define SCOUT\_SOUND\_NONE 9
- #define SCOUT\_SOUND\_TOUCH1\_PRES 10
- #define SCOUT\_SOUND\_TOUCH1\_REL 11
- #define SCOUT\_SOUND\_TOUCH2\_PRES 12
- #define SCOUT\_SOUND\_TOUCH2\_REL 13
- #define SCOUT\_SOUND\_ENTER\_BRIGHT 14
- #define SCOUT\_SOUND\_ENTER\_NORMAL 15
- #define SCOUT\_SOUND\_ENTER\_DARK 16
- #define SCOUT\_SOUND\_1\_BLINK 17
- #define SCOUT\_SOUND\_2\_BLINK 18
- #define SCOUT\_SOUND\_COUNTER1 19
- #define SCOUT\_SOUND\_COUNTER2 20
- #define SCOUT\_SOUND\_TIMER1 21
- #define SCOUT\_SOUND\_TIMER2 22
- #define SCOUT\_SOUND\_TIMER3 23
- #define SCOUT\_SOUND\_MAIL\_RECEIVED 24
- #define SCOUT\_SOUND\_SPECIAL1 25
- #define SCOUT\_SOUND\_SPECIAL2 26
- #define SCOUT\_SOUND\_SPECIAL3 27
- #define SCOUT\_SNDSET\_NONE 0
- #define SCOUT\_SNDSET\_BASIC 1
- #define SCOUT\_SNDSET\_BUG 2
- #define SCOUT\_SNDSET\_ALARM 3

- #define SCOUT SNDSET RANDOM 4
- #define SCOUT SNDSET SCIENCE 5
- #define SCOUT MODE STANDALONE 0
- #define SCOUT MODE POWER 1
- #define SCOUT MR NO MOTION 0
- #define SCOUT MR FORWARD 1
- #define SCOUT MR ZIGZAG 2
- #define SCOUT MR CIRCLE RIGHT 3
- #define SCOUT MR CIRCLE LEFT 4
- #define SCOUT MR LOOP A 5
- #define SCOUT MR LOOP B 6
- #define SCOUT MR LOOP AB 7
- #define SCOUT TR IGNORE 0
- #define SCOUT TR REVERSE 1
- #define SCOUT TR AVOID 2
- #define SCOUT TR WAIT FOR 3
- #define SCOUT TR OFF WHEN 4
- #define SCOUT LR IGNORE 0
- #define SCOUT LR SEEK LIGHT 1
- #define SCOUT LR SEEK DARK 2
- #define SCOUT LR AVOID 3
- #define SCOUT LR WAIT FOR 4
- #define SCOUT LR OFF WHEN 5
- #define SCOUT TGS SHORT 0
- #define SCOUT TGS MEDIUM 1
- #define SCOUT TGS LONG 2
- #define SCOUT FXR NONE 0
- #define SCOUT FXR BUG 1
- #define SCOUT FXR ALARM 2
- #define SCOUT FXR RANDOM 3
- #define SCOUT FXR SCIENCE 4
- #define RCX\_VariableSrc 0
- #define RCX\_TimerSrc 1
- #define RCX\_ConstantSrc 2
- #define RCX\_OutputStatusSrc 3
- #define RCX\_RandomSrc 4
- #define RCX\_ProgramSlotSrc 8
- #define RCX\_InputValueSrc 9
- #define RCX\_InputTypeSrc 10
- #define RCX\_InputModeSrc 11
- #define RCX\_InputRawSrc 12
- #define RCX\_InputBooleanSrc 13
- #define RCX\_WatchSrc 14
- #define RCX\_MessageSrc 15
- #define RCX\_GlobalMotorStatusSrc 17
- #define RCX\_ScoutRulesSrc 18
- #define RCX\_ScoutLightParamsSrc 19
- #define RCX\_ScoutTimerLimitSrc 20
- #define RCX\_CounterSrc 21
- #define RCX\_ScoutCounterLimitSrc 22
- #define RCX\_TaskEventsSrc 23

- #define RCX\_ScoutEventFBSrc 24
- #define RCX\_EventStateSrc 25
- #define RCX\_TenMSTimerSrc 26
- #define RCX\_ClickCounterSrc 27
- #define RCX\_UpperThresholdSrc 28
- #define RCX\_LowerThresholdSrc 29
- #define RCX\_HysteresisSrc 30
- #define RCX\_DurationSrc 31
- #define RCX\_UARTSetupSrc 33
- #define RCX\_BatteryLevelSrc 34
- #define RCX\_FirmwareVersionSrc 35
- #define RCX\_IndirectVarSrc 36
- #define RCX\_DatalogSrcIndirectSrc 37
- #define RCX\_DatalogSrcDirectSrc 38
- #define RCX\_DatalogValueIndirectSrc 39
- #define RCX\_DatalogValueDirectSrc 40
- #define RCX\_DatalogRawIndirectSrc 41
- #define RCX\_DatalogRawDirectSrc 42
- #define RCX\_PingOp 0x10
- #define RCX\_BatteryLevelOp 0x30
- #define RCX\_DeleteTasksOp 0x40
- #define RCX\_StopAllTasksOp 0x50
- #define RCX\_PBTurnOffOp 0x60
- #define RCX\_DeleteSubsOp 0x70
- #define RCX\_ClearSoundOp 0x80
- #define RCX\_ClearMsgOp 0x90
- #define RCX\_LSCalibrateOp 0xc0
- #define RCX\_MuteSoundOp 0xd0
- #define RCX\_UnmuteSoundOp 0xe0
- #define RCX\_ClearAllEventsOp 0x06
- #define RCX\_OnOffFloatOp 0x21
- #define RCX\_IRModeOp 0x31
- #define RCX\_PlaySoundOp 0x51
- #define RCX\_DeleteTaskOp 0x61
- #define RCX\_StartTaskOp 0x71
- #define RCX\_StopTaskOp 0x81
- #define RCX\_SelectProgramOp 0x91
- #define RCX\_ClearTimerOp 0xa1
- #define RCX\_AutoOffOp 0xb1
- #define RCX\_DeleteSubOp 0xc1
- #define RCX\_ClearSensorOp 0xd1
- #define RCX\_OutputDirOp 0xe1
- #define RCX\_PlayToneVarOp 0x02
- #define RCX\_PollOp 0x12
- #define RCX\_SetWatchOp 0x22
- #define RCX\_InputTypeOp 0x32
- #define RCX\_InputModeOp 0x42
- #define RCX\_SetDatalogOp 0x52
- #define RCX\_DatalogOp 0x62
- #define RCX\_SendUARTDataOp 0xc2
- #define RCX\_RemoteOp 0xd2

- #define RCX\_VLLOp 0xe2
- #define RCX\_DirectEventOp 0x03
- #define RCX\_OutputPowerOp 0x13
- #define RCX\_PlayToneOp 0x23
- #define RCX\_DisplayOp 0x33
- #define RCX\_PollMemoryOp 0x63
- #define RCX\_SetFeedbackOp 0x83
- #define RCX\_SetEventOp 0x93
- #define RCX\_GOutputPowerOp 0xa3
- #define RCX\_LSUpperThreshOp 0xb3
- #define RCX\_LSLowerThreshOp 0xc3
- #define RCX\_LSHysteresisOp 0xd3
- #define RCX\_LSBlinkTimeOp 0xe3
- #define RCX\_CalibrateEventOp 0x04
- #define RCX\_SetVarOp 0x14
- #define RCX\_SumVarOp 0x24
- #define RCX\_SubVarOp 0x34
- #define RCX\_DivVarOp 0x44
- #define RCX\_MulVarOp 0x54
- #define RCX\_SgnVarOp 0x64
- #define RCX\_AbsVarOp 0x74
- #define RCX\_AndVarOp 0x84
- #define RCX\_OrVarOp 0x94
- #define RCX\_UploadDatalogOp 0xa4
- #define RCX\_SetTimerLimitOp 0xc4
- #define RCX\_SetCounterOp 0xd4
- #define RCX\_SetSourceValueOp 0x05
- #define RCX\_UnlockOp 0x15
- #define RCX\_BootModeOp 0x65
- #define RCX\_UnlockFirmOp 0xa5
- #define RCX\_ScoutRulesOp 0xd5
- #define RCX\_ViewSourceValOp 0xe5
- #define RCX\_ScoutOp 0x47
- #define RCX\_SoundOp 0x57
- #define RCX\_GOutputModeOp 0x67
- #define RCX\_GOutputDirOp 0x77
- #define RCX\_LightOp 0x87
- #define RCX\_IncCounterOp 0x97
- #define RCX\_DecCounterOp 0xa7
- #define RCX\_ClearCounterOp 0xb7
- #define RCX\_SetPriorityOp 0xd7
- #define RCX\_MessageOp 0xf7
- #define PF\_CMD\_STOP 0
- #define PF\_CMD\_FLOAT 0
- #define PF\_CMD\_FWD 1
- #define PF\_CMD\_REV 2
- #define PF\_CMD\_BRAKE 3
- #define PF\_CHANNEL\_1 0
- #define PF\_CHANNEL\_2 1
- #define PF\_CHANNEL\_3 2
- #define PF\_CHANNEL\_4 3

- #define PF\_MODE\_TRAIN 0
- #define PF\_MODE\_COMBO\_DIRECT 1
- #define PF\_MODE\_SINGLE\_PIN\_CONT 2
- #define PF\_MODE\_SINGLE\_PIN\_TIME 3
- #define PF\_MODE\_COMBO\_PWM 4
- #define PF\_MODE\_SINGLE\_OUTPUT\_PWM 4
- #define PF\_MODE\_SINGLE\_OUTPUT\_CST 6
- #define TRAIN\_FUNC\_STOP 0
- #define TRAIN\_FUNC\_INCR\_SPEED 1
- #define TRAIN\_FUNC\_DECR\_SPEED 2
- #define TRAIN\_FUNC\_TOGGLE\_LIGHT 4
- #define TRAIN\_CHANNEL\_1 0
- #define TRAIN\_CHANNEL\_2 1
- #define TRAIN\_CHANNEL\_3 2
- #define TRAIN\_CHANNEL\_ALL 3
- #define PF\_OUT\_A 0
- #define PF\_OUT\_B 1
- #define PF\_PIN\_C1 0
- #define PF\_PIN\_C2 1
- #define PF\_FUNC\_NOCHANGE 0
- #define PF\_FUNC\_CLEAR 1
- #define PF\_FUNC\_SET 2
- #define PF\_FUNC\_TOGGLE 3
- #define PF\_CST\_CLEAR1\_CLEAR2 0
- #define PF\_CST\_SET1\_CLEAR2 1
- #define PF\_CST\_CLEAR1\_SET2 2
- #define PF\_CST\_SET1\_SET2 3
- #define PF\_CST\_INCREMENT\_PWM 4
- #define PF\_CST\_DECREMENT\_PWM 5
- #define PF\_CST\_FULL\_FWD 6
- #define PF\_CST\_FULL\_REV 7
- #define PF\_CST\_TOGGLE\_DIR 8
- #define PF\_PWM\_FLOAT 0
- #define PF\_PWM\_FWD1 1
- #define PF\_PWM\_FWD2 2
- #define PF\_PWM\_FWD3 3
- #define PF\_PWM\_FWD4 4
- #define PF\_PWM\_FWD5 5
- #define PF\_PWM\_FWD6 6
- #define PF\_PWM\_FWD7 7
- #define PF\_PWM\_BRAKE 8
- #define PF\_PWM\_REV7 9
- #define PF\_PWM\_REV6 10
- #define PF\_PWM\_REV5 11
- #define PF\_PWM\_REV4 12
- #define PF\_PWM\_REV3 13
- #define PF\_PWM\_REV2 14
- #define PF\_PWM\_REV1 15
- #define HT\_ADDR\_IRSEEKER 0x02
- #define HT\_ADDR\_IRSEEKER2 0x10
- #define HT\_ADDR\_IRRECEIVER 0x02

- #define HT\_ADDR\_COMPASS 0x02
- #define HT\_ADDR\_ACCEL 0x02
- #define HT\_ADDR\_COLOR 0x02
- #define HT\_ADDR\_COLOR2 0x02
- #define HT\_ADDR\_IRLINK 0x02
- #define HT\_ADDR\_ANGLE 0x02
- #define HT\_ADDR\_BAROMETRIC 0x02
- #define HT\_ADDR\_PROTOBOARD 0x02
- #define HT\_ADDR\_SUPERPRO 0x10
- #define HT\_ADDR\_PIR 0x02
- #define HTIR2\_MODE\_1200 0
- #define HTIR2\_MODE\_600 1
- #define HTIR2\_REG\_MODE 0x41
- #define HTIR2\_REG\_DCDIR 0x42
- #define HTIR2\_REG\_DC01 0x43
- #define HTIR2\_REG\_DC02 0x44
- #define HTIR2\_REG\_DC03 0x45
- #define HTIR2\_REG\_DC04 0x46
- #define HTIR2\_REG\_DC05 0x47
- #define HTIR2\_REG\_DCAVG 0x48
- #define HTIR2\_REG\_ACDIR 0x49
- #define HTIR2\_REG\_AC01 0x4A
- #define HTIR2\_REG\_AC02 0x4B
- #define HTIR2\_REG\_AC03 0x4C
- #define HTIR2\_REG\_AC04 0x4D
- #define HTIR2\_REG\_AC05 0x4E
- #define HT\_CH1\_A 0
- #define HT\_CH1\_B 1
- #define HT\_CH2\_A 2
- #define HT\_CH2\_B 3
- #define HT\_CH3\_A 4
- #define HT\_CH3\_B 5
- #define HT\_CH4\_A 6
- #define HT\_CH4\_B 7
- #define HT\_CMD\_COLOR2\_ACTIVE 0x00
- #define HT\_CMD\_COLOR2\_PASSIVE 0x01
- #define HT\_CMD\_COLOR2\_RAW 0x03
- #define HT\_CMD\_COLOR2\_50HZ 0x35
- #define HT\_CMD\_COLOR2\_60HZ 0x36
- #define HT\_CMD\_COLOR2\_BLCAL 0x42
- #define HT\_CMD\_COLOR2\_WBCAL 0x43
- #define HT\_CMD\_COLOR2\_FAR 0x46
- #define HT\_CMD\_COLOR2\_LED\_HI 0x48
- #define HT\_CMD\_COLOR2\_LED\_LOW 0x4C
- #define HT\_CMD\_COLOR2\_NEAR 0x4E
- #define HTANGLE\_MODE\_NORMAL 0x00
- #define HTANGLE\_MODE\_CALIBRATE 0x43
- #define HTANGLE\_MODE\_RESET 0x52
- #define HTANGLE\_REG\_MODE 0x41
- #define HTANGLE\_REG\_DCDIR 0x42
- #define HTANGLE\_REG\_DC01 0x43

- #define HTANGLE\_REG\_DC02 0x44
- #define HTANGLE\_REG\_DC03 0x45
- #define HTANGLE\_REG\_DC04 0x46
- #define HTANGLE\_REG\_DC05 0x47
- #define HTANGLE\_REG\_DCAVG 0x48
- #define HTANGLE\_REG\_ACDIR 0x49
- #define HTBAR\_REG\_COMMAND 0x40
- #define HTBAR\_REG\_TEMPERATURE 0x42
- #define HTBAR\_REG\_PRESSURE 0x44
- #define HTBAR\_REG\_CALIBRATION 0x46
- #define HTPROTO\_REG\_A0 0x42
- #define HTPROTO\_REG\_A1 0x44
- #define HTPROTO\_REG\_A2 0x46
- #define HTPROTO\_REG\_A3 0x48
- #define HTPROTO\_REG\_A4 0x4A
- #define HTPROTO\_REG\_DIN 0x4C
- #define HTPROTO\_REG\_DOUT 0x4D
- #define HTPROTO\_REG\_DCTRL 0x4E
- #define HTPROTO\_REG\_SRATE 0x4F
- #define HTPROTO\_A0 0x42
- #define HTPROTO\_A1 0x44
- #define HTPROTO\_A2 0x46
- #define HTPROTO\_A3 0x48
- #define HTPROTO\_A4 0x4A
- #define HTSPRO\_REG\_CTRL 0x40
- #define HTSPRO\_REG\_A0 0x42
- #define HTSPRO\_REG\_A1 0x44
- #define HTSPRO\_REG\_A2 0x46
- #define HTSPRO\_REG\_A3 0x48
- #define HTSPRO\_REG\_DIN 0x4C
- #define HTSPRO\_REG\_DOUT 0x4D
- #define HTSPRO\_REG\_DCTRL 0x4E
- #define HTSPRO\_REG\_STROBE 0x50
- #define HTSPRO\_REG\_LED 0x51
- #define HTSPRO\_REG\_DAC0\_MODE 0x52
- #define HTSPRO\_REG\_DAC0\_FREQ 0x53
- #define HTSPRO\_REG\_DAC0\_VOLTAGE 0x55
- #define HTSPRO\_REG\_DAC1\_MODE 0x57
- #define HTSPRO\_REG\_DAC1\_FREQ 0x58
- #define HTSPRO\_REG\_DAC1\_VOLTAGE 0x5A
- #define HTSPRO\_REG\_DLADDRESS 0x60
- #define HTSPRO\_REG\_DLDATA 0x62
- #define HTSPRO\_REG\_DLCHKSUM 0x6A
- #define HTSPRO\_REG\_DLCONTROL 0x6B
- #define HTSPRO\_REG\_MEMORY\_20 0x80
- #define HTSPRO\_REG\_MEMORY\_21 0x84
- #define HTSPRO\_REG\_MEMORY\_22 0x88
- #define HTSPRO\_REG\_MEMORY\_23 0x8C
- #define HTSPRO\_REG\_MEMORY\_24 0x90
- #define HTSPRO\_REG\_MEMORY\_25 0x94
- #define HTSPRO\_REG\_MEMORY\_26 0x98

- #define HTSPRO\_REG\_MEMORY\_27 0x9C
- #define HTSPRO\_REG\_MEMORY\_28 0xA0
- #define HTSPRO\_REG\_MEMORY\_29 0xA4
- #define HTSPRO\_REG\_MEMORY\_2A 0xA8
- #define HTSPRO\_REG\_MEMORY\_2B 0xAC
- #define HTSPRO\_REG\_MEMORY\_2C 0xB0
- #define HTSPRO\_REG\_MEMORY\_2D 0xB4
- #define HTSPRO\_REG\_MEMORY\_2E 0xB8
- #define HTSPRO\_REG\_MEMORY\_2F 0xBC
- #define HTSPRO\_REG\_MEMORY\_30 0xC0
- #define HTSPRO\_REG\_MEMORY\_31 0xC4
- #define HTSPRO\_REG\_MEMORY\_32 0xC8
- #define HTSPRO\_REG\_MEMORY\_33 0xCC
- #define HTSPRO\_REG\_MEMORY\_34 0xD0
- #define HTSPRO\_REG\_MEMORY\_35 0xD4
- #define HTSPRO\_REG\_MEMORY\_36 0xD8
- #define HTSPRO\_REG\_MEMORY\_37 0xDC
- #define HTSPRO\_REG\_MEMORY\_38 0xE0
- #define HTSPRO\_REG\_MEMORY\_39 0xE4
- #define HTSPRO\_REG\_MEMORY\_3A 0xE8
- #define HTSPRO\_REG\_MEMORY\_3B 0xEC
- #define HTSPRO\_REG\_MEMORY\_3C 0xF0
- #define HTSPRO\_REG\_MEMORY\_3D 0xF4
- #define HTSPRO\_REG\_MEMORY\_3E 0xF8
- #define HTSPRO\_REG\_MEMORY\_3F 0xFC
- #define HTSPRO\_A0 0x42
- #define HTSPRO\_A1 0x44
- #define HTSPRO\_A2 0x46
- #define HTSPRO\_A3 0x48
- #define HTSPRO\_DAC0 0x52
- #define HTSPRO\_DAC1 0x57
- #define LED\_BLUE 0x02
- #define LED\_RED 0x01
- #define LED\_NONE 0x00
- #define DAC\_MODE\_DCOUT 0
- #define DAC\_MODE\_SINEWAVE 1
- #define DAC\_MODE\_SQUAREWAVE 2
- #define DAC\_MODE\_SAWPOSWAVE 3
- #define DAC\_MODE\_SAWNNEGWAVE 4
- #define DAC\_MODE\_TRIANGLEWAVE 5
- #define DAC\_MODE\_PWMVOLTAGE 6
- #define DAC\_MODE\_RESTART\_MASK 0x80
- #define DIGI\_PIN0 0x01
- #define DIGI\_PIN1 0x02
- #define DIGI\_PIN2 0x04
- #define DIGI\_PIN3 0x08
- #define DIGI\_PIN4 0x10
- #define DIGI\_PIN5 0x20
- #define DIGI\_PIN6 0x40
- #define DIGI\_PIN7 0x80
- #define STROBE\_SO 0x01

- #define STROBE\_S1 0x02
- #define STROBE\_S2 0x04
- #define STROBE\_S3 0x08
- #define STROBE\_READ 0x10
- #define STROBE\_WRITE 0x20
- #define HTPIR\_REG\_DEADBAND 0x41
- #define HTPIR\_REG\_READING 0x42
- #define MS\_CMD\_ENERGIZED 0x45
- #define MS\_CMD\_DEENERGIZED 0x44
- #define MS\_CMD\_ADPA\_ON 0x4E
- #define MS\_CMD\_ADPA\_OFF 0x4F
- #define MS\_ADDR\_RTCLOCK 0xD0
- #define MS\_ADDR\_DISTNX 0x02
- #define MS\_ADDR\_NRLINK 0x02
- #define MS\_ADDR\_ACCLNX 0x02
- #define MS\_ADDR\_CMPSNX 0x02
- #define MS\_ADDR\_PSPNX 0x02
- #define MS\_ADDR\_LINELDR 0x02
- #define MS\_ADDR\_NXTCAM 0x02
- #define MS\_ADDR\_NXTHID 0x04
- #define MS\_ADDR\_NXTSERVO 0xB0
- #define MS\_ADDR\_NXTSERVO\_EM 0x40
- #define MS\_ADDR\_PFMATE 0x48
- #define MS\_ADDR\_MTRMUX 0xB4
- #define MS\_ADDR\_NXTMMX 0x06
- #define MS\_ADDR\_IVSENS 0x12
- #define MS\_ADDR\_RXMUX 0x7E
- #define MS\_ADDR\_NUMERICPAD 0xB4
- #define MS\_ADDR\_TOUCHPANEL 0x04
- #define DIST\_CMD\_GP2D12 0x31
- #define DIST\_CMD\_GP2D120 0x32
- #define DIST\_CMD\_GP2YA21 0x33
- #define DIST\_CMD\_GP2YA02 0x34
- #define DIST\_CMD\_CUSTOM 0x35
- #define DIST\_REG\_DIST 0x42
- #define DIST\_REG\_VOLT 0x44
- #define DIST\_REG\_MODULE\_TYPE 0x50
- #define DIST\_REG\_NUM\_POINTS 0x51
- #define DIST\_REG\_DIST\_MIN 0x52
- #define DIST\_REG\_DIST\_MAX 0x54
- #define DIST\_REG\_VOLT1 0x56
- #define DIST\_REG\_DIST1 0x58
- #define PSP\_CMD\_DIGITAL 0x41
- #define PSP\_CMD\_ANALOG 0x73
- #define PSP\_REG\_BTNSET1 0x42
- #define PSP\_REG\_BTNSET2 0x43
- #define PSP\_REG\_XLEFT 0x44
- #define PSP\_REG\_YLEFT 0x45
- #define PSP\_REG\_XRIGHT 0x46
- #define PSP\_REG\_YRIGHT 0x47
- #define PSP\_BTNSET1\_LEFT 0x80

- #define PSP\_BTNSET1\_DOWN 0x40
- #define PSP\_BTNSET1\_RIGHT 0x20
- #define PSP\_BTNSET1\_UP 0x10
- #define PSP\_BTNSET1\_START 0x08
- #define PSP\_BTNSET1\_R3 0x04
- #define PSP\_BTNSET1\_L3 0x02
- #define PSP\_BTNSET1\_SELECT 0x01
- #define PSP\_BTNSET2\_SQUARE 0x80
- #define PSP\_BTNSET2\_CROSS 0x40
- #define PSP\_BTNSET2\_CIRCLE 0x20
- #define PSP\_BTNSET2\_TRIANGLE 0x10
- #define PSP\_BTNSET2\_R1 0x08
- #define PSP\_BTNSET2\_L1 0x04
- #define PSP\_BTNSET2\_R2 0x02
- #define PSP\_BTNSET2\_L2 0x01
- #define NRLINK\_CMD\_2400 0x44
- #define NRLINK\_CMD\_FLUSH 0x46
- #define NRLINK\_CMD\_4800 0x48
- #define NRLINK\_CMD\_IR\_LONG 0x4C
- #define NRLINK\_CMD\_IR\_SHORT 0x53
- #define NRLINK\_CMD\_RUN\_MACRO 0x52
- #define NRLINK\_CMD\_TX\_RAW 0x55
- #define NRLINK\_CMD\_SET\_RCX 0x58
- #define NRLINK\_CMD\_SET\_TRAIN 0x54
- #define NRLINK\_CMD\_SET\_PF 0x50
- #define NRLINK\_REG\_BYTES 0x40
- #define NRLINK\_REG\_DATA 0x42
- #define NRLINK\_REG\_EEPROM 0x50
- #define ACCL\_CMD\_X\_CAL 0x58
- #define ACCL\_CMD\_Y\_CAL 0x59
- #define ACCL\_CMD\_Z\_CAL 0x5a
- #define ACCL\_CMD\_X\_CAL\_END 0x78
- #define ACCL\_CMD\_Y\_CAL\_END 0x79
- #define ACCL\_CMD\_Z\_CAL\_END 0x7a
- #define ACCL\_CMD\_RESET\_CAL 0x52
- #define ACCL\_REG\_SENS\_LVL 0x19
- #define ACCL\_REG\_X\_TILT 0x42
- #define ACCL\_REG\_Y\_TILT 0x43
- #define ACCL\_REG\_Z\_TILT 0x44
- #define ACCL\_REG\_X\_ACCEL 0x45
- #define ACCL\_REG\_Y\_ACCEL 0x47
- #define ACCL\_REG\_Z\_ACCEL 0x49
- #define ACCL\_REG\_X\_OFFSET 0x4b
- #define ACCL\_REG\_X\_RANGE 0x4d
- #define ACCL\_REG\_Y\_OFFSET 0x4f
- #define ACCL\_REG\_Y\_RANGE 0x51
- #define ACCL\_REG\_Z\_OFFSET 0x53
- #define ACCL\_REG\_Z\_RANGE 0x55
- #define ACCL\_SENSITIVITY\_LEVEL\_1 0x31
- #define ACCL\_SENSITIVITY\_LEVEL\_2 0x32
- #define ACCL\_SENSITIVITY\_LEVEL\_3 0x33

- #define ACCL\_SENSITIVITY\_LEVEL\_4 0x34
- #define PFMATE\_REG\_CMD 0x41
- #define PFMATE\_REG\_CHANNEL 0x42
- #define PFMATE\_REG\_MOTORS 0x43
- #define PFMATE\_REG\_A\_CMD 0x44
- #define PFMATE\_REG\_A\_SPEED 0x45
- #define PFMATE\_REG\_B\_CMD 0x46
- #define PFMATE\_REG\_B\_SPEED 0x47
- #define PFMATE\_CMD\_GO 0x47
- #define PFMATE\_CMD\_RAW 0x52
- #define PFMATE\_MOTORS\_BOTH 0x00
- #define PFMATE\_MOTORS\_A 0x01
- #define PFMATE\_MOTORS\_B 0x02
- #define PFMATE\_CHANNEL\_1 1
- #define PFMATE\_CHANNEL\_2 2
- #define PFMATE\_CHANNEL\_3 3
- #define PFMATE\_CHANNEL\_4 4
- #define NXTSERVO\_REG\_VOLTAGE 0x41
- #define NXTSERVO\_REG\_CMD 0x41
- #define NXTSERVO\_REG\_S1\_POS 0x42
- #define NXTSERVO\_REG\_S2\_POS 0x44
- #define NXTSERVO\_REG\_S3\_POS 0x46
- #define NXTSERVO\_REG\_S4\_POS 0x48
- #define NXTSERVO\_REG\_S5\_POS 0x4A
- #define NXTSERVO\_REG\_S6\_POS 0x4C
- #define NXTSERVO\_REG\_S7\_POS 0x4E
- #define NXTSERVO\_REG\_S8\_POS 0x50
- #define NXTSERVO\_REG\_S1\_SPEED 0x52
- #define NXTSERVO\_REG\_S2\_SPEED 0x53
- #define NXTSERVO\_REG\_S3\_SPEED 0x54
- #define NXTSERVO\_REG\_S4\_SPEED 0x55
- #define NXTSERVO\_REG\_S5\_SPEED 0x56
- #define NXTSERVO\_REG\_S6\_SPEED 0x57
- #define NXTSERVO\_REG\_S7\_SPEED 0x58
- #define NXTSERVO\_REG\_S8\_SPEED 0x59
- #define NXTSERVO\_REG\_S1\_QPOS 0x5A
- #define NXTSERVO\_REG\_S2\_QPOS 0x5B
- #define NXTSERVO\_REG\_S3\_QPOS 0x5C
- #define NXTSERVO\_REG\_S4\_QPOS 0x5D
- #define NXTSERVO\_REG\_S5\_QPOS 0x5E
- #define NXTSERVO\_REG\_S6\_QPOS 0x5F
- #define NXTSERVO\_REG\_S7\_QPOS 0x60
- #define NXTSERVO\_REG\_S8\_QPOS 0x61
- #define NXTSERVO\_EM\_REG\_CMD 0x00
- #define NXTSERVO\_EM\_REG\_EEPROM\_START 0x21
- #define NXTSERVO\_EM\_REG\_EEPROM\_END 0xFF
- #define NXTSERVO\_POS\_CENTER 1500
- #define NXTSERVO\_POS\_MIN 500
- #define NXTSERVO\_POS\_MAX 2500
- #define NXTSERVO\_QPOS\_CENTER 150
- #define NXTSERVO\_QPOS\_MIN 50

- #define NXTSERVO\_QPOS\_MAX 250
- #define NXTSERVO\_SERVO\_1 0
- #define NXTSERVO\_SERVO\_2 1
- #define NXTSERVO\_SERVO\_3 2
- #define NXTSERVO\_SERVO\_4 3
- #define NXTSERVO\_SERVO\_5 4
- #define NXTSERVO\_SERVO\_6 5
- #define NXTSERVO\_SERVO\_7 6
- #define NXTSERVO\_SERVO\_8 7
- #define NXTSERVO\_CMD\_INIT 0x49
- #define NXTSERVO\_CMD\_RESET 0x53
- #define NXTSERVO\_CMD\_HALT 0x48
- #define NXTSERVO\_CMD\_RESUME 0x52
- #define NXTSERVO\_CMD\_GOTO 0x47
- #define NXTSERVO\_CMD\_PAUSE 0x50
- #define NXTSERVO\_CMD\_EDIT1 0x45
- #define NXTSERVO\_CMD\_EDIT2 0x4D
- #define NXTSERVO\_EM\_CMD\_QUIT 0x51
- #define NXTHID\_REG\_CMD 0x41
- #define NXTHID\_REG\_MODIFIER 0x42
- #define NXTHID\_REG\_DATA 0x43
- #define NXTHID\_MOD\_NONE 0x00
- #define NXTHID\_MOD\_LEFT\_CTRL 0x01
- #define NXTHID\_MOD\_LEFT\_SHIFT 0x02
- #define NXTHID\_MOD\_LEFT\_ALT 0x04
- #define NXTHID\_MOD\_LEFT\_GUI 0x08
- #define NXTHID\_MOD\_RIGHT\_CTRL 0x10
- #define NXTHID\_MOD\_RIGHT\_SHIFT 0x20
- #define NXTHID\_MOD\_RIGHT\_ALT 0x40
- #define NXTHID\_MOD\_RIGHT\_GUI 0x80
- #define NXTHID\_CMD\_ASCII 0x41
- #define NXTHID\_CMD\_DIRECT 0x44
- #define NXTHID\_CMD\_TRANSMIT 0x54
- #define NXTPM\_REG\_CMD 0x41
- #define NXTPM\_REG\_CURRENT 0x42
- #define NXTPM\_REG\_VOLTAGE 0x44
- #define NXTPM\_REG\_CAPACITY 0x46
- #define NXTPM\_REG\_POWER 0x48
- #define NXTPM\_REG\_TOTALPOWER 0x4A
- #define NXTPM\_REG\_MAXCURRENT 0x4E
- #define NXTPM\_REG\_MINCURRENT 0x50
- #define NXTPM\_REG\_MAXVOLTAGE 0x52
- #define NXTPM\_REG\_MINVOLTAGE 0x54
- #define NXTPM\_REG\_TIME 0x56
- #define NXTPM\_REG\_USERGAIN 0x5A
- #define NXTPM\_REG\_GAIN 0x5E
- #define NXTPM\_REG\_ERRORCOUNT 0x5F
- #define NXTPM\_CMD\_RESET 0x52
- #define NXTSE\_ZONE\_NONE 0
- #define NXTSE\_ZONE\_FRONT 1
- #define NXTSE\_ZONE\_LEFT 2

- #define NXTSE\_ZONE\_RIGHT 3
- #define NXTLL\_REG\_CMD 0x41
- #define NXTLL\_REG\_STEERING 0x42
- #define NXTLL\_REG\_AVERAGE 0x43
- #define NXTLL\_REG\_RESULT 0x44
- #define NXTLL\_REG\_SETPOINT 0x45
- #define NXTLL\_REG\_KP\_VALUE 0x46
- #define NXTLL\_REG\_KI\_VALUE 0x47
- #define NXTLL\_REG\_KD\_VALUE 0x48
- #define NXTLL\_REG\_CALIBRATED 0x49
- #define NXTLL\_REG\_WHITELIMITS 0x51
- #define NXTLL\_REG\_BLACKLIMITS 0x59
- #define NXTLL\_REG\_KP\_FACTOR 0x61
- #define NXTLL\_REG\_KI\_FACTOR 0x62
- #define NXTLL\_REG\_KD\_FACTOR 0x63
- #define NXTLL\_REG\_WHITEDATA 0x64
- #define NXTLL\_REG\_BLACKDATA 0x6C
- #define NXTLL\_REG\_RAWVOLTAGE 0x74
- #define NXTLL\_CMD\_USA 0x41
- #define NXTLL\_CMD\_BLACK 0x42
- #define NXTLL\_CMD\_POWERDOWN 0x44
- #define NXTLL\_CMD\_EUROPEAN 0x45
- #define NXTLL\_CMD\_INVERT 0x49
- #define NXTLL\_CMD\_POWERUP 0x50
- #define NXTLL\_CMD\_RESET 0x52
- #define NXTLL\_CMD\_SNAPSHOT 0x53
- #define NXTLL\_CMD\_UNIVERSAL 0x55
- #define NXTLL\_CMD\_WHITE 0x57
- #define NXTNP\_REG\_BUTTONS 0x00
- #define NXTTP\_REG\_CMD 0x41
- #define NXTTP\_CMD\_USA 0x41
- #define RFID\_MODE\_STOP 0
- #define RFID\_MODE\_SINGLE 1
- #define RFID\_MODE\_CONTINUOUS 2
- #define CT\_ADDR\_RFID 0x04
- #define CT\_REG\_STATUS 0x32
- #define CT\_REG\_MODE 0x41
- #define CT\_REG\_DATA 0x42
- #define DI\_ADDR\_DGPS 0x06
- #define DGPS\_REG\_TIME 0x00
- #define DGPS\_REG\_STATUS 0x01
- #define DGPS\_REG\_LATITUDE 0x02
- #define DGPS\_REG\_LONGITUDE 0x04
- #define DGPS\_REG\_VELOCITY 0x06
- #define DGPS\_REG\_HEADING 0x07
- #define DGPS\_REG\_DISTANCE 0x08
- #define DGPS\_REG\_WAYANGLE 0x09
- #define DGPS\_REG\_LASTANGLE 0x0A
- #define DGPS\_REG\_SETLATITUDE 0x0B
- #define DGPS\_REG\_SETLONGITUDE 0x0C
- #define DI\_ADDR\_GYRO 0xD2

- #define DI\_ADDR\_ACCL 0x3A
- #define DIGYRO\_REG\_WHOAMI 0x0F
- #define DIGYRO\_REG\_CTRL1 0x20
- #define DIGYRO\_REG\_CTRL2 0x21
- #define DIGYRO\_REG\_CTRL3 0x22
- #define DIGYRO\_REG\_CTRL4 0x23
- #define DIGYRO\_REG\_CTRL5 0x24
- #define DIGYRO\_REG\_REFERENCE 0x25
- #define DIGYRO\_REG\_OUTTEMP 0x26
- #define DIGYRO\_REG\_STATUS 0x27
- #define DIGYRO\_REG\_XLOW 0x28
- #define DIGYRO\_REG\_XHIGH 0x29
- #define DIGYRO\_REG\_YLOW 0x2A
- #define DIGYRO\_REG\_YHIGH 0x2B
- #define DIGYRO\_REG\_ZLOW 0x2C
- #define DIGYRO\_REG\_ZHIGH 0x2D
- #define DIGYRO\_REG\_FIFOCTRL 0x2E
- #define DIGYRO\_REG\_FIFOSRC 0x2F
- #define DIGYRO\_REG\_INT1\_CFG 0x30
- #define DIGYRO\_REG\_INT1\_SRC 0x31
- #define DIGYRO\_REG\_INT1\_XHI 0x32
- #define DIGYRO\_REG\_INT1\_XLO 0x33
- #define DIGYRO\_REG\_INT1\_YHI 0x34
- #define DIGYRO\_REG\_INT1\_YLO 0x35
- #define DIGYRO\_REG\_INT1\_ZHI 0x36
- #define DIGYRO\_REG\_INT1\_ZLO 0x37
- #define DIGYRO\_REG\_INT1\_DUR 0x38
- #define DIGYRO\_REG\_CTRL1AUTO 0xA0
- #define DIGYRO\_REG\_TEMPAUTO 0xA6
- #define DIGYRO\_REG\_XLOWBURST 0xA8
- #define DIGYRO\_REG\_YLOWBURST 0xAA
- #define DIGYRO\_REG\_ZLOWBURST 0xAC
- #define DIGYRO\_CTRL1\_XENABLE 0x01
- #define DIGYRO\_CTRL1\_YENABLE 0x02
- #define DIGYRO\_CTRL1\_ZENABLE 0x04
- #define DIGYRO\_CTRL1\_POWERDOWN 0x00
- #define DIGYRO\_CTRL1\_NORMAL 0x08
- #define DIGYRO\_CTRL1\_BANDWIDTH\_1 0x00
- #define DIGYRO\_CTRL1\_BANDWIDTH\_2 0x10
- #define DIGYRO\_CTRL1\_BANDWIDTH\_3 0x20
- #define DIGYRO\_CTRL1\_BANDWIDTH\_4 0x30
- #define DIGYRO\_CTRL1\_DATARATE\_100 0x00
- #define DIGYRO\_CTRL1\_DATARATE\_200 0x40
- #define DIGYRO\_CTRL1\_DATARATE\_400 0x80
- #define DIGYRO\_CTRL1\_DATARATE\_800 0xC0
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_8 0x00
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_4 0x01
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_2 0x02
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_1 0x03
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_05 0x04
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_02 0x05

- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_01 0x06
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_005 0x07
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_002 0x08
- #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_001 0x09
- #define DIGYRO\_CTRL2\_HPMODE\_RESET 0x00
- #define DIGYRO\_CTRL2\_HPMODE\_REFSIG 0x10
- #define DIGYRO\_CTRL2\_HPMODE\_NORMAL 0x20
- #define DIGYRO\_CTRL2\_HPMODE\_AUTOINT 0x30
- #define DIGYRO\_CTRL3\_INT1\_ENABLE 0x80
- #define DIGYRO\_CTRL3\_INT1\_BOOT 0x40
- #define DIGYRO\_CTRL3\_INT1\_LOWACTIVE 0x20
- #define DIGYRO\_CTRL3\_OPENDRAIN 0x10
- #define DIGYRO\_CTRL3\_INT2\_DATAREADY 0x08
- #define DIGYRO\_CTRL3\_INT2\_WATERMARK 0x04
- #define DIGYRO\_CTRL3\_INT2\_OVERRUN 0x02
- #define DIGYRO\_CTRL3\_INT2\_EMPTY 0x01
- #define DIGYRO\_CTRL4\_BLOCKDATA 0x80
- #define DIGYRO\_CTRL4\_BIGENDIAN 0x40
- #define DIGYRO\_CTRL4\_SCALE\_250 0x00
- #define DIGYRO\_CTRL4\_SCALE\_500 0x10
- #define DIGYRO\_CTRL4\_SCALE\_2000 0x30
- #define DIGYRO\_CTRL5\_REBOOTMEM 0x80
- #define DIGYRO\_CTRL5\_FIFOENABLE 0x40
- #define DIGYRO\_CTRL5\_HPENABLE 0x10
- #define DIGYRO\_CTRL5\_OUT\_SEL\_1 0x00
- #define DIGYRO\_CTRL5\_OUT\_SEL\_2 0x01
- #define DIGYRO\_CTRL5\_OUT\_SEL\_3 0x02
- #define DIGYRO\_CTRL5\_INT1\_SEL\_1 0x00
- #define DIGYRO\_CTRL5\_INT1\_SEL\_2 0x04
- #define DIGYRO\_CTRL5\_INT1\_SEL\_3 0x08
- #define DIGYRO\_FIFOCTRL\_BYPASS 0x00
- #define DIGYRO\_FIFOCTRL\_FIFO 0x20
- #define DIGYRO\_FIFOCTRL\_STREAM 0x40
- #define DIGYRO\_FIFOCTRL\_STREAM2FIFO 0x60
- #define DIGYRO\_FIFOCTRL\_BYPASS2STREAM 0x80
- #define DIGYRO\_FIFOCTRL\_WATERMARK\_MASK 0x1F
- #define DIGYRO\_STATUS\_XDATA 0x01
- #define DIGYRO\_STATUS\_YDATA 0x02
- #define DIGYRO\_STATUS\_ZDATA 0x04
- #define DIGYRO\_STATUS\_XYZDATA 0x08
- #define DIGYRO\_STATUS\_XOVER 0x10
- #define DIGYRO\_STATUS\_YOVER 0x20
- #define DIGYRO\_STATUS\_ZOVER 0x40
- #define DIGYRO\_STATUS\_XYZOVER 0x80
- #define DIACCL\_REG\_XLOW 0x00
- #define DIACCL\_REG\_XHIGH 0x01
- #define DIACCL\_REG\_YLOW 0x02
- #define DIACCL\_REG\_YHIGH 0x03
- #define DIACCL\_REG\_ZLOW 0x04
- #define DIACCL\_REG\_ZHIGH 0x05
- #define DIACCL\_REG\_X8 0x06

- #define DIACCL\_REG\_Y8 0x07
- #define DIACCL\_REG\_Z8 0x08
- #define DIACCL\_REG\_STATUS 0x09
- #define DIACCL\_REG\_DETECTSRC 0x0A
- #define DIACCL\_REG\_OUTTEMP 0x0B
- #define DIACCL\_REG\_I2CADDR 0x0D
- #define DIACCL\_REG\_USERINFO 0x0E
- #define DIACCL\_REG\_WHOAMI 0x0F
- #define DIACCL\_REG\_XLOWDRIFT 0x10
- #define DIACCL\_REG\_XHIGHDRIFT 0x11
- #define DIACCL\_REG\_YLOWDRIFT 0x12
- #define DIACCL\_REG\_YHIGHDRIFT 0x13
- #define DIACCL\_REG\_ZLOWDRIFT 0x14
- #define DIACCL\_REG\_ZHIGHDRIFT 0x15
- #define DIACCL\_REG\_MODECTRL 0x16
- #define DIACCL\_REG\_INTLATCH 0x17
- #define DIACCL\_REG\_CTRL1 0x18
- #define DIACCL\_REG\_CTRL2 0x19
- #define DIACCL\_REG\_LVLDETTHR 0x1A
- #define DIACCL\_REG\_PLSDETTHR 0x1B
- #define DIACCL\_REG\_PLSDURVAL 0x1C
- #define DIACCL\_REG\_LATENCYTM 0x1D
- #define DIACCL\_REG\_TIMEWINDOW 0x1E
- #define DIACCL\_STATUS\_DATAREADY 0x01
- #define DIACCL\_STATUS\_DATAOVER 0x02
- #define DIACCL\_STATUS\_PARITYERR 0x04
- #define DIACCL\_MODE\_STANDBY 0x00
- #define DIACCL\_MODE\_MEASURE 0x01
- #define DIACCL\_MODE\_LVLDetect 0x02
- #define DIACCL\_MODE\_PLSDETECT 0x03
- #define DIACCL\_MODE\_GLVL8 0x00
- #define DIACCL\_MODE\_GLVL2 0x04
- #define DIACCL\_MODE\_GLVL4 0x08
- #define DIACCL\_INTERRUPT\_LATCH\_CLEAR1 0x01
- #define DIACCL\_INTERRUPT\_LATCH\_CLEAR2 0x02
- #define DIACCL\_CTRL1\_INT2TOINT1 0x01
- #define DIACCL\_CTRL1\_LEVELPULSE 0x00
- #define DIACCL\_CTRL1\_PULSELEVEL 0x02
- #define DIACCL\_CTRL1\_PULSEPULSE 0x04
- #define DIACCL\_CTRL1\_NO\_XDETECT 0x08
- #define DIACCL\_CTRL1\_NO\_YDETECT 0x10
- #define DIACCL\_CTRL1\_NO\_ZDETECT 0x20
- #define DIACCL\_CTRL1\_THRESH\_INT 0x40
- #define DIACCL\_CTRL1\_FILT\_BW125 0x80
- #define DIACCL\_CTRL2\_LVLPOl\_NEGAND 0x01
- #define DIACCL\_CTRL2\_DETPOL\_NEGAND 0x02
- #define DIACCL\_CTRL2\_DRIVE\_STRONG 0x04
- #define MI\_ADDR\_XG1300L 0x02
- #define XG1300L\_REG\_ANGLE 0x42
- #define XG1300L\_REG\_TURNRATE 0x44
- #define XG1300L\_REG\_XAXIS 0x46

- #define XG1300L\_REG\_YAXIS 0x48
- #define XG1300L\_REG\_ZAXIS 0x4A
- #define XG1300L\_REG\_RESET 0x60
- #define XG1300L\_REG\_2G 0x61
- #define XG1300L\_REG\_4G 0x62
- #define XG1300L\_REG\_8G 0x63
- #define XG1300L\_SCALE\_2G 0x01
- #define XG1300L\_SCALE\_4G 0x02
- #define XG1300L\_SCALE\_8G 0x04
- #define RICImgPoint(\_X, \_Y) (\_X)&0xFF, (\_X)>>8, (\_Y)&0xFF, (\_Y)>>8
 

Output an RIC ImgPoint structure.
- #define RICImgRect(\_Pt, \_W, \_H) \_Pt, (\_W)&0xFF, (\_W)>>8, (\_H)&0xFF, (\_H)>>8
 

Output an RIC ImgRect structure.
- #define RICOpDescription(\_Options, \_Width, \_Height) 8, 0, 0, 0, (\_Options)&0xFF, (\_Options)>>8, (\_Width)&0xFF, (\_Width)>>8, (\_Height)&0xFF, (\_Height)>>8
 

Output an RIC Description opcode.
- #define RICOpCopyBits(\_CopyOptions, \_DataAddr, \_SrcRect, \_DstPoint) 18, 0, 3, 0, (\_CopyOptions)&0xFF, (\_CopyOptions)>>8, (\_DataAddr)&0xFF, (\_DataAddr)>>8, \_SrcRect, \_DstPoint
 

Output an RIC CopyBits opcode.
- #define RICOpPixel(\_CopyOptions, \_Point, \_Value) 10, 0, 4, 0, (\_CopyOptions)&0xFF, (\_CopyOptions)>>8, \_Point, (\_Value)&0xFF, (\_Value)>>8
 

Output an RIC Pixel opcode.
- #define RICOpLine(\_CopyOptions, \_Point1, \_Point2) 12, 0, 5, 0, (\_CopyOptions)&0xFF, (\_CopyOptions)>>8, \_Point1, \_Point2
 

Output an RIC Line opcode.
- #define RICOpRect(\_CopyOptions, \_Point, \_Width, \_Height) 12, 0, 6, 0, (\_CopyOptions)&0xFF, (\_CopyOptions)>>8, \_Point, (\_Width)&0xFF, (\_Width)>>8, (\_Height)&0xFF, (\_Height)>>8
 

Output an RIC Rect opcode.
- #define RICOpCircle(\_CopyOptions, \_Point, \_Radius) 10, 0, 7, 0, (\_CopyOptions)&0xFF, (\_CopyOptions)>>8, \_Point, (\_Radius)&0xFF, (\_Radius)>>8
 

Output an RIC Circle opcode.
- #define RICOpNumBox(\_CopyOptions, \_Point, \_Value) 10, 0, 8, 0, (\_CopyOptions)&0xFF, (\_CopyOptions)>>8, \_Point, (\_Value)&0xFF, (\_Value)>>8
 

Output an RIC NumBox opcode.
- #define RICOpSprite(\_DataAddr, \_Rows, \_BytesPerRow, \_SpriteData) ((\_Rows\*\_BytesPerRow)+(((\_Rows\*\_BytesPerRow)%2)+8)&0xFF, ((\_Rows\*\_BytesPerRow)+(((\_Rows\*\_BytesPerRow)%2)+8))>>8, 1, 0, (\_DataAddr)&0xFF, (\_DataAddr)>>8, (\_Rows)&0xFF, (\_Rows)>>8, (\_BytesPerRow)&0xFF, (\_BytesPerRow)>>8, \_SpriteData
 

Output an RIC Sprite opcode.
- #define RICSpriteData(...) \_\_VA\_ARGS\_\_
 

Output RIC sprite data.
- #define RICOpVarMap(\_DataAddr, \_MapCount, \_MapFunction) ((\_MapCount\*4)+6)&0xFF, ((\_MapCount\*4)+6)>>8, 2, 0, (\_DataAddr)&0xFF, (\_DataAddr)>>8, (\_MapCount)&0xFF, (\_MapCount)>>8, \_MapFunction
 

Output an RIC VarMap opcode.
- #define RICMapElement(\_Domain, \_Range) (\_Domain)&0xFF, (\_Domain)>>8, (\_Range)&0xFF, (\_Range)>>8
 

Output an RIC map element.
- #define RICMapFunction(\_MapElement,...) \_MapElement, \_\_VA\_ARGS\_\_
 

Output an RIC VarMap function.

- `#define RICArg(_arg) ((_arg)|0x1000)`  
*Output an RIC parameterized argument.*
- `#define RICMapArg(_mapidx, _arg) ((_arg)|0x1000|(((_mapidx)&0xF)<<8))`  
*Output an RIC parameterized and mapped argument.*
- `#define RICOpPolygon(_CopyOptions, _Count, _ThePoints) ((_Count*4)+6)&0xFF, ((_Count*4)+6)>>8, 10, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, (_Count)&0xFF, (_Count)>>8, _ThePoints)`  
*Output an RIC Polygon opcode.*
- `#define RICPolygonPoints(_pPoint1, _pPoint2,...) _pPoint1, _pPoint2, __VA_ARGS__`  
*Output RIC polygon points.*
- `#define RICOpEllipse(_CopyOptions, _Point, _RadiusX, _RadiusY) 12, 0, 9, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point, (_RadiusX)&0xFF, (_RadiusX)>>8, (_RadiusY)&0xFF, (_RadiusY)>>8`  
*Output an RIC Ellipse opcode.*
- `#define CHAR_BIT 8`
- `#define SCHAR_MIN -128`
- `#define SCHAR_MAX 127`
- `#define UCHAR_MAX 255`
- `#define CHAR_MIN -128`
- `#define CHAR_MAX 127`
- `#define SHRT_MIN -32768`
- `#define SHRT_MAX 32767`
- `#define USHRT_MAX 65535`
- `#define INT_MIN -32768`
- `#define INT_MAX 32767`
- `#define UINT_MAX 65535`
- `#define LONG_MIN -2147483648`
- `#define LONG_MAX 2147483647`
- `#define ULONG_MAX 4294967295`
- `#define RAND_MAX 2147483646`
- `#define GL_POLYGON 1`
- `#define GL_LINE 2`
- `#define GL_POINT 3`
- `#define GL_CIRCLE 4`
- `#define GL_TRANSLATE_X 1`
- `#define GL_TRANSLATE_Y 2`
- `#define GL_TRANSLATE_Z 3`
- `#define GL_ROTATE_X 4`
- `#define GL_ROTATE_Y 5`
- `#define GL_ROTATE_Z 6`
- `#define GL_SCALE_X 7`
- `#define GL_SCALE_Y 8`
- `#define GL_SCALE_Z 9`
- `#define GL_CIRCLE_SIZE 1`
- `#define GL_CULL_MODE 2`
- `#define GL_CAMERA_DEPTH 3`
- `#define GL_ZOOM_FACTOR 4`
- `#define GL_CULL_BACK 2`
- `#define GL_CULL_FRONT 3`
- `#define GL_CULL_NONE 4`

### 6.2.1 Detailed Description

Constants and macros common to both NBC and NXC. [NBCCommon.h](#) contains declarations for the NBC and NXC NXT API functions.

#### License:

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Initial Developer of this code is John Hansen. Portions created by John Hansen are Copyright (C) 2009-2013 John Hansen. All Rights Reserved.

#### Author

John Hansen (bricxcc\_at\_comcast.net)

#### Date

2013-02-16

#### Version

74

### 6.2.2 Macro Definition Documentation

#### 6.2.2.1 #define ACCL\_CMD\_RESET\_CAL 0x52

Reset to factory calibration

#### 6.2.2.2 #define ACCL\_CMD\_X\_CAL 0x58

Acquire X-axis calibration point

#### 6.2.2.3 #define ACCL\_CMD\_X\_CAL\_END 0x78

Acquire X-axis calibration point and end calibration

#### 6.2.2.4 #define ACCL\_CMD\_Y\_CAL 0x59

Acquire Y-axis calibration point

#### 6.2.2.5 #define ACCL\_CMD\_Y\_CAL\_END 0x79

Acquire Y-axis calibration point and end calibration

#### 6.2.2.6 #define ACCL\_CMD\_Z\_CAL 0x5a

Acquire Z-axis calibration point

#### 6.2.2.7 #define ACCL\_CMD\_Z\_CAL\_END 0x7a

Acquire Z-axis calibration point and end calibration

6.2.2.8 #define ACCL\_REG\_SENS\_LVL 0x19

The current sensitivity

6.2.2.9 #define ACCL\_REG\_X\_ACCEL 0x45

The X-axis acceleration data

6.2.2.10 #define ACCL\_REG\_X\_OFFSET 0x4b

The X-axis offset

6.2.2.11 #define ACCL\_REG\_X\_RANGE 0x4d

The X-axis range

6.2.2.12 #define ACCL\_REG\_X\_TILT 0x42

The X-axis tilt data

6.2.2.13 #define ACCL\_REG\_Y\_ACCEL 0x47

The Y-axis acceleration data

6.2.2.14 #define ACCL\_REG\_Y\_OFFSET 0x4f

The Y-axis offset

6.2.2.15 #define ACCL\_REG\_Y\_RANGE 0x51

The Y-axis range

6.2.2.16 #define ACCL\_REG\_Y\_TILT 0x43

The Y-axis tilt data

6.2.2.17 #define ACCL\_REG\_Z\_ACCEL 0x49

The Z-axis acceleration data

6.2.2.18 #define ACCL\_REG\_Z\_OFFSET 0x53

The Z-axis offset

6.2.2.19 #define ACCL\_REG\_Z\_RANGE 0x55

The Z-axis range

6.2.2.20 #define ACCL\_REG\_Z\_TILT 0x44

The Z-axis tilt data

6.2.2.21 #define ACCL\_SENSITIVITY\_LEVEL\_1 0x31

The ACCL-Nx sensitivity level 1

6.2.2.22 #define ACCL\_SENSITIVITY\_LEVEL\_2 0x32

The ACCL-Nx sensitivity level 2

6.2.2.23 #define ACCL\_SENSITIVITY\_LEVEL\_3 0x33

The ACCL-Nx sensitivity level 3

6.2.2.24 #define ACCL\_SENSITIVITY\_LEVEL\_4 0x34

The ACCL-Nx sensitivity level 4

6.2.2.25 #define ActualSpeedField 3

Actual speed field.

Contains the actual power level (-100 to 100). Read only. Return the percent of full power the firmware is applying to the output. This may vary from the PowerField value when auto-regulation code in the firmware responds to a load on the output.

6.2.2.26 #define BITMAP\_1 0

Bitmap 1

6.2.2.27 #define BITMAP\_2 1

Bitmap 2

6.2.2.28 #define BITMAP\_3 2

Bitmap 3

6.2.2.29 #define BITMAP\_4 3

Bitmap 4

6.2.2.30 #define BITMAPS 4

The number of bitmap bits

6.2.2.31 #define BlockTachoCountField 13

NXT-G block tachometer count field.

Contains the current NXT-G block tachometer count. Read only. Return the block-relative position counter value for the specified port. Refer to the [UpdateFlagsField](#) description for information about how to use block-relative position counts. Set the [UF\\_UPDATE\\_RESET\\_BLOCK\\_COUNT](#) flag in [UpdateFlagsField](#) to request that the firmware reset the BlockTachoCountField. The sign of BlockTachoCountField indicates the direction of rotation. Positive values indicate forward rotation and negative values indicate reverse rotation. Forward and reverse depend on the orientation of the motor.

6.2.2.32 #define BREAKOUT\_REQ 3

VM should break out of current thread

6.2.2.33 #define BT\_ARM\_CMD\_MODE 1

BtState constant bluetooth command mode

6.2.2.34 #define BT\_ARM\_DATA\_MODE 2

BtState constant bluetooth data mode

6.2.2.35 #define BT\_ARM\_OFF 0

BtState constant bluetooth off

6.2.2.36 #define BT\_BRICK\_PORT\_OPEN 0x02

BtStateStatus port open bit

6.2.2.37 #define BT\_BRICK\_VISIBILITY 0x01

BtStateStatus brick visibility bit

6.2.2.38 #define BT\_CMD\_BYTE 1

Size of Bluetooth command

6.2.2.39 #define BT\_CMD\_READY 0x02

A constant representing bluetooth direct command

6.2.2.40 #define BT\_CONNECTION\_0\_ENABLE 0x10

BtStateStatus connection 0 enable/disable bit

6.2.2.41 #define BT\_CONNECTION\_1\_ENABLE 0x20

BtStateStatus connection 1 enable/disable bit

6.2.2.42 #define BT\_CONNECTION\_2\_ENABLE 0x40

BtStateStatus connection 2 enable/disable bit

6.2.2.43 #define BT\_CONNECTION\_3\_ENABLE 0x80

BtStateStatus connection 3 enable/disable bit

6.2.2.44 #define BT\_DEFAULT\_INQUIRY\_MAX 0

Bluetooth default inquiry Max (0 == unlimited)

6.2.2.45 #define BT\_DEFAULT\_INQUIRY\_TIMEOUT\_LO 15

Bluetooth inquiry timeout (15\*1.28 sec = 19.2 sec)

6.2.2.46 #define BT\_DEVICE\_AWAY 0x80

Bluetooth device away

6.2.2.47 #define BT\_DEVICE\_EMPTY 0x00

Bluetooth device table empty

6.2.2.48 #define BT\_DEVICE\_KNOWN 0x02

Bluetooth device known

6.2.2.49 #define BT\_DEVICE\_NAME 0x40

Bluetooth device name

6.2.2.50 #define BT\_DEVICE\_UNKNOWN 0x01

Bluetooth device unknown

6.2.2.51 #define BT\_DISABLE 0x01

BtHwStatus bluetooth disable

6.2.2.52 #define BT\_ENABLE 0x00

BtHwStatus bluetooth enable

6.2.2.53 #define BTN1 0

The exit button.

6.2.2.54 #define BTN2 1

The right button.

6.2.2.55 #define BTN3 2

The left button.

6.2.2.56 #define BTN4 3

The enter button.

6.2.2.57 #define BTNCENTER BTN4

The enter button.

6.2.2.58 #define BTNEXTIT BTN1

The exit button.

6.2.2.59 #define BTNLEFT BTN3

The left button.

6.2.2.60 #define BTNRIGHT BTN2

The right button.

6.2.2.61 #define BTNSTATE\_LONG\_PRESSED\_EV 0x04

Button is in the long pressed state.

6.2.2.62 #define BTNSTATE\_LONG\_RELEASED\_EV 0x08

Button is in the long released state.

6.2.2.63 #define BTNSTATE\_NONE 0x10

The default button state.

6.2.2.64 #define BTNSTATE\_PRESSED\_EV 0x01

Button is in the pressed state.

6.2.2.65 #define BTNSTATE\_PRESSED\_STATE 0x80

A bitmask for the button pressed state

6.2.2.66 #define BTNSTATE\_SHORT\_RELEASED\_EV 0x02

Button is in the short released state.

6.2.2.67 #define ButtonModuleID 0x00040001

The button module ID

6.2.2.68 #define ButtonModuleName "Button.mod"

The button module name

6.2.2.69 #define ButtonOffsetLongPressCnt( *b* ) (((*b*)\*8)+1)

Offset to the LongPressCnt field. This field stores the long press count.

6.2.2.70 #define ButtonOffsetLongRelCnt( *b* ) (((*b*)\*8)+3)

Offset to the LongRelCnt field. This field stores the long release count.

6.2.2.71 #define ButtonOffsetPressedCnt( *b* ) (((*b*)\*8)+0)

Offset to the PressedCnt field. This field stores the press count.

6.2.2.72 #define ButtonOffsetRelCnt( *b* ) (((*b*)\*8)+4)

Offset to the RelCnt field. This field stores the release count.

6.2.2.73 #define ButtonOffsetShortRelCnt( *b* ) (((*b*)\*8)+2)

Offset to the ShortRelCnt field. This field stores the short release count.

6.2.2.74 #define ButtonOffsetState( *b* ) ((*b*)+32)

Offset to the State field. This field stores the current button state.

6.2.2.75 #define CHAR\_BIT 8

The number of bits in the char type

6.2.2.76 #define CHAR\_MAX 127

The maximum value of the char type

6.2.2.77 #define CHAR\_MIN -128

The minimum value of the char type

6.2.2.78 #define CLUMP\_DONE 1

VM has finished executing thread

6.2.2.79 #define CLUMP\_SUSPEND 2

VM should suspend thread

6.2.2.80 #define ColorSensorRead 34

Read data from the NXT 2.0 color sensor

6.2.2.81 #define COM\_CHANNEL\_FOUR\_ACTIVE 0x08

Low speed channel 4 is active

6.2.2.82 #define COM\_CHANNEL\_NONE\_ACTIVE 0x00

None of the low speed channels are active

6.2.2.83 #define COM\_CHANNEL\_ONE\_ACTIVE 0x01

Low speed channel 1 is active

6.2.2.84 #define COM\_CHANNEL\_THREE\_ACTIVE 0x04

Low speed channel 3 is active

6.2.2.85 #define COM\_CHANNEL\_TWO\_ACTIVE 0x02

Low speed channel 2 is active

6.2.2.86 #define CommandModuleID 0x00010001

The command module ID

6.2.2.87 #define CommandModuleName "Command.mod"

The command module name

6.2.2.88 #define CommandOffsetActivateFlag 30

Offset to the activate flag

6.2.2.89 #define CommandOffsetAwake 29

Offset to the VM's awake state

6.2.2.90 #define CommandOffsetDeactivateFlag 31

Offset to the deactivate flag

6.2.2.91 #define CommandOffsetFileName 32

Offset to the running program's filename

6.2.2.92 #define CommandOffsetFormatString 0

Offset to the format string

6.2.2.93 #define CommandOffsetMemoryPool 52

Offset to the VM's memory pool

6.2.2.94 #define CommandOffsetOffsetDS 24

Offset to the running program's data space (DS)

6.2.2.95 #define CommandOffsetOffsetDVA 26

Offset to the running program's DOPE vector address (DVA)

6.2.2.96 #define CommandOffsetPRCHandler 16

Offset to the RC Handler function pointer

6.2.2.97 #define CommandOffsetProgStatus 28

Offset to the running program's status

6.2.2.98 #define CommandOffsetSyncTick 32824

Offset to the VM sync tick

6.2.2.99 #define CommandOffsetSyncTime 32820

Offset to the VM sync time

6.2.2.100 #define CommandOffsetTick 20

Offset to the VM's current tick

6.2.2.101 #define CommBTCheckStatus 28

Check the bluetooth status

6.2.2.102 #define CommBTConnection 36

Connect or disconnect to a known bluetooth device

6.2.2.103 #define CommBTOnOff 35

Turn the bluetooth radio on or off

6.2.2.104 #define CommBTRead 30

Read from a bluetooth connection

6.2.2.105 #define CommBTWrite 29

Write to a bluetooth connections

6.2.2.106 #define CommExecuteFunction 81

Execute one of the Comm module's internal functions

6.2.2.107 #define CommHSCheckStatus 39

Check the status of the hi-speed port

6.2.2.108 #define CommHSControl 88

Control the hi-speed port

6.2.2.109 #define CommHSRead 38

Read data from the hi-speed port

6.2.2.110 #define CommHSWrite 37

Write data to the hi-speed port

6.2.2.111 #define CommLSCheckStatus 23

Check the status of a lowspeed (aka I2C) device

6.2.2.112 #define CommLSRead 22

Read from a lowspeed (aka I2C) device

6.2.2.113 #define CommLSSWrite 21

Write to a lowspeed (aka I2C) device

6.2.2.114 #define CommLSSWriteEx 89

Write to a lowspeed (aka I2C) device with optional restart on read

6.2.2.115 #define CommModuleID 0x00050001

The Comm module ID

6.2.2.116 #define CommModuleName "Comm.mod"

The Comm module name

6.2.2.117 #define CommOffsetBrickDataBdAddr 1144

Offset to Bluetooth address (7 bytes)

6.2.2.118 #define CommOffsetBrickDataBluecoreVersion 1142

Offset to Bluecore version (2 bytes)

6.2.2.119 #define CommOffsetBrickDataBtHwStatus 1152

Offset to BtHwStatus (1 byte)

6.2.2.120 #define CommOffsetBrickDataBtStateStatus 1151

Offset to BtStateStatus (1 byte)

6.2.2.121 #define CommOffsetBrickDataName 1126

Offset to brick name (16 bytes)

6.2.2.122 #define CommOffsetBrickDataTimeOutValue 1153

Offset to data timeout value (1 byte)

6.2.2.123 #define CommOffsetBtConnectTableBdAddr( *p* ) (((*p*)\*47)+974)

Offset to Bluetooth connect table address (7 bytes)

6.2.2.124 #define CommOffsetBtConnectTableClassOfDevice( *p* ) (((*p*)\*47)+954)

Offset to Bluetooth connect table device class (4 bytes)

6.2.2.125 #define CommOffsetBtConnectTableHandleNr( *p* ) (((*p*)\*47)+981)

Offset to Bluetooth connect table handle (1 byte)

6.2.2.126 #define CommOffsetBtConnectTableLinkQuality( *p* ) (((*p*)\*47)+983)

Offset to Bluetooth connect table link quality (1 byte)

6.2.2.127 #define CommOffsetBtConnectTableName( *p* ) (((*p*)\*47)+938)

Offset to Bluetooth connect table name (16 bytes)

6.2.2.128 #define CommOffsetBtConnectTablePinCode( *p* ) (((*p*)\*47)+958)

Offset to Bluetooth connect table pin code (16 bytes)

6.2.2.129 #define CommOffsetBtConnectTableStreamStatus( *p* ) (((*p*)\*47)+982)

Offset to Bluetooth connect table stream status (1 byte)

6.2.2.130 #define CommOffsetBtDataMode 1898

Offset to Bluetooth data mode (1 byte)

6.2.2.131 #define CommOffsetBtDeviceCnt 1889

Offset to Bluetooth device count (1 byte)

6.2.2.132 #define CommOffsetBtDeviceNameCnt 1890

Offset to Bluetooth device name count (1 byte)

6.2.2.133 #define CommOffsetBtDeviceTableBdAddr( p ) (((p)\*31)+28)

Offset to Bluetooth device table address (7 bytes)

6.2.2.134 #define CommOffsetBtDeviceTableClassOfDevice( p ) (((p)\*31)+24)

Offset to Bluetooth device table device class (4 bytes)

6.2.2.135 #define CommOffsetBtDeviceTableDeviceStatus( p ) (((p)\*31)+35)

Offset to Bluetooth device table status (1 byte)

6.2.2.136 #define CommOffsetBtDeviceTableName( p ) (((p)\*31)+8)

Offset to BT device table name (16 bytes)

6.2.2.137 #define CommOffsetBtInBufBuf 1157

Offset to Bluetooth input buffer data (128 bytes)

6.2.2.138 #define CommOffsetBtInBufInPtr 1285

Offset to Bluetooth input buffer front pointer (1 byte)

6.2.2.139 #define CommOffsetBtInBufOutPtr 1286

Offset to Bluetooth output buffer back pointer (1 byte)

6.2.2.140 #define CommOffsetBtOutBufBuf 1289

Offset to Bluetooth output buffer offset data (128 bytes)

6.2.2.141 #define CommOffsetBtOutBufInPtr 1417

Offset to Bluetooth output buffer front pointer (1 byte)

6.2.2.142 #define CommOffsetBtOutBufOutPtr 1418

Offset to Bluetooth output buffer back pointer (1 byte)

6.2.2.143 #define CommOffsetHsAddress 1895

Offset to High Speed address (1 byte)

6.2.2.144 #define CommOffsetHsDataMode 1899

Offset to High Speed data mode (1 byte)

6.2.2.145 #define CommOffsetHsFlags 1891

Offset to High Speed flags (1 byte)

6.2.2.146 #define CommOffsetHsInBufBuf 1421

Offset to High Speed input buffer data (128 bytes)

6.2.2.147 #define CommOffsetHsInBufInPtr 1549

Offset to High Speed input buffer front pointer (1 byte)

6.2.2.148 #define CommOffsetHsInBufOutPtr 1550

Offset to High Speed input buffer back pointer (1 byte)

6.2.2.149 #define CommOffsetHsMode 1896

Offset to High Speed mode (2 bytes)

6.2.2.150 #define CommOffsetHsOutBufBuf 1553

Offset to High Speed output buffer data (128 bytes)

6.2.2.151 #define CommOffsetHsOutBufInPtr 1681

Offset to High Speed output buffer front pointer (1 byte)

6.2.2.152 #define CommOffsetHsOutBufOutPtr 1682

Offset to High Speed output buffer back pointer (1 byte)

6.2.2.153 #define CommOffsetHsSpeed 1892

Offset to High Speed speed (1 byte)

6.2.2.154 #define CommOffsetHsState 1893

Offset to High Speed state (1 byte)

6.2.2.155 #define CommOffsetPFunc 0

Offset to the Comm module first function pointer (4 bytes)

6.2.2.156 #define CommOffsetPFuncTwo 4

Offset to the Comm module second function pointer (4 bytes)

6.2.2.157 #define CommOffsetUsbInBufBuf 1685

Offset to Usb input buffer data (64 bytes)

6.2.2.158 #define CommOffsetUsbInBufInPtr 1749

Offset to Usb input buffer front pointer (1 byte)

6.2.2.159 #define CommOffsetUsbInBufOutPtr 1750

Offset to Usb input buffer back pointer (1 byte)

6.2.2.160 #define CommOffsetUsbOutBufBuf 1753

Offset to Usb output buffer data (64 bytes)

6.2.2.161 #define CommOffsetUsbOutBufInPtr 1817

Offset to Usb output buffer front pointer (1 byte)

6.2.2.162 #define CommOffsetUsbOutBufOutPtr 1818

Offset to Usb output buffer back pointer (1 byte)

6.2.2.163 #define CommOffsetUsbPollBufBuf 1821

Offset to Usb Poll buffer data (64 bytes)

6.2.2.164 #define CommOffsetUsbPollBufInPtr 1885

Offset to Usb Poll buffer front pointer (1 byte)

6.2.2.165 #define CommOffsetUsbPollBufOutPtr 1886

Offset to Usb Poll buffer back pointer (1 byte)

6.2.2.166 #define CommOffsetUsbState 1894

Offset to Usb State (1 byte)

6.2.2.167 #define ComputeCalibValue 42

Compute a calibration value

6.2.2.168 #define CONN\_BT0 0x0

Bluetooth connection 0

6.2.2.169 #define CONN\_BT1 0x1

Bluetooth connection 1

6.2.2.170 #define CONN\_BT2 0x2

Bluetooth connection 2

6.2.2.171 #define CONN\_BT3 0x3

Bluetooth connection 3

6.2.2.172 #define CONN\_HS4 0x4

RS485 (hi-speed) connection (port 4, all devices)

6.2.2.173 #define CONN\_HS\_1 0x5

RS485 (hi-speed) connection (port 4, device address 1)

6.2.2.174 #define CONN\_HS\_2 0x6

RS485 (hi-speed) connection (port 4, device address 2)

6.2.2.175 #define CONN\_HS\_3 0x7

RS485 (hi-speed) connection (port 4, device address 3)

6.2.2.176 #define CONN\_HS\_4 0x8

RS485 (hi-speed) connection (port 4, device address 4)

6.2.2.177 #define CONN\_HS\_5 0x9

RS485 (hi-speed) connection (port 4, device address 5)

6.2.2.178 #define CONN\_HS\_6 0xa

RS485 (hi-speed) connection (port 4, device address 6)

6.2.2.179 #define CONN\_HS\_7 0xb

RS485 (hi-speed) connection (port 4, device address 7)

6.2.2.180 #define CONN\_HS\_8 0xc

RS485 (hi-speed) connection (port 4, device address 8)

6.2.2.181 #define CONN\_HS\_ALL 0x4

RS485 (hi-speed) connection (port 4, all devices)

6.2.2.182 #define CT\_ADDR\_RFID 0x04

RFID I2C address

6.2.2.183 #define CT\_REG\_DATA 0x42

RFID data register

6.2.2.184 #define CT\_REG\_MODE 0x41

RFID mode register

6.2.2.185 #define CT\_REG\_STATUS 0x32

RFID status register

6.2.2.186 #define DAC\_MODE\_DCOUT 0

Steady (DC) voltage output.

6.2.2.187 #define DAC\_MODE\_PWMVOLTAGE 6

PWM square wave output.

6.2.2.188 #define DAC\_MODE\_RESTART\_MASK 0x80

Add mask to DAC mode constants to force waveform generation from the start of the wave table.

6.2.2.189 #define DAC\_MODE\_SAWNEGWAVE 4

Negative going sawtooth output.

6.2.2.190 #define DAC\_MODE\_SAWPOSWAVE 3

Positive going sawtooth output.

6.2.2.191 #define DAC\_MODE\_SINEWAVE 1

Sine wave output.

6.2.2.192 #define DAC\_MODE\_SQUAREWAVE 2

Square wave output.

6.2.2.193 #define DAC\_MODE\_TRIANGLEWAVE 5

Triangle wave output.

6.2.2.194 #define DATA\_MODE\_GPS 0x01

Use GPS data mode

6.2.2.195 #define DATA\_MODE\_MASK 0x07

A mask for the data mode bits.

6.2.2.196 #define DATA\_MODE\_NXT 0x00

Use NXT data mode

6.2.2.197 #define DATA\_MODE\_RAW 0x02

Use RAW data mode

6.2.2.198 #define DATA\_MODE\_UPDATE 0x08

Indicates that the data mode has been changed.

6.2.2.199 #define DatalogGetTimes 45

Get datalog timing information

6.2.2.200 #define DatalogWrite 44

Write to the datalog

6.2.2.201 #define DEGREES\_PER\_RADIAN 180/PI

Used for converting from radians to degrees

6.2.2.202 #define DGPS\_REG\_DISTANCE 0x08

Read distance to current waypoint in meters.

6.2.2.203 #define DGPS\_REG\_HEADING 0x07

Read heading in degrees.

6.2.2.204 #define DGPS\_REG\_LASTANGLE 0x0A

Read angle travelled since last request, resets the request coordinates on the GPS sensor, sends the angle of travel since last reset.

6.2.2.205 #define DGPS\_REG\_LATITUDE 0x02

Read integer latitude.(dddddd; Positive = North; Negative = South).

6.2.2.206 #define DGPS\_REG\_LONGITUDE 0x04

Read integer longitude (dddddd; Positive = East; Negative = West).

6.2.2.207 #define DGPS\_REG\_SETLATITUDE 0x0B

Set waypoint latitude as a 4 byte integer.

6.2.2.208 #define DGPS\_REG\_SETLONGITUDE 0x0C

Set waypoint longitude as a 4 byte integer.

6.2.2.209 #define DGPS\_REG\_STATUS 0x01

Read status of the GPS (0 - invalid signal, 1 - valid signal).

6.2.2.210 #define DGPS\_REG\_TIME 0x00

Read time in UTC (hhmmss).

6.2.2.211 #define DGPS\_REG\_VELOCITY 0x06

Read velocity in cm/s.

6.2.2.212 #define DGPS\_REG\_WAYANGLE 0x09

Read angle to current waypoint in degrees.

6.2.2.213 #define DI\_ADDR\_ACCL 0x3A

Dexter Industries DIMU Accelerometer I2C address

6.2.2.214 #define DI\_ADDR\_DGPS 0x06

Dexter Industries DGPS I2C address

6.2.2.215 #define DI\_ADDR\_GYRO 0xD2

Dexter Industries DIMU Gyro I2C address

6.2.2.216 #define DIACCL\_CTRL1\_FILT\_BW125 0x80

Accelerometer digital filter band width is 125 Hz.

6.2.2.217 #define DIACCL\_CTRL1\_INT2TOINT1 0x01

Accelerometer INT2 pin is routed to INT1 bit in Detection Source Register (\$0A) and INT1 pin is routed to INT2 bit in Detection Source Register (\$0A)

6.2.2.218 #define DIACCL\_CTRL1\_LEVELPULSE 0x00

Accelerometer INT1 register is detecting Level while INT2 is detecting pulse

6.2.2.219 #define DIACCL\_CTRL1\_NO\_XDETECT 0x08

Accelerometer disable x-axis detection.

6.2.2.220 #define DIACCL\_CTRL1\_NO\_YDETECT 0x10

Accelerometer disable y-axis detection.

6.2.2.221 #define DIACCL\_CTRL1\_NO\_ZDETECT 0x20

Accelerometer disable z-axis detection.

6.2.2.222 #define DIACCL\_CTRL1\_PULSELEVEL 0x02

Accelerometer INT1 Register is detecting Pulse while INT2 is detecting Level

6.2.2.223 #define DIACCL\_CTRL1\_PULSEPULSE 0x04

Accelerometer INT1 Register is detecting a Single Pulse and INT2 is detecting Single Pulse (if 2nd Time Window = 0) or if there is a latency time window and second time window > 0 then INT2 will detect the double pulse only.

6.2.2.224 #define DIACCL\_CTRL1\_THRESH\_INT 0x40

Accelerometer threshold value can be an integer.

6.2.2.225 #define DIACCL\_CTRL2\_DETPOL\_NEGAND 0x02

Accelerometer pulse detection polarity is negative and detecting condition is AND all 3 axes

6.2.2.226 #define DIACCL\_CTRL2\_DRIVE\_STRONG 0x04

Accelerometer strong drive strength on SDA/SDO pin

6.2.2.227 #define DIACCL\_CTRL2\_LVLPOLE\_NEGAND 0x01

Accelerometer level detection polarity is negative and detecting condition is AND all 3 axes

6.2.2.228 #define DIACCL\_INTERRUPT\_LATCH\_CLEAR1 0x01

Accelerometer clear interrupt 1

6.2.2.229 #define DIACCL\_INTERRUPT\_LATCH\_CLEAR2 0x02

Accelerometer clear interrupt 2

6.2.2.230 #define DIACCL\_MODE\_GLVL2 0x04

Accelerometer 2G measurement range

6.2.2.231 #define DIACCL\_MODE\_GLVL4 0x08

Accelerometer 4G measurement range

6.2.2.232 #define DIACCL\_MODE\_GLVL8 0x00

Accelerometer 8G measurement range

6.2.2.233 #define DIACCL\_MODE\_LVLDETECT 0x02

Accelerometer level detect mode

6.2.2.234 #define DIACCL\_MODE\_MEASURE 0x01

Accelerometer measurement mode

6.2.2.235 #define DIACCL\_MODE\_PLSDETECT 0x03

Accelerometer pulse detect mode

6.2.2.236 #define DIACCL\_MODE\_STANDBY 0x00

Accelerometer standby mode

6.2.2.237 #define DIACCL\_REG\_CTRL1 0x18

Accelerometer control register 1 (read/write)

6.2.2.238 #define DIACCL\_REG\_CTRL2 0x19

Accelerometer control register 1 (read/write)

6.2.2.239 #define DIACCL\_REG\_DETECTSRC 0x0A

Accelerometer detection source register (read only)

6.2.2.240 #define DIACCL\_REG\_I2CADDR 0x0D

Accelerometer I2C address register (read only)

6.2.2.241 #define DIACCL\_REG\_INTLATCH 0x17

Accelerometer interrupt latch reset register (read/write)

6.2.2.242 #define DIACCL\_REG\_LATENCYTM 0x1D

Accelerometer latency time value register (read/write)

6.2.2.243 #define DIACCL\_REG\_LVLDETTHR 0x1A

Accelerometer level detection threshold limit value register (read/write)

6.2.2.244 #define DIACCL\_REG\_MODECTRL 0x16

Accelerometer mode control register (read/write)

6.2.2.245 #define DIACCL\_REG\_OUTTEMP 0x0B

Accelerometer temperature output register (read only)

6.2.2.246 #define DIACCL\_REG\_PLSDETTHR 0x1B

Accelerometer pulse detection threshold limit value register (read/write)

6.2.2.247 #define DIACCL\_REG\_PLSDURVAL 0x1C

Accelerometer pulse duration value register (read/write)

6.2.2.248 #define DIACCL\_REG\_STATUS 0x09

Accelerometer status register (read only)

6.2.2.249 #define DIACCL\_REG\_TIMEWINDOW 0x1E

Accelerometer time window for 2nd pulse value register (read/write)

6.2.2.250 #define DIACCL\_REG\_USERINFO 0x0E

Accelerometer user information register (read only)

6.2.2.251 #define DIACCL\_REG\_WHOAMI 0x0F

Accelerometer device identification register (read only)

6.2.2.252 #define DIACCL\_REG\_X8 0x06

Accelerometer x-axis 8-bit register (read only)

6.2.2.253 #define DIACCL\_REG\_XHIGH 0x01

Accelerometer x-axis high byte register (read only)

6.2.2.254 #define DIACCL\_REG\_XHIGHDRIFT 0x11

Accelerometer x-axis offset drift high byte register (read/write)

6.2.2.255 #define DIACCL\_REG\_XLOW 0x00

Accelerometer x-axis low byte register (read only)

6.2.2.256 #define DIACCL\_REG\_XLOWDRIFT 0x10

Accelerometer x-axis offset drift low byte register (read/write)

6.2.2.257 #define DIACCL\_REG\_Y8 0x07

Accelerometer x-axis 8-bit register (read only)

6.2.2.258 #define DIACCL\_REG\_YHIGH 0x03

Accelerometer y-axis high byte register (read only)

6.2.2.259 #define DIACCL\_REG\_YHIGHDRIFT 0x13

Accelerometer y-axis offset drift high byte register (read/write)

6.2.2.260 #define DIACCL\_REG\_YLOW 0x02

Accelerometer y-axis low byte register (read only)

6.2.2.261 #define DIACCL\_REG\_YLOWDRIFT 0x12

Accelerometer y-axis offset drift low byte register (read/write)

6.2.2.262 #define DIACCL\_REG\_Z8 0x08

Accelerometer x-axis 8-bit register (read only)

6.2.2.263 #define DIACCL\_REG\_ZHIGH 0x05

Accelerometer z-axis high byte register (read only)

6.2.2.264 #define DIACCL\_REG\_ZHIGHDRIFT 0x15

Accelerometer z-axis offset drift high byte register (read/write)

6.2.2.265 #define DIACCL\_REG\_ZLOW 0x04

Accelerometer z-axis low byte register (read only)

6.2.2.266 #define DIACCL\_REG\_ZLOWDRIFT 0x14

Accelerometer z-axis offset drift low byte register (read/write)

6.2.2.267 #define DIACCL\_STATUS\_DATAOVER 0x02

Accelerometer data is overwritten

6.2.2.268 #define DIACCL\_STATUS\_DATAREADY 0x01

Accelerometer data is ready

6.2.2.269 #define DIACCL\_STATUS\_PARITYERR 0x04

Accelerometer parity error is detected in trim data

6.2.2.270 #define DIGI\_PIN0 0x01

Access digital pin 0 (B0)

6.2.2.271 #define DIGI\_PIN1 0x02

Access digital pin 1 (B1)

6.2.2.272 #define DIGI\_PIN2 0x04

Access digital pin 2 (B2)

6.2.2.273 #define DIGI\_PIN3 0x08

Access digital pin 3 (B3)

6.2.2.274 #define DIGI\_PIN4 0x10

Access digital pin 4 (B4)

6.2.2.275 #define DIGI\_PIN5 0x20

Access digital pin 5 (B5)

6.2.2.276 #define DIGI\_PIN6 0x40

Access digital pin 6 (B6)

6.2.2.277 #define DIGI\_PIN7 0x80

Access digital pin 7 (B7)

6.2.2.278 #define DIGYRO\_CTRL1\_BANDWIDTH\_1 0x00

Gyro LPF2 cut-off frequency bandwidth level 1 (12.5hz, 12.5hz, 20hz, 30hz)

6.2.2.279 #define DIGYRO\_CTRL1\_BANDWIDTH\_2 0x10

Gyro LPF2 cut-off frequency bandwidth level 2 (12.5hz, 25hz, 50hz, 70hz)

6.2.2.280 #define DIGYRO\_CTRL1\_BANDWIDTH\_3 0x20

Gyro LPF2 cut-off frequency bandwidth level 3 (20hz, 25hz, 50hz, 110hz)

6.2.2.281 #define DIGYRO\_CTRL1\_BANDWIDTH\_4 0x30

Gyro LPF2 cut-off frequency bandwidth level 4 (30hz, 35hz, 50hz, 110hz)

6.2.2.282 #define DIGYRO\_CTRL1\_DATARATE\_100 0x00

Gyro output data rate 100 hz

6.2.2.283 #define DIGYRO\_CTRL1\_DATARATE\_200 0x40

Gyro output data rate 200 hz

6.2.2.284 #define DIGYRO\_CTRL1\_DATARATE\_400 0x80

Gyro output data rate 400 hz

6.2.2.285 #define DIGYRO\_CTRL1\_DATARATE\_800 0xC0

Gyro output data rate 800 hz

6.2.2.286 #define DIGYRO\_CTRL1\_NORMAL 0x08

Gyro disable power down mode

6.2.2.287 #define DIGYRO\_CTRL1\_POWERDOWN 0x00

Gyro enable power down mode

6.2.2.288 #define DIGYRO\_CTRL1\_XENABLE 0x01

Gyro enable X axis

6.2.2.289 #define DIGYRO\_CTRL1\_YENABLE 0x02

Gyro enable Y axis

6.2.2.290 #define DIGYRO\_CTRL1\_ZENABLE 0x04

Gyro enable Z axis

6.2.2.291 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_001 0x09

Gyro high pass filter cutoff frequency 0.01 hz

6.2.2.292 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_002 0x08

Gyro high pass filter cutoff frequency 0.02 hz

6.2.2.293 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_005 0x07

Gyro high pass filter cutoff frequency 0.05 hz

6.2.2.294 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_01 0x06

Gyro high pass filter cutoff frequency 0.1 hz

6.2.2.295 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_02 0x05

Gyro high pass filter cutoff frequency 0.2 hz

6.2.2.296 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_05 0x04

Gyro high pass filter cutoff frequency 0.5 hz

6.2.2.297 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_1 0x03

Gyro high pass filter cutoff frequency 1 hz

6.2.2.298 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_2 0x02

Gyro high pass filter cutoff frequency 2 hz

6.2.2.299 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_4 0x01

Gyro high pass filter cutoff frequency 4 hz

6.2.2.300 #define DIGYRO\_CTRL2\_CUTOFF\_FREQ\_8 0x00

Gyro high pass filter cutoff frequency 8 hz

6.2.2.301 #define DIGYRO\_CTRL2\_HPMODE\_AUTOINT 0x30

Gyro high pass filter autoreset on interrupt event mode

6.2.2.302 #define DIGYRO\_CTRL2\_HPMODE\_NORMAL 0x20

Gyro high pass filter normal mode

6.2.2.303 #define DIGYRO\_CTRL2\_HPMODE\_REFSIG 0x10

Gyro high pass filter reference signal mode

6.2.2.304 #define DIGYRO\_CTRL2\_HPMODE\_RESET 0x00

Gyro high pass filter reset mode

6.2.2.305 #define DIGYRO\_CTRL3\_INT1\_BOOT 0x40

Gyro boot status available on INT1

6.2.2.306 #define DIGYRO\_CTRL3\_INT1\_ENABLE 0x80

Gyro interrupt enable on INT1 pin

6.2.2.307 #define DIGYRO\_CTRL3\_INT1\_LOWACTIVE 0x20

Gyro interrupt active low on INT1

6.2.2.308 #define DIGYRO\_CTRL3\_INT2\_DATAREADY 0x08

Gyro data ready on DRDY/INT2

6.2.2.309 #define DIGYRO\_CTRL3\_INT2\_EMPTY 0x01

Gyro FIFO empty interrupt on DRDY/INT2

6.2.2.310 #define DIGYRO\_CTRL3\_INT2\_OVERRUN 0x02

Gyro FIFO overrun interrupt on DRDY/INT2

6.2.2.311 #define DIGYRO\_CTRL3\_INT2\_WATERMARK 0x04

Gyro FIFO watermark interrupt on DRDY/INT2

6.2.2.312 #define DIGYRO\_CTRL3\_OPENDRAIN 0x10

Gyro use open drain rather than push-pull

6.2.2.313 #define DIGYRO\_CTRL4\_BIGENDIAN 0x40

Gyro use big endian - MSB/LSB rather than LSB/MSB in output registers

6.2.2.314 #define DIGYRO\_CTRL4\_BLOCKDATA 0x80

Gyro block data update - output registers are not updated until MSB and LSB reading

6.2.2.315 #define DIGYRO\_CTRL4\_SCALE\_2000 0x30

Gyro 2000 degrees per second scale

6.2.2.316 #define DIGYRO\_CTRL4\_SCALE\_250 0x00

Gyro 250 degrees per second scale

6.2.2.317 #define DIGYRO\_CTRL4\_SCALE\_500 0x10

Gyro 500 degrees per second scale

6.2.2.318 #define DIGYRO\_CTRL5\_FIFOENABLE 0x40

Gyro enable FIFO

6.2.2.319 #define DIGYRO\_CTRL5\_HPENABLE 0x10

Gyro enable high pass filter

6.2.2.320 #define DIGYRO\_CTRL5\_INT1\_SEL\_1 0x00

Gyro non-high-pass-filtered data are used for interrupt generation

6.2.2.321 #define DIGYRO\_CTRL5\_INT1\_SEL\_2 0x04

Gyro high-pass-filtered data are used for interrupt generation

6.2.2.322 #define DIGYRO\_CTRL5\_INT1\_SEL\_3 0x08

Gyro low-pass-filtered data are used for interrupt generation

6.2.2.323 #define DIGYRO\_CTRL5\_OUT\_SEL\_1 0x00

Gyro data in data registers and FIFO are not high-pass filtered

6.2.2.324 #define DIGYRO\_CTRL5\_OUT\_SEL\_2 0x01

Gyro data in data registers and FIFO are high-pass filtered

6.2.2.325 #define DIGYRO\_CTRL5\_OUT\_SEL\_3 0x02

Gyro data in data registers and FIFO are low-pass filtered by LPF2

6.2.2.326 #define DIGYRO\_CTRL5\_REBOOTMEM 0x80

Gyro reboot memory content

6.2.2.327 #define DIGYRO\_FIFOCTRL\_BYPASS 0x00

Gyro FIFO bypass mode

6.2.2.328 #define DIGYRO\_FIFOCTRL\_BYPASS2STREAM 0x80

Gyro FIFO bypass-to-stream mode

6.2.2.329 #define DIGYRO\_FIFOCTRL\_FIFO 0x20

Gyro FIFO mode

6.2.2.330 #define DIGYRO\_FIFOCTRL\_STREAM 0x40

Gyro FIFO stream mode

6.2.2.331 #define DIGYRO\_FIFOCTRL\_STREAM2FIFO 0x60

Gyro FIFO stream-to-FIFO mode

6.2.2.332 #define DIGYRO\_FIFOCTRL\_WATERMARK\_MASK 0x1F

Gyro FIFO threshold. Watermark level setting mask (values from 0x00 to 0x1F)

6.2.2.333 #define DIGYRO\_REG\_CTRL1 0x20

Gyro control register 1

6.2.2.334 #define DIGYRO\_REG\_CTRL1AUTO 0xA0

Gyro control register 1 - auto increment write

6.2.2.335 #define DIGYRO\_REG\_CTRL2 0x21

Gyro control register 2

6.2.2.336 #define DIGYRO\_REG\_CTRL3 0x22

Gyro control register 3

6.2.2.337 #define DIGYRO\_REG\_CTRL4 0x23

Gyro control register 4

6.2.2.338 #define DIGYRO\_REG\_CTRL5 0x24

Gyro control register 5

6.2.2.339 #define DIGYRO\_REG\_FIFOCTRL 0x2E

Gyro FIFO control register

6.2.2.340 #define DIGYRO\_REG\_FIFOSRC 0x2F

Gyro FIFO source register (read only)

6.2.2.341 #define DIGYRO\_REG\_INT1\_CFG 0x30

Gyro interrupt 1 config register

6.2.2.342 #define DIGYRO\_REG\_INT1\_DUR 0x38

Gyro interrupt 1 duration register

6.2.2.343 #define DIGYRO\_REG\_INT1\_SRC 0x31

Gyro interrupt 1 source register

6.2.2.344 #define DIGYRO\_REG\_INT1\_XHI 0x32

Gyro interrupt 1 x-axis high threshold register

6.2.2.345 #define DIGYRO\_REG\_INT1\_XLO 0x33

Gyro interrupt 1 x-axis low threshold register

6.2.2.346 #define DIGYRO\_REG\_INT1\_YHI 0x34

Gyro interrupt 1 y-axis high threshold register

6.2.2.347 #define DIGYRO\_REG\_INT1\_YLO 0x35

Gyro interrupt 1 y-axis low threshold register

6.2.2.348 #define DIGYRO\_REG\_INT1\_ZHI 0x36

Gyro interrupt 1 z-axis high threshold register

6.2.2.349 #define DIGYRO\_REG\_INT1\_ZLO 0x37

Gyro interrupt 1 z-axis low threshold register

6.2.2.350 #define DIGYRO\_REG\_OUTTEMP 0x26

Gyro temperature register (read only) - stores temperature data

6.2.2.351 #define DIGYRO\_REG\_REFERENCE 0x25

Gyro reference register - stores the reference value used for interrupt generation

6.2.2.352 #define DIGYRO\_REG\_STATUS 0x27

Gyro status register (read only)

6.2.2.353 #define DIGYRO\_REG\_TEMPAUTO 0xA6

Gyro temperature register - read burst mode (read only)

6.2.2.354 #define DIGYRO\_REG\_WHOAMI 0x0F

Gyro device identification register (read only)

6.2.2.355 #define DIGYRO\_REG\_XHIGH 0x29

Gyro x-axis high byte register (read only)

6.2.2.356 #define DIGYRO\_REG\_XLOW 0x28

Gyro x-axis low byte register (read only)

6.2.2.357 #define DIGYRO\_REG\_XLOWBURST 0xA8

Gyro x-axis low byte register - read burst mode (read only)

6.2.2.358 #define DIGYRO\_REG\_YHIGH 0x2B

Gyro y-axis high byte register (read only)

6.2.2.359 #define DIGYRO\_REG\_YLOW 0x2A

Gyro y-axis low byte register (read only)

6.2.2.360 #define DIGYRO\_REG\_YLOWBURST 0xAA

Gyro y-axis low byte register - read burst mode (read only)

6.2.2.361 #define DIGYRO\_REG\_ZHIGH 0x2D

Gyro z-axis high byte register (read only)

6.2.2.362 #define DIGYRO\_REG\_ZLOW 0x2C

Gyro z-axis low byte register (read only)

6.2.2.363 #define DIGYRO\_REG\_ZLOWBURST 0xAC

Gyro y-axis low byte register - read burst mode (read only)

6.2.2.364 #define DIGYRO\_STATUS\_XDATA 0x01

Gyro X-axis new data available

6.2.2.365 #define DIGYRO\_STATUS\_XOVER 0x10

Gyro X-axis data overrun - new data for the X-axis has overwritten the previous one

6.2.2.366 #define DIGYRO\_STATUS\_XYZDATA 0x08

Gyro X, Y, or Z-axis new data available - a new set of data is available

6.2.2.367 #define DIGYRO\_STATUS\_XYZOVER 0x80

Gyro X, Y, or Z-axis data overrun - new data has overwritten the previous one before it was read

6.2.2.368 #define DIGYRO\_STATUS\_YDATA 0x02

Gyro Y-axis new data available

6.2.2.369 #define DIGYRO\_STATUS\_YOVER 0x20

Gyro Y-axis data overrun - new data for the Y-axis has overwritten the previous one

6.2.2.370 #define DIGYRO\_STATUS\_ZDATA 0x04

Gyro Z-axis new data available

6.2.2.371 #define DIGYRO\_STATUS\_ZOVER 0x40

Gyro Z-axis data overrun - new data for the Z-axis has overwritten the previous one

6.2.2.372 #define DISPLAY\_BUSY 0x80

R - Refresh in progress

6.2.2.373 #define DISPLAY\_CHAR 0x04

W - draw char (actual font) (CMD,TRUE,X1,Y1,Char,x)

6.2.2.374 #define DISPLAY\_CONTRAST\_DEFAULT 0x5A

Default display contrast value

6.2.2.375 #define DISPLAY\_CONTRAST\_MAX 0x7F

Maximum display contrast value

6.2.2.376 #define DISPLAY\_ERASE\_ALL 0x00

W - erase entire screen (CMD,x,x,x,x,x)

6.2.2.377 #define DISPLAY\_ERASE\_LINE 0x05

W - erase a single line (CMD,x,LINE,x,x,x)

6.2.2.378 #define DISPLAY\_FILL\_REGION 0x06

W - fill screen region (CMD,TRUE/FALSE,X1,Y1,X2,Y2)

6.2.2.379 #define DISPLAY\_FRAME 0x07

W - draw a frame (on/off) (CMD,TRUE/FALSE,X1,Y1,X2,Y2)

6.2.2.380 #define DISPLAY\_HEIGHT 64

The height of the LCD screen in pixels

6.2.2.381 #define DISPLAY\_HORIZONTAL\_LINE 0x02

W - draw horizontal line (CMD,TRUE/FALSE,X1,Y1,X2,x)

6.2.2.382 #define DISPLAY\_MENUICONS\_X\_DIFF 31

Display menu icons x delta

6.2.2.383 #define DISPLAY\_MENUICONS\_X\_OFFSET 7

Display menu icons x offset

6.2.2.384 #define DISPLAY\_MENUICONS\_Y 40

Display menu icons y value

6.2.2.385 #define DISPLAY\_ON 0x01

W - Display on

6.2.2.386 #define DISPLAY\_PIXEL 0x01

W - set pixel (on/off) (CMD,TRUE/FALSE,X,Y,x,x)

6.2.2.387 #define DISPLAY\_POPUP 0x08

W - Use popup display memory

6.2.2.388 #define DISPLAY\_REFRESH 0x02

W - Enable refresh

6.2.2.389 #define DISPLAY\_REFRESH\_DISABLED 0x40

R - Refresh disabled

6.2.2.390 #define DISPLAY\_VERTICAL\_LINE 0x03

W - draw vertical line (CMD,TRUE/FALSE,X1,Y1,x,Y2)

6.2.2.391 #define DISPLAY\_WIDTH 100

The width of the LCD screen in pixels

6.2.2.392 #define DisplayExecuteFunction 80

Execute one of the Display module's internal functions

6.2.2.393 #define DisplayModuleID 0x000A0001

The display module ID

6.2.2.394 #define DisplayModuleName "Display.mod"

The display module name

6.2.2.395 #define DisplayOffsetContrast 1719

Adjust the display contrast with this field (NBC/NXC)

6.2.2.396 #define DisplayOffsetDisplay 104

Display content copied to physical display every 17 mS

6.2.2.397 #define DisplayOffsetEraseMask 4

Section erase mask (executed first)

6.2.2.398 #define DisplayOffsetFlags 117

Update flags enumerated above

6.2.2.399 #define DisplayOffsetNormal( *l, w* ) (((*l*)\*100)+(*w*)+119)

Raw display memory for normal screen

6.2.2.400 #define DisplayOffsetPBitmaps( *p* ) (((*p*)\*4)+68)

Pointer to free bitmap files

6.2.2.401 #define DisplayOffsetPFont 12

Pointer to font file

6.2.2.402 #define DisplayOffsetPFunc 0

Simple draw entry

6.2.2.403 #define DisplayOffsetPMenulcons( *p* ) (((*p*)\*4)+88)

Pointer to menu icon images (NULL == none)

6.2.2.404 #define DisplayOffsetPMenuText 84

Pointer to menu icon text (NULL == none)

6.2.2.405 #define DisplayOffsetPopup( *l, w* ) (((*l*)\*100)+(*w*)+919)

Raw display memory for popup screen

6.2.2.406 #define DisplayOffsetPScreens( *p* ) (((*p*)\*4)+56)

Pointer to screen bitmap file

6.2.2.407 #define DisplayOffsetPStatusIcons 52

Pointer to status icon collection file

6.2.2.408 #define DisplayOffsetPStatusText 48

Pointer to status text string

6.2.2.409 #define DisplayOffsetPStepIcons 100

Pointer to step icon collection file

6.2.2.410 #define DisplayOffsetPTextLines( *p* ) (((*p*)\*4)+16)

Pointer to text strings

6.2.2.411 #define DisplayOffsetStatusIcons( *p* ) ((*p*)+108)

Index in status icon collection file (index = 0 -> none)

6.2.2.412 #define DisplayOffsetStepIcons( p ) ((p)+112)

Index in step icon collection file (index = 0 -> none)

6.2.2.413 #define DisplayOffsetTextLinesCenterFlags 118

Mask to center TextLines

6.2.2.414 #define DisplayOffsetUpdateMask 8

Section update mask (executed next)

6.2.2.415 #define DIST\_CMD\_CUSTOM 0x35

Set the DIST-Nx to a custom mode

6.2.2.416 #define DIST\_CMD\_GP2D12 0x31

Set the DIST-Nx to GP2D12 mode

6.2.2.417 #define DIST\_CMD\_GP2D120 0x32

Set the DIST-Nx to GP2D120 mode

6.2.2.418 #define DIST\_CMD\_GP2YA02 0x34

Set the DIST-Nx to GP2YA02 mode

6.2.2.419 #define DIST\_CMD\_GP2YA21 0x33

Set the DIST-Nx to GP2YA21 mode

6.2.2.420 #define DIST\_REG\_DIST 0x42

The DIST-Nx distance register

6.2.2.421 #define DIST\_REG\_DIST1 0x58

The DIST-Nx distance 1 register

6.2.2.422 #define DIST\_REG\_DIST\_MAX 0x54

The DIST-Nx maximum distance register

6.2.2.423 #define DIST\_REG\_DIST\_MIN 0x52

The DIST-Nx minimum distance register

6.2.2.424 #define DIST\_REG\_MODULE\_TYPE 0x50

The DIST-Nx module type register

6.2.2.425 #define DIST\_REG\_NUM\_POINTS 0x51

The DIST-Nx number of data points in Custom curve register

6.2.2.426 #define DIST\_REG\_VOLT 0x44

The DIST-Nx voltage register

6.2.2.427 #define DIST\_REG\_VOLT1 0x56

The DIST-Nx voltage 1 register

6.2.2.428 #define DRAW\_OPT\_CLEAR (0x0004)

Clear pixels while drawing (aka draw in white)

6.2.2.429 #define DRAW\_OPT\_CLEAR\_EOL (0x1000)

When drawing text, clear to the end of the line after drawing the text

6.2.2.430 #define DRAW\_OPT\_CLEAR\_EXCEPT\_STATUS\_SCREEN (0x0002)

Clear the screen except for the status line before drawing

6.2.2.431 #define DRAW\_OPT\_CLEAR\_LINE (0x0800)

When drawing text, clear the entire line before drawing the text

6.2.2.432 #define DRAW\_OPT\_CLEAR\_PIXELS (0x0004)

Clear pixels while drawing (aka draw in white)

6.2.2.433 #define DRAW\_OPT\_CLEAR\_SCREEN\_MODES (0x0003)

Bit mask for the clear screen modes

6.2.2.434 #define DRAW\_OPT\_CLEAR\_WHOLE\_SCREEN (0x0001)

Clear the entire screen before drawing

6.2.2.435 #define DRAW\_OPT\_FILL\_SHAPE (0x0020)

Fill the shape while drawing (rectangle, circle, ellipses, and polygon)

6.2.2.436 #define DRAW\_OPT\_FONT\_DIR\_B2TL (0x0100)

Font bottom to top left align

6.2.2.437 #define DRAW\_OPT\_FONT\_DIR\_B2TR (0x0140)

Font bottom to top right align

6.2.2.438 #define DRAW\_OPT\_FONT\_DIR\_L2RB (0x0000)

Font left to right bottom align

6.2.2.439 #define DRAW\_OPT\_FONT\_DIR\_L2RT (0x0040)

Font left to right top align

6.2.2.440 `#define DRAW_OPT_FONT_DIR_R2LB (0x0080)`

Font right to left bottom align

6.2.2.441 `#define DRAW_OPT_FONT_DIR_R2LT (0x00C0)`

Font right to left top align

6.2.2.442 `#define DRAW_OPT_FONT_DIR_T2BL (0x0180)`

Font top to bottom left align

6.2.2.443 `#define DRAW_OPT_FONT_DIR_T2BR (0x01C0)`

Font top to bottom right align

6.2.2.444 `#define DRAW_OPT_FONT_DIRECTIONS (0x01C0)`

Bit mask for the font direction bits

6.2.2.445 `#define DRAW_OPT_FONT_WRAP (0x0200)`

Option to have text wrap in [FontNumOut](#) and [FontTextOut](#) calls

6.2.2.446 `#define DRAW_OPT_INVERT (0x0004)`

Invert text or graphics

6.2.2.447 `#define DRAW_OPT_LOGICAL_AND (0x0008)`

Draw pixels using a logical AND operation

6.2.2.448 `#define DRAW_OPT_LOGICAL_COPY (0x0000)`

Draw pixels using a logical copy operation

6.2.2.449 `#define DRAW_OPT_LOGICAL_OPERATIONS (0x0018)`

Bit mask for the logical drawing operations

6.2.2.450 `#define DRAW_OPT_LOGICAL_OR (0x0010)`

Draw pixels using a logical OR operation

6.2.2.451 `#define DRAW_OPT_LOGICAL_XOR (0x0018)`

Draw pixels using a logical XOR operation

6.2.2.452 `#define DRAW_OPT_NORMAL (0x0000)`

Normal drawing

6.2.2.453 `#define DRAW_OPT_POLYGON_POLYLINE (0x0400)`

When drawing polygons, do not close (i.e., draw a polyline instead)

**6.2.2.454 #define DrawCircle 16**

Draw a circle on the LCD screen

**6.2.2.455 #define DrawEllipse 94**

Draw an ellipse on the LCD screen

**6.2.2.456 #define DrawFont 95**

Draw text using a custom RIC-based font to the LCD screen

**6.2.2.457 #define DrawGraphic 18**

Draw a graphic image on the LCD screen

**6.2.2.458 #define DrawGraphicArray 92**

Draw a graphic image from a byte array to the LCD screen

**6.2.2.459 #define DrawLine 15**

Draw a line on the LCD screen

**6.2.2.460 #define DrawPoint 14**

Draw a single pixel on the LCD screen

**6.2.2.461 #define DrawPolygon 93**

Draw a polygon on the LCD screen

**6.2.2.462 #define DrawRect 17**

Draw a rectangle on the LCD screen

**6.2.2.463 #define DrawText 13**

Draw text to one of 8 LCD lines

**6.2.2.464 #define EMETER\_REG\_AIN 0x0c**

The register address for amps in

**6.2.2.465 #define EMETER\_REG\_AOUT 0x10**

The register address for amps out

**6.2.2.466 #define EMETER\_REG\_JOULES 0x12**

The register address for joules

**6.2.2.467 #define EMETER\_REG\_VIN 0x0a**

The register address for voltage in

6.2.2.468 #define EMETER\_REG\_VOUT 0x0e

The register address for voltage out

6.2.2.469 #define EMETER\_REG\_WIN 0x14

The register address for watts in

6.2.2.470 #define EMETER\_REG\_WOUT 0x16

The register address for watts out

6.2.2.471 #define EOF -1

A constant representing end of file

6.2.2.472 #define ERR\_ARG -1

0xFF Bad arguments

6.2.2.473 #define ERR\_BAD\_POOL\_SIZE -10

0xF6 VarsCmd.PoolSize > POOL\_MAX\_SIZE

6.2.2.474 #define ERR\_BAD\_PTR -6

0xFA Someone passed us a bad pointer!

6.2.2.475 #define ERR\_CLUMP\_COUNT -7

0xF9 (FileClumpCount == 0 || FileClumpCount >= NOT\_A\_CLUMP)

6.2.2.476 #define ERR\_COMM\_BUFFER\_FULL -34

0xDE No room in comm buffer

6.2.2.477 #define ERR\_COMM\_BUS\_ERR -35

0xDD Something went wrong on the communications bus

6.2.2.478 #define ERR\_COMM\_CHAN\_INVALID -33

0xDF Specified channel/connection is not valid

6.2.2.479 #define ERR\_COMM\_CHAN\_NOT\_READY -32

0xE0 Specified channel/connection not configured or busy

6.2.2.480 #define ERR\_DEFAULT\_OFFSETS -14

0xF2 (DefaultsOffset != FileOffsets.DynamicDefaults) || (DefaultsOffset + FileOffsets.DynamicDefaultsSize != FileOffsets.DSDefaultsSize)

6.2.2.481 #define ERR\_FILE -3

0xFD Malformed file contents

6.2.2.482 #define **ERR\_INSANE\_OFFSET** -9

0xF7 CurrOffset != (DataSize - VarsCmd.CodespaceCount \* 2)

6.2.2.483 #define **ERR\_INSTR** -2

0xFE Illegal bytecode instruction

6.2.2.484 #define **ERR\_INVALID\_FIELD** -17

0xEF Attempted to access invalid field of a structure

6.2.2.485 #define **ERR\_INVALID\_PORT** -16

0xF0 Bad input or output port specified

6.2.2.486 #define **ERR\_INVALID\_QUEUE** -18

0xEE Illegal queue ID specified

6.2.2.487 #define **ERR\_INVALID\_SIZE** -19

0xED Illegal size specified

6.2.2.488 #define **ERR\_LOADER\_ERR** -11

0xF5 LOADER\_ERR(Lstatus) != SUCCESS || pData == NULL || DataSize == 0

6.2.2.489 #define **ERR\_MEM** -5

0xFB Insufficient memory available

6.2.2.490 #define **ERR\_MEMMGR\_FAIL** -15

0xF1 (UBYTE \*)VarsCmd.MemMgr.pDopeVectorArray != VarsCmd.pDataspace + DV\_ARRAY[0].Offset

6.2.2.491 #define **ERR\_NO\_ACTIVE\_CLUMP** -13

0xF3 VarsCmd.RunQ.Head == NOT\_A\_CLUMP

6.2.2.492 #define **ERR\_NO\_CODE** -8

0xF8 VarsCmd.CodespaceCount == 0

6.2.2.493 #define **ERR\_NO\_PROG** -20

0xEC No active program

6.2.2.494 #define **ERR\_NON\_FATAL** -16

Fatal errors are greater than this value

6.2.2.495 #define **ERR\_RC\_BAD\_PACKET** -65

0xBF Clearly insane packet

6.2.2.496 #define ERR\_RC\_FAILED -67

0xBD Request failed (i.e. specified file not found)

6.2.2.497 #define ERR\_RC\_ILLEGAL\_VAL -64

0xC0 Data contains out-of-range values

6.2.2.498 #define ERR\_RC\_UNKNOWN\_CMD -66

0xBE Unknown command opcode

6.2.2.499 #define ERR\_SPOTCHECK\_FAIL -12

0xF4 ((UBYTE\*)(VarsCmd.pCodespace) < pData) (c\_cmd.c 1893)

6.2.2.500 #define ERR\_VER -4

0xFC Version mismatch between firmware and compiler

6.2.2.501 #define FALSE 0

A false value

6.2.2.502 #define FileClose 5

Close the specified file

6.2.2.503 #define FileDelete 8

Delete a file

6.2.2.504 #define FileFindFirst 83

Start a search for a file using a filename pattern

6.2.2.505 #define FileFindNext 84

Continue searching for a file

6.2.2.506 #define FileOpenAppend 2

Open a file for appending to the end of the file

6.2.2.507 #define FileOpenRead 0

Open a file for reading

6.2.2.508 #define FileOpenReadLinear 87

Open a linear file for reading

6.2.2.509 #define FileOpenWrite 1

Open a file for writing (creates a new file)

6.2.2.510 #define FileOpenWriteLinear 85

Open a linear file for writing

6.2.2.511 #define FileOpenWriteNonLinear 86

Open a non-linear file for writing

6.2.2.512 #define FileRead 3

Read from the specified file

6.2.2.513 #define FileRename 7

Rename a file

6.2.2.514 #define FileResize 91

Resize a file (not yet implemented)

6.2.2.515 #define FileResolveHandle 6

Get a file handle for the specified filename if it is already open

6.2.2.516 #define FileSeek 90

Seek to a specific position in an open file

6.2.2.517 #define FileTell 98

Return the current file position in an open file

6.2.2.518 #define FileWrite 4

Write to the specified file

6.2.2.519 #define FRAME\_SELECT 0

Center icon select frame

6.2.2.520 #define FREQUENCY\_MAX 14080

Maximum frequency [Hz]

6.2.2.521 #define FREQUENCY\_MIN 220

Minimum frequency [Hz]

6.2.2.522 #define GetStartTick 25

Get the current system tick count

6.2.2.523 #define GL\_CAMERA\_DEPTH 3

Set the camera depth.

---

6.2.2.524 #define GL\_CIRCLE 4

Use circle mode.

6.2.2.525 #define GL\_CIRCLE\_SIZE 1

Set the circle size.

6.2.2.526 #define GL\_CULL\_BACK 2

Cull lines in back.

6.2.2.527 #define GL\_CULL\_FRONT 3

Cull lines in front.

6.2.2.528 #define GL\_CULL\_MODE 2

Set the cull mode.

6.2.2.529 #define GL\_CULL\_NONE 4

Do not cull any lines.

6.2.2.530 #define GL\_LINE 2

Use line mode.

6.2.2.531 #define GL\_POINT 3

Use point mode.

6.2.2.532 #define GL\_POLYGON 1

Use polygon mode.

6.2.2.533 #define GL\_ROTATE\_X 4

Rotate around the X axis.

6.2.2.534 #define GL\_ROTATE\_Y 5

Rotate around the Y axis.

6.2.2.535 #define GL\_ROTATE\_Z 6

Rotate around the Z axis.

6.2.2.536 #define GL\_SCALE\_X 7

Scale along the X axis.

6.2.2.537 #define GL\_SCALE\_Y 8

Scale along the Y axis.

6.2.2.538 #define GL\_SCALE\_Z 9

Scale along the Z axis.

6.2.2.539 #define GL\_TRANSLATE\_X 1

Translate along the X axis.

6.2.2.540 #define GL\_TRANSLATE\_Y 2

Translate along the Y axis.

6.2.2.541 #define GL\_TRANSLATE\_Z 3

Translate along the Z axis.

6.2.2.542 #define GL\_ZOOM\_FACTOR 4

Set the zoom factor.

6.2.2.543 #define HS\_ADDRESS\_1 1

HsAddress device address 1

6.2.2.544 #define HS\_ADDRESS\_2 2

HsAddress device address 2

6.2.2.545 #define HS\_ADDRESS\_3 3

HsAddress device address 3

6.2.2.546 #define HS\_ADDRESS\_4 4

HsAddress device address 4

6.2.2.547 #define HS\_ADDRESS\_5 5

HsAddress device address 5

6.2.2.548 #define HS\_ADDRESS\_6 6

HsAddress device address 6

6.2.2.549 #define HS\_ADDRESS\_7 7

HsAddress device address 7

6.2.2.550 #define HS\_ADDRESS\_8 8

HsAddress device address 8

6.2.2.551 #define HS\_ADDRESS\_ALL 0

HsAddress all devices

6.2.2.552 #define HS\_BAUD\_115200 12

HsSpeed 115200 Baud

6.2.2.553 #define HS\_BAUD\_1200 0

HsSpeed 1200 Baud

6.2.2.554 #define HS\_BAUD\_14400 6

HsSpeed 14400 Baud

6.2.2.555 #define HS\_BAUD\_19200 7

HsSpeed 19200 Baud

6.2.2.556 #define HS\_BAUD\_230400 13

HsSpeed 230400 Baud

6.2.2.557 #define HS\_BAUD\_2400 1

HsSpeed 2400 Baud

6.2.2.558 #define HS\_BAUD\_28800 8

HsSpeed 28800 Baud

6.2.2.559 #define HS\_BAUD\_3600 2

HsSpeed 3600 Baud

6.2.2.560 #define HS\_BAUD\_38400 9

HsSpeed 38400 Baud

6.2.2.561 #define HS\_BAUD\_460800 14

HsSpeed 460800 Baud

6.2.2.562 #define HS\_BAUD\_4800 3

HsSpeed 4800 Baud

6.2.2.563 #define HS\_BAUD\_57600 10

HsSpeed 57600 Baud

6.2.2.564 #define HS\_BAUD\_7200 4

HsSpeed 7200 Baud

6.2.2.565 #define HS\_BAUD\_76800 11

HsSpeed 76800 Baud

6.2.2.566 #define HS\_BAUD\_921600 15

HsSpeed 921600 Baud

6.2.2.567 #define HS\_BAUD\_9600 5

HsSpeed 9600 Baud

6.2.2.568 #define HS\_BAUD\_DEFAULT 15

HsSpeed default Baud (921600)

6.2.2.569 #define HS\_BYTES\_REMAINING 16

HsState bytes remaining to be sent

6.2.2.570 #define HS\_CMD\_READY 0x04

A constant representing high speed direct command

6.2.2.571 #define HS\_CTRL\_EXIT 2

Ddisable the high speed port

6.2.2.572 #define HS\_CTRL\_INIT 0

Enable the high speed port

6.2.2.573 #define HS\_CTRL\_UART 1

Setup the high speed port UART configuration

6.2.2.574 #define HS\_DEFAULT 6

HsState default

6.2.2.575 #define HS\_DISABLE 4

HsState disable

6.2.2.576 #define HS\_ENABLE 5

HsState enable

6.2.2.577 #define HS\_INIT\_RECEIVER 2

HsState initialize receiver

6.2.2.578 #define HS\_INITIALISE 1

HsState initialize

6.2.2.579 #define HS\_MODE\_10\_STOP 0x0000

HsMode 1 stop bit

6.2.2.580 #define HS\_MODE\_15\_STOP 0x1000

HsMode 1.5 stop bits

6.2.2.581 #define HS\_MODE\_20\_STOP 0x2000

HsMode 2 stop bits

6.2.2.582 #define HS\_MODE\_5\_DATA 0x0000

HsMode 5 data bits

6.2.2.583 #define HS\_MODE\_6\_DATA 0x0040

HsMode 6 data bits

6.2.2.584 #define HS\_MODE\_7\_DATA 0x0080

HsMode 7 data bits

6.2.2.585 #define HS\_MODE\_7E1 (HS\_MODE\_7\_DATA|HS\_MODE\_E\_PARITY|HS\_MODE\_10\_STOP)

HsMode 7 data bits, even parity, 1 stop bit

6.2.2.586 #define HS\_MODE\_8\_DATA 0x00C0

HsMode 8 data bits

6.2.2.587 #define HS\_MODE\_8N1 (HS\_MODE\_8\_DATA|HS\_MODE\_N\_PARITY|HS\_MODE\_10\_STOP)

HsMode 8 data bits, no parity, 1 stop bit

6.2.2.588 #define HS\_MODE\_DEFAULT HS\_MODE\_8N1

HsMode default mode (8 data bits, no parity, 1 stop bit)

6.2.2.589 #define HS\_MODE\_E\_PARITY 0x0000

HsMode Even parity

6.2.2.590 #define HS\_MODE\_M\_PARITY 0x0600

HsMode Mark parity

6.2.2.591 #define HS\_MODE\_MASK 0x3EC0

HsMode mode mask

6.2.2.592 #define HS\_MODE\_N\_PARITY 0x0800

HsMode No parity

6.2.2.593 #define HS\_MODE\_O\_PARITY 0x0200

HsMode Odd parity

6.2.2.594 #define HS\_MODE\_S\_PARITY 0x0400

HsMode Space parity

6.2.2.595 #define HS\_MODE\_UART\_RS232 0x1

HsMode UART in normal or RS232 mode

6.2.2.596 #define HS\_MODE\_UART\_RS485 0x0

HsMode UART in default or RS485 mode

6.2.2.597 #define HS\_SEND\_DATA 3

HsState send data

6.2.2.598 #define HS\_UART\_MASK 0x000F

HsMode UART mask

6.2.2.599 #define HS\_UPDATE 1

HsFlags high speed update required

6.2.2.600 #define HT\_ADDR\_ACCEL 0x02

HiTechnic Accel I2C address

6.2.2.601 #define HT\_ADDR\_ANGLE 0x02

HiTechnic Angle I2C address

6.2.2.602 #define HT\_ADDR\_BAROMETRIC 0x02

HiTechnic Barometric I2C address

6.2.2.603 #define HT\_ADDR\_COLOR 0x02

HiTechnic Color I2C address

6.2.2.604 #define HT\_ADDR\_COLOR2 0x02

HiTechnic Color2 I2C address

6.2.2.605 #define HT\_ADDR\_COMPASS 0x02

HiTechnic Compass I2C address

6.2.2.606 #define HT\_ADDR\_IRLINK 0x02

HiTechnic IRLink I2C address

6.2.2.607 #define HT\_ADDR\_IRRECEIVER 0x02

HiTechnic IRReceiver I2C address

6.2.2.608 #define HT\_ADDR\_IRSEEKER 0x02

HiTechnic IRSeeker I2C address

6.2.2.609 #define HT\_ADDR\_IRSEEKER2 0x10

HiTechnic IRSeeker2 I2C address

6.2.2.610 #define HT\_ADDR\_PIR 0x02

HiTechnic PIR (Passive Infrared) I2C address

6.2.2.611 #define HT\_ADDR\_PROTOBOARD 0x02

HiTechnic Prototype board I2C address

6.2.2.612 #define HT\_ADDR\_SUPERPRO 0x10

HiTechnic SuperPro board I2C address

6.2.2.613 #define HT\_CH1\_A 0

Use IRReceiver channel 1 output A

6.2.2.614 #define HT\_CH1\_B 1

Use IRReceiver channel 1 output B

6.2.2.615 #define HT\_CH2\_A 2

Use IRReceiver channel 2 output A

6.2.2.616 #define HT\_CH2\_B 3

Use IRReceiver channel 2 output B

6.2.2.617 #define HT\_CH3\_A 4

Use IRReceiver channel 3 output A

6.2.2.618 #define HT\_CH3\_B 5

Use IRReceiver channel 3 output B

6.2.2.619 #define HT\_CH4\_A 6

Use IRReceiver channel 4 output A

6.2.2.620 #define HT\_CH4\_B 7

Use IRReceiver channel 4 output B

6.2.2.621 #define HT\_CMD\_COLOR2\_50HZ 0x35

Set the Color2 sensor to 50Hz mode

6.2.2.622 #define HT\_CMD\_COLOR2\_60HZ 0x36

Set the Color2 sensor to 60Hz mode

6.2.2.623 #define HT\_CMD\_COLOR2\_ACTIVE 0x00

Set the Color2 sensor to active mode

6.2.2.624 #define HT\_CMD\_COLOR2\_BLCAL 0x42

Set the Color2 sensor to black level calibration mode

6.2.2.625 #define HT\_CMD\_COLOR2\_FAR 0x46

Set the Color2 sensor to far mode

6.2.2.626 #define HT\_CMD\_COLOR2\_LED\_HI 0x48

Set the Color2 sensor to LED high mode

6.2.2.627 #define HT\_CMD\_COLOR2\_LED\_LOW 0x4C

Set the Color2 sensor to LED low mode

6.2.2.628 #define HT\_CMD\_COLOR2\_NEAR 0x4E

Set the Color2 sensor to near mode

6.2.2.629 #define HT\_CMD\_COLOR2\_PASSIVE 0x01

Set the Color2 sensor to passive mode

6.2.2.630 #define HT\_CMD\_COLOR2\_RAW 0x03

Set the Color2 sensor to raw mode

6.2.2.631 #define HT\_CMD\_COLOR2\_WBCAL 0x43

Set the Color2 sensor to white level calibration mode

6.2.2.632 #define HTANGLE\_MODE\_CALIBRATE 0x43

Resets 0 degree position to current shaft angle

6.2.2.633 #define HTANGLE\_MODE\_NORMAL 0x00

Normal angle measurement mode

6.2.2.634 #define HTANGLE\_MODE\_RESET 0x52

Resets the accumulated angle

6.2.2.635 #define HTANGLE\_REG\_ACDIR 0x49

Angle 16 bit revolutions per minute, low byte register

6.2.2.636 #define HTANGLE\_REG\_DC01 0x43

Angle current angle (1 degree adder) register

6.2.2.637 #define HTANGLE\_REG\_DC02 0x44

Angle 32 bit accumulated angle, high byte register

6.2.2.638 #define HTANGLE\_REG\_DC03 0x45

Angle 32 bit accumulated angle, mid byte register

6.2.2.639 #define HTANGLE\_REG\_DC04 0x46

Angle 32 bit accumulated angle, mid byte register

6.2.2.640 #define HTANGLE\_REG\_DC05 0x47

Angle 32 bit accumulated angle, low byte register

6.2.2.641 #define HTANGLE\_REG\_DCAVG 0x48

Angle 16 bit revolutions per minute, high byte register

6.2.2.642 #define HTANGLE\_REG\_DCDIR 0x42

Angle current angle (2 degree increments) register

6.2.2.643 #define HTANGLE\_REG\_MODE 0x41

Angle mode register

6.2.2.644 #define HTBAR\_REG\_CALIBRATION 0x46

Barometric sensor calibration register (2 bytes msb/lsb)

6.2.2.645 #define HTBAR\_REG\_COMMAND 0x40

Barometric sensor command register

6.2.2.646 #define HTBAR\_REG\_PRESSURE 0x44

Barometric sensor pressure register (2 bytes msb/lsb)

6.2.2.647 #define HTBAR\_REG\_TEMPERATURE 0x42

Barometric sensor temperature register (2 bytes msb/lsb)

6.2.2.648 #define HTIR2\_MODE\_1200 0

Set IRSeeker2 to 1200 mode

6.2.2.649 #define HTIR2\_MODE\_600 1

Set IRSeeker2 to 600 mode

6.2.2.650 #define HTIR2\_REG\_AC01 0x4A

IRSeeker2 AC 01 register

6.2.2.651 #define HTIR2\_REG\_AC02 0x4B

IRSeeker2 AC 02 register

6.2.2.652 #define HTIR2\_REG\_AC03 0x4C

IRSeeker2 AC 03 register

6.2.2.653 #define HTIR2\_REG\_AC04 0x4D

IRSeeker2 AC 04 register

6.2.2.654 #define HTIR2\_REG\_AC05 0x4E

IRSeeker2 AC 05 register

6.2.2.655 #define HTIR2\_REG\_ACDIR 0x49

IRSeeker2 AC direction register

6.2.2.656 #define HTIR2\_REG\_DC01 0x43

IRSeeker2 DC 01 register

6.2.2.657 #define HTIR2\_REG\_DC02 0x44

IRSeeker2 DC 02 register

6.2.2.658 #define HTIR2\_REG\_DC03 0x45

IRSeeker2 DC 03 register

6.2.2.659 #define HTIR2\_REG\_DC04 0x46

IRSeeker2 DC 04 register

6.2.2.660 #define HTIR2\_REG\_DC05 0x47

IRSeeker2 DC 05 register

6.2.2.661 #define HTIR2\_REG\_DCAVG 0x48

IRSeeker2 DC average register

6.2.2.662 #define HTIR2\_REG\_DCDIR 0x42

IRSeeker2 DC direction register

6.2.2.663 #define HTIR2\_REG\_MODE 0x41

IRSeeker2 mode register

6.2.2.664 #define HTPIR\_REG\_DEADBAND 0x41  
PIR sensor deadband register

6.2.2.665 #define HTPIR\_REG\_READING 0x42  
PIR sensor value register (signed byte)

6.2.2.666 #define HTPROTO\_A0 0x42  
Read Prototype board analog input 0

6.2.2.667 #define HTPROTO\_A1 0x44  
Read Prototype board analog input 1

6.2.2.668 #define HTPROTO\_A2 0x46  
Read Prototype board analog input 2

6.2.2.669 #define HTPROTO\_A3 0x48  
Read Prototype board analog input 3

6.2.2.670 #define HTPROTO\_A4 0x4A  
Read Prototype board analog input 4

6.2.2.671 #define HTPROTO\_REG\_A0 0x42  
Prototype board analog 0 register (2 bytes msb/lsb)

6.2.2.672 #define HTPROTO\_REG\_A1 0x44  
Prototype board analog 1 register (2 bytes msb/lsb)

6.2.2.673 #define HTPROTO\_REG\_A2 0x46  
Prototype board analog 2 register (2 bytes msb/lsb)

6.2.2.674 #define HTPROTO\_REG\_A3 0x48  
Prototype board analog 3 register (2 bytes msb/lsb)

6.2.2.675 #define HTPROTO\_REG\_A4 0x4A  
Prototype board analog 4 register (2 bytes msb/lsb)

6.2.2.676 #define HTPROTO\_REG\_DCTRL 0x4E  
Prototype board digital pin control register (6 bits)

6.2.2.677 #define HTPROTO\_REG\_DIN 0x4C  
Prototype board digital pin input register (6 bits)

6.2.2.678 #define HTPROTO\_REG\_DOUT 0x4D

Prototype board digital pin output register (6 bits)

6.2.2.679 #define HTPROTO\_REG\_SRATE 0x4F

Prototype board sample rate register

6.2.2.680 #define HTSPRO\_A0 0x42

Read SuperPro analog input 0

6.2.2.681 #define HTSPRO\_A1 0x44

Read SuperPro analog input 1

6.2.2.682 #define HTSPRO\_A2 0x46

Read SuperPro analog input 2

6.2.2.683 #define HTSPRO\_A3 0x48

Read SuperPro analog input 3

6.2.2.684 #define HTSPRO\_DAC0 0x52

Set SuperPro analog output 0 configuration

6.2.2.685 #define HTSPRO\_DAC1 0x57

Set SuperPro analog output 1 configuration

6.2.2.686 #define HTSPRO\_REG\_A0 0x42

SuperPro analog 0 register (10 bits)

6.2.2.687 #define HTSPRO\_REG\_A1 0x44

SuperPro analog 1 register (10 bits)

6.2.2.688 #define HTSPRO\_REG\_A2 0x46

SuperPro analog 2 register (10 bits)

6.2.2.689 #define HTSPRO\_REG\_A3 0x48

SuperPro analog 3 register (10 bits)

6.2.2.690 #define HTSPRO\_REG\_CTRL 0x40

SuperPro program control register

6.2.2.691 #define HTSPRO\_REG\_DAC0\_FREQ 0x53

SuperPro analog output 0 frequency register (2 bytes msb/lsb)

6.2.2.692 #define HTSPRO\_REG\_DAC0\_MODE 0x52

SuperPro analog output 0 mode register

6.2.2.693 #define HTSPRO\_REG\_DAC0\_VOLTAGE 0x55

SuperPro analog output 0 voltage register (10 bits)

6.2.2.694 #define HTSPRO\_REG\_DAC1\_FREQ 0x58

SuperPro analog output 1 frequency register (2 bytes msb/lsb)

6.2.2.695 #define HTSPRO\_REG\_DAC1\_MODE 0x57

SuperPro analog output 1 mode register

6.2.2.696 #define HTSPRO\_REG\_DAC1\_VOLTAGE 0x5A

SuperPro analog output 1 voltage register (10 bits)

6.2.2.697 #define HTSPRO\_REG\_DCTRL 0x4E

SuperPro digital pin control register (8 bits)

6.2.2.698 #define HTSPRO\_REG\_DIN 0x4C

SuperPro digital pin input register (8 bits)

6.2.2.699 #define HTSPRO\_REG\_DLADDRESS 0x60

SuperPro download address register (2 bytes msb/lsb)

6.2.2.700 #define HTSPRO\_REG\_DLCKSUM 0x6A

SuperPro download checksum register

6.2.2.701 #define HTSPRO\_REG\_DLCONTROL 0x6B

SuperPro download control register

6.2.2.702 #define HTSPRO\_REG\_DLDATA 0x62

SuperPro download data register (8 bytes)

6.2.2.703 #define HTSPRO\_REG\_DOUT 0x4D

SuperPro digital pin output register (8 bits)

6.2.2.704 #define HTSPRO\_REG\_LED 0x51

SuperPro LED control register

6.2.2.705 #define HTSPRO\_REG\_MEMORY\_20 0x80

SuperPro memory address 0x20 register (4 bytes msb/lsb)

6.2.2.706 #define HTSPRO\_REG\_MEMORY\_21 0x84

SuperPro memory address 0x21 register (4 bytes msb/lsb)

6.2.2.707 #define HTSPRO\_REG\_MEMORY\_22 0x88

SuperPro memory address 0x22 register (4 bytes msb/lsb)

6.2.2.708 #define HTSPRO\_REG\_MEMORY\_23 0x8C

SuperPro memory address 0x23 register (4 bytes msb/lsb)

6.2.2.709 #define HTSPRO\_REG\_MEMORY\_24 0x90

SuperPro memory address 0x24 register (4 bytes msb/lsb)

6.2.2.710 #define HTSPRO\_REG\_MEMORY\_25 0x94

SuperPro memory address 0x25 register (4 bytes msb/lsb)

6.2.2.711 #define HTSPRO\_REG\_MEMORY\_26 0x98

SuperPro memory address 0x26 register (4 bytes msb/lsb)

6.2.2.712 #define HTSPRO\_REG\_MEMORY\_27 0x9C

SuperPro memory address 0x27 register (4 bytes msb/lsb)

6.2.2.713 #define HTSPRO\_REG\_MEMORY\_28 0xA0

SuperPro memory address 0x28 register (4 bytes msb/lsb)

6.2.2.714 #define HTSPRO\_REG\_MEMORY\_29 0xA4

SuperPro memory address 0x29 register (4 bytes msb/lsb)

6.2.2.715 #define HTSPRO\_REG\_MEMORY\_2A 0xA8

SuperPro memory address 0x2A register (4 bytes msb/lsb)

6.2.2.716 #define HTSPRO\_REG\_MEMORY\_2B 0xAC

SuperPro memory address 0x2B register (4 bytes msb/lsb)

6.2.2.717 #define HTSPRO\_REG\_MEMORY\_2C 0xB0

SuperPro memory address 0x2C register (4 bytes msb/lsb)

6.2.2.718 #define HTSPRO\_REG\_MEMORY\_2D 0xB4

SuperPro memory address 0x2D register (4 bytes msb/lsb)

6.2.2.719 #define HTSPRO\_REG\_MEMORY\_2E 0xB8

SuperPro memory address 0x2E register (4 bytes msb/lsb)

6.2.2.720 #define HTSPRO\_REG\_MEMORY\_2F 0xBC

SuperPro memory address 0x2F register (4 bytes msb/lsb)

6.2.2.721 #define HTSPRO\_REG\_MEMORY\_30 0xC0

SuperPro memory address 0x30 register (4 bytes msb/lsb)

6.2.2.722 #define HTSPRO\_REG\_MEMORY\_31 0xC4

SuperPro memory address 0x31 register (4 bytes msb/lsb)

6.2.2.723 #define HTSPRO\_REG\_MEMORY\_32 0xC8

SuperPro memory address 0x32 register (4 bytes msb/lsb)

6.2.2.724 #define HTSPRO\_REG\_MEMORY\_33 0xCC

SuperPro memory address 0x33 register (4 bytes msb/lsb)

6.2.2.725 #define HTSPRO\_REG\_MEMORY\_34 0xD0

SuperPro memory address 0x34 register (4 bytes msb/lsb)

6.2.2.726 #define HTSPRO\_REG\_MEMORY\_35 0xD4

SuperPro memory address 0x35 register (4 bytes msb/lsb)

6.2.2.727 #define HTSPRO\_REG\_MEMORY\_36 0xD8

SuperPro memory address 0x36 register (4 bytes msb/lsb)

6.2.2.728 #define HTSPRO\_REG\_MEMORY\_37 0xDC

SuperPro memory address 0x37 register (4 bytes msb/lsb)

6.2.2.729 #define HTSPRO\_REG\_MEMORY\_38 0xE0

SuperPro memory address 0x38 register (4 bytes msb/lsb)

6.2.2.730 #define HTSPRO\_REG\_MEMORY\_39 0xE4

SuperPro memory address 0x39 register (4 bytes msb/lsb)

6.2.2.731 #define HTSPRO\_REG\_MEMORY\_3A 0xE8

SuperPro memory address 0x3A register (4 bytes msb/lsb)

6.2.2.732 #define HTSPRO\_REG\_MEMORY\_3B 0xEC

SuperPro memory address 0x3B register (4 bytes msb/lsb)

6.2.2.733 #define HTSPRO\_REG\_MEMORY\_3C 0xF0

SuperPro memory address 0x3C register (4 bytes msb/lsb)

6.2.2.734 #define HTSPRO\_REG\_MEMORY\_3D 0xF4

SuperPro memory address 0x3D register (4 bytes msb/lsb)

6.2.2.735 #define HTSPRO\_REG\_MEMORY\_3E 0xF8

SuperPro memory address 0x3E register (4 bytes msb/lsb)

6.2.2.736 #define HTSPRO\_REG\_MEMORY\_3F 0xFC

SuperPro memory address 0x3F register (4 bytes msb/lsb)

6.2.2.737 #define HTSPRO\_REG\_STROBE 0x50

SuperPro strobe control register

6.2.2.738 #define I2C\_ADDR\_DEFAULT 0x02

Standard NXT I2C device address

6.2.2.739 #define I2C\_OPTION\_FAST 0x08

Fast I2C speed

6.2.2.740 #define I2C\_OPTION\_NORESTART 0x04

Use no restart on I2C read

6.2.2.741 #define I2C\_OPTION\_STANDARD 0x00

Standard I2C speed

6.2.2.742 #define I2C\_REG\_CMD 0x41

Standard NXT I2C device command register

6.2.2.743 #define I2C\_REG\_DEVICE\_ID 0x10

Standard NXT I2C device ID register

6.2.2.744 #define I2C\_REG\_VENDOR\_ID 0x08

Standard NXT I2C vendor ID register

6.2.2.745 #define I2C\_REG\_VERSION 0x00

Standard NXT I2C version register

6.2.2.746 #define IN\_1 0x00

Input port 1

6.2.2.747 #define IN\_2 0x01

Input port 2

6.2.2.748 #define IN\_3 0x02

Input port 3

6.2.2.749 #define IN\_4 0x03

Input port 4

6.2.2.750 #define IN\_MODE\_ANGLESTEP 0xE0

RCX rotation sensor (16 ticks per revolution)

6.2.2.751 #define IN\_MODE\_BOOLEAN 0x20

Boolean value (0 or 1)

6.2.2.752 #define IN\_MODE\_CELSIUS 0xA0

RCX temperature sensor value in degrees celcius

6.2.2.753 #define IN\_MODE\_FAHRENHEIT 0xC0

RCX temperature sensor value in degrees fahrenheit

6.2.2.754 #define IN\_MODE\_MODEMASK 0xE0

Mask for the mode without any slope value

6.2.2.755 #define IN\_MODE\_PCTFULLSCALE 0x80

Scaled value from 0 to 100

6.2.2.756 #define IN\_MODE\_PERIODCOUNTER 0x60

Counts the number of boolean periods

6.2.2.757 #define IN\_MODE\_RAW 0x00

Raw value from 0 to 1023

6.2.2.758 #define IN\_MODE\_SLOPEMASK 0x1F

Mask for slope parameter added to mode

6.2.2.759 #define IN\_MODE\_TRANSITIONCNT 0x40

Counts the number of boolean transitions

6.2.2.760 #define IN\_TYPE\_ANGLE 0x04

RCX rotation sensor

6.2.2.761 #define IN\_TYPE\_COLORBLUE 0x10

NXT 2.0 color sensor with blue light

6.2.2.762 #define IN\_TYPE\_COLOREXIT 0x12

NXT 2.0 color sensor internal state

6.2.2.763 #define IN\_TYPE\_COLORFULL 0x0D

NXT 2.0 color sensor in full color mode

6.2.2.764 #define IN\_TYPE\_COLORGREEN 0x0F

NXT 2.0 color sensor with green light

6.2.2.765 #define IN\_TYPE\_COLORNONE 0x11

NXT 2.0 color sensor with no light

6.2.2.766 #define IN\_TYPE\_COLORRED 0x0E

NXT 2.0 color sensor with red light

6.2.2.767 #define IN\_TYPE\_CUSTOM 0x09

NXT custom sensor

6.2.2.768 #define IN\_TYPE\_HISPEED 0x0C

NXT Hi-speed port (only S4)

6.2.2.769 #define IN\_TYPE\_LIGHT\_ACTIVE 0x05

NXT light sensor with light

6.2.2.770 #define IN\_TYPE\_LIGHT\_INACTIVE 0x06

NXT light sensor without light

6.2.2.771 #define IN\_TYPE\_LOWSPEED 0x0A

NXT I2C digital sensor

6.2.2.772 #define IN\_TYPE\_LOWSPEED\_9V 0x0B

NXT I2C digital sensor with 9V power

6.2.2.773 #define IN\_TYPE\_NO\_SENSOR 0x00

No sensor configured

6.2.2.774 #define IN\_TYPE\_REFLECTION 0x03

RCX light sensor

6.2.2.775 #define IN\_TYPE\_SOUND\_DB 0x07

NXT sound sensor with dB scaling

6.2.2.776 #define IN\_TYPE\_SOUND\_DBIA 0x08

NXT sound sensor with dBA scaling

6.2.2.777 #define IN\_TYPE\_SWITCH 0x01

NXT or RCX touch sensor

6.2.2.778 #define IN\_TYPE\_TEMPERATURE 0x02

RCX temperature sensor

6.2.2.779 #define INPUT\_BLACKCOLOR 1

The color value is black

6.2.2.780 #define INPUT\_BLANK 3

Access the blank value from color sensor value arrays

6.2.2.781 #define INPUT\_BLUE 2

Access the blue value from color sensor value arrays

6.2.2.782 #define INPUT\_BLUECOLOR 2

The color value is blue

6.2.2.783 #define INPUT\_CAL\_POINT\_0 0

Calibration point 0

6.2.2.784 #define INPUT\_CAL\_POINT\_1 1

Calibration point 1

6.2.2.785 #define INPUT\_CAL\_POINT\_2 2

Calibration point 2

6.2.2.786 #define INPUT\_CUSTOM9V 0x01

Custom sensor 9V

6.2.2.787 #define INPUT\_CUSTOMACTIVE 0x02

Custom sensor active

6.2.2.788 #define INPUT\_CUSTOMINACTIVE 0x00

Custom sensor inactive

6.2.2.789 #define INPUT\_DIGI0 0x01

Digital pin 0

6.2.2.790 #define INPUT\_DIGI1 0x02

Digital pin 1

6.2.2.791 #define INPUT\_GREEN 1

Access the green value from color sensor value arrays

6.2.2.792 #define INPUT\_GREENCOLOR 3

The color value is green

6.2.2.793 #define INPUT\_INVALID\_DATA 0x01

Invalid data flag

6.2.2.794 #define INPUT\_NO\_OF\_COLORS 4

The number of entries in the color sensor value arrays

6.2.2.795 #define INPUT\_NO\_OF\_POINTS 3

The number of calibration points

6.2.2.796 #define INPUT\_PINCMD\_CLEAR 0x02

Clear digital pin(s)

6.2.2.797 #define INPUT\_PINCMD\_DIR 0x00

Set digital pin(s) direction

6.2.2.798 #define INPUT\_PINCMD\_MASK 0x03

Mask for the two bits used by pin function commands

6.2.2.799 #define INPUT\_PINCMD\_READ 0x03

Read digital pin(s)

6.2.2.800 #define INPUT\_PINCMD\_SET 0x01

Set digital pin(s)

6.2.2.801 #define INPUT\_PINCMD\_WAIT( *\_usec* ) ((*\_usec*)<<2)

A wait value in microseconds that can be added after one of the above commands by ORing with the command

6.2.2.802 #define INPUT\_PINDIR\_INPUT 0x04

Use with the direction command to set direction to output. OR this with the pin value.

6.2.2.803 #define INPUT\_PINDIR\_OUTPUT 0x00

Use with the direction command to set direction to input. OR this with the pin value.

6.2.2.804 #define INPUT\_RED 0

Access the red value from color sensor value arrays

6.2.2.805 #define INPUT\_REDCOLOR 5

The color value is red

6.2.2.806 #define INPUT\_RESETCAL 0x80

Unused calibration state constant

6.2.2.807 #define INPUT\_RUNNINGCAL 0x20

Unused calibration state constant

6.2.2.808 #define INPUT\_SENSORCAL 0x01

The state returned while the color sensor is calibrating

6.2.2.809 #define INPUT\_SENSOROFF 0x02

The state returned once calibration has completed

6.2.2.810 #define INPUT\_STARTCAL 0x40

Unused calibration state constant

6.2.2.811 #define INPUT\_WHITECOLOR 6

The color value is white

6.2.2.812 #define INPUT\_YELLOWCOLOR 4

The color value is yellow

6.2.2.813 #define InputModeField 1

Input mode field. Contains one of the sensor mode constants. Read/write.

6.2.2.814 #define InputModuleID 0x00030001

The input module ID

6.2.2.815 #define InputModuleName "Input.mod"

The input module name.

6.2.2.816 #define InputOffsetADRaw( p ) (((p)\*20)+2)

Read the AD raw sensor value (2 bytes) uword

6.2.2.817 #define InputOffsetColorADRaw( p, nc ) (80+((p)\*84)+52+((nc)\*2))

Read AD raw color sensor values

6.2.2.818 #define InputOffsetColorBoolean( *p*, *nc* ) (80+((*p*)\*84)+76+((*nc*)\*2))

Read color sensor boolean values

6.2.2.819 #define InputOffsetColorCalibration( *p*, *np*, *nc* ) (80+((*p*)\*84)+0+((*np*)\*16)+((*nc*)\*4))

Read/write color calibration point values

6.2.2.820 #define InputOffsetColorCalibrationState( *p* ) (80+((*p*)\*84)+80)

Read color sensor calibration state

6.2.2.821 #define InputOffsetColorCallLimits( *p*, *np* ) (80+((*p*)\*84)+48+((*np*)\*2))

Read/write color calibration limits

6.2.2.822 #define InputOffsetColorSensorRaw( *p*, *nc* ) (80+((*p*)\*84)+60+((*nc*)\*2))

Read raw color sensor values

6.2.2.823 #define InputOffsetColorSensorValue( *p*, *nc* ) (80+((*p*)\*84)+68+((*nc*)\*2))

Read scaled color sensor values

6.2.2.824 #define InputOffsetCustomActiveStatus( *p* ) (((*p*)\*20)+15)

Read/write the active or inactive state of the custom sensor

6.2.2.825 #define InputOffsetCustomPctFullScale( *p* ) (((*p*)\*20)+14)

Read/write the Pct full scale of the custom sensor

6.2.2.826 #define InputOffsetCustomZeroOffset( *p* ) (((*p*)\*20)+0)

Read/write the zero offset of a custom sensor (2 bytes) uword

6.2.2.827 #define InputOffsetDigiPinsDir( *p* ) (((*p*)\*20)+11)

Read/write the direction of the Digital pins (1 is output, 0 is input)

6.2.2.828 #define InputOffsetDigiPinsIn( *p* ) (((*p*)\*20)+12)

Read/write the status of the digital pins

6.2.2.829 #define InputOffsetDigiPinsOut( *p* ) (((*p*)\*20)+13)

Read/write the output level of the digital pins

6.2.2.830 #define InputOffsetInvalidData( *p* ) (((*p*)\*20)+16)

Indicates whether data is invalid (1) or valid (0)

6.2.2.831 #define InputOffsetSensorBoolean( *p* ) (((*p*)\*20)+10)

Read the sensor boolean value

6.2.2.832 #define InputOffsetSensorMode( *p* ) (((*p*)\*20)+9)

Read/write the sensor mode

6.2.2.833 #define InputOffsetSensorRaw( *p* ) (((*p*)\*20)+4)

Read the raw sensor value (2 bytes) *uword*

6.2.2.834 #define InputOffsetSensorType( *p* ) (((*p*)\*20)+8)

Read/write the sensor type

6.2.2.835 #define InputOffsetSensorValue( *p* ) (((*p*)\*20)+6)

Read/write the scaled sensor value (2 bytes) *sword*

6.2.2.836 #define InputPinFunction 77

Execute the Input module's pin function

6.2.2.837 #define INT\_MAX 32767

The maximum value of the int type

6.2.2.838 #define INT\_MIN -32768

The minimum value of the int type

6.2.2.839 #define INTF\_BTOFF 13

Turn off the bluetooth radio

6.2.2.840 #define INTF\_BTON 12

Turn on the bluetooth radio

6.2.2.841 #define INTF\_CONNECT 3

Connect to one of the known devices

6.2.2.842 #define INTF\_CONNECTBYNAME 18

Connect to a bluetooth device by name

6.2.2.843 #define INTF\_CONNECTREQ 17

Connection request from another device

6.2.2.844 #define INTF\_DISCONNECT 4

Disconnect from one of the connected devices

6.2.2.845 #define INTF\_DISCONNECTALL 5

Disconnect all devices

6.2.2.846 #define INTF\_EXTREAD 15

External read request

6.2.2.847 #define INTF\_FACTORYRESET 11

Reset bluetooth settings to factory values

6.2.2.848 #define INTF\_OPENSTREAM 9

Open a bluetooth stream

6.2.2.849 #define INTF\_PINREQ 16

Bluetooth PIN request

6.2.2.850 #define INTF\_REMOVEDEVICE 6

Remove a device from the known devices table

6.2.2.851 #define INTF\_SEARCH 1

Search for bluetooth devices

6.2.2.852 #define INTF\_SENDDATA 10

Send data over a bluetooth connection

6.2.2.853 #define INTF\_SENDFILE 0

Send a file via bluetooth to another device

6.2.2.854 #define INTF\_SETBTNAME 14

Set the bluetooth name

6.2.2.855 #define INTF\_SETCMDMODE 8

Set bluetooth into command mode

6.2.2.856 #define INTF\_STOPSEARCH 2

Stop searching for bluetooth devices

6.2.2.857 #define INTF\_VISIBILITY 7

Set the bluetooth visibility on or off

6.2.2.858 #define InvalidDataField 5

Invalid data field. Contains a boolean value indicating whether the sensor data is valid or not. Read/write.

6.2.2.859 #define IOCTRL\_BOOT 0xA55A

Reboot the NXT into SAMBA mode

6.2.2.860 #define IOCTRL\_POWERDOWN 0x5A00

Power down the NXT

6.2.2.861 #define IOCtrlModuleID 0x00060001

The IOCtrl module ID

6.2.2.862 #define IOCtrlModuleName "IOCtrl.mod"

The IOCtrl module name

6.2.2.863 #define IOCtrlOffsetPowerOn 0

Offset to power on field

6.2.2.864 #define IOMapRead 32

Read data from one of the firmware module's IOMap structures using the module's name

6.2.2.865 #define IOMapReadByID 78

Read data from one of the firmware module's IOMap structures using the module's ID

6.2.2.866 #define IOMapWrite 33

Write data to one of the firmware module's IOMap structures using the module's name

6.2.2.867 #define IOMapWriteByID 79

Write data to one of the firmware module's IOMap structures using the module's ID

6.2.2.868 #define JOY\_BTN\_01 0x00000001

Joystick button 1

6.2.2.869 #define JOY\_BTN\_02 0x00000002

Joystick button 2

6.2.2.870 #define JOY\_BTN\_03 0x00000004

Joystick button 3

6.2.2.871 #define JOY\_BTN\_04 0x00000008

Joystick button 4

6.2.2.872 #define JOY\_BTN\_05 0x00000010

Joystick button 5

6.2.2.873 #define JOY\_BTN\_06 0x00000020

Joystick button 6

6.2.2.874 #define JOY\_BTN\_07 0x00000040

Joystick button 7

6.2.2.875 #define JOY\_BTN\_08 0x00000080

Joystick button 8

6.2.2.876 #define JOY\_BTN\_09 0x00000100

Joystick button 9

6.2.2.877 #define JOY\_BTN\_10 0x00000200

Joystick button 10

6.2.2.878 #define JOY\_BTN\_11 0x00000400

Joystick button 11

6.2.2.879 #define JOY\_BTN\_12 0x00000800

Joystick button 12

6.2.2.880 #define JOY\_BTN\_13 0x00001000

Joystick button 13

6.2.2.881 #define JOY\_BTN\_14 0x00002000

Joystick button 14

6.2.2.882 #define JOY\_BTN\_15 0x00004000

Joystick button 15

6.2.2.883 #define JOY\_BTN\_16 0x00008000

Joystick button 16

6.2.2.884 #define JOY\_BTN\_17 0x00010000

Joystick button 17

6.2.2.885 #define JOY\_BTN\_18 0x00020000

Joystick button 18

6.2.2.886 #define JOY\_BTN\_19 0x00040000

Joystick button 19

6.2.2.887 #define JOY\_BTN\_20 0x00080000

Joystick button 20

6.2.2.888 #define JOY\_BTN\_21 0x00100000

Joystick button 21

6.2.2.889 #define JOY\_BTN\_22 0x00200000

Joystick button 22

6.2.2.890 #define JOY\_BTN\_23 0x00400000

Joystick button 23

6.2.2.891 #define JOY\_BTN\_24 0x00800000

Joystick button 24

6.2.2.892 #define JOY\_BTN\_25 0x01000000

Joystick button 25

6.2.2.893 #define JOY\_BTN\_26 0x02000000

Joystick button 26

6.2.2.894 #define JOY\_BTN\_27 0x04000000

Joystick button 27

6.2.2.895 #define JOY\_BTN\_28 0x08000000

Joystick button 28

6.2.2.896 #define JOY\_BTN\_29 0x10000000

Joystick button 29

6.2.2.897 #define JOY\_BTN\_30 0x20000000

Joystick button 30

6.2.2.898 #define JOY\_BTN\_31 0x40000000

Joystick button 31

6.2.2.899 #define JOY\_BTN\_32 0x80000000

Joystick button 32

6.2.2.900 #define JOY\_POV\_BACKWARD 18000

Joystick POV backward

6.2.2.901 #define JOY\_POV\_BOTLEFT 22500

Joystick POV bottom left

6.2.2.902 #define JOY\_POV\_BOTRIGHT 13500

Joystick POV bottom right

6.2.2.903 #define JOY\_POV\_CENTERED 65535

Joystick POV centered

6.2.2.904 #define JOY\_POV\_FORWARD 0

Joystick POV forward

6.2.2.905 #define JOY\_POV\_LEFT 27000

Joystick POV left

6.2.2.906 #define JOY\_POV\_RIGHT 9000

Joystick POV right

6.2.2.907 #define JOY\_POV\_TOPLEFT 31500

Joystick POV top left

6.2.2.908 #define JOY\_POV\_TOPRIGHT 4500

Joystick POV top right

6.2.2.909 #define KeepAlive 31

Reset the NXT sleep timer

6.2.2.910 #define LCD\_LINE1 56

The 1st line of the LCD screen

6.2.2.911 #define LCD\_LINE2 48

The 2nd line of the LCD screen

6.2.2.912 #define LCD\_LINE3 40

The 3rd line of the LCD screen

6.2.2.913 #define LCD\_LINE4 32

The 4th line of the LCD screen

6.2.2.914 #define LCD\_LINE5 24

The 5th line of the LCD screen

6.2.2.915 #define LCD\_LINE6 16

The 6th line of the LCD screen

6.2.2.916 #define LCD\_LINE7 8

The 7th line of the LCD screen

6.2.2.917 #define LCD\_LINE8 0

The 8th line of the LCD screen

6.2.2.918 #define LDR\_APPENDNOTPOSSIBLE 0x8D00

Only datafiles can be appended to.

6.2.2.919 #define LDR\_BTBUSY 0x9400

The bluetooth system is busy.

6.2.2.920 #define LDR\_BTCONNECTFAIL 0x9500

Bluetooth connection attempt failed.

6.2.2.921 #define LDR\_BTTIMEOUT 0x9600

A timeout in the bluetooth system has occurred.

6.2.2.922 #define LDR\_CMD\_BOOTCMD 0x97

Reboot the NXT into SAMBA mode

6.2.2.923 #define LDR\_CMD\_BTFACTORYRESET 0xA4

Reset bluetooth configuration to factory defaults

6.2.2.924 #define LDR\_CMD\_BTGETADR 0x9A

Get the NXT's bluetooth brick address

6.2.2.925 #define LDR\_CMD\_CLOSE 0x84

Close a file handle

6.2.2.926 #define LDR\_CMD\_CLOSEMODHANDLE 0x92

Close a module handle

6.2.2.927 #define LDR\_CMD\_CROPDATAFILE 0x8D

Crop a data file to its used space

6.2.2.928 #define LDR\_CMD\_DELETE 0x85

Delete a file

6.2.2.929 #define LDR\_CMD\_DELETEUSERFLASH 0xA0

Delete all files from user flash memory

6.2.2.930 #define LDR\_CMD\_DEVICEINFO 0x9B

Read device information

6.2.2.931 #define LDR\_CMD\_FINDFIRST 0x86

Find the first file matching the specified pattern

6.2.2.932 #define LDR\_CMD\_FINDFIRSTMODULE 0x90

Find the first module matching the specified pattern

6.2.2.933 #define LDR\_CMD\_FINDNEXT 0x87

Find the next file matching the specified pattern

6.2.2.934 #define LDR\_CMD\_FINDNEXTMODULE 0x91

Find the next module matching the specified pattern

6.2.2.935 #define LDR\_CMD\_IOMAPREAD 0x94

Read data from a module IOMAP

6.2.2.936 #define LDR\_CMD\_IOMAPWRITE 0x95

Write data to a module IOMAP

6.2.2.937 #define LDR\_CMD\_OPENAPPENDDATA 0x8C

Open a data file for appending

6.2.2.938 #define LDR\_CMD\_OPENREAD 0x80

Open a file for reading

6.2.2.939 #define LDR\_CMD\_OPENREADLINEAR 0x8A

Open a linear file for reading

6.2.2.940 #define LDR\_CMD\_OPENWRITE 0x81

Open a file for writing

6.2.2.941 #define LDR\_CMD\_OPENWRITEDATA 0x8B

Open a data file for writing

6.2.2.942 #define LDR\_CMD\_OPENWRITELINEAR 0x89

Open a linear file for writing

6.2.2.943 #define LDR\_CMD\_POLLCMD 0xA2

Poll command

6.2.2.944 #define LDR\_CMD\_POLLCMDLEN 0xA1

Read poll command length

6.2.2.945 #define LDR\_CMD\_READ 0x82

Read from a file

6.2.2.946 #define LDR\_CMD\_RENAMEFILE 0xA3

Rename a file

6.2.2.947 #define LDR\_CMD\_RESIZEDATAFILE 0xD0

Resize a data file

6.2.2.948 #define LDR\_CMD\_SEEKFROMCURRENT 0xD2

Seek from the current position

6.2.2.949 #define LDR\_CMD\_SEEKFROMEND 0xD3

Seek from the end of the file

6.2.2.950 #define LDR\_CMD\_SEEKFROMSTART 0xD1

Seek from the start of the file

6.2.2.951 #define LDR\_CMD\_SETBRICKNAME 0x98

Set the NXT's brick name

6.2.2.952 #define LDR\_CMD\_VERSIONS 0x88

Read firmware version information

6.2.2.953 #define LDR\_CMD\_WRITE 0x83

Write to a file

6.2.2.954 #define LDR\_ENDOFFILE 0x8500

The end of the file has been reached.

6.2.2.955 #define LDR\_EOFEXPECTED 0x8400

EOF expected.

6.2.2.956 #define LDR\_FILEEXISTS 0x8F00

A file with the same name already exists.

6.2.2.957 #define LDR\_FILEISBUSY 0x8B00

The file is already being used.

6.2.2.958 #define LDR\_FILEISFULL 0x8E00

The allocated file size has been filled.

6.2.2.959 #define LDR\_FILENOFOUND 0x8700

No files matched the search criteria.

6.2.2.960 #define LDR\_FILETX\_CLOSEERROR 0x9B00

Error transmitting file: attempt to close file failed.

6.2.2.961 #define LDR\_FILETX\_DSTEXISTS 0x9800

Error transmitting file: destination file exists.

6.2.2.962 #define LDR\_FILETX\_SRCMISSING 0x9900

Error transmitting file: source file is missing.

6.2.2.963 #define LDR\_FILETX\_STREAMERROR 0x9A00

Error transmitting file: a stream error occurred.

6.2.2.964 #define LDR\_FILETX\_TIMEOUT 0x9700

Error transmitting file: a timeout occurred.

6.2.2.965 #define LDR\_HANDLEALREADYCLOSED 0x8800

The file handle has already been closed.

6.2.2.966 #define LDR\_ILLEGALFILENAME 0x9200

Filename length to long or attempted open a system file (\*.rxe, \*.rtm, or \*.sys) for writing as a datafile.

6.2.2.967 #define LDR\_ILLEGALHANDLE 0x9300

Invalid file handle.

6.2.2.968 #define LDR\_INPROGRESS 0x0001

The function is executing but has not yet completed.

6.2.2.969 #define LDR\_INVALIDSEEK 0x9C00

Invalid file seek operation.

6.2.2.970 #define LDR\_MODULENOTFOUND 0x9000

No modules matched the specified search criteria.

6.2.2.971 #define LDR\_NOLINEARSPACE 0x8900

Not enough linear flash memory is available.

6.2.2.972 #define LDR\_NMOREFILES 0x8300

The maximum number of files has been reached.

6.2.2.973 #define LDR\_NMOREHANDLES 0x8100

All available file handles are in use.

6.2.2.974 #define LDR\_NOSPACE 0x8200

Not enough free flash memory for the specified file size.

6.2.2.975 #define LDR\_NOTLINEARFILE 0x8600

The specified file is not linear.

6.2.2.976 #define LDR\_NOWRITEBUFFERS 0x8C00

No more write buffers are available.

6.2.2.977 #define LDR\_OUTOFBOUNDARY 0x9100

Specified IOMap offset is outside the bounds of the IOMap.

6.2.2.978 #define LDR\_REQPIN 0x0002

A PIN exchange request is in progress.

6.2.2.979 #define LDR\_SUCCESS 0x0000

The function completed successfully.

6.2.2.980 #define LDR\_UNDEFINEDERROR 0x8A00

An undefined error has occurred.

6.2.2.981 #define LED\_BLUE 0x02

Turn on the blue onboard LED.

6.2.2.982 #define LED\_NONE 0x00

Turn off the onboard LEDs.

6.2.2.983 #define LED\_RED 0x01

Turn on the red onboard LED.

6.2.2.984 #define LEGO\_ADDR\_EMETER 0x04

The LEGO e-meter sensor's I2C address

6.2.2.985 #define LEGO\_ADDR\_TEMP 0x98

The LEGO temperature sensor's I2C address

6.2.2.986 #define LEGO\_ADDR\_US 0x02

The LEGO ultrasonic sensor's I2C address

6.2.2.987 #define ListFiles 47

List files that match the specified filename pattern

6.2.2.988 #define LoaderExecuteFunction 82

Execute one of the Loader module's internal functions

6.2.2.989 #define LoaderModuleID 0x00090001

The Loader module ID

6.2.2.990 #define LoaderModuleName "Loader.mod"

The Loader module name

6.2.2.991 #define LoaderOffsetFreeUserFlash 4

Offset to the amount of free user flash

6.2.2.992 #define LoaderOffsetPFunc 0

Offset to the Loader module function pointer

6.2.2.993 #define LONG\_MAX 2147483647

The maximum value of the long type

6.2.2.994 #define LONG\_MIN -2147483648

The minimum value of the long type

6.2.2.995 #define LOWSPEED\_CH\_NOT\_READY 1

Lowspeed port is not ready

6.2.2.996 #define LOWSPEED\_COMMUNICATING 3

Channel is actively communicating

6.2.2.997 #define LOWSPEED\_DATA\_RECEIVED 3

Lowspeed port is in data received mode

6.2.2.998 #define LOWSPEED\_DONE 5

Channel is done communicating

6.2.2.999 #define LOWSPEED\_ERROR 4

Channel is in an error state

6.2.2.1000 #define LOWSPEED\_IDLE 0

Channel is idle

6.2.2.1001 #define LOWSPEED\_INIT 1

Channel is being initialized

6.2.2.1002 #define LOWSPEED\_LOAD\_BUFFER 2

Channel buffer is loading

6.2.2.1003 #define LOWSPEED\_NO\_ERROR 0

Lowspeed port has no error

6.2.2.1004 #define LOWSPEED RECEIVING 2

Lowspeed port is in receiving mode

6.2.2.1005 #define LOWSPEED\_RX\_ERROR 3

Lowspeed port encountered an error while receiving data

6.2.2.1006 #define LOWSPEED\_TRANSMITTING 1

Lowspeed port is in transmitting mode

6.2.2.1007 #define LOWSPEED\_TX\_ERROR 2

Lowspeed port encountered an error while transmitting data

6.2.2.1008 #define LowSpeedModuleID 0x000B0001

The low speed module ID

6.2.2.1009 #define LowSpeedModuleName "Low Speed.mod"

The low speed module name

6.2.2.1010 #define LowSpeedOffsetChannelState( p ) ((p)+156)

R - Lowspeed channel state (1 byte)

6.2.2.1011 #define LowSpeedOffsetErrorType( p ) ((p)+160)

R - Lowspeed port error type (1 byte)

6.2.2.1012 #define LowSpeedOffsetInBufBuf( p ) (((p)\*19)+0)

RW - Input buffer data buffer field offset (16 bytes)

6.2.2.1013 #define LowSpeedOffsetInBufBytesToRx( p ) (((p)\*19)+18)

RW - Input buffer bytes to receive field offset (1 byte)

6.2.2.1014 #define LowSpeedOffsetInBufInPtr( *p* ) (((*p*)\*19)+16)

RW - Input buffer in pointer field offset (1 byte)

6.2.2.1015 #define LowSpeedOffsetInBufOutPtr( *p* ) (((*p*)\*19)+17)

RW - Input buffer out pointer field offset (1 byte)

6.2.2.1016 #define LowSpeedOffsetMode( *p* ) ((*p*)+152)

R - Lowspeed port mode (1 byte)

6.2.2.1017 #define LowSpeedOffsetNoRestartOnRead 166

RW - Lowspeed option for no restart on read (all channels) (NBC/NXC)

6.2.2.1018 #define LowSpeedOffsetOutBufBuf( *p* ) (((*p*)\*19)+76)

RW - Output buffer data buffer field offset (16 bytes)

6.2.2.1019 #define LowSpeedOffsetOutBufBytesToRx( *p* ) (((*p*)\*19)+94)

RW - Output buffer bytes to receive field offset (1 byte)

6.2.2.1020 #define LowSpeedOffsetOutBufInPtr( *p* ) (((*p*)\*19)+92)

RW - Output buffer in pointer field offset (1 byte)

6.2.2.1021 #define LowSpeedOffsetOutBufOutPtr( *p* ) (((*p*)\*19)+93)

RW - Output buffer out pointer field offset (1 byte)

6.2.2.1022 #define LowSpeedOffsetSpeed 165

R - Lowspeed speed (unused)

6.2.2.1023 #define LowSpeedOffsetState 164

R - Lowspeed state (all channels)

6.2.2.1024 #define LR\_COULD\_NOT\_SAVE 0x51

Bluetooth list result could not save

6.2.2.1025 #define LR\_ENTRY\_REMOVED 0x53

Bluetooth list result entry removed

6.2.2.1026 #define LR\_STORE\_IS\_FULL 0x52

Bluetooth list result store is full

6.2.2.1027 #define LR\_SUCCESS 0x50

Bluetooth list result success

6.2.2.1028 #define LR\_UNKNOWN\_ADDR 0x54

Bluetooth list result unknown address

6.2.2.1029 #define LSREAD\_NO\_RESTART\_1 0x01

No restart on read for channel 1

6.2.2.1030 #define LSREAD\_NO\_RESTART\_2 0x02

No restart on read for channel 2

6.2.2.1031 #define LSREAD\_NO\_RESTART\_3 0x04

No restart on read for channel 3

6.2.2.1032 #define LSREAD\_NO\_RESTART\_4 0x08

No restart on read for channel 4

6.2.2.1033 #define LSREAD\_NO\_RESTART\_MASK 0x10

No restart mask

6.2.2.1034 #define LSREAD\_RESTART\_ALL 0x00

Restart on read for all channels (default)

6.2.2.1035 #define LSREAD\_RESTART\_NONE 0x0F

No restart on read for all channels

6.2.2.1036 #define MAILBOX1 0

Mailbox number 1

6.2.2.1037 #define MAILBOX10 9

Mailbox number 10

6.2.2.1038 #define MAILBOX2 1

Mailbox number 2

6.2.2.1039 #define MAILBOX3 2

Mailbox number 3

6.2.2.1040 #define MAILBOX4 3

Mailbox number 4

6.2.2.1041 #define MAILBOX5 4

Mailbox number 5

6.2.2.1042 #define MAILBOX6 5

Mailbox number 6

6.2.2.1043 #define MAILBOX7 6

Mailbox number 7

6.2.2.1044 #define MAILBOX8 7

Mailbox number 8

6.2.2.1045 #define MAILBOX9 8

Mailbox number 9

6.2.2.1046 #define MAX\_BT\_MSG\_SIZE 60000

Max Bluetooth Message Size

6.2.2.1047 #define MaxAccelerationField 17

MaxAcceleration field.

Contains the current max acceleration value. Read/write. Set the maximum acceleration to be used during position regulation.

#### Warning

This option requires the enhanced NBC/NXC firmware version 1.31+

6.2.2.1048 #define MaxSpeedField 16

MaxSpeed field.

Contains the current max speed value. Read/write. Set the maximum speed to be used during position regulation.

#### Warning

This option requires the enhanced NBC/NXC firmware version 1.31+

6.2.2.1049 #define MemoryManager 96

Read memory manager information, optionally compacting the dataspace first

6.2.2.1050 #define MENUICON\_CENTER 1

Center icon

6.2.2.1051 #define MENUICON\_LEFT 0

Left icon

6.2.2.1052 #define MENUICON\_RIGHT 2

Right icon

6.2.2.1053 #define MENUICONS 3

The number of menu icons

6.2.2.1054 #define MENUTEXT 2

Center icon text

6.2.2.1055 #define MessageRead 27

Read a message from a mailbox

6.2.2.1056 #define MessageWrite 26

Write a message to a mailbox

6.2.2.1057 #define MI\_ADDR\_XG1300L 0x02

XG1300L I2C address

6.2.2.1058 #define MIN\_1 60000

1 minute

6.2.2.1059 #define MS\_1 1

1 millisecond

6.2.2.1060 #define MS\_10 10

10 milliseconds

6.2.2.1061 #define MS\_100 100

100 milliseconds

6.2.2.1062 #define MS\_150 150

150 milliseconds

6.2.2.1063 #define MS\_2 2

2 milliseconds

6.2.2.1064 #define MS\_20 20

20 milliseconds

6.2.2.1065 #define MS\_200 200

200 milliseconds

6.2.2.1066 #define MS\_250 250

250 milliseconds

6.2.2.1067 #define MS\_3 3

3 milliseconds

6.2.2.1068 #define MS\_30 30

30 milliseconds

6.2.2.1069 #define MS\_300 300

300 milliseconds

6.2.2.1070 #define MS\_350 350

350 milliseconds

6.2.2.1071 #define MS\_4 4

4 milliseconds

6.2.2.1072 #define MS\_40 40

40 milliseconds

6.2.2.1073 #define MS\_400 400

400 milliseconds

6.2.2.1074 #define MS\_450 450

450 milliseconds

6.2.2.1075 #define MS\_5 5

5 milliseconds

6.2.2.1076 #define MS\_50 50

50 milliseconds

6.2.2.1077 #define MS\_500 500

500 milliseconds

6.2.2.1078 #define MS\_6 6

6 milliseconds

6.2.2.1079 #define MS\_60 60

60 milliseconds

6.2.2.1080 #define MS\_600 600

600 milliseconds

6.2.2.1081 #define MS\_7 7

7 milliseconds

6.2.2.1082 #define MS\_70 70

70 milliseconds

6.2.2.1083 #define MS\_700 700

700 milliseconds

6.2.2.1084 #define MS\_8 8

8 milliseconds

6.2.2.1085 #define MS\_80 80

80 milliseconds

6.2.2.1086 #define MS\_800 800

800 milliseconds

6.2.2.1087 #define MS\_9 9

9 milliseconds

6.2.2.1088 #define MS\_90 90

90 milliseconds

6.2.2.1089 #define MS\_900 900

900 milliseconds

6.2.2.1090 #define MS\_ADDR\_ACCLNX 0x02

MindSensors ACCL-Nx I2C address

6.2.2.1091 #define MS\_ADDR\_CMPSNX 0x02

MindSensors CMPS-Nx I2C address

6.2.2.1092 #define MS\_ADDR\_DISTNX 0x02

MindSensors DIST-Nx I2C address

6.2.2.1093 #define MS\_ADDR\_IVSENS 0x12

MindSensors IVSens (NXTPowerMeter) I2C address

6.2.2.1094 #define MS\_ADDR\_LINELDR 0x02

MindSensors LineLdr I2C address

6.2.2.1095 #define MS\_ADDR\_MTRMUX 0xB4

MindSensors MTRMux I2C address

6.2.2.1096 #define MS\_ADDR\_NRLINK 0x02

MindSensors NRLink I2C address

6.2.2.1097 #define MS\_ADDR\_NUMERICPAD 0xB4

MindSensors NumericPad I2C address

6.2.2.1098 #define MS\_ADDR\_NXTCAM 0x02

MindSensors NXTCam I2C address

6.2.2.1099 #define MS\_ADDR\_NXTHID 0x04

MindSensors NXTHID I2C address

6.2.2.1100 #define MS\_ADDR\_NXTMMX 0x06

MindSensors NXTMMX I2C address

6.2.2.1101 #define MS\_ADDR\_NXTSERVO 0xB0

MindSensors NXTServo I2C address

6.2.2.1102 #define MS\_ADDR\_NXTSERVO\_EM 0x40

MindSensors NXTServo in edit macro mode I2C address

6.2.2.1103 #define MS\_ADDR\_PFMATE 0x48

MindSensors PFMate I2C address

6.2.2.1104 #define MS\_ADDR\_PSPNX 0x02

MindSensors PSP-Nx I2C address

6.2.2.1105 #define MS\_ADDR\_RTCLOCK 0xD0

MindSensors RTClock I2C address

6.2.2.1106 #define MS\_ADDR\_RXMUX 0x7E

MindSensors RXMux I2C address

6.2.2.1107 #define MS\_ADDR\_TOUCHPANEL 0x04

MindSensors TouchPanel I2C address

6.2.2.1108 #define MS\_CMD\_ADPA\_OFF 0x4F

Turn MindSensors ADPA mode off

6.2.2.1109 #define MS\_CMD\_ADPA\_ON 0x4E

Turn MindSensors ADPA mode on

6.2.2.1110 #define MS\_CMD\_DEENERGIZED 0x44

De-energize the MindSensors device

6.2.2.1111 #define MS\_CMD\_ENERGIZED 0x45

Energize the MindSensors device

6.2.2.1112 #define NA 0xFFFF

The specified argument does not apply (aka unwired)

6.2.2.1113 #define NO\_ERR 0

Successful execution of the specified command

6.2.2.1114 #define NO\_OF\_BTNS 4

The number of NXT buttons.

6.2.2.1115 #define NormalizedValueField 3

Normalized value field. Contains the current normalized analog sensor value. Read only.

6.2.2.1116 #define NOTE\_EIGHT (NOTE\_WHOLE/8)

The duration of an eighth note (ms)

6.2.2.1117 #define NOTE\_HALF (NOTE\_WHOLE/2)

The duration of a half note (ms)

6.2.2.1118 #define NOTE\_QUARTER (NOTE\_WHOLE/4)

The duration of a quarter note (ms)

6.2.2.1119 #define NOTE\_SIXTEEN (NOTE\_WHOLE/16)

The duration of an sixteenth note (ms)

6.2.2.1120 #define NOTE\_WHOLE 1000

The duration of a whole note (ms)

6.2.2.1121 #define NRLINK\_CMD\_2400 0x44

Set NRLink to 2400 baud

6.2.2.1122 #define NRLINK\_CMD\_4800 0x48

Set NRLink to 4800 baud

6.2.2.1123 #define NRLINK\_CMD\_FLUSH 0x46

Flush the NRLink

6.2.2.1124 #define NRLINK\_CMD\_IR\_LONG 0x4C

Set the NRLink to long range IR

6.2.2.1125 #define NRLINK\_CMD\_IR\_SHORT 0x53

Set the NRLink to short range IR

6.2.2.1126 #define NRLINK\_CMD\_RUN\_MACRO 0x52

Run an NRLink macro

6.2.2.1127 #define NRLINK\_CMD\_SET\_PF 0x50

Set the NRLink to Power Function mode

6.2.2.1128 #define NRLINK\_CMD\_SET\_RCX 0x58

Set the NRLink to RCX mode

6.2.2.1129 #define NRLINK\_CMD\_SET\_TRAIN 0x54

Set the NRLink to IR Train mode

6.2.2.1130 #define NRLINK\_CMD\_TX\_RAW 0x55

Set the NRLink to transmit raw bytes

6.2.2.1131 #define NRLINK\_REG\_BYTES 0x40

The NRLink bytes register

6.2.2.1132 #define NRLINK\_REG\_DATA 0x42

The NRLink data register

6.2.2.1133 #define NRLINK\_REG\_EEPROM 0x50

The NRLink eeprom register

6.2.2.1134 #define NULL 0

A constant representing NULL

6.2.2.1135 #define NXTHID\_CMD\_ASCII 0x41

Use ASCII data mode. In ASCII mode no non-printable characters can be sent.

6.2.2.1136 #define NXTHID\_CMD\_DIRECT 0x44

Use direct data mode In direct mode any character can be sent.

6.2.2.1137 #define NXTHID\_CMD\_TRANSMIT 0x54

NXTHID transmit command.

6.2.2.1138 #define NXTHID\_MOD\_LEFT\_ALT 0x04

NXTHID left alt modifier.

6.2.2.1139 #define NXTHID\_MOD\_LEFT\_CTRL 0x01

NXTHID left control modifier.

6.2.2.1140 #define NXTHID\_MOD\_LEFT\_GUI 0x08

NXTHID left gui modifier.

6.2.2.1141 #define NXTHID\_MOD\_LEFT\_SHIFT 0x02

NXTHID left shift modifier.

6.2.2.1142 #define NXTHID\_MOD\_NONE 0x00

NXTHID no modifier.

6.2.2.1143 #define NXTHID\_MOD\_RIGHT\_ALT 0x40

NXTHID right alt modifier.

6.2.2.1144 #define NXTHID\_MOD\_RIGHT\_CTRL 0x10

NXTHID right control modifier.

6.2.2.1145 #define NXTHID\_MOD\_RIGHT\_GUI 0x80

NXTHID right gui modifier.

6.2.2.1146 #define NXTHID\_MOD\_RIGHT\_SHIFT 0x20

NXTHID right shift modifier.

6.2.2.1147 #define NXTHID\_REG\_CMD 0x41

NXTHID command register. See [MindSensors NXTHID commands group](#).

6.2.2.1148 #define NXTHID\_REG\_DATA 0x43

NXTHID data register.

6.2.2.1149 #define NXTHID\_REG\_MODIFIER 0x42

NXTHID modifier register. See [MindSensors NXTHID modifier keys group](#).

6.2.2.1150 #define NXTLL\_CMD\_BLACK 0x42

Black calibration.

6.2.2.1151 #define NXTLL\_CMD\_EUROPEAN 0x45

European power frequency. (50hz)

6.2.2.1152 #define NXTLL\_CMD\_INVERT 0x49

Invert color.

6.2.2.1153 #define NXTLL\_CMD\_POWERDOWN 0x44

Power down the device.

6.2.2.1154 #define NXTLL\_CMD\_POWERUP 0x50

Power up the device.

6.2.2.1155 #define NXTLL\_CMD\_RESET 0x52

Reset inversion.

6.2.2.1156 #define NXTLL\_CMD\_SNAPSHOT 0x53

Setpoint based on snapshot (automatically sets invert if needed).

6.2.2.1157 #define NXTLL\_CMD\_UNIVERSAL 0x55

Universal power frequency. The sensor auto adjusts for any frequency. This is the default mode.

6.2.2.1158 #define NXTLL\_CMD\_USA 0x41

USA power frequency. (60hz)

6.2.2.1159 #define NXTLL\_CMD\_WHITE 0x57

White balance calibration.

6.2.2.1160 #define NXTLL\_REG\_AVERAGE 0x43

NXTLineLeader average result register.

6.2.2.1161 #define NXTLL\_REG\_BLACKDATA 0x6C

NXTLineLeader black calibration data registers. 8 bytes.

6.2.2.1162 #define NXTLL\_REG\_BLACKLIMITS 0x59

NXTLineLeader black limit registers. 8 bytes.

6.2.2.1163 #define NXTLL\_REG\_CALIBRATED 0x49

NXTLineLeader calibrated sensor reading registers. 8 bytes.

6.2.2.1164 #define NXTLL\_REG\_CMD 0x41

NXTLineLeader command register. See the [MindSensors NXTLineLeader commands](#) group.

6.2.2.1165 #define NXTLL\_REG\_KD\_FACTOR 0x63

NXTLineLeader Kd factor register. Default = 32.

6.2.2.1166 #define NXTLL\_REG\_KD\_VALUE 0x48

NXTLineLeader Kd value register. Default = 8.

6.2.2.1167 #define NXTLL\_REG\_KI\_FACTOR 0x62

NXTLineLeader Ki factor register. Default = 32.

6.2.2.1168 #define NXTLL\_REG\_KI\_VALUE 0x47

NXTLineLeader Ki value register. Default = 0.

6.2.2.1169 #define NXTLL\_REG\_KP\_FACTOR 0x61

NXTLineLeader Kp factor register. Default = 32.

6.2.2.1170 #define NXTLL\_REG\_KP\_VALUE 0x46

NXTLineLeader Kp value register. Default = 25.

6.2.2.1171 #define NXTLL\_REG\_RAWVOLTAGE 0x74

NXTLineLeader uncalibrated sensor voltage registers. 16 bytes.

6.2.2.1172 #define NXTLL\_REG\_RESULT 0x44

NXTLineLeader result register (sensor bit values).

6.2.2.1173 #define NXTLL\_REG\_SETPOINT 0x45

NXTLineLeader user settable average (setpoint) register. Default = 45.

6.2.2.1174 #define NXTLL\_REG\_STEERING 0x42

NXTLineLeader steering register.

6.2.2.1175 #define NXTLL\_REG\_WHITEDATA 0x64

NXTLineLeader white calibration data registers. 8 bytes.

6.2.2.1176 #define NXTLL\_REG\_WHITELIMITS 0x51

NXTLineLeader white limit registers. 8 bytes.

6.2.2.1177 #define NXTNP\_REG\_BUTTONS 0x00

NXTNumericPad buttons register.

6.2.2.1178 #define NXTPM\_CMD\_RESET 0x52

Reset counters.

6.2.2.1179 #define NXTPM\_REG\_CAPACITY 0x46

NXTPowerMeter capacity used since last reset register. (2 bytes)

6.2.2.1180 #define NXTPM\_REG\_CMD 0x41

NXTPowerMeter command register. See the [MindSensors NXTPowerMeter commands](#) group.

6.2.2.1181 #define NXTPM\_REG\_CURRENT 0x42

NXTPowerMeter present current in mA register. (2 bytes)

6.2.2.1182 #define NXTPM\_REG\_ERRORCOUNT 0x5F

NXTPowerMeter error count register. (2 bytes)

6.2.2.1183 #define NXTPM\_REG\_GAIN 0x5E

NXTPowerMeter gain register. (1 byte)

6.2.2.1184 #define NXTPM\_REG\_MAXCURRENT 0x4E

NXTPowerMeter max current register. (2 bytes)

6.2.2.1185 #define NXTPM\_REG\_MAXVOLTAGE 0x52

NXTPowerMeter max voltage register. (2 bytes)

6.2.2.1186 #define NXTPM\_REG\_MINCURRENT 0x50

NXTPowerMeter min current register. (2 bytes)

6.2.2.1187 #define NXTPM\_REG\_MINVOLTAGE 0x54

NXTPowerMeter min voltage register. (2 bytes)

6.2.2.1188 #define NXTPM\_REG\_POWER 0x48

NXTPowerMeter present power register. (2 bytes)

6.2.2.1189 #define NXTPM\_REG\_TIME 0x56

NXTPowerMeter time register. (4 bytes)

6.2.2.1190 #define NXTPM\_REG\_TOTALPOWER 0x4A

NXTPowerMeter total power consumed since last reset register. (4 bytes)

6.2.2.1191 #define NXTPM\_REG\_USERGAIN 0x5A

NXTPowerMeter user gain register. Not yet implemented. (4 bytes)

6.2.2.1192 #define NXTPM\_REG\_VOLTAGE 0x44

NXTPowerMeter present voltage in mV register. (2 bytes)

6.2.2.1193 #define NXTSE\_ZONE\_FRONT 1

Obstacle zone front.

6.2.2.1194 #define NXTSE\_ZONE\_LEFT 2

Obstacle zone left.

6.2.2.1195 #define NXTSE\_ZONE\_NONE 0

Obstacle zone none.

6.2.2.1196 #define NXTSE\_ZONE\_RIGHT 3

Obstacle zone right.

6.2.2.1197 #define NXTSERVO\_CMD\_EDIT1 0x45

Edit Macro (part 1 of 2 character command sequence)

6.2.2.1198 #define NXTSERVO\_CMD\_EDIT2 0x4D

Edit Macro (part 2 of 2 character command sequence)

6.2.2.1199 #define NXTSERVO\_CMD\_GOTO 0x47

Goto EEPROM position x. This command re-initializes the macro environment.

6.2.2.1200 #define NXTSERVO\_CMD\_HALT 0x48

Halt Macro. This command re-initializes the macro environment.

6.2.2.1201 #define NXTSERVO\_CMD\_INIT 0x49

Store the initial speed and position properties of the servo motor 'n'. Current speed and position values of the nth servo is read from the servo speed register and servo position register and written to permanent memory.

6.2.2.1202 #define NXTSERVO\_CMD\_PAUSE 0x50

Pause Macro. This command will pause the macro, and save the environment for subsequent resumption.

6.2.2.1203 #define NXTSERVO\_CMD\_RESET 0x53

Reset servo properties to factory default. Initial Position of servos to 1500, and speed to 0.

6.2.2.1204 #define NXTSERVO\_CMD\_RESUME 0x52

Resume macro Execution. This command resumes macro where it was paused last, using the same environment.

6.2.2.1205 #define NXTSERVO\_EM\_CMD\_QUIT 0x51

Exit edit macro mode

6.2.2.1206 #define NXTSERVO\_EM\_REG\_CMD 0x00

NXTServo in macro edit mode command register.

6.2.2.1207 #define NXTSERVO\_EM\_REG\_EEPROM\_END 0xFF

NXTServo in macro edit mode EEPROM end register.

6.2.2.1208 #define NXTSERVO\_EM\_REG\_EEPROM\_START 0x21

NXTServo in macro edit mode EEPROM start register.

6.2.2.1209 #define NXTSERVO\_POS\_CENTER 1500

Center position for 1500us servos.

6.2.2.1210 #define NXTSERVO\_POS\_MAX 2500

Maximum position for 1500us servos.

6.2.2.1211 #define NXTSERVO\_POS\_MIN 500

Minimum position for 1500us servos.

6.2.2.1212 #define NXTSERVO\_QPOS\_CENTER 150

Center quick position for 1500us servos.

6.2.2.1213 #define NXTSERVO\_QPOS\_MAX 250

Maximum quick position for 1500us servos.

6.2.2.1214 #define NXTSERVO\_QPOS\_MIN 50

Minimum quick position for 1500us servos.

6.2.2.1215 #define NXTSERVO\_REG\_CMD 0x41

NXTServo command register. See [MindSensors NXTServo commands](#) group. (write only)

6.2.2.1216 #define NXTSERVO\_REG\_S1\_POS 0x42

NXTServo servo 1 position register.

6.2.2.1217 #define NXTSERVO\_REG\_S1\_QPOS 0x5A

NXTServo servo 1 quick position register. (write only)

6.2.2.1218 #define NXTSERVO\_REG\_S1\_SPEED 0x52

NXTServo servo 1 speed register.

6.2.2.1219 #define NXTSERVO\_REG\_S2\_POS 0x44

NXTServo servo 2 position register.

6.2.2.1220 #define NXTSERVO\_REG\_S2\_QPOS 0x5B

NXTServo servo 2 quick position register. (write only)

6.2.2.1221 #define NXTSERVO\_REG\_S2\_SPEED 0x53

NXTServo servo 2 speed register.

6.2.2.1222 #define NXTSERVO\_REG\_S3\_POS 0x46

NXTServo servo 3 position register.

6.2.2.1223 #define NXTSERVO\_REG\_S3\_QPOS 0x5C

NXTServo servo 3 quick position register. (write only)

6.2.2.1224 #define NXTSERVO\_REG\_S3\_SPEED 0x54

NXTServo servo 3 speed register.

6.2.2.1225 #define NXTSERVO\_REG\_S4\_POS 0x48

NXTServo servo 4 position register.

6.2.2.1226 #define NXTSERVO\_REG\_S4\_QPOS 0x5D

NXTServo servo 4 quick position register. (write only)

6.2.2.1227 #define NXTSERVO\_REG\_S4\_SPEED 0x55

NXTServo servo 4 speed register.

6.2.2.1228 #define NXTSERVO\_REG\_S5\_POS 0x4A

NXTServo servo 5 position register.

6.2.2.1229 #define NXTSERVO\_REG\_S5\_QPOS 0x5E

NXTServo servo 5 quick position register. (write only)

6.2.2.1230 #define NXTSERVO\_REG\_S5\_SPEED 0x56

NXTServo servo 5 speed register.

6.2.2.1231 #define NXTSERVO\_REG\_S6\_POS 0x4C

NXTServo servo 6 position register.

6.2.2.1232 #define NXTSERVO\_REG\_S6\_QPOS 0x5F

NXTServo servo 6 quick position register. (write only)

6.2.2.1233 #define NXTSERVO\_REG\_S6\_SPEED 0x57

NXTServo servo 6 speed register.

6.2.2.1234 #define NXTSERVO\_REG\_S7\_POS 0x4E

NXTServo servo 7 position register.

6.2.2.1235 #define NXTSERVO\_REG\_S7\_QPOS 0x60

NXTServo servo 7 quick position register. (write only)

6.2.2.1236 #define NXTSERVO\_REG\_S7\_SPEED 0x58

NXTServo servo 7 speed register.

6.2.2.1237 #define NXTSERVO\_REG\_S8\_POS 0x50

NXTServo servo 8 position register.

6.2.2.1238 #define NXTSERVO\_REG\_S8\_QPOS 0x61

NXTServo servo 8 quick position register. (write only)

6.2.2.1239 #define NXTSERVO\_REG\_S8\_SPEED 0x59

NXTServo servo 8 speed register.

6.2.2.1240 #define NXTSERVO\_REG\_VOLTAGE 0x41

Battery voltage register. (read only)

6.2.2.1241 #define NXTSERVO\_SERVO\_1 0

NXTServo server number 1.

6.2.2.1242 #define NXTSERVO\_SERVO\_2 1

NXTServo server number 2.

6.2.2.1243 #define NXTSERVO\_SERVO\_3 2

NXTServo server number 3.

6.2.2.1244 #define NXTSERVO\_SERVO\_4 3

NXTServo server number 4.

6.2.2.1245 #define NXTSERVO\_SERVO\_5 4

NXTServo server number 5.

6.2.2.1246 #define NXTSERVO\_SERVO\_6 5

NXTServo server number 6.

6.2.2.1247 #define NXTSERVO\_SERVO\_7 6

NXTServo server number 7.

6.2.2.1248 #define NXTSERVO\_SERVO\_8 7

NXTServo server number 8.

6.2.2.1249 #define NXTTP\_CMD\_USA 0x41

USA power frequency. (60hz)

6.2.2.1250 #define NXTTP\_REG\_CMD 0x41

NXTTTouchPanel command register. See the [MindSensors NXTTTouchPanel commands](#) group.

6.2.2.1251 #define OPARR\_MAX 0x05

Calculate the maximum value of the elements in the numeric input array

6.2.2.1252 #define OPARR\_MEAN 0x01

Calculate the mean value for the elements in the numeric input array

6.2.2.1253 #define OPARR\_MIN 0x04

Calculate the minimum value of the elements in the numeric input array

6.2.2.1254 #define OPARR\_SORT 0x06

Sort the elements in the numeric input array

6.2.2.1255 #define OPARR\_STD 0x03

Calculate the standard deviation of the elements in the numeric input array

6.2.2.1256 #define OPARR\_SUM 0x00

Calculate the sum of the elements in the numeric input array

6.2.2.1257 #define OPARR\_SUMSQR 0x02

Calculate the sum of the squares of the elements in the numeric input array

6.2.2.1258 #define OPARR\_TOLOWER 0x08

Lowercase the input string

6.2.2.1259 #define OPARR\_TOUPPER 0x07

Uppercase the input string

6.2.2.1260 #define OUT\_A 0x00

Output port A

6.2.2.1261 #define OUT\_AB 0x03

Output ports A and B

6.2.2.1262 #define OUT\_ABC 0x06

Output ports A, B, and C

6.2.2.1263 #define OUT\_AC 0x04

Output ports A and C

6.2.2.1264 #define OUT\_B 0x01

Output port B

6.2.2.1265 #define OUT\_BC 0x05

Output ports B and C

6.2.2.1266 #define OUT\_C 0x02

Output port C

6.2.2.1267 #define OUT\_MODE\_BRAKE 0x02

Uses electronic braking to outputs

6.2.2.1268 #define OUT\_MODE\_COAST 0x00

No power and no braking so motors rotate freely.

6.2.2.1269 #define OUT\_MODE\_MOTORON 0x01

Enables PWM power to the outputs given the power setting

6.2.2.1270 #define OUT\_MODE\_REGMETHOD 0xF0

Mask for unimplemented regulation mode

6.2.2.1271 #define OUT\_MODE\_REGULATED 0x04

Enables active power regulation using the regulation mode value

6.2.2.1272 #define OUT\_OPTION\_HOLDATLIMIT 0x10

Option to have the firmware hold the motor when it reaches the tachometer limit

6.2.2.1273 #define OUT\_OPTION\_RAMPDOWNTOLIMIT 0x20

Option to have the firmware rampdown the motor power as it approaches the tachometer limit

6.2.2.1274 #define OUT\_REGMODE\_IDLE 0

No motor regulation.

6.2.2.1275 #define OUT\_REGMODE\_POS 4

Regulate a motor's position.

6.2.2.1276 #define OUT\_REGMODE\_SPEED 1

Regulate a motor's speed (aka power).

6.2.2.1277 #define OUT\_REGMODE\_SYNC 2

Synchronize the rotation of two motors.

6.2.2.1278 #define OUT\_REGOPTION\_NO\_SATURATION 0x01

Do not limit intermediary regulation results

6.2.2.1279 #define OUT\_RUNSTATE\_HOLD 0x60

Set motor run state to hold at the current position.

6.2.2.1280 #define OUT\_RUNSTATE\_IDLE 0x00

Disable all power to motors.

6.2.2.1281 #define OUT\_RUNSTATE\_RAMPDOWN 0x40

Enable ramping down from a current power to a new (lower) power over a specified [TachoLimitField](#) goal.

6.2.2.1282 #define OUT\_RUNSTATE\_RAMPUP 0x10

Enable ramping up from a current power to a new (higher) power over a specified [TachoLimitField](#) goal.

6.2.2.1283 #define OUT\_RUNSTATE\_RUNNING 0x20

Enable power to motors at the specified power level.

6.2.2.1284 #define OutputModeField 1

Mode field.

Contains a combination of the output mode constants. Read/write. The [OUT\\_MODE\\_MOTORON](#) bit must be set in order for power to be applied to the motors. Add [OUT\\_MODE\\_BRAKE](#) to enable electronic braking. Braking means that the output voltage is not allowed to float between active PWM pulses. It improves the accuracy of motor output but uses more battery power. To use motor regulation include [OUT\\_MODE\\_REGULATED](#) in the [OutputModeField](#) value. Use [UF\\_UPDATE\\_MODE](#) with [UpdateFlagsField](#) to commit changes to this field.

6.2.2.1285 #define OutputModuleID 0x00020001

The output module ID

6.2.2.1286 #define OutputModuleName "Output.mod"

The output module name

6.2.2.1287 #define OutputOffsetActualSpeed( p ) (((p)\*32)+21)

R - Holds the current motor speed (1 byte) sbyte

6.2.2.1288 #define OutputOffsetBlockTachoCount( p ) (((p)\*32)+4)

R - Holds current number of counts for the current output block (4 bytes) slong

6.2.2.1289 #define OutputOffsetFlags( p ) (((p)\*32)+18)

RW - Holds flags for which data should be updated (1 byte) ubyte

6.2.2.1290 #define OutputOffsetMaxAccel( *p* ) (((*p*)\*32)+31)

RW - holds the maximum acceleration for position regulation (1 byte) sbyte (enhanced NBC/NXC firmware only)

6.2.2.1291 #define OutputOffsetMaxSpeed( *p* ) (((*p*)\*32)+30)

RW - holds the maximum speed for position regulation (1 byte) sbyte (enhanced NBC/NXC firmware only)

6.2.2.1292 #define OutputOffsetMode( *p* ) (((*p*)\*32)+19)

RW - Holds motor mode: Run, Break, regulated, ... (1 byte) ubyte

6.2.2.1293 #define OutputOffsetMotorRPM( *p* ) (((*p*)\*32)+16)

Not updated, will be removed later !! (2 bytes) sword

6.2.2.1294 #define OutputOffsetOptions( *p* ) (((*p*)\*32)+29)

RW - holds extra motor options related to the tachometer limit (1 byte) ubyte (enhanced NBC/NXC firmware only)

6.2.2.1295 #define OutputOffsetOverloaded( *p* ) (((*p*)\*32)+27)

R - True if the motor has been overloaded within speed control regulation (1 byte) ubyte

6.2.2.1296 #define OutputOffsetRegDParameter( *p* ) (((*p*)\*32)+24)

RW - Holds the D-constant used in the regulation (1 byte) ubyte

6.2.2.1297 #define OutputOffsetRegIParameter( *p* ) (((*p*)\*32)+23)

RW - Holds the I-constant used in the regulation (1 byte) ubyte

6.2.2.1298 #define OutputOffsetRegMode( *p* ) (((*p*)\*32)+26)

RW - Tells which regulation mode should be used (1 byte) ubyte

6.2.2.1299 #define OutputOffsetRegPParameter( *p* ) (((*p*)\*32)+22)

RW - Holds the P-constant used in the regulation (1 byte) ubyte

6.2.2.1300 #define OutputOffsetRegulationOptions 97

use for position regulation options (1 byte) ubyte (enhanced NBC/NXC firmware only)

6.2.2.1301 #define OutputOffsetRegulationTime 96

use for frequency of checking regulation mode (1 byte) ubyte (enhanced NBC/NXC firmware only)

6.2.2.1302 #define OutputOffsetRotationCount( *p* ) (((*p*)\*32)+8)

R - Holds current number of counts for the rotation counter to the output (4 bytes) slong

6.2.2.1303 #define OutputOffsetRunState( *p* ) (((*p*)\*32)+25)

RW - Holds the current motor run state in the output module (1 byte) ubyte

6.2.2.1304 #define OutputOffsetSpeed( p ) (((p)\*32)+20)

RW - Holds the wanted speed (1 byte) sbyte

6.2.2.1305 #define OutputOffsetSyncTurnParameter( p ) (((p)\*32)+28)

RW - Holds the turning parameter need within MoveBlock (1 byte) sbyte

6.2.2.1306 #define OutputOffsetTachoCount( p ) (((p)\*32)+0)

R - Holds current number of counts, since last reset, updated every 1 mS (4 bytes) ulong

6.2.2.1307 #define OutputOffsetTachoLimit( p ) (((p)\*32)+12)

RW - Holds number of counts to travel, 0 => Run forever (4 bytes) ulong

6.2.2.1308 #define OutputOptionsField 15

Options field.

Contains a combination of the output options constants. Read/write. Set options for how the output module will act when a tachometer limit is reached. Option constants can be combined with bitwise OR. Use OUT\_OPTION\_HOLDATLIMIT to have the output module hold the motor when it reaches the tachometer limit. Use OUT\_OPTION\_RAMPDOWNTOLIMIT to have the output module ramp down the motor power as it approaches the tachometer limit.

#### Warning

This option requires the enhanced NBC/NXC firmware version 1.31+

6.2.2.1309 #define OverloadField 9

Overload field.

Contains a boolean value which is TRUE if the motor is overloaded. Read only. This field will have a value of 1 (true) if the firmware speed regulation cannot overcome a physical load on the motor. In other words, the motor is turning more slowly than expected. If the motor speed can be maintained in spite of loading then this field value is zero (false). In order to use this field the motor must have a non-idle [RunStateField](#), an [OutputModeField](#) which includes [OUT\\_MODE\\_MOTORON](#) and [OUT\\_MODE\\_REGULATED](#), and its [RegModeField](#) must be set to [OUT\\_REGMODE\\_SPEED](#).

6.2.2.1310 #define PF\_CHANNEL\_1 0

Power function channel 1

6.2.2.1311 #define PF\_CHANNEL\_2 1

Power function channel 2

6.2.2.1312 #define PF\_CHANNEL\_3 2

Power function channel 3

6.2.2.1313 #define PF\_CHANNEL\_4 3

Power function channel 4

6.2.2.1314 #define PF\_CMD\_BRAKE 3

Power function command brake

6.2.2.1315 #define PF\_CMD\_FLOAT 0

Power function command float (same as stop)

6.2.2.1316 #define PF\_CMD\_FWD 1

Power function command forward

6.2.2.1317 #define PF\_CMD\_REV 2

Power function command reverse

6.2.2.1318 #define PF\_CMD\_STOP 0

Power function command stop

6.2.2.1319 #define PF\_CST\_CLEAR1\_CLEAR2 0

Power function CST clear 1 and clear 2

6.2.2.1320 #define PF\_CST\_CLEAR1\_SET2 2

Power function CST clear 1 and set 2

6.2.2.1321 #define PF\_CST\_DECREMENT\_PWM 5

Power function CST decrement PWM

6.2.2.1322 #define PF\_CST\_FULL\_FWD 6

Power function CST full forward

6.2.2.1323 #define PF\_CST\_FULL\_REV 7

Power function CST full reverse

6.2.2.1324 #define PF\_CST\_INCREMENT\_PWM 4

Power function CST increment PWM

6.2.2.1325 #define PF\_CST\_SET1\_CLEAR2 1

Power function CST set 1 and clear 2

6.2.2.1326 #define PF\_CST\_SET1\_SET2 3

Power function CST set 1 and set 2

6.2.2.1327 #define PF\_CST\_TOGGLE\_DIR 8

Power function CST toggle direction

6.2.2.1328 #define PF\_FUNC\_CLEAR 1

Power function single pin - clear

6.2.2.1329 #define PF\_FUNC\_NOCHANGE 0

Power function single pin - no change

6.2.2.1330 #define PF\_FUNC\_SET 2

Power function single pin - set

6.2.2.1331 #define PF\_FUNC\_TOGGLE 3

Power function single pin - toggle

6.2.2.1332 #define PF\_MODE\_COMBO\_DIRECT 1

Power function mode combo direct

6.2.2.1333 #define PF\_MODE\_COMBO\_PWM 4

Power function mode combo pulse width modulation (PWM)

6.2.2.1334 #define PF\_MODE\_SINGLE\_OUTPUT\_CST 6

Power function mode single output clear, set, toggle (CST)

6.2.2.1335 #define PF\_MODE\_SINGLE\_OUTPUT\_PWM 4

Power function mode single output pulse width modulation (PWM)

6.2.2.1336 #define PF\_MODE\_SINGLE\_PIN\_CONT 2

Power function mode single pin continuous

6.2.2.1337 #define PF\_MODE\_SINGLE\_PIN\_TIME 3

Power function mode single pin timed

6.2.2.1338 #define PF\_MODE\_TRAIN 0

Power function mode IR Train

6.2.2.1339 #define PF\_OUT\_A 0

Power function output A

6.2.2.1340 #define PF\_OUT\_B 1

Power function output B

6.2.2.1341 #define PF\_PIN\_C1 0

Power function pin C1

6.2.2.1342 #define PF\_PIN\_C2 1

Power function pin C2

6.2.2.1343 #define PF\_PWM\_BRAKE 8

Power function PWM brake

6.2.2.1344 #define PF\_PWM\_FLOAT 0

Power function PWM float

6.2.2.1345 #define PF\_PWM\_FWD1 1

Power function PWM foward level 1

6.2.2.1346 #define PF\_PWM\_FWD2 2

Power function PWM foward level 2

6.2.2.1347 #define PF\_PWM\_FWD3 3

Power function PWM foward level 3

6.2.2.1348 #define PF\_PWM\_FWD4 4

Power function PWM foward level 4

6.2.2.1349 #define PF\_PWM\_FWD5 5

Power function PWM foward level 5

6.2.2.1350 #define PF\_PWM\_FWD6 6

Power function PWM foward level 6

6.2.2.1351 #define PF\_PWM\_FWD7 7

Power function PWM foward level 7

6.2.2.1352 #define PF\_PWM\_REV1 15

Power function PWM reverse level 1

6.2.2.1353 #define PF\_PWM\_REV2 14

Power function PWM reverse level 2

6.2.2.1354 #define PF\_PWM\_REV3 13

Power function PWM reverse level 3

6.2.2.1355 #define PF\_PWM\_REV4 12

Power function PWM reverse level 4

6.2.2.1356 #define PF\_PWM\_REV5 11

Power function PWM reverse level 5

6.2.2.1357 #define PF\_PWM\_REV6 10

Power function PWM reverse level 6

6.2.2.1358 #define PF\_PWM\_REV7 9

Power function PWM reverse level 7

6.2.2.1359 #define PFMATE\_CHANNEL\_1 1

Power function channel 1

6.2.2.1360 #define PFMATE\_CHANNEL\_2 2

Power function channel 2

6.2.2.1361 #define PFMATE\_CHANNEL\_3 3

Power function channel 3

6.2.2.1362 #define PFMATE\_CHANNEL\_4 4

Power function channel 4

6.2.2.1363 #define PFMATE\_CMD\_GO 0x47

Send IR signal to IR receiver

6.2.2.1364 #define PFMATE\_CMD\_RAW 0x52

Send raw IR signal to IR receiver

6.2.2.1365 #define PFMATE\_MOTORS\_A 0x01

Control only motor A

6.2.2.1366 #define PFMATE\_MOTORS\_B 0x02

Control only motor B

6.2.2.1367 #define PFMATE\_MOTORS\_BOTH 0x00

Control both motors

6.2.2.1368 #define PFMATE\_REG\_A\_CMD 0x44

PF command for motor A? (PF\_CMD\_FLOAT, PF\_CMD\_FWD, PF\_CMD\_REV, PF\_CMD\_BRAKE)

6.2.2.1369 #define PFMATE\_REG\_A\_SPEED 0x45

PF speed for motor A? (0-7)

6.2.2.1370 #define PFMATE\_REG\_B\_CMD 0x46

PF command for motor B? (PF\_CMD\_FLOAT, PF\_CMD\_FWD, PF\_CMD\_REV, PF\_CMD\_BRAKE)

6.2.2.1371 #define PFMATE\_REG\_B\_SPEED 0x47

PF speed for motor B? (0-7)

6.2.2.1372 #define PFMATE\_REG\_CHANNEL 0x42

PF channel? 1, 2, 3, or 4

6.2.2.1373 #define PFMATE\_REG\_CMD 0x41

PFMate command

6.2.2.1374 #define PFMATE\_REG\_MOTORS 0x43

PF motors? (0 = both, 1 = A, 2 = B)

6.2.2.1375 #define PI 3.141593

A constant for PI

6.2.2.1376 #define PID\_0 0

PID zero

6.2.2.1377 #define PID\_1 32

PID one

6.2.2.1378 #define PID\_2 64

PID two

6.2.2.1379 #define PID\_3 96

PID three

6.2.2.1380 #define PID\_4 128

PID four

6.2.2.1381 #define PID\_5 160

PID five

6.2.2.1382 #define PID\_6 192

PID six

6.2.2.1383 #define PID\_7 224

PID seven

6.2.2.1384 #define POOL\_MAX\_SIZE 32768

Maximum size of memory pool, in bytes

6.2.2.1385 #define PowerField 2

Power field.

Contains the desired power level (-100 to 100). Read/write. Specify the power level of the output. The absolute value of PowerField is a percentage of the full power of the motor. The sign of PowerField controls the rotation direction. Positive values tell the firmware to turn the motor forward, while negative values turn the motor backward. Use [UF\\_UPDATE\\_SPEED](#) with [UpdateFlagsField](#) to commit changes to this field.

6.2.2.1386 #define PROG\_ABORT 4

Program has been aborted

6.2.2.1387 #define PROG\_ERROR 3

A program error has occurred

6.2.2.1388 #define PROG\_IDLE 0

Program state is idle

6.2.2.1389 #define PROG\_OK 1

Program state is okay

6.2.2.1390 #define PROG\_RESET 5

Program has been reset

6.2.2.1391 #define PROG\_RUNNING 2

Program is running

6.2.2.1392 #define PSP\_BTNSET1\_DOWN 0x40

The PSP-Nx button set 1 down arrow

6.2.2.1393 #define PSP\_BTNSET1\_L3 0x02

The PSP-Nx button set 1 L3

6.2.2.1394 #define PSP\_BTNSET1\_LEFT 0x80

The PSP-Nx button set 1 left arrow

6.2.2.1395 #define PSP\_BTNSET1\_R3 0x04

The PSP-Nx button set 1 R3

6.2.2.1396 #define PSP\_BTNSET1\_RIGHT 0x20

The PSP-Nx button set 1 right arrow

6.2.2.1397 #define PSP\_BTNSET1\_SELECT 0x01

The PSP-Nx button set 1 select

6.2.2.1398 #define PSP\_BTNSET1\_START 0x08

The PSP-Nx button set 1 start

6.2.2.1399 #define PSP\_BTNSET1\_UP 0x10

The PSP-Nx button set 1 up arrow

6.2.2.1400 #define PSP\_BTNSET2\_CIRCLE 0x20

The PSP-Nx button set 2 circle

6.2.2.1401 #define PSP\_BTNSET2\_CROSS 0x40

The PSP-Nx button set 2 cross

6.2.2.1402 #define PSP\_BTNSET2\_L1 0x04

The PSP-Nx button set 2 L1

6.2.2.1403 #define PSP\_BTNSET2\_L2 0x01

The PSP-Nx button set 2 L2

6.2.2.1404 #define PSP\_BTNSET2\_R1 0x08

The PSP-Nx button set 2 R1

6.2.2.1405 #define PSP\_BTNSET2\_R2 0x02

The PSP-Nx button set 2 R2

6.2.2.1406 #define PSP\_BTNSET2\_SQUARE 0x80

The PSP-Nx button set 2 square

6.2.2.1407 #define PSP\_BTNSET2\_TRIANGLE 0x10

The PSP-Nx button set 2 triangle

6.2.2.1408 #define PSP\_CMD\_ANALOG 0x73

Set the PSP-Nx to analog mode

6.2.2.1409 #define PSP\_CMD\_DIGITAL 0x41

Set the PSP-Nx to digital mode

6.2.2.1410 #define PSP\_REG\_BTNSET1 0x42

The PSP-Nx button set 1 register

6.2.2.1411 #define PSP\_REG\_BTNSET2 0x43

The PSP-Nx button set 2 register

6.2.2.1412 #define PSP\_REG\_XLEFT 0x44

The PSP-Nx X left register

6.2.2.1413 #define PSP\_REG\_XRIGHT 0x46

The PSP-Nx X right register

6.2.2.1414 #define PSP\_REG\_YLEFT 0x45

The PSP-Nx Y left register

6.2.2.1415 #define PSP\_REG\_YRIGHT 0x47

The PSP-Nx Y right register

6.2.2.1416 #define RADIANS\_PER\_DEGREE PI/180

Used for converting from degrees to radians

6.2.2.1417 #define RAND\_MAX 2147483646

The maximum long random number returned by rand

6.2.2.1418 #define RandomEx 99

Generate a random number or seed the RNG.

6.2.2.1419 #define RandomNumber 24

Generate a random number

6.2.2.1420 #define RawValueField 2

Raw value field. Contains the current raw analog sensor value. Read only.

6.2.2.1421 #define RC\_PROP\_BTTONOFF 0x0

Set/get whether bluetooth is on or off

6.2.2.1422 #define RC\_PROP\_DEBUGGING 0xF

Set/get enhanced firmware debugging information (NBC/NXC)

6.2.2.1423 #define RC\_PROP\_SLEEP\_TIMEOUT 0x2

Set/get the NXT sleep timeout value (times 60000)

6.2.2.1424 #define RC\_PROP\_SOUND\_LEVEL 0x1

Set/get the NXT sound level

6.2.2.1425 #define RCX\_AbsVarOp 0x74

Absolute value function

6.2.2.1426 #define RCX\_AndVarOp 0x84

AND function

6.2.2.1427 #define RCX\_AutoOffOp 0xb1

Set auto off timer

6.2.2.1428 #define RCX\_BatteryLevelOp 0x30

Read the battery level

6.2.2.1429 #define RCX\_BatteryLevelSrc 34

The RCX battery level source

6.2.2.1430 #define RCX\_BootModeOp 0x65

Set into book mode

6.2.2.1431 #define RCX\_CalibrateEventOp 0x04

Calibrate event

6.2.2.1432 #define RCX\_ClearAllEventsOp 0x06

Clear all events

6.2.2.1433 #define RCX\_ClearCounterOp 0xb7

Clear a counter

6.2.2.1434 #define RCX\_ClearMsgOp 0x90

Clear message

6.2.2.1435 #define RCX\_ClearSensorOp 0xd1

Clear a sensor

6.2.2.1436 #define RCX\_ClearSoundOp 0x80

Clear sound

6.2.2.1437 #define RCX\_ClearTimerOp 0xa1

Clear a timer

6.2.2.1438 #define RCX\_ClickCounterSrc 27

The RCX event click counter source

6.2.2.1439 #define RCX\_ConstantSrc 2

The RCX constant value source

6.2.2.1440 #define RCX\_CounterSrc 21

The RCX counter source

6.2.2.1441 #define RCX\_DatalogOp 0x62

Datalog the specified source/value

6.2.2.1442 #define RCX\_DatalogRawDirectSrc 42

The RCX direct datalog raw source

6.2.2.1443 #define RCX\_DatalogRawIndirectSrc 41

The RCX indirect datalog raw source

6.2.2.1444 #define RCX\_DatalogSrcDirectSrc 38

The RCX direct datalog source source

6.2.2.1445 #define RCX\_DatalogSrcIndirectSrc 37

The RCX indirect datalog source source

6.2.2.1446 #define RCX\_DatalogValueDirectSrc 40

The RCX direct datalog value source

6.2.2.1447 #define RCX\_DatalogValueIndirectSrc 39

The RCX indirect datalog value source

6.2.2.1448 #define RCX\_DecCounterOp 0xa7

Decrement a counter

6.2.2.1449 #define RCX\_DeleteSubOp 0xc1

Delete a subroutine

6.2.2.1450 #define RCX\_DeleteSubsOp 0x70

Delete subroutines

6.2.2.1451 #define RCX\_DeleteTaskOp 0x61

Delete a task

6.2.2.1452 #define RCX\_DeleteTasksOp 0x40

Delete tasks

6.2.2.1453 #define RCX\_DirectEventOp 0x03

Fire an event

6.2.2.1454 #define RCX\_DisplayOp 0x33

Set LCD display value

6.2.2.1455 #define RCX\_DivVarOp 0x44

Divide function

6.2.2.1456 #define RCX\_DurationSrc 31

The RCX event duration source

6.2.2.1457 #define RCX\_EventStateSrc 25

The RCX event static source

6.2.2.1458 #define RCX\_FirmwareVersionSrc 35

The RCX firmware version source

6.2.2.1459 #define RCX\_GlobalMotorStatusSrc 17

The RCX global motor status source

6.2.2.1460 #define RCX\_GOutputDirOp 0x77

Set global motor direction

6.2.2.1461 #define RCX\_GOutputModeOp 0x67

Set global motor mode

6.2.2.1462 #define RCX\_GOutputPowerOp 0xa3

Set global motor power levels

6.2.2.1463 #define RCX\_HysteresisSrc 30

The RCX event hysteresis source

6.2.2.1464 #define RCX\_IncCounterOp 0x97

Increment a counter

6.2.2.1465 #define RCX\_IndirectVarSrc 36

The RCX indirect variable source

6.2.2.1466 #define RCX\_InputBooleanSrc 13

The RCX input boolean source

6.2.2.1467 #define RCX\_InputModeOp 0x42

Set the input mode

6.2.2.1468 #define RCX\_InputModeSrc 11

The RCX input mode source

6.2.2.1469 #define RCX\_InputRawSrc 12

The RCX input raw source

6.2.2.1470 #define RCX\_InputTypeOp 0x32

Set the input type

6.2.2.1471 #define RCX\_InputTypeSrc 10

The RCX input type source

6.2.2.1472 #define RCX\_InputValueSrc 9

The RCX input value source

6.2.2.1473 #define RCX\_IRModeOp 0x31

Set the IR transmit mode

6.2.2.1474 #define RCX\_LightOp 0x87

Light opcode

6.2.2.1475 #define RCX\_LowerThresholdSrc 29

The RCX event lower threshold source

6.2.2.1476 #define RCX\_LSBlinkTimeOp 0xe3

Set the light sensor blink time

6.2.2.1477 #define RCX\_LSCalibrateOp 0xc0

Calibrate the light sensor

6.2.2.1478 #define RCX\_LSHysteresisOp 0xd3

Set the light sensor hysteresis

6.2.2.1479 #define RCX\_LSLowerThreshOp 0xc3

Set the light sensor lower threshold

6.2.2.1480 #define RCX\_LSUpprThreshOp 0xb3

Set the light sensor upper threshold

6.2.2.1481 #define RCX\_MessageOp 0xf7

Set message

6.2.2.1482 #define RCX\_MessageSrc 15

The RCX message source

6.2.2.1483 #define RCX\_MulVarOp 0x54

Multiply function

6.2.2.1484 #define RCX\_MuteSoundOp 0xd0

Mute sound

6.2.2.1485 #define RCX\_OnOffFloatOp 0x21

Control motor state - on, off, float

6.2.2.1486 #define RCX\_OrVarOp 0x94

OR function

6.2.2.1487 #define RCX\_OUT\_A 0x01

RCX Output A

6.2.2.1488 #define RCX\_OUT\_AB 0x03

RCX Outputs A and B

6.2.2.1489 #define RCX\_OUT\_ABC 0x07

RCX Outputs A, B, and C

6.2.2.1490 #define RCX\_OUT\_AC 0x05

RCX Outputs A and C

6.2.2.1491 #define RCX\_OUT\_B 0x02

RCX Output B

6.2.2.1492 #define RCX\_OUT\_BC 0x06

RCX Outputs B and C

6.2.2.1493 #define RCX\_OUT\_C 0x04

RCX Output C

6.2.2.1494 #define RCX\_OUT\_FLOAT 0

Set RCX output to float

6.2.2.1495 #define RCX\_OUT\_FULL 7

Set RCX output power level to full

6.2.2.1496 #define RCX\_OUT\_FWD 0x80

Set RCX output direction to forward

6.2.2.1497 #define RCX\_OUT\_HALF 3

Set RCX output power level to half

6.2.2.1498 #define RCX\_OUT\_LOW 0

Set RCX output power level to low

6.2.2.1499 #define RCX\_OUT\_OFF 0x40

Set RCX output to off

6.2.2.1500 #define RCX\_OUT\_ON 0x80

Set RCX output to on

6.2.2.1501 #define RCX\_OUT\_REV 0

Set RCX output direction to reverse

6.2.2.1502 #define RCX\_OUT\_TOGGLE 0x40

Set RCX output direction to toggle

6.2.2.1503 #define RCX\_OutputDirOp 0xe1

Set the motor direction

6.2.2.1504 #define RCX\_OutputPowerOp 0x13

Set the motor power level

6.2.2.1505 #define RCX\_OutputStatusSrc 3

The RCX output status source

6.2.2.1506 #define RCX\_PBTurnOffOp 0x60

Turn off the brick

6.2.2.1507 #define RCX\_PingOp 0x10

Ping the brick

6.2.2.1508 #define RCX\_PlaySoundOp 0x51

Play a sound

6.2.2.1509 #define RCX\_PlayToneOp 0x23

Play a tone

6.2.2.1510 #define RCX\_PlayToneVarOp 0x02

Play a tone using a variable

6.2.2.1511 #define RCX\_PollMemoryOp 0x63

Poll a memory location

6.2.2.1512 #define RCX\_PollOp 0x12

Poll a source/value combination

6.2.2.1513 #define RCX\_ProgramSlotSrc 8

The RCX program slot source

6.2.2.1514 #define RCX\_RandomSrc 4

The RCX random number source

6.2.2.1515 #define RCX\_RemoteKeysReleased 0x0000

All remote keys have been released

6.2.2.1516 #define RCX\_RemoteOp 0xd2

Execute simulated remote control buttons

6.2.2.1517 #define RCX\_RemoteOutABackward 0x4000

Set output A backward

6.2.2.1518 #define RCX\_RemoteOutAForward 0x0800

Set output A forward

6.2.2.1519 #define RCX\_RemoteOutBBackward 0x8000

Set output B backward

6.2.2.1520 #define RCX\_RemoteOutBForward 0x1000

Set output B forward

6.2.2.1521 #define RCX\_RemoteOutCBackward 0x0001

Set output C backward

6.2.2.1522 #define RCX\_RemoteOutCForward 0x2000

Set output C forward

6.2.2.1523 #define RCX\_RemotePBMessage1 0x0100

Send PB message 1

6.2.2.1524 #define RCX\_RemotePBMessage2 0x0200

Send PB message 2

6.2.2.1525 #define RCX\_RemotePBMessage3 0x0400

Send PB message 3

6.2.2.1526 #define RCX\_RemotePlayASound 0x0080

Play a sound

6.2.2.1527 #define RCX\_RemoteSelProgram1 0x0002

Select program 1

6.2.2.1528 #define RCX\_RemoteSelProgram2 0x0004

Select program 2

6.2.2.1529 #define RCX\_RemoteSelProgram3 0x0008

Select program 3

6.2.2.1530 #define RCX\_RemoteSelProgram4 0x0010

Select program 4

6.2.2.1531 #define RCX\_RemoteSelProgram5 0x0020

Select program 5

6.2.2.1532 #define RCX\_RemoteStopOutOff 0x0040

Stop and turn off outputs

6.2.2.1533 #define RCX\_ScoutCounterLimitSrc 22

The Scout counter limit source

6.2.2.1534 #define RCX\_ScoutEventFBSrc 24

The Scout event feedback source

6.2.2.1535 #define RCX\_ScoutLightParamsSrc 19

The Scout light parameters source

6.2.2.1536 #define RCX\_ScoutOp 0x47

Scout opcode

6.2.2.1537 #define RCX\_ScoutRulesOp 0xd5

Set Scout rules

6.2.2.1538 #define RCX\_ScoutRulesSrc 18

The Scout rules source

6.2.2.1539 #define RCX\_ScoutTimerLimitSrc 20

The Scout timer limit source

6.2.2.1540 #define RCX\_SelectProgramOp 0x91

Select a program slot

6.2.2.1541 #define RCX\_SendUARTDataOp 0xc2

Send data via IR using UART settings

6.2.2.1542 #define RCX\_SetCounterOp 0xd4

Set counter value

6.2.2.1543 #define RCX\_SetDatalogOp 0x52

Set the datalog size

6.2.2.1544 #define RCX\_SetEventOp 0x93

Set an event

6.2.2.1545 #define RCX\_SetFeedbackOp 0x83

Set Scout feedback

6.2.2.1546 #define RCX\_SetPriorityOp 0xd7

Set task priority

6.2.2.1547 #define RCX\_SetSourceValueOp 0x05

Set a source/value

6.2.2.1548 #define RCX\_SetTimerLimitOp 0xc4

Set timer limit

6.2.2.1549 #define RCX\_SetVarOp 0x14

Set function

6.2.2.1550 #define RCX\_SetWatchOp 0x22

Set the watch source/value

6.2.2.1551 #define RCX\_SgnVarOp 0x64

Sign function

6.2.2.1552 #define RCX\_SoundOp 0x57

Sound opcode

6.2.2.1553 #define RCX\_StartTaskOp 0x71

Start a task

6.2.2.1554 #define RCX\_StopAllTasksOp 0x50

Stop all tasks

6.2.2.1555 #define RCX\_StopTaskOp 0x81

Stop a task

6.2.2.1556 #define RCX\_SubVarOp 0x34

Subtract function

6.2.2.1557 #define RCX\_SumVarOp 0x24

Sum function

6.2.2.1558 #define RCX\_TaskEventsSrc 23

The RCX task events source

6.2.2.1559 #define RCX\_TenMSTimerSrc 26

The RCX 10ms timer source

6.2.2.1560 #define RCX\_TimerSrc 1

The RCX timer source

6.2.2.1561 #define RCX\_UARTSetupSrc 33

The RCX UART setup source

6.2.2.1562 #define RCX\_UnlockFirmOp 0xa5

Unlock the firmware

6.2.2.1563 #define RCX\_UnlockOp 0x15

Unlock the brick

6.2.2.1564 #define RCX\_UnmuteSoundOp 0xe0

Unmute sound

6.2.2.1565 #define RCX\_UploadDatalogOp 0xa4

Upload datalog contents

6.2.2.1566 #define RCX\_UpperThresholdSrc 28

The RCX event upper threshold source

6.2.2.1567 #define RCX\_VariableSrc 0

The RCX variable source

6.2.2.1568 #define RCX\_ViewSourceValOp 0xe5

View a source/value

6.2.2.1569 #define RCX\_VLLOp 0xe2

Send visual light link (VLL) data

6.2.2.1570 #define RCX\_WatchSrc 14

The RCX watch source

6.2.2.1571 #define ReadButton 20

Read the current button state

6.2.2.1572 #define ReadLastResponse 97

Read the last response packet received by the NXT. Optionally clear the value after reading it.

6.2.2.1573 #define ReadSemData 40

Read motor semaphore data

6.2.2.1574 #define RegDValueField 12

Derivative field.

Contains the derivative constant for the PID motor controller. Read/write. This field specifies the derivative term used in the internal proportional-integral-derivative (PID) control algorithm. Set [UF\\_UPDATE\\_PID\\_VALUES](#) to commit changes to RegPValue, RegIValue, and RegDValue simultaneously.

6.2.2.1575 #define RegIValueField 11

Integral field.

Contains the integral constant for the PID motor controller. Read/write. This field specifies the integral term used in the internal proportional-integral-derivative (PID) control algorithm. Set [UF\\_UPDATE\\_PID\\_VALUES](#) to commit changes to RegPValue, RegIValue, and RegDValue simultaneously.

6.2.2.1576 #define RegModeField 8

Regulation mode field.

Contains one of the regulation mode constants. Read/write. This field specifies the regulation mode to use with the specified port(s). It is ignored if the [OUT\\_MODE\\_REGULATED](#) bit is not set in the [OutputModeField](#) field. Unlike [OutputModeField](#), RegModeField is not a bitfield. Only one regulation mode value can be set at a time. Speed regulation

means that the firmware tries to maintain a certain speed based on the [PowerField](#) setting. The firmware adjusts the PWM duty cycle if the motor is affected by a physical load. This adjustment is reflected by the value of the [ActualSpeedField](#) property. When using speed regulation, do not set [PowerField](#) to its maximum value since the firmware cannot adjust to higher power levels in that situation. Synchronization means the firmware tries to keep two motors in sync regardless of physical loads. Use this mode to maintain a straight path for a mobile robot automatically. Also use this mode with the [TurnRatioField](#) property to provide proportional turning. Set [OUT\\_REGMODE\\_SYNC](#) on at least two motor ports in order for synchronization to function. Setting [OUT\\_REGMODE\\_SYNC](#) on all three motor ports will result in only the first two ([OUT\\_A](#) and [OUT\\_B](#)) being synchronized.

6.2.2.1577 #define RegPValueField 10

Proportional field.

Contains the proportional constant for the PID motor controller. Read/write. This field specifies the proportional term used in the internal proportional-integral-derivative (PID) control algorithm. Set [UF\\_UPDATE\\_PID\\_VALUES](#) to commit changes to RegPValue, RegIValue, and RegDValue simultaneously.

6.2.2.1578 #define RESET\_ALL 0x68

Reset all three tachometer counters

6.2.2.1579 #define RESET\_BLOCK\_COUNT 0x20

Reset the NXT-G block tachometer counter

6.2.2.1580 #define RESET\_BLOCKANDTACHO 0x28

Reset both the internal counter and the NXT-G block counter

6.2.2.1581 #define RESET\_COUNT 0x08

Reset the internal tachometer counter

6.2.2.1582 #define RESET\_NONE 0x00

No counters will be reset

6.2.2.1583 #define RESET\_ROTATION\_COUNT 0x40

Reset the rotation counter

6.2.2.1584 #define RFID\_MODE\_CONTINUOUS 2

Configure the RFID device for continuous reading

6.2.2.1585 #define RFID\_MODE\_SINGLE 1

Configure the RFID device for a single reading

6.2.2.1586 #define RFID\_MODE\_STOP 0

Stop the RFID device

6.2.2.1587 #define RICArg( \_arg ) ((.arg)|0x1000)

Output an RIC parameterized argument.

**Parameters**

<code>_arg</code>	The argument that you want to parameterize.
-------------------	---

6.2.2.1588 #define RICImgPoint( `_X`, `_Y` ) (`_X`&0xFF, (`_X`)>>8, (`_Y`)&0xFF, (`_Y`)>>8)

Output an RIC ImgPoint structure.

**Parameters**

<code>_X</code>	The X coordinate.
<code>_Y</code>	The Y coordinate.

6.2.2.1589 #define RICImgRect( `_Pt`, `_W`, `_H` ) `_Pt`, (`_W`&0xFF, (`_W`)>>8, (`_H`)&0xFF, (`_H`)>>8)

Output an RIC ImgRect structure.

**Parameters**

<code>_Pt</code>	An ImgPoint. See <a href="#">RICImgPoint</a> .
<code>_W</code>	The rectangle width.
<code>_H</code>	The rectangle height.

6.2.2.1590 #define RICMapArg( `_mapidx`, `_arg` ) ((`_arg`)|0x1000|(((`_mapidx`)&0xF)<<8))

Output an RIC parameterized and mapped argument.

**Parameters**

<code>_mapidx</code>	The varmap data address.
<code>_arg</code>	The parameterized argument you want to pass through a varmap.

6.2.2.1591 #define RICMapElement( `_Domain`, `_Range` ) (`_Domain`&0xFF, (`_Domain`)>>8, (`_Range`)&0xFF, (`_Range`)>>8)

Output an RIC map element.

**Parameters**

<code>_Domain</code>	The map element domain.
<code>_Range</code>	The map element range.

6.2.2.1592 #define RICMapFunction( `_MapElement`, ... ) `_MapElement`, \_\_VA\_ARGS\_\_

Output an RIC VarMap function.

**Parameters**

<code>_MapElement</code>	An entry in the varmap function. At least 2 elements are required. See <a href="#">RICMapElement</a> .
--------------------------	--

6.2.2.1593 #define RICOpCircle( `_CopyOptions`, `_Point`, `_Radius` ) 10, 0, 7, 0, (`_CopyOptions`)&0xFF, (`_CopyOptions`)>>8, `_Point`, (`_Radius`)&0xFF, (`_Radius`)>>8

Output an RIC Circle opcode.

## Parameters

<i>_CopyOptions</i>	Circle copy options. See <a href="#">Drawing option constants</a> .
<i>_Point</i>	The circle's center point. See <a href="#">RICImgPoint</a> .
<i>_Radius</i>	The circle's radius.

```
6.2.2.1594 #define RICOpCopyBits( _CopyOptions, _DataAddr, _SrcRect, _DstPoint ) 18, 0, 3, 0, (_CopyOptions)&0xFF,
(_CopyOptions)>>8, (_DataAddr)&0xFF, (_DataAddr)>>8, _SrcRect, _DstPoint
```

Output an RIC CopyBits opcode.

## Parameters

<i>_CopyOptions</i>	CopyBits copy options. See <a href="#">Drawing option constants</a> .
<i>_DataAddr</i>	The address of the sprite from which to copy data.
<i>_SrcRect</i>	The rectangular portion of the sprite to copy. See <a href="#">RICImgRect</a> .
<i>_DstPoint</i>	The LCD coordinate to which to copy the data. See <a href="#">RICImgPoint</a> .

```
6.2.2.1595 #define RICOpDescription( _Options, _Width, _Height ) 8, 0, 0, 0, (_Options)&0xFF, (_Options)>>8, (_Width)&0xFF,
(_Width)>>8, (_Height)&0xFF, (_Height)>>8
```

Output an RIC Description opcode.

## Parameters

<i>_Options</i>	RIC options.
<i>_Width</i>	The total RIC width.
<i>_Height</i>	The total RIC height.

```
6.2.2.1596 #define RICOpEllipse( _CopyOptions, _Point, _RadiusX, _RadiusY ) 12, 0, 9, 0, (_CopyOptions)&0xFF,
(_CopyOptions)>>8, _Point, (_RadiusX)&0xFF, (_RadiusX)>>8, (_RadiusY)&0xFF, (_RadiusY)>>8
```

Output an RIC Ellipse opcode.

## Parameters

<i>_CopyOptions</i>	Ellipse copy options. See <a href="#">Drawing option constants</a> .
<i>_Point</i>	The center of the ellipse. See <a href="#">RICImgPoint</a> .
<i>_RadiusX</i>	The x-axis radius of the ellipse.
<i>_RadiusY</i>	The y-axis radius of the ellipse.

```
6.2.2.1597 #define RICOpLine( _CopyOptions, _Point1, _Point2 ) 12, 0, 5, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point1,
_Point2
```

Output an RIC Line opcode.

## Parameters

<i>_CopyOptions</i>	Line copy options. See <a href="#">Drawing option constants</a> .
<i>_Point1</i>	The starting point of the line. See <a href="#">RICImgPoint</a> .
<i>_Point2</i>	The ending point of the line. See <a href="#">RICImgPoint</a> .

```
6.2.2.1598 #define RICOpNumBox( _CopyOptions, _Point, _Value ) 10, 0, 8, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point,
(_Value)&0xFF, (_Value)>>8
```

Output an RIC NumBox opcode.

#### Parameters

<i>_CopyOptions</i>	NumBox copy options. See <a href="#">Drawing option constants</a> .
<i>_Point</i>	The numbox bottom left corner. See <a href="#">RICImgPoint</a> .
<i>_Value</i>	The number to draw.

```
6.2.2.1599 #define RICOpPixel( _CopyOptions, _Point, _Value ) 10, 0, 4, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point,
(_Value)&0xFF, (_Value)>>8
```

Output an RIC Pixel opcode.

#### Parameters

<i>_CopyOptions</i>	Pixel copy options. See <a href="#">Drawing option constants</a> .
<i>_Point</i>	The pixel coordinate. See <a href="#">RICImgPoint</a> .
<i>_Value</i>	The pixel value (unused).

```
6.2.2.1600 #define RICOpPolygon( _CopyOptions, _Count, _ThePoints ) ((_Count*4)+6)&0xFF, ((_Count*4)+6)>>8, 10, 0,
(_CopyOptions)&0xFF, (_CopyOptions)>>8, (_Count)&0xFF, (_Count)>>8, _ThePoints
```

Output an RIC Polygon opcode.

#### Parameters

<i>_CopyOptions</i>	Polygon copy options. See <a href="#">Drawing option constants</a> .
<i>_Count</i>	The number of points in the polygon.
<i>_ThePoints</i>	The list of polygon points. See <a href="#">RICPolygonPoints</a> .

```
6.2.2.1601 #define RICOpRect( _CopyOptions, _Point, _Width, _Height ) 12, 0, 6, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8,
_Point, (_Width)&0xFF, (_Width)>>8, (_Height)&0xFF, (_Height)>>8
```

Output an RIC Rect opcode.

#### Parameters

<i>_CopyOptions</i>	Rect copy options. See <a href="#">Drawing option constants</a> .
<i>_Point</i>	The rectangle's top left corner. See <a href="#">RICImgPoint</a> .
<i>_Width</i>	The rectangle's width.
<i>_Height</i>	The rectangle's height.

```
6.2.2.1602 #define RICOpSprite( _DataAddr, _Rows, _BytesPerRow, _SpriteData ) ((_Rows*_BytesPerRow)+(((_Rows*_
BytesPerRow)%2)+8)&0xFF, ((_Rows*_BytesPerRow)+(((_Rows*_BytesPerRow)%2)+8))>>8, 1, 0, (_DataAddr)&0xFF,
(_DataAddr)>>8, (_Rows)&0xFF, (_Rows)>>8, (_BytesPerRow)&0xFF, (_BytesPerRow)>>8, _SpriteData
```

Output an RIC Sprite opcode.

#### Parameters

<i>_DataAddr</i>	The address of the sprite.
<i>_Rows</i>	The number of rows of data.
<i>_BytesPerRow</i>	The number of bytes per row.
<i>_SpriteData</i>	The actual sprite data. See <a href="#">RICSpriteData</a> .

6.2.2.1603 #define RICOpVarMap( *\_DataAddr*, *\_MapCount*, *\_MapFunction* ) ((*\_MapCount*\*4)+6)&0xFF, ((*\_MapCount*\*4)+6)>>8, 2, 0, (*\_DataAddr*)&0xFF, (*\_DataAddr*)>>8, (*\_MapCount*)&0xFF, (*\_MapCount*)>>8, *\_MapFunction*

Output an RIC VarMap opcode.

#### Parameters

<i>_DataAddr</i>	The address of the varmap.
<i>_MapCount</i>	The number of points in the function.
<i>_MapFunction</i>	The definition of the varmap function. See <a href="#">RICMapFunction</a> .

6.2.2.1604 #define RICPolygonPoints( *\_pPoint1*, *\_pPoint2*, ... ) *\_pPoint1*, *\_pPoint2*, \_\_VA\_ARGS\_\_

Output RIC polygon points.

#### Parameters

<i>_pPoint1</i>	The first polygon point. See <a href="#">RICImgPoint</a> .
<i>_pPoint2</i>	The second polygon point (at least 3 points are required). See <a href="#">RICImgPoint</a> .

6.2.2.1605 #define RICSpriteData( ... ) \_\_VA\_ARGS\_\_

Output RIC sprite data.

6.2.2.1606 #define ROTATE\_QUEUE 5

VM should rotate queue

6.2.2.1607 #define RotationCountField 14

Rotation counter field.

Contains the current rotation count. Read only. Return the program-relative position counter value for the specified port. Refer to the [UpdateFlagsField](#) description for information about how to use program-relative position counts. Set the [UF\\_UPDATE\\_RESET\\_ROTATION\\_COUNT](#) flag in [UpdateFlagsField](#) to request that the firmware reset the RotationCountField. The sign of RotationCountField indicates the direction of rotation. Positive values indicate forward rotation and negative values indicate reverse rotation. Forward and reverse depend on the orientation of the motor.

6.2.2.1608 #define RunStateField 6

Run state field.

Contains one of the run state constants. Read/write. Use this field to specify the running state of an output. Set the RunStateField to [OUT\\_RUNSTATE\\_RUNNING](#) to enable power to any output. Use [OUT\\_RUNSTATE\\_RAMPUP](#) to enable automatic ramping to a new [PowerField](#) level greater than the current [PowerField](#) level. Use [OUT\\_RUNSTATE\\_RAMPDOWN](#) to enable automatic ramping to a new [PowerField](#) level less than the current [PowerField](#) level. Both the rampup and rampdown bits must be used in conjunction with appropriate [TachoLimitField](#) and [PowerField](#) values. In this case the firmware smoothly increases or decreases the actual power to the new [PowerField](#) level over the total number of degrees of rotation specified in [TachoLimitField](#).

6.2.2.1609 #define SAMPLERATE\_DEFAULT 8000

Default sample rate [sps]

6.2.2.1610 #define SAMPLERATE\_MAX 16000

Max sample rate [sps]

6.2.2.1611 #define SAMPLERATE\_MIN 2000

Min sample rate [sps]

6.2.2.1612 #define ScaledValueField 4

Scaled value field. Contains the current scaled analog sensor value. Read/write.

6.2.2.1613 #define SCHAR\_MAX 127

The maximum value of the signed char type

6.2.2.1614 #define SCHAR\_MIN -128

The minimum value of the signed char type

6.2.2.1615 #define SCOUT\_FXR\_ALARM 2

Alarm special effects

6.2.2.1616 #define SCOUT\_FXR\_BUG 1

Bug special effects

6.2.2.1617 #define SCOUT\_FXR\_NONE 0

No special effects

6.2.2.1618 #define SCOUT\_FXR\_RANDOM 3

Random special effects

6.2.2.1619 #define SCOUT\_FXR\_SCIENCE 4

Science special effects

6.2.2.1620 #define SCOUT\_LIGHT\_OFF 0

Turn off the scout light

6.2.2.1621 #define SCOUT\_LIGHT\_ON 0x80

Turn on the scout light

6.2.2.1622 #define SCOUT\_LR\_AVOID 3

Light rule avoid

6.2.2.1623 #define SCOUT\_LR\_IGNORE 0

Light rule ignore

6.2.2.1624 #define SCOUT\_LR\_OFF\_WHEN 5

Light rule off when

6.2.2.1625 #define SCOUT\_LR\_SEEK\_DARK 2

Light rule seek dark

6.2.2.1626 #define SCOUT\_LR\_SEEK\_LIGHT 1

Light rule seek light

6.2.2.1627 #define SCOUT\_LR\_WAIT\_FOR 4

Light rule wait for

6.2.2.1628 #define SCOUT\_MODE\_POWER 1

Enter power mode

6.2.2.1629 #define SCOUT\_MODE\_STANDALONE 0

Enter stand alone mode

6.2.2.1630 #define SCOUT\_MR\_CIRCLE\_LEFT 4

Motion rule circle left

6.2.2.1631 #define SCOUT\_MR\_CIRCLE\_RIGHT 3

Motion rule circle right

6.2.2.1632 #define SCOUT\_MR\_FORWARD 1

Motion rule forward

6.2.2.1633 #define SCOUT\_MR\_LOOP\_A 5

Motion rule loop A

6.2.2.1634 #define SCOUT\_MR\_LOOP\_AB 7

Motion rule loop A then B

6.2.2.1635 #define SCOUT\_MR\_LOOP\_B 6

Motion rule loop B

6.2.2.1636 #define SCOUT\_MR\_NO\_MOTION 0

Motion rule none

6.2.2.1637 #define SCOUT\_MR\_ZIGZAG 2

Motion rule zigzag

6.2.2.1638 #define SCOUT SNDSET\_ALARM 3

Set sound set to alarm

6.2.2.1639 #define SCOUT SNDSET\_BASIC 1

Set sound set to basic

6.2.2.1640 #define SCOUT SNDSET\_BUG 2

Set sound set to bug

6.2.2.1641 #define SCOUT SNDSET\_NONE 0

Set sound set to none

6.2.2.1642 #define SCOUT SNDSET\_RANDOM 4

Set sound set to random

6.2.2.1643 #define SCOUT SNDSET\_SCIENCE 5

Set sound set to science

6.2.2.1644 #define SCOUT\_SOUND\_1\_BLINK 17

Play the Scout 1 blink sound

6.2.2.1645 #define SCOUT\_SOUND\_2\_BLINK 18

Play the Scout 2 blink sound

6.2.2.1646 #define SCOUT\_SOUND\_COUNTER1 19

Play the Scout counter 1 sound

6.2.2.1647 #define SCOUT\_SOUND\_COUNTER2 20

Play the Scout counter 2 sound

6.2.2.1648 #define SCOUT\_SOUND\_ENTER\_BRIGHT 14

Play the Scout enter bright sound

6.2.2.1649 #define SCOUT\_SOUND\_ENTER\_DARK 16

Play the Scout enter dark sound

6.2.2.1650 #define SCOUT\_SOUND\_ENTER\_NORMAL 15

Play the Scout enter normal sound

6.2.2.1651 #define SCOUT\_SOUND\_ENTERSA 7

Play the Scout enter standalone sound

6.2.2.1652 #define SCOUT\_SOUND\_KEYERROR 8

Play the Scout key error sound

6.2.2.1653 #define SCOUT\_SOUND\_MAIL\_RECEIVED 24

Play the Scout mail received sound

6.2.2.1654 #define SCOUT\_SOUND\_NONE 9

Play the Scout none sound

6.2.2.1655 #define SCOUT\_SOUND\_REMOTE 6

Play the Scout remote sound

6.2.2.1656 #define SCOUT\_SOUND\_SPECIAL1 25

Play the Scout special 1 sound

6.2.2.1657 #define SCOUT\_SOUND\_SPECIAL2 26

Play the Scout special 2 sound

6.2.2.1658 #define SCOUT\_SOUND\_SPECIAL3 27

Play the Scout special 3 sound

6.2.2.1659 #define SCOUT\_SOUND\_TIMER1 21

Play the Scout timer 1 sound

6.2.2.1660 #define SCOUT\_SOUND\_TIMER2 22

Play the Scout timer 2 sound

6.2.2.1661 #define SCOUT\_SOUND\_TIMER3 23

Play the Scout timer 3 sound

6.2.2.1662 #define SCOUT\_SOUND\_TOUCH1\_PRES 10

Play the Scout touch 1 pressed sound

6.2.2.1663 #define SCOUT\_SOUND\_TOUCH1\_REL 11

Play the Scout touch 1 released sound

6.2.2.1664 #define SCOUT\_SOUND\_TOUCH2\_PRES 12

Play the Scout touch 2 pressed sound

6.2.2.1665 #define SCOUT\_SOUND\_TOUCH2\_REL 13

Play the Scout touch 2 released sound

6.2.2.1666 #define SCOUT\_TGS\_LONG 2

Transmit level long

6.2.2.1667 #define SCOUT\_TGS\_MEDIUM 1

Transmit level medium

6.2.2.1668 #define SCOUT\_TGS\_SHORT 0

Transmit level short

6.2.2.1669 #define SCOUT\_TR\_AVOID 2

Touch rule avoid

6.2.2.1670 #define SCOUT\_TR\_IGNORE 0

Touch rule ignore

6.2.2.1671 #define SCOUT\_TR\_OFF\_WHEN 4

Touch rule off when

6.2.2.1672 #define SCOUT\_TR\_REVERSE 1

Touch rule reverse

6.2.2.1673 #define SCOUT\_TR\_WAIT\_FOR 3

Touch rule wait for

6.2.2.1674 #define SCREEN\_BACKGROUND 0

Entire screen

6.2.2.1675 #define SCREEN\_LARGE 1

Entire screen except status line

6.2.2.1676 #define SCREEN\_MODE\_CLEAR 0x01

Clear the screen

#### See Also

[SetScreenMode\(\)](#)

6.2.2.1677 #define SCREEN\_MODE\_RESTORE 0x00

Restore the screen

**See Also**

[SetScreenMode\(\)](#)

**6.2.2.1678 #define SCREEN\_SMALL 2**

Screen between menu icons and status line

**6.2.2.1679 #define SCREENS 3**

The number of screen bits

**6.2.2.1680 #define SEC\_1 1000**

1 second

**6.2.2.1681 #define SEC\_10 10000**

10 seconds

**6.2.2.1682 #define SEC\_15 15000**

15 seconds

**6.2.2.1683 #define SEC\_2 2000**

2 seconds

**6.2.2.1684 #define SEC\_20 20000**

20 seconds

**6.2.2.1685 #define SEC\_3 3000**

3 seconds

**6.2.2.1686 #define SEC\_30 30000**

30 seconds

**6.2.2.1687 #define SEC\_4 4000**

4 seconds

**6.2.2.1688 #define SEC\_5 5000**

5 seconds

**6.2.2.1689 #define SEC\_6 6000**

6 seconds

**6.2.2.1690 #define SEC\_7 7000**

7 seconds

6.2.2.1691 #define SEC\_8 8000

8 seconds

6.2.2.1692 #define SEC\_9 9000

9 seconds

6.2.2.1693 #define SetScreenMode 19

Set the screen mode

6.2.2.1694 #define SetSleepTimeoutVal 46

Set the NXT sleep timeout value

6.2.2.1695 #define SHRT\_MAX 32767

The maximum value of the short type

6.2.2.1696 #define SHRT\_MIN -32768

The minimum value of the short type

6.2.2.1697 #define SIZE\_OF\_BDADDR 7

Size of Bluetooth Address

6.2.2.1698 #define SIZE\_OF\_BRICK\_NAME 8

Size of NXT Brick name

6.2.2.1699 #define SIZE\_OF\_BT\_CONNECT\_TABLE 4

Size of Bluetooth connection table – Index 0 is always incoming connection

6.2.2.1700 #define SIZE\_OF\_BT\_DEVICE\_TABLE 30

Size of Bluetooth device table

6.2.2.1701 #define SIZE\_OF\_BT\_NAME 16

Size of Bluetooth name

6.2.2.1702 #define SIZE\_OF\_BT\_PINCODE 16

Size of Bluetooth PIN

6.2.2.1703 #define SIZE\_OF\_BTBUF 128

Size of Bluetooth buffer

6.2.2.1704 #define SIZE\_OF\_CLASS\_OF\_DEVICE 4

Size of class of device

6.2.2.1705 #define SIZE\_OF\_HSBUF 128

Size of High Speed Port 4 buffer

6.2.2.1706 #define SIZE\_OF\_USBBUF 64

Size of USB Buffer in bytes

6.2.2.1707 #define SIZE\_OF\_USBDATA 62

Size of USB Buffer available for data

6.2.2.1708 #define SOUND\_CLICK 0

Play the standard key click sound

6.2.2.1709 #define SOUND\_DOUBLE\_BEEP 1

Play the standard double beep sound

6.2.2.1710 #define SOUND\_DOWN 2

Play the standard sweep down sound

6.2.2.1711 #define SOUND\_FAST\_UP 5

Play the standard fast up sound

6.2.2.1712 #define SOUND\_FLAGS\_IDLE 0x00

R - Sound is idle

6.2.2.1713 #define SOUND\_FLAGS\_RUNNING 0x02

R - Currently processing a tone or file

6.2.2.1714 #define SOUND\_FLAGS\_UPDATE 0x01

W - Make changes take effect

6.2.2.1715 #define SOUND\_LOW\_BEEP 4

Play the standard low beep sound

6.2.2.1716 #define SOUND\_MODE\_LOOP 0x01

W - Play file until writing SOUND\_STATE\_STOP into SoundState

6.2.2.1717 #define SOUND\_MODE\_ONCE 0x00

W - Only play file once

6.2.2.1718 #define SOUND\_MODE\_TONE 0x02

W - Play tone specified in Frequency for Duration ms

6.2.2.1719 #define SOUND\_STATE\_FILE 0x02

R - Processing a file of sound/melody data

6.2.2.1720 #define SOUND\_STATE\_IDLE 0x00

R - Idle, ready for start sound (SOUND\_UPDATE)

6.2.2.1721 #define SOUND\_STATE\_STOP 0x04

W - Stop sound immediately and close hardware

6.2.2.1722 #define SOUND\_STATE\_TONE 0x03

R - Processing a play tone request

6.2.2.1723 #define SOUND\_UP 3

Play the standard sweep up sound

6.2.2.1724 #define SoundGetState 11

Get the current sound module state

6.2.2.1725 #define SoundModuleID 0x00080001

The sound module ID

6.2.2.1726 #define SoundModuleName "Sound.mod"

The sound module name

6.2.2.1727 #define SoundOffsetDuration 2

RW - Tone duration [mS] (2 bytes)

6.2.2.1728 #define SoundOffsetFlags 26

RW - Play flag - described above (1 byte) [SoundFlags constants](#)

6.2.2.1729 #define SoundOffsetFreq 0

RW - Tone frequency [Hz] (2 bytes)

6.2.2.1730 #define SoundOffsetMode 28

RW - Play mode - described above (1 byte) [SoundMode constants](#)

6.2.2.1731 #define SoundOffsetSampleRate 4

RW - Sound file sample rate [2000..16000] (2 bytes)

6.2.2.1732 #define SoundOffsetSoundFilename 6

RW - Sound/melody filename (20 bytes)

6.2.2.1733 #define SoundOffsetState 27

RW - Play state - described above (1 byte) [SoundState constants](#)

6.2.2.1734 #define SoundOffsetVolume 29

RW - Sound/melody volume [0..4] 0 = off (1 byte)

6.2.2.1735 #define SoundPlayFile 9

Play a sound or melody file

6.2.2.1736 #define SoundPlayTone 10

Play a simple tone with the specified frequency and duration

6.2.2.1737 #define SoundSetState 12

Set the sound module state

6.2.2.1738 #define SPECIALS 5

The number of special bit values

6.2.2.1739 #define STAT\_COMM\_PENDING 32

Pending setup operation in progress

6.2.2.1740 #define STAT\_MSG\_EMPTY\_MAILBOX 64

Specified mailbox contains no new messages

6.2.2.1741 #define STATUSICON\_BATTERY 3

Battery status icon collection

6.2.2.1742 #define STATUSICON\_BLUETOOTH 0

BlueTooth status icon collection

6.2.2.1743 #define STATUSICON\_USB 1

USB status icon collection

6.2.2.1744 #define STATUSICON\_VM 2

VM status icon collection

6.2.2.1745 #define STATUSICONS 4

The number of status icons

6.2.2.1746 #define STATUSTEXT 1

Status text (BT name)

6.2.2.1747 #define STEPICON\_1 0

Left most step icon

6.2.2.1748 #define STEPICON\_2 1

Step icon #2

6.2.2.1749 #define STEPICON\_3 2

Step icon #3

6.2.2.1750 #define STEPICON\_4 3

Step icon #4

6.2.2.1751 #define STEPICON\_5 4

Right most step icon

6.2.2.1752 #define STEPICONS 5

The number of step icons

6.2.2.1753 #define STEPLINE 3

Step collection lines

6.2.2.1754 #define STOP\_REQ 4

VM should stop executing program

6.2.2.1755 #define STROBE\_READ 0x10

Access read pin (RD)

6.2.2.1756 #define STROBE\_S0 0x01

Access strobe 0 pin (S0)

6.2.2.1757 #define STROBE\_S1 0x02

Access strobe 1 pin (S1)

6.2.2.1758 #define STROBE\_S2 0x04

Access strobe 2 pin (S2)

6.2.2.1759 #define STROBE\_S3 0x08

Access strobe 3 pin (S3)

6.2.2.1760 #define STROBE\_WRITE 0x20

Access write pin (WR)

**6.2.2.1761 #define TachoCountField 4**

Internal tachometer count field.

Contains the current internal tachometer count. Read only. Return the internal position counter value for the specified output. The internal count is reset automatically when a new goal is set using the [TachoLimitField](#) and the [UF\\_UPDATE\\_TACHO\\_LIMIT](#) flag. Set the [UF\\_UPDATE\\_RESET\\_COUNT](#) flag in [UpdateFlagsField](#) to reset TachoCountField and cancel any [TachoLimitField](#). The sign of TachoCountField indicates the motor rotation direction.

**6.2.2.1762 #define TachoLimitField 5**

Tachometer limit field.

Contains the current tachometer limit. Read/write. Specify the number of degrees the motor should rotate. Use [UF\\_UPDATE\\_TACHO\\_LIMIT](#) with the [UpdateFlagsField](#) field to commit changes to the TachoLimitField. The value of this field is a relative distance from the current motor position at the moment when the [UF\\_UPDATE\\_TACHO\\_LIMIT](#) flag is processed.

**6.2.2.1763 #define TEMP\_FQ\_1 0x00**

Set fault queue to 1 fault before alert

**6.2.2.1764 #define TEMP\_FQ\_2 0x08**

Set fault queue to 2 faults before alert

**6.2.2.1765 #define TEMP\_FQ\_4 0x10**

Set fault queue to 4 faults before alert

**6.2.2.1766 #define TEMP\_FQ\_6 0x18**

Set fault queue to 6 faults before alert

**6.2.2.1767 #define TEMP\_OS\_ONESHOT 0x80**

Set the sensor into oneshot mode. When the device is in shutdown mode this will start a single temperature conversion. The device returns to shutdown mode when it completes.

**6.2.2.1768 #define TEMP\_POL\_HIGH 0x04**

Set polarity of ALERT pin to be active HIGH

**6.2.2.1769 #define TEMP\_POL\_LOW 0x00**

Set polarity of ALERT pin to be active LOW

**6.2.2.1770 #define TEMP\_REG\_CONFIG 0x01**

The register for reading/writing sensor configuration values

**6.2.2.1771 #define TEMP\_REG\_TEMP 0x00**

The register where temperature values can be read

**6.2.2.1772 #define TEMP\_REG\_THIGH 0x03**

The register for reading/writing a user-defined high temperature limit

6.2.2.1773 #define TEMP\_REG\_TLOW 0x02

The register for reading/writing a user-defined low temperature limit

6.2.2.1774 #define TEMP\_RES\_10BIT 0x20

Set the temperature conversion resolution to 10 bit

6.2.2.1775 #define TEMP\_RES\_11BIT 0x40

Set the temperature conversion resolution to 11 bit

6.2.2.1776 #define TEMP\_RES\_12BIT 0x60

Set the temperature conversion resolution to 12 bit

6.2.2.1777 #define TEMP\_RES\_9BIT 0x00

Set the temperature conversion resolution to 9 bit

6.2.2.1778 #define TEMP\_SD\_CONTINUOUS 0x00

Set the sensor mode to continuous

6.2.2.1779 #define TEMP\_SD\_SHUTDOWN 0x01

Set the sensor mode to shutdown. The device will shut down after the current conversion is completed.

6.2.2.1780 #define TEMP\_TM\_COMPARATOR 0x00

Set the thermostat mode to comparator

6.2.2.1781 #define TEMP\_TM\_INTERRUPT 0x02

Set the thermostat mode to interrupt

6.2.2.1782 #define TEXTLINE\_1 0

Text line 1

6.2.2.1783 #define TEXTLINE\_2 1

Text line 2

6.2.2.1784 #define TEXTLINE\_3 2

Text line 3

6.2.2.1785 #define TEXTLINE\_4 3

Text line 4

6.2.2.1786 #define TEXTLINE\_5 4

Text line 5

6.2.2.1787 #define TEXTLINE\_6 5

Text line 6

6.2.2.1788 #define TEXTLINE\_7 6

Text line 7

6.2.2.1789 #define TEXTLINE\_8 7

Text line 8

6.2.2.1790 #define TEXTLINES 8

The number of text lines on the LCD

6.2.2.1791 #define TIMES\_UP 6

VM time is up

6.2.2.1792 #define TONE\_A3 220

Third octave A

6.2.2.1793 #define TONE\_A4 440

Fourth octave A

6.2.2.1794 #define TONE\_A5 880

Fifth octave A

6.2.2.1795 #define TONE\_A6 1760

Sixth octave A

6.2.2.1796 #define TONE\_A7 3520

Seventh octave A

6.2.2.1797 #define TONE\_AS3 233

Third octave A sharp

6.2.2.1798 #define TONE\_AS4 466

Fourth octave A sharp

6.2.2.1799 #define TONE\_AS5 932

Fifth octave A sharp

6.2.2.1800 #define TONE\_AS6 1865

Sixth octave A sharp

6.2.2.1801 #define TONE\_AS7 3729

Seventh octave A sharp

6.2.2.1802 #define TONE\_B3 247

Third octave B

6.2.2.1803 #define TONE\_B4 494

Fourth octave B

6.2.2.1804 #define TONE\_B5 988

Fifth octave B

6.2.2.1805 #define TONE\_B6 1976

Sixth octave B

6.2.2.1806 #define TONE\_B7 3951

Seventh octave B

6.2.2.1807 #define TONE\_C3 131

Third octave C

6.2.2.1808 #define TONE\_C4 262

Fourth octave C

6.2.2.1809 #define TONE\_C5 523

Fifth octave C

6.2.2.1810 #define TONE\_C6 1047

Sixth octave C

6.2.2.1811 #define TONE\_C7 2093

Seventh octave C

6.2.2.1812 #define TONE\_CS3 139

Third octave C sharp

6.2.2.1813 #define TONE\_CS4 277

Fourth octave C sharp

6.2.2.1814 #define TONE\_CS5 554

Fifth octave C sharp

6.2.2.1815 #define TONE\_CS6 1109

Sixth octave C sharp

6.2.2.1816 #define TONE\_CS7 2217

Seventh octave C sharp

6.2.2.1817 #define TONE\_D3 147

Third octave D

6.2.2.1818 #define TONE\_D4 294

Fourth octave D

6.2.2.1819 #define TONE\_D5 587

Fifth octave D

6.2.2.1820 #define TONE\_D6 1175

Sixth octave D

6.2.2.1821 #define TONE\_D7 2349

Seventh octave D

6.2.2.1822 #define TONE\_DS3 156

Third octave D sharp

6.2.2.1823 #define TONE\_DS4 311

Fourth octave D sharp

6.2.2.1824 #define TONE\_DS5 622

Fifth octave D sharp

6.2.2.1825 #define TONE\_DS6 1245

Sixth octave D sharp

6.2.2.1826 #define TONE\_DS7 2489

Seventh octave D sharp

6.2.2.1827 #define TONE\_E3 165

Third octave E

6.2.2.1828 #define TONE\_E4 330

Fourth octave E

6.2.2.1829 #define TONE\_E5 659

Fifth octave E

6.2.2.1830 #define TONE\_E6 1319

Sixth octave E

6.2.2.1831 #define TONE\_E7 2637

Seventh octave E

6.2.2.1832 #define TONE\_F3 175

Third octave F

6.2.2.1833 #define TONE\_F4 349

Fourth octave F

6.2.2.1834 #define TONE\_F5 698

Fifth octave F

6.2.2.1835 #define TONE\_F6 1397

Sixth octave F

6.2.2.1836 #define TONE\_F7 2794

Seventh octave F

6.2.2.1837 #define TONE\_FS3 185

Third octave F sharp

6.2.2.1838 #define TONE\_FS4 370

Fourth octave F sharp

6.2.2.1839 #define TONE\_FS5 740

Fifth octave F sharp

6.2.2.1840 #define TONE\_FS6 1480

Sixth octave F sharp

6.2.2.1841 #define TONE\_FS7 2960

Seventh octave F sharp

6.2.2.1842 #define TONE\_G3 196

Third octave G

6.2.2.1843 #define TONE\_G4 392

Fourth octave G

6.2.2.1844 #define TONE\_G5 784

Fifth octave G

6.2.2.1845 #define TONE\_G6 1568

Sixth octave G

6.2.2.1846 #define TONE\_G7 3136

Seventh octave G

6.2.2.1847 #define TONE\_GS3 208

Third octave G sharp

6.2.2.1848 #define TONE\_GS4 415

Fourth octave G sharp

6.2.2.1849 #define TONE\_GS5 831

Fifth octave G sharp

6.2.2.1850 #define TONE\_GS6 1661

Sixth octave G sharp

6.2.2.1851 #define TONE\_GS7 3322

Seventh octave G sharp

6.2.2.1852 #define TOPLINE 4

Top status underline

6.2.2.1853 #define TRAIN\_CHANNEL\_1 0

IR Train channel 1

6.2.2.1854 #define TRAIN\_CHANNEL\_2 1

IR Train channel 2

6.2.2.1855 #define TRAIN\_CHANNEL\_3 2

IR Train channel 3

6.2.2.1856 #define TRAIN\_CHANNEL\_ALL 3

IR Train channel all

6.2.2.1857 #define TRAIN\_FUNC\_DECR\_SPEED 2

PF/IR Train function decrement speed

6.2.2.1858 #define TRAIN\_FUNC\_INCR\_SPEED 1

PF/IR Train function increment speed

6.2.2.1859 #define TRAIN\_FUNC\_STOP 0

PF/IR Train function stop

6.2.2.1860 #define TRAIN\_FUNC\_TOGGLE\_LIGHT 4

PF/IR Train function toggle light

6.2.2.1861 #define TRUE 1

A true value

6.2.2.1862 #define TurnRatioField 7

Turn ratio field.

Contains the current turn ratio. Only applicable when synchronizing multiple motors. Read/write. Use this field to specify a proportional turning ratio. This field must be used in conjunction with other field values: [OutputModeField](#) must include [OUT\\_MODE\\_MOTORON](#) and [OUT\\_MODE\\_REGULATED](#), [RegModeField](#) must be set to [OUT\\_REGMODE\\_SYNC](#), [RunStateField](#) must not be [OUT\\_RUNSTATE\\_IDLE](#), and [PowerField](#) must be non-zero. There are only three valid combinations of left and right motors for use with TurnRatioField: [OUT\\_AB](#), [OUT\\_BC](#), and [OUT\\_AC](#). In each of these three options the first motor listed is considered to be the left motor and the second motor is the right motor, regardless of the physical configuration of the robot. Negative turn ratio values shift power toward the left motor while positive values shift power toward the right motor. An absolute value of 50 usually results in one motor stopping. An absolute value of 100 usually results in two motors turning in opposite directions at equal power.

6.2.2.1863 #define TypeField 0

Type field. Contains one of the sensor type constants. Read/write.

6.2.2.1864 #define UCHAR\_MAX 255

The maximum value of the unsigned char type

6.2.2.1865 #define UF\_PENDING\_UPDATES 0x80

Are there any pending motor updates?

6.2.2.1866 #define UF\_UPDATE\_MODE 0x01

Commits changes to the [OutputModeField](#) output property

6.2.2.1867 #define UF\_UPDATE\_PID\_VALUES 0x10

Commits changes to the PID motor regulation properties

6.2.2.1868 #define UF\_UPDATE\_RESET\_BLOCK\_COUNT 0x20

Resets the NXT-G block-relative rotation counter

6.2.2.1869 #define UF\_UPDATE\_RESET\_COUNT 0x08

Resets all rotation counters, cancels the current goal, and resets the rotation error-correction system

6.2.2.1870 #define UF\_UPDATE\_RESET\_ROTATION\_COUNT 0x40

Resets the program-relative (user) rotation counter

6.2.2.1871 #define UF\_UPDATE\_SPEED 0x02

Commits changes to the [PowerField](#) output property

6.2.2.1872 #define UF\_UPDATE\_TACHO\_LIMIT 0x04

Commits changes to the [TachoLimitField](#) output property

6.2.2.1873 #define UI\_BT\_CONNECT\_REQUEST 0x40

RW - BT get connect accept in progress

6.2.2.1874 #define UI\_BT\_ERROR\_ATTENTION 0x08

W - BT error attention

6.2.2.1875 #define UI\_BT\_PIN\_REQUEST 0x80

RW - BT get pin code

6.2.2.1876 #define UI\_BT\_STATE\_CONNECTED 0x02

RW - BT connected to something

6.2.2.1877 #define UI\_BT\_STATE\_OFF 0x04

RW - BT power off

6.2.2.1878 #define UI\_BT\_STATE\_VISIBLE 0x01

RW - BT visible

6.2.2.1879 #define UI\_BUTTON\_ENTER 2

W - Insert enter button

6.2.2.1880 #define UI\_BUTTON\_EXIT 4

W - Insert exit button

6.2.2.1881 #define UI\_BUTTON\_LEFT 1

W - Insert left arrow button

6.2.2.1882 #define UI\_BUTTON\_NONE 0

R - Button inserted are executed

6.2.2.1883 #define UI\_BUTTON\_RIGHT 3

W - Insert right arrow button

6.2.2.1884 #define UI\_FLAGS\_BUSY 0x40

R - UI busy running or datalogging (popup disabled)

6.2.2.1885 #define UI\_FLAGS\_DISABLE\_EXIT 0x04

RW - Disable exit button

6.2.2.1886 #define UI\_FLAGS\_DISABLE\_LEFT\_RIGHT\_ENTER 0x02

RW - Disable left, right and enter button

6.2.2.1887 #define UI\_FLAGS\_ENABLE\_STATUS\_UPDATE 0x80

W - Enable status line to be updated

6.2.2.1888 #define UI\_FLAGS\_EXECUTE\_LMS\_FILE 0x20

W - Execute LMS file in "LMSfilename" (Try It)

6.2.2.1889 #define UI\_FLAGS\_REDRAW\_STATUS 0x08

W - Redraw entire status line

6.2.2.1890 #define UI\_FLAGS\_RESET\_SLEEP\_TIMER 0x10

W - Reset sleep timeout timer

6.2.2.1891 #define UI\_FLAGS\_UPDATE 0x01

W - Make changes take effect

6.2.2.1892 #define UI\_STATE\_BT\_ERROR 16

R - BT error

6.2.2.1893 #define UI\_STATE\_CONNECT\_REQUEST 12

RW - Request for connection accept

6.2.2.1894 #define UI\_STATE\_DRAW\_MENU 6

RW - Execute function and draw menu icons

6.2.2.1895 #define UI\_STATE\_ENTER\_PRESSED 10

RW - Load selected function and next menu id

6.2.2.1896 #define UI\_STATE\_EXECUTE\_FILE 13

RW - Execute file in "LMSfilename"

6.2.2.1897 #define UI\_STATE\_EXECUTING\_FILE 14

R - Executing file in "LMSfilename"

6.2.2.1898 #define UI\_STATE\_EXIT\_PRESSED 11

RW - Load selected function and next menu id

6.2.2.1899 #define UI\_STATE\_INIT\_DISPLAY 0

RW - Init display and load font, menu etc.

6.2.2.1900 #define UI\_STATE\_INIT\_INTRO 2

R - Display intro

6.2.2.1901 #define UI\_STATE\_INIT\_LOW\_BATTERY 1

R - Low battery voltage at power on

6.2.2.1902 #define UI\_STATE\_INIT\_MENU 4

RW - Init menu system

6.2.2.1903 #define UI\_STATE\_INIT\_WAIT 3

RW - Wait for initialization end

6.2.2.1904 #define UI\_STATE\_LEFT\_PRESSED 8

RW - Load selected function and next menu id

6.2.2.1905 #define UI\_STATE\_LOW\_BATTERY 15

R - Low battery at runtime

6.2.2.1906 #define UI\_STATE\_NEXT\_MENU 5

RW - Next menu icons ready for drawing

6.2.2.1907 #define UI\_STATE\_RIGHT\_PRESSED 9

RW - Load selected function and next menu id

6.2.2.1908 #define UI\_STATE\_TEST\_BUTTONS 7

RW - Wait for buttons to be pressed

6.2.2.1909 #define UI\_VM\_IDLE 0

VM\_IDLE: Just sitting around. Request to run program will lead to ONE of the VM\_RUN\* states.

6.2.2.1910 #define UI\_VM\_RESET1 4

VM\_RESET1: Initialize state variables and some I/O devices – executed when programs end

6.2.2.1911 #define UI\_VM\_RESET2 5

VM\_RESET2: Final clean up and return to IDLE

6.2.2.1912 #define UI\_VM\_RUN\_FREE 1

VM\_RUN\_FREE: Attempt to run as many instructions as possible within our timeslice

6.2.2.1913 #define UI\_VM\_RUN\_PAUSE 3

VM\_RUN\_PAUSE: Program still "active", but someone has asked us to pause

6.2.2.1914 #define UI\_VM\_RUN\_SINGLE 2

VM\_RUN\_SINGLE: Run exactly one instruction per timeslice

6.2.2.1915 #define UIModuleID 0x000C0001

The Ui module ID

6.2.2.1916 #define UIModuleName "Ui.mod"

The Ui module name

6.2.2.1917 #define UINT\_MAX 65535

The maximum value of the unsigned int type

6.2.2.1918 #define UIOffsetAbortFlag 40

RW - Long Abort (true == use long press to abort) (1 byte)

6.2.2.1919 #define UIOffsetBatteryState 30

W - Battery state (0..4 capacity) (1 byte)

6.2.2.1920 #define UIOffsetBatteryVoltage 4

R - Battery voltage in millivolts (2 bytes)

6.2.2.1921 #define UIOffsetBluetoothState 31

W - Bluetooth state (0=on, 1=visible, 2=conn, 3=conn.visible, 4=off, 5=dfu) (1 byte)

6.2.2.1922 #define UIOffsetButton 28

RW - Insert button (buttons enumerated above) (1 byte)

6.2.2.1923 #define UIOffsetError 37

W - Error code (1 byte)

6.2.2.1924 #define UIOffsetFlags 26

RW - Update command flags (flags enumerated above) (1 byte)

6.2.2.1925 #define UIOffsetForceOff 39

W - Force off (> 0 = off) (1 byte)

6.2.2.1926 #define UIOffsetLMSfilename 6

W - LMS filename to execute (Try It) (20 bytes)

6.2.2.1927 #define UIOffsetOBPPointer 38

W - Actual OBP step (0 - 4) (1 byte)

6.2.2.1928 #define UIOffsetPMenu 0

W - Pointer to menu file (4 bytes)

6.2.2.1929 #define UIOffsetRechargeable 35

R - Rechargeable battery (0 = no, 1 = yes) (1 byte)

6.2.2.1930 #define UIOffsetRunState 29

W - VM Run state (0 = stopped, 1 = running) (1 byte)

6.2.2.1931 #define UIOffsetSleepTimeout 33

RW - Sleep timeout time (min) (1 byte)

6.2.2.1932 #define UIOffsetSleepTimer 34

RW - Sleep timer (min) (1 byte)

6.2.2.1933 #define UIOffsetState 27

RW - UI state (states enumerated above) (1 byte)

6.2.2.1934 #define UIOffsetUsbState 32

W - Usb state (0=disconnected, 1=connected, 2=working) (1 byte)

6.2.2.1935 #define UIOffsetVolume 36

RW - Volume used in UI (0 - 4) (1 byte)

6.2.2.1936 #define ULONG\_MAX 4294967295

The maximum value of the unsigned long type

6.2.2.1937 #define UpdateCalibCacheInfo 43

Update sensor calibration cache information

6.2.2.1938 #define UpdateFlagsField 0

Update flags field.

Contains a combination of the update flag constants. Read/write. Use [UF\\_UPDATE\\_MODE](#), [UF\\_UPDATE\\_SPEED](#), [UF\\_UPDATE\\_TACHO\\_LIMIT](#), and [UF\\_UPDATE\\_PID\\_VALUES](#) along with other fields to commit changes to the state

of outputs. Set the appropriate flags after setting one or more of the output fields in order for the changes to actually go into affect.

6.2.2.1939 #define US\_CMD\_CONTINUOUS 0x02

Command to put the ultrasonic sensor into continuous polling mode (default)

6.2.2.1940 #define US\_CMD\_EVENTCAPTURE 0x03

Command to put the ultrasonic sensor into event capture mode

6.2.2.1941 #define US\_CMD\_OFF 0x00

Command to turn off the ultrasonic sensor

6.2.2.1942 #define US\_CMD\_SINGLESHOT 0x01

Command to put the ultrasonic sensor into single shot mode

6.2.2.1943 #define US\_CMD\_WARMRESET 0x04

Command to warm reset the ultrasonic sensor

6.2.2.1944 #define US\_REG\_ACTUAL\_ZERO 0x50

The register address used to store the actual zero value

6.2.2.1945 #define US\_REG\_CM\_INTERVAL 0x40

The register address used to store the CM interval

6.2.2.1946 #define US\_REG\_FACTORY\_ACTUAL\_ZERO 0x11

The register address containing the factory setting for the actual zero value

6.2.2.1947 #define US\_REG\_FACTORY\_SCALE\_DIVISOR 0x13

The register address containing the factory setting for the scale divisor value

6.2.2.1948 #define US\_REG\_FACTORY\_SCALE\_FACTOR 0x12

The register address containing the factory setting for the scale factor value

6.2.2.1949 #define US\_REG\_MEASUREMENT\_UNITS 0x14

The register address containing the measurement units (degrees C or F)

6.2.2.1950 #define US\_REG\_SCALE\_DIVISOR 0x52

The register address used to store the scale divisor value

6.2.2.1951 #define US\_REG\_SCALE\_FACTOR 0x51

The register address used to store the scale factor value

6.2.2.1952 #define USB\_CMD\_READY 0x01

A constant representing usb direct command

6.2.2.1953 #define USB\_PROTOCOL\_OVERHEAD 2

Size of USB Overhead in bytes – Command type byte + Command

6.2.2.1954 #define USHRT\_MAX 65535

The maximum value of the unsigned short type

6.2.2.1955 #define VT\_A1\_FLOAT 0x1A

Variable type 1d array of float

6.2.2.1956 #define VT\_A1\_SBYTE 0x12

Variable type 1d array of signed byte

6.2.2.1957 #define VT\_A1\_SLONG 0x16

Variable type 1d array of signed long

6.2.2.1958 #define VT\_A1\_STRUCT 0x17

Variable type 1d array of structure

6.2.2.1959 #define VT\_A1\_SWORD 0x14

Variable type 1d array of signed word

6.2.2.1960 #define VT\_A1\_UBYTE 0x11

Variable type 1d array of unsigned byte

6.2.2.1961 #define VT\_A1 ULONG 0x15

Variable type 1d array of unsigned long

6.2.2.1962 #define VT\_A1\_UWORD 0x13

Variable type 1d array of unsigned word

6.2.2.1963 #define VT\_A2\_FLOAT 0x2A

Variable type 2d array of float

6.2.2.1964 #define VT\_A2\_SBYTE 0x22

Variable type 2d array of signed byte

6.2.2.1965 #define VT\_A2\_SLONG 0x26

Variable type 2d array of signed long

6.2.2.1966 #define VT\_A2\_STRUCT 0x27

Variable type 2d array of structure

6.2.2.1967 #define VT\_A2\_SWORD 0x24

Variable type 2d array of signed word

6.2.2.1968 #define VT\_A2\_UBYTE 0x21

Variable type 2d array of unsigned byte

6.2.2.1969 #define VT\_A2 ULONG 0x25

Variable type 2d array of unsigned long

6.2.2.1970 #define VT\_A2\_UWORD 0x23

Variable type 2d array of unsigned word

6.2.2.1971 #define VT\_ARRAY\_MASK 0xF0

Variable type array mask

6.2.2.1972 #define VT\_FLOAT 0x0A

Variable type float

6.2.2.1973 #define VT\_MUTEX 0x09

Variable type mutex

6.2.2.1974 #define VT\_SBYTE 0x02

Variable type signed byte

6.2.2.1975 #define VT\_SLONG 0x06

Variable type signed long

6.2.2.1976 #define VT\_STRUCT 0x08

Variable type structure

6.2.2.1977 #define VT\_SWORD 0x04

Variable type signed word

6.2.2.1978 #define VT\_UBYTE 0x01

Variable type unsigned byte

6.2.2.1979 #define VT ULONG 0x05

Variable type unsigned long

6.2.2.1980 #define VT\_UWORD 0x03

Variable type unsigned word

6.2.2.1981 #define WriteSemData 41

Write motor semaphore data

6.2.2.1982 #define XG1300L\_REG\_2G 0x61

Select +/- 2G accelerometer range.

6.2.2.1983 #define XG1300L\_REG\_4G 0x62

Select +/- 4G accelerometer range.

6.2.2.1984 #define XG1300L\_REG\_8G 0x63

Select +/- 8G accelerometer range.

6.2.2.1985 #define XG1300L\_REG\_ANGLE 0x42

Read accumulated angle (2 bytes little endian) in 1/100s of degrees.

6.2.2.1986 #define XG1300L\_REG\_RESET 0x60

Reset the XG1300L device.

6.2.2.1987 #define XG1300L\_REG\_TURNRATE 0x44

Read rate of turn (2 bytes little endian) in 1/100s of degrees/second.

6.2.2.1988 #define XG1300L\_REG\_XAXIS 0x46

Read x-axis acceleration (2 bytes little endian) in m/s<sup>2</sup> scaled by 100/ACC\_RANGE\*2, where ACC\_RANGE is 2, 4, or 8.

6.2.2.1989 #define XG1300L\_REG\_YAXIS 0x48

Read y-axis acceleration (2 bytes little endian) in m/s<sup>2</sup> scaled by 100/ACC\_RANGE\*2, where ACC\_RANGE is 2, 4, or 8.

6.2.2.1990 #define XG1300L\_REG\_ZAXIS 0x4A

Read z-axis acceleration (2 bytes little endian) in m/s<sup>2</sup> scaled by 100/ACC\_RANGE\*2, where ACC\_RANGE is 2, 4, or 8.

6.2.2.1991 #define XG1300L\_SCALE\_2G 0x01

Select +/- 2G accelerometer range.

6.2.2.1992 #define XG1300L\_SCALE\_4G 0x02

Select +/- 4G accelerometer range.

6.2.2.1993 #define XG1300L\_SCALE\_8G 0x04

Select +/- 8G accelerometer range.

## Index

A simple 3D graphics library, [206](#)  
glAddToAngleX, [207](#)  
glAddToAngleY, [207](#)  
glAddToAngleZ, [207](#)  
glAddVertex, [207](#)  
glBegin, [208](#)  
glBeginObject, [208](#)  
glBeginRender, [208](#)  
glBox, [208](#)  
glCallObject, [208](#)  
glCos32768, [209](#)  
glCube, [209](#)  
glEnd, [209](#)  
glEndObject, [209](#)  
glFinishRender, [209](#)  
glInit, [209](#)  
glObjectAction, [209](#)  
glPyramid, [210](#)  
glSet, [210](#)  
glSetAngleX, [210](#)  
glSetAngleY, [210](#)  
glSetAngleZ, [210](#)  
glSin32768, [211](#)  
ACCL\_CMD\_X\_CAL  
    MindSensors ACCL-Nx constants, [615](#)  
    NBCCCommon.h, [735](#)  
ACCL\_CMD\_Y\_CAL  
    MindSensors ACCL-Nx constants, [615](#)  
    NBCCCommon.h, [735](#)  
ACCL\_CMD\_Z\_CAL  
    MindSensors ACCL-Nx constants, [616](#)  
    NBCCCommon.h, [735](#)  
ACCL\_REG\_SENS\_LVL  
    NBCCCommon.h, [735](#)  
ACCL\_REG\_X\_ACCEL  
    MindSensors ACCL-Nx constants, [616](#)  
    NBCCCommon.h, [736](#)  
ACCL\_REG\_X\_OFFSET  
    NBCCCommon.h, [736](#)  
ACCL\_REG\_X\_RANGE  
    MindSensors ACCL-Nx constants, [616](#)  
    NBCCCommon.h, [736](#)  
ACCL\_REG\_X\_TILT  
    MindSensors ACCL-Nx constants, [616](#)  
    NBCCCommon.h, [736](#)  
ACCL\_REG\_Y\_ACCEL  
    MindSensors ACCL-Nx constants, [616](#)  
    NBCCCommon.h, [736](#)  
ACCL\_REG\_Y\_OFFSET  
    NBCCCommon.h, [736](#)  
ACCL\_REG\_Y\_RANGE  
    MindSensors ACCL-Nx constants, [616](#)  
    NBCCCommon.h, [736](#)  
ACCL\_REG\_Y\_TILT  
    MindSensors ACCL-Nx constants, [616](#)  
    NBCCCommon.h, [736](#)  
ACCL\_REG\_Z\_ACCEL  
    MindSensors ACCL-Nx constants, [616](#)  
    NBCCCommon.h, [736](#)  
ACCL\_REG\_Z\_OFFSET  
    NBCCCommon.h, [736](#)  
ACCL\_REG\_Z\_RANGE  
    MindSensors ACCL-Nx constants, [617](#)  
    NBCCCommon.h, [736](#)  
ACCL\_REG\_Z\_TILT  
    MindSensors ACCL-Nx constants, [617](#)  
    NBCCCommon.h, [736](#)  
ACCLNxCalibrateX  
    MindSensors API Functions, [133](#)  
ACCLNxCalibrateXEnd  
    MindSensors API Functions, [133](#)  
ACCLNxCalibrateY  
    MindSensors API Functions, [133](#)  
ACCLNxCalibrateYEnd  
    MindSensors API Functions, [133](#)  
ACCLNxCalibrateZ  
    MindSensors API Functions, [133](#)  
ACCLNxCalibrateZEnd  
    MindSensors API Functions, [134](#)  
ACCLNxResetCalibration  
    MindSensors API Functions, [134](#)  
ActualSpeedField  
    NBCCCommon.h, [737](#)  
    Output field constants, [458](#)  
Array operation constants, [363](#)  
    OPARR\_MAX, [363](#)  
    OPARR\_MEAN, [363](#)  
    OPARR\_MIN, [363](#)  
    OPARR\_SORT, [363](#)  
    OPARR\_STD, [363](#)  
    OPARR\_SUM, [363](#)  
    OPARR\_SUMSQR, [363](#)  
    OPARR\_TOLOWER, [363](#)  
    OPARR\_TOUPPER, [363](#)  
BITMAP\_1  
    Display module constants, [483](#)  
    NBCCCommon.h, [737](#)  
BITMAP\_2  
    Display module constants, [483](#)  
    NBCCCommon.h, [737](#)  
BITMAP\_3

Display module constants, 483  
NBCCCommon.h, 737

**BITMAP\_4**  
Display module constants, 483  
NBCCCommon.h, 737

**BITMAPS**  
Display module constants, 483  
NBCCCommon.h, 737

**BREAKOUT\_REQ**  
NBCCCommon.h, 737  
VM state constants, 381

**BT\_ARM\_CMD\_MODE**  
Bluetooth State constants, 508  
NBCCCommon.h, 737

**BT\_ARM\_DATA\_MODE**  
Bluetooth State constants, 508  
NBCCCommon.h, 737

**BT\_ARM\_OFF**  
Bluetooth State constants, 508  
NBCCCommon.h, 738

**BT\_BRICK\_PORT\_OPEN**  
Bluetooth state status constants, 510

**BT\_BRICK\_VISIBILITY**  
Bluetooth state status constants, 510

**BT\_CMD\_BYTE**  
Miscellaneous Comm module constants, 502  
NBCCCommon.h, 738

**BT\_CMD\_READY**  
Comm module status code constants, 531  
NBCCCommon.h, 738

**BT\_DEVICE\_AWAY**  
Device status constants, 527  
NBCCCommon.h, 738

**BT\_DEVICE\_EMPTY**  
Device status constants, 527  
NBCCCommon.h, 738

**BT\_DEVICE\_KNOWN**  
Device status constants, 527  
NBCCCommon.h, 738

**BT\_DEVICE\_NAME**  
Device status constants, 527  
NBCCCommon.h, 739

**BT\_DEVICE\_UNKNOWN**  
Device status constants, 527  
NBCCCommon.h, 739

**BT\_DISABLE**  
Bluetooth hardware status constants, 513  
NBCCCommon.h, 739

**BT\_ENABLE**  
Bluetooth hardware status constants, 513  
NBCCCommon.h, 739

**BTN1**  
Button name constants, 416  
NBCCCommon.h, 739

**BTN2**  
Button name constants, 416  
NBCCCommon.h, 739

**BTN3**  
Button name constants, 416  
NBCCCommon.h, 739

**BTN4**  
Button name constants, 416  
NBCCCommon.h, 739

**BTNCENTER**  
Button name constants, 416  
NBCCCommon.h, 739

**BTNEXTIT**  
Button name constants, 416  
NBCCCommon.h, 739

**BTNLEFT**  
Button name constants, 416  
NBCCCommon.h, 739

**BTNRIGHT**  
Button name constants, 416  
NBCCCommon.h, 739

**BTNSTATE\_NONE**  
ButtonState constants, 418  
NBCCCommon.h, 740

**BTNSTATE\_PRESSED\_EV**  
ButtonState constants, 418

**bcd2dec**  
cmath API, 358

**BlockTachoCountField**  
NBCCCommon.h, 737  
Output field constants, 458

Bluetooth hardware status constants, 513

**BT\_DISABLE**, 513  
**BT\_ENABLE**, 513

Bluetooth State constants, 508

**BT\_ARM\_CMD\_MODE**, 508  
**BT\_ARM\_DATA\_MODE**, 508  
**BT\_ARM\_OFF**, 508

Bluetooth state status constants, 510

**BT\_BRICK\_PORT\_OPEN**, 510  
**BT\_BRICK\_VISIBILITY**, 510

BluetoothState constants, 426

**UI\_BT\_PIN\_REQUEST**, 426  
**UI\_BT\_STATE\_OFF**, 426

**BluetoothStatus**  
Comm module functions, 301

**BluetoothWrite**  
Comm module functions, 301

**Button module**, 73

**Button module constants**, 415

**Button module functions**, 285

GetButtonLongPressCount, 285  
GetButtonLongReleaseCount, 286  
GetButtonPressCount, 286

GetButtonReleaseCount, 286  
GetButtonShortReleaseCount, 286  
GetButtonState, 286  
ReadButtonEx, 286  
SetButtonLongPressCount, 287  
SetButtonLongReleaseCount, 287  
SetButtonPressCount, 287  
SetButtonReleaseCount, 287  
SetButtonShortReleaseCount, 288  
SetButtonState, 288  
Button module IOMAP offsets, 419  
ButtonOffsetLongPressCnt, 419  
ButtonOffsetLongRelCnt, 419  
ButtonOffsetPressedCnt, 419  
ButtonOffsetRelCnt, 419  
ButtonOffsetShortRelCnt, 419  
ButtonOffsetState, 419  
Button name constants, 416  
BTN1, 416  
BTN2, 416  
BTN3, 416  
BTN4, 416  
BTNCENTER, 416  
BTNEXIT, 416  
BTNLEFT, 416  
BTNRIGHT, 416  
NO\_OF\_BTNS, 417  
ButtonModuleID  
    NBCCCommon.h, 740  
    NXT firmware module IDs, 201  
ButtonModuleName  
    NBCCCommon.h, 740  
    NXT firmware module names, 199  
ButtonOffsetLongPressCnt  
    Button module IOMAP offsets, 419  
    NBCCCommon.h, 740  
ButtonOffsetLongRelCnt  
    Button module IOMAP offsets, 419  
    NBCCCommon.h, 740  
ButtonOffsetPressedCnt  
    Button module IOMAP offsets, 419  
    NBCCCommon.h, 740  
ButtonOffsetRelCnt  
    Button module IOMAP offsets, 419  
    NBCCCommon.h, 740  
ButtonOffsetShortRelCnt  
    Button module IOMAP offsets, 419  
    NBCCCommon.h, 740  
ButtonOffsetState  
    Button module IOMAP offsets, 419  
    NBCCCommon.h, 740  
ButtonState constants, 418  
    BTNSTATE\_NONE, 418  
CHAR\_BIT  
    Data type limits, 687  
    NBCCCommon.h, 740  
CHAR\_MAX  
    Data type limits, 687  
    NBCCCommon.h, 740  
CHAR\_MIN  
    Data type limits, 687  
    NBCCCommon.h, 741  
CLUMP\_DONE  
    NBCCCommon.h, 741  
    VM state constants, 381  
CLUMP\_SUSPEND  
    NBCCCommon.h, 741  
    VM state constants, 381  
CONN\_BT0  
    NBCCCommon.h, 747  
    Remote connection constants, 511  
CONN\_BT1  
    NBCCCommon.h, 747  
    Remote connection constants, 511  
CONN\_BT2  
    NBCCCommon.h, 747  
    Remote connection constants, 511  
CONN\_BT3  
    NBCCCommon.h, 747  
    Remote connection constants, 511  
CONN\_HS4  
    NBCCCommon.h, 747  
    Remote connection constants, 511  
CONN\_HS\_1  
    NBCCCommon.h, 747  
    Remote connection constants, 511  
CONN\_HS\_2  
    NBCCCommon.h, 747  
    Remote connection constants, 511  
CONN\_HS\_3  
    NBCCCommon.h, 748  
    Remote connection constants, 512  
CONN\_HS\_4  
    NBCCCommon.h, 748  
    Remote connection constants, 512  
CONN\_HS\_5  
    NBCCCommon.h, 748  
    Remote connection constants, 512  
CONN\_HS\_6  
    NBCCCommon.h, 748  
    Remote connection constants, 512  
CONN\_HS\_7  
    NBCCCommon.h, 748  
    Remote connection constants, 512  
CONN\_HS\_8  
    NBCCCommon.h, 748  
    Remote connection constants, 512

CONN\_HS\_ALL  
  NBCCCommon.h, 748  
  Remote connection constants, 512

CT\_ADDR\_RFID  
  Codatex RFID sensor constants, 653  
  NBCCCommon.h, 748

CT\_REG\_DATA  
  Codatex RFID sensor constants, 653  
  NBCCCommon.h, 748

CT\_REG\_MODE  
  Codatex RFID sensor constants, 653  
  NBCCCommon.h, 748

CT\_REG\_STATUS  
  Codatex RFID sensor constants, 653  
  NBCCCommon.h, 748

CircleOut  
  Display module functions, 252

CircleOutEx  
  Display module functions, 252

ClearLine  
  Display module functions, 252

ClearScreen  
  Display module functions, 252

ClearSensor  
  Input module functions, 228

CloseFile  
  Loader module functions, 349

cmath API, 358  
  bcd2dec, 358

Coast  
  Output module functions, 214

CoastEx  
  Output module functions, 214

Codatex API Functions, 181  
  RFIDInit, 181  
  RFIDMode, 181  
  RFIDRead, 182  
  RFIDReadContinuous, 182  
  RFIDReadSingle, 182  
  RFIDStatus, 182  
  RFIDStop, 183

Codatex device constants, 652

Codatex RFID sensor constants, 653  
  CT\_ADDR\_RFID, 653  
  CT\_REG\_DATA, 653  
  CT\_REG\_MODE, 653  
  CT\_REG\_STATUS, 653

Codatex RFID sensor modes, 654  
  RFID\_MODE\_SINGLE, 654  
  RFID\_MODE\_STOP, 654

Color calibration constants, 441  
  INPUT\_CAL\_POINT\_0, 441  
  INPUT\_CAL\_POINT\_1, 441  
  INPUT\_CAL\_POINT\_2, 441

  INPUT\_NO\_OF\_POINTS, 441

Color calibration state constants, 440  
  INPUT\_RESETCAL, 440  
  INPUT\_RUNNINGCAL, 440  
  INPUT\_SENSORCAL, 440  
  INPUT\_SENSOROFF, 440  
  INPUT\_STARTCAL, 440

Color sensor array indices, 438  
  INPUT\_BLANK, 438  
  INPUT\_BLUE, 438  
  INPUT\_GREEN, 438  
  INPUT\_NO\_OF\_COLORS, 438  
  INPUT\_RED, 438

Color values, 439  
  INPUT\_BLACKCOLOR, 439  
  INPUT\_BLUECOLOR, 439  
  INPUT\_GREENCOLOR, 439  
  INPUT\_REDCOLOR, 439  
  INPUT\_WHITECOLOR, 439  
  INPUT\_YELLOWCOLOR, 439

ColorSensorRead  
  NBCCCommon.h, 741  
  System Call function constants, 366

Comm module, 72

Comm module constants, 501

Comm module functions, 296  
  BluetoothStatus, 301  
  BluetoothWrite, 301  
  GetBTConnectionAddress, 302  
  GetBTConnectionClass, 303  
  GetBTConnectionHandleNum, 303  
  GetBTConnectionLinkQuality, 303  
  GetBTConnectionName, 303  
  GetBTConnectionPinCode, 303  
  GetBTConnectionStreamStatus, 304  
  GetBTDataMode, 304  
  GetBTDeviceAddress, 304  
  GetBTDeviceClass, 304  
  GetBTDeviceCount, 305  
  GetBTDeviceName, 305  
  GetBTDeviceNameCount, 305  
  GetBTDeviceStatus, 305  
  GetBTInputBuffer, 306  
  GetBTInputBufferInPtr, 306  
  GetBTInputBufferOutPtr, 306  
  GetBTOutputBuffer, 306  
  GetBTOutputBufferInPtr, 307  
  GetBTOutputBufferOutPtr, 307  
  GetBrickDataAddress, 301  
  GetBrickDataBluecoreVersion, 301  
  GetBrickDataBtHardwareStatus, 301  
  GetBrickDataBtStateStatus, 302  
  GetBrickDataName, 302  
  GetBrickDataTimeoutValue, 302

GetHSAddress, 307  
GetHSDMode, 308  
GetHSFlags, 308  
GetHSInputBuffer, 308  
GetHSInputBufferInPtr, 308  
GetHSInputBufferOutPtr, 309  
GetHSMode, 309  
GetHSOutputBuffer, 309  
GetHSOutputBufferInPtr, 310  
GetHSOutputBufferOutPtr, 310  
GetHSSpeed, 310  
GetHSState, 310  
GetUSBInputBuffer, 311  
GetUSBInputBufferInPtr, 311  
GetUSBInputBufferOutPtr, 311  
GetUSBOOutputBuffer, 311  
GetUSBOOutputBufferInPtr, 312  
GetUSBOOutputBufferOutPtr, 312  
GetUSBPollBuffer, 312  
GetUSBPollBufferInPtr, 312  
GetUSBPollBufferOutPtr, 313  
GetUSBState, 313  
RS485Control, 315  
RS485Disable, 316  
RS485Enable, 316  
RS485Initialize, 316  
RS485Read, 317  
RS485ReadEx, 317  
RS485Status, 317  
RS485Uart, 317  
RS485Write, 318  
ReceiveMessage, 313  
ReceiveRemoteBool, 314  
ReceiveRemoteMessageEx, 314  
ReceiveRemoteNumber, 314  
ReceiveRemoteString, 314  
RemoteConnectionIdle, 315  
RemoteConnectionWrite, 315  
SendMessage, 318  
SendRS485Bool, 320  
SendRS485Number, 320  
SendRS485String, 320  
SendRemoteBool, 318  
SendRemoteNumber, 318  
SendRemoteString, 319  
SendResponseBool, 319  
SendResponseNumber, 319  
SendResponseString, 319  
SetBTDataMode, 321  
SetBTInputBuffer, 321  
SetBTInputBufferInPtr, 321  
SetBTInputBufferOutPtr, 321  
SetBTOutputBuffer, 321  
SetBTOutputBufferInPtr, 322  
SetBTOutputBufferOutPtr, 322  
SetBTOutputBufferOutPtr, 322  
SetHSAddress, 322  
SetHSDMode, 322  
SetHSFlags, 322  
SetHSInputBuffer, 323  
SetHSInputBufferInPtr, 323  
SetHSInputBufferOutPtr, 323  
SetHSMode, 323  
SetHSOutputBuffer, 323  
SetHSOutputBufferInPtr, 324  
SetHSOutputBufferOutPtr, 324  
SetHSSpeed, 324  
SetHSState, 324  
SetUSBInputBuffer, 324  
SetUSBInputBufferInPtr, 325  
SetUSBInputBufferOutPtr, 325  
SetUSBOOutputBuffer, 325  
SetUSBOOutputBufferInPtr, 325  
SetUSBOOutputBufferOutPtr, 325  
SetUSBPollBuffer, 325  
SetUSBPollBufferInPtr, 326  
SetUSBPollBufferOutPtr, 326  
SetUSBState, 326  
UseRS485, 326  
Comm module IOMAP offsets, 532  
  CommOffsetBrickDataBdAddr, 533  
  CommOffsetBrickDataBluecoreVersion, 533  
  CommOffsetBrickDataBtHwStatus, 533  
  CommOffsetBrickDataBtStateStatus, 533  
  CommOffsetBrickDataName, 533  
  CommOffsetBrickDataTimeOutValue, 533  
  CommOffsetBtConnectTableBdAddr, 533  
  CommOffsetBtConnectTableClassOfDevice, 533  
  CommOffsetBtConnectTableHandleNr, 533  
  CommOffsetBtConnectTableLinkQuality, 533  
  CommOffsetBtConnectTableName, 533  
  CommOffsetBtConnectTablePinCode, 534  
  CommOffsetBtConnectTableStreamStatus, 534  
  CommOffsetBtDataMode, 534  
  CommOffsetBtDeviceCnt, 534  
  CommOffsetBtDeviceNameCnt, 534  
  CommOffsetBtDeviceTableBdAddr, 534  
  CommOffsetBtDeviceTableClassOfDevice, 534  
  CommOffsetBtDeviceTableDeviceStatus, 534  
  CommOffsetBtDeviceTableName, 534  
  CommOffsetBtInBufBuf, 534  
  CommOffsetBtInBufInPtr, 534  
  CommOffsetBtInBufOutPtr, 534  
  CommOffsetBtOutBufBuf, 534  
  CommOffsetBtOutBuflnPtr, 534  
  CommOffsetBtOutBufOutPtr, 535  
  CommOffsetHsAddress, 535  
  CommOffsetHsDataMode, 535  
  CommOffsetHsFlags, 535

CommOffsetHsInBufBuf, 535  
CommOffsetHsInBufInPtr, 535  
CommOffsetHsInBufOutPtr, 535  
CommOffsetHsMode, 535  
CommOffsetHsOutBufBuf, 535  
CommOffsetHsOutBufInPtr, 535  
CommOffsetHsOutBufOutPtr, 535  
CommOffsetHsSpeed, 535  
CommOffsetHsState, 535  
CommOffsetPFunc, 535  
CommOffsetPFuncTwo, 536  
CommOffsetUsbInBufBuf, 536  
CommOffsetUsbInBufInPtr, 536  
CommOffsetUsbInBufOutPtr, 536  
CommOffsetUsbOutBufBuf, 536  
CommOffsetUsbOutBufInPtr, 536  
CommOffsetUsbOutBufOutPtr, 536  
CommOffsetUsbPollBufBuf, 536  
CommOffsetUsbPollBufInPtr, 536  
CommOffsetUsbPollBufOutPtr, 536  
CommOffsetUsbState, 536

Comm module interface function constants, 528

INTF\_BTOFF, 528  
INTF\_BTON, 528  
INTF\_CONNECT, 528  
INTF\_CONNECTBYNAME, 528  
INTF\_CONNECTREQ, 528  
INTF\_DISCONNECT, 529  
INTF\_DISCONNECTALL, 529  
INTF\_EXTREAD, 529  
INTF\_FACTORYRESET, 529  
INTF\_OPENSTREAM, 529  
INTF\_PINREQ, 529  
INTF\_REMOVEDevice, 529  
INTF\_SEARCH, 529  
INTF\_SENDDATA, 529  
INTF\_SENDFILE, 529  
INTF\_SETBTNAME, 529  
INTF\_SETCMDMODE, 529  
INTF\_STOPSEARCH, 529  
INTF\_VISIBILITY, 529

Comm module status code constants, 531

BT\_CMD\_READY, 531  
HS\_CMD\_READY, 531  
LR\_COULD\_NOT\_SAVE, 531  
LR\_ENTRY\_REMOVED, 531  
LR\_STORE\_IS\_FULL, 531  
LR\_SUCCESS, 531  
LR\_UNKNOWN\_ADDR, 531  
USB\_CMD\_READY, 531

CommBTCheckStatus

NBCCommon.h, 742  
System Call function constants, 366

CommBTConnection

NBCCommon.h, 742  
System Call function constants, 366

CommBTOnOff

NBCCommon.h, 742  
System Call function constants, 366

CommBTRead

NBCCommon.h, 742  
System Call function constants, 366

CommBTWrite

NBCCommon.h, 743  
System Call function constants, 366

CommExecuteFunction

NBCCommon.h, 743  
System Call function constants, 367

CommHSCheckStatus

NBCCommon.h, 743  
System Call function constants, 367

CommHSControl

NBCCommon.h, 743  
System Call function constants, 367

CommHSRead

NBCCommon.h, 743  
System Call function constants, 367

CommHSWrite

NBCCommon.h, 743  
System Call function constants, 367

CommLSCheckStatus

NBCCommon.h, 743  
System Call function constants, 367

CommLSRead

NBCCommon.h, 743  
System Call function constants, 367

CommLSWrite

NBCCommon.h, 743  
System Call function constants, 367

CommLSWriteEx

NBCCommon.h, 743  
System Call function constants, 367

CommModuleID

NBCCommon.h, 743  
NXT firmware module IDs, 201

CommModuleName

NBCCommon.h, 743  
NXT firmware module names, 199

CommOffsetBrickDataBdAddr

Comm module IOMAP offsets, 533

NBCCommon.h, 743

CommOffsetBrickDataBluecoreVersion

Comm module IOMAP offsets, 533

NBCCommon.h, 743

CommOffsetBrickDataBtHwStatus

Comm module IOMAP offsets, 533

NBCCommon.h, 744

CommOffsetBrickDataBtStateStatus

Comm module IOMAP offsets, 533  
NBCCCommon.h, 744

CommOffsetBrickDataName  
Comm module IOMAP offsets, 533  
NBCCCommon.h, 744

CommOffsetBrickDataTimeOutValue  
Comm module IOMAP offsets, 533  
NBCCCommon.h, 744

CommOffsetBtConnectTableBdAddr  
Comm module IOMAP offsets, 533  
NBCCCommon.h, 744

CommOffsetBtConnectTableClassOfDevice  
Comm module IOMAP offsets, 533  
NBCCCommon.h, 744

CommOffsetBtConnectTableHandleNr  
Comm module IOMAP offsets, 533  
NBCCCommon.h, 744

CommOffsetBtConnectTableLinkQuality  
Comm module IOMAP offsets, 533  
NBCCCommon.h, 744

CommOffsetBtConnectTableName  
Comm module IOMAP offsets, 533  
NBCCCommon.h, 744

CommOffsetBtConnectTablePinCode  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 744

CommOffsetBtConnectTableStreamStatus  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 744

CommOffsetBtDataMode  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 744

CommOffsetBtDeviceCnt  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 744

CommOffsetBtDeviceNameCnt  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 744

CommOffsetBtDeviceTableBdAddr  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 745

CommOffsetBtDeviceTableClassOfDevice  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 745

CommOffsetBtDeviceTableDeviceStatus  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 745

CommOffsetBtDeviceTableName  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 745

CommOffsetBtInBufBuf  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 745

CommOffsetBtInBufInPtr  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 745

CommOffsetBtOutBufBuf  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 745

CommOffsetBtOutBufInPtr  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 745

CommOffsetBtOutBufOutPtr  
Comm module IOMAP offsets, 534  
NBCCCommon.h, 745

CommOffsetBsAddress  
Comm module IOMAP offsets, 535  
NBCCCommon.h, 745

CommOffsetBsDataMode  
Comm module IOMAP offsets, 535  
NBCCCommon.h, 745

CommOffsetBsFlags  
Comm module IOMAP offsets, 535  
NBCCCommon.h, 745

CommOffsetBsInBufBuf  
Comm module IOMAP offsets, 535  
NBCCCommon.h, 745

CommOffsetBsInBufInPtr  
Comm module IOMAP offsets, 535  
NBCCCommon.h, 746

CommOffsetBsInBufOutPtr  
Comm module IOMAP offsets, 535  
NBCCCommon.h, 746

CommOffsetBsMode  
Comm module IOMAP offsets, 535  
NBCCCommon.h, 746

CommOffsetBsOutBufBuf  
Comm module IOMAP offsets, 535  
NBCCCommon.h, 746

CommOffsetBsOutBufInPtr  
Comm module IOMAP offsets, 535  
NBCCCommon.h, 746

CommOffsetBsOutBufOutPtr  
Comm module IOMAP offsets, 535  
NBCCCommon.h, 746

CommOffsetBsSpeed  
Comm module IOMAP offsets, 535  
NBCCCommon.h, 746

CommOffsetBsState  
Comm module IOMAP offsets, 535  
NBCCCommon.h, 746

CommOffsetPFunc  
Comm module IOMAP offsets, 535  
NBCCCommon.h, 746

CommOffsetPFuncTwo

Comm module IOMAP offsets, 536  
NBCCCommon.h, 746

CommOffsetUsbInBufBuf  
Comm module IOMAP offsets, 536  
NBCCCommon.h, 746

CommOffsetUsbInBufInPtr  
Comm module IOMAP offsets, 536  
NBCCCommon.h, 746

CommOffsetUsbInBufOutPtr  
Comm module IOMAP offsets, 536  
NBCCCommon.h, 746

CommOffsetUsbOutBufBuf  
Comm module IOMAP offsets, 536  
NBCCCommon.h, 746

CommOffsetUsbOutBufInPtr  
Comm module IOMAP offsets, 536  
NBCCCommon.h, 747

CommOffsetUsbOutBufOutPtr  
Comm module IOMAP offsets, 536  
NBCCCommon.h, 747

CommOffsetUsbPollBufBuf  
Comm module IOMAP offsets, 536  
NBCCCommon.h, 747

CommOffsetUsbPollBufInPtr  
Comm module IOMAP offsets, 536  
NBCCCommon.h, 747

CommOffsetUsbPollBufOutPtr  
Comm module IOMAP offsets, 536  
NBCCCommon.h, 747

CommOffsetUsbState  
Comm module IOMAP offsets, 536  
NBCCCommon.h, 747

Command module, 69

Command module constants, 70  
NO\_ERR, 70  
POOL\_MAX\_SIZE, 70  
STAT\_COMM\_PENDING, 70

Command module functions, 271  
GetButtonModuleValue, 273  
GetCommModuleBytes, 274  
GetCommModuleValue, 274  
GetCommandModuleBytes, 273  
GetCommandModuleValue, 273  
GetDisplayModuleBytes, 274  
GetDisplayModuleValue, 274  
GetFirstTick, 275  
GetIOMapBytes, 275  
GetIOMapBytesByID, 275  
GetIOMapValue, 276  
GetIOMapValueByID, 276  
GetInputModuleValue, 275  
GetLastResponseInfo, 276  
GetLoaderModuleValue, 277  
GetLowSpeedModuleBytes, 277

GetLowSpeedModuleValue, 277  
GetMemoryInfo, 277

GetOutputModuleValue, 278  
GetSoundModuleValue, 278

GetUIModuleValue, 278  
ResetSleepTimer, 279

SetButtonModuleValue, 279  
SetCommModuleBytes, 279  
SetCommModuleValue, 280

SetCommandModuleBytes, 279  
SetCommandModuleValue, 279

SetDisplayModuleBytes, 280  
SetDisplayModuleValue, 280

SetIOCtrlModuleValue, 281

SetIOMapBytes, 281  
SetIOMapBytesByID, 281

SetIOMapValue, 282  
SetIOMapValueByID, 282

SetInputModuleValue, 280

SetLoaderModuleValue, 282

SetLowSpeedModuleBytes, 282

SetLowSpeedModuleValue, 283

SetOutputModuleValue, 283

SetSoundModuleBytes, 283

SetSoundModuleValue, 283

SetUIModuleValue, 284

Wait, 284

Command module IOMAP offsets, 388  
CommandOffsetActivateFlag, 388  
CommandOffsetAwake, 388  
CommandOffsetDeactivateFlag, 388  
CommandOffsetFileName, 388  
CommandOffsetFormatString, 388  
CommandOffsetMemoryPool, 388  
CommandOffsetOffsetDS, 388  
CommandOffsetOffsetDVA, 388  
CommandOffsetPRCHandler, 389  
CommandOffsetProgStatus, 389  
CommandOffsetSyncTick, 389  
CommandOffsetSyncTime, 389  
CommandOffsetTick, 389

CommandFlags constants, 421  
UI\_FLAGS\_BUSY, 421  
UI\_FLAGS\_UPDATE, 421

CommandModuleID  
NBCCCommon.h, 741  
NXT firmware module IDs, 201

CommandModuleName  
NBCCCommon.h, 741  
NXT firmware module names, 199

CommandOffsetActivateFlag  
Command module IOMAP offsets, 388  
NBCCCommon.h, 741

CommandOffsetAwake

Command module IOMAP offsets, 388  
NBCCCommon.h, 741

CommandOffsetDeactivateFlag  
Command module IOMAP offsets, 388  
NBCCCommon.h, 741

CommandOffsetFileName  
Command module IOMAP offsets, 388  
NBCCCommon.h, 742

CommandOffsetFormatString  
Command module IOMAP offsets, 388  
NBCCCommon.h, 742

CommandOffsetMemoryPool  
Command module IOMAP offsets, 388  
NBCCCommon.h, 742

CommandOffsetOffsetDS  
Command module IOMAP offsets, 388  
NBCCCommon.h, 742

CommandOffsetOffsetDVA  
Command module IOMAP offsets, 388  
NBCCCommon.h, 742

CommandOffsetPRCHandler  
Command module IOMAP offsets, 389  
NBCCCommon.h, 742

CommandOffsetProgStatus  
Command module IOMAP offsets, 389  
NBCCCommon.h, 742

CommandOffsetSyncTick  
Command module IOMAP offsets, 389  
NBCCCommon.h, 742

CommandOffsetSyncTime  
Command module IOMAP offsets, 389  
NBCCCommon.h, 742

CommandOffsetTick  
Command module IOMAP offsets, 389  
NBCCCommon.h, 742

Communications specific errors, 385  
ERR\_COMM\_BUS\_ERR, 385

Comparison Constants, 60  
EQ, 61  
GT, 61  
GTEQ, 61  
LT, 61  
LTEQ, 61  
NEQ, 61

ComputeCalibValue  
NBCCCommon.h, 747  
System Call function constants, 367

ConfigureTemperatureSensor  
LowSpeed module functions, 239

Constants to use with the Input module's Pin function, 445  
INPUT\_PINCMD\_CLEAR, 445  
INPUT\_PINCMD\_DIR, 445  
INPUT\_PINCMD\_MASK, 445  
INPUT\_PINCMD\_READ, 445

INPUT\_PINCMD\_SET, 445  
INPUT\_PINCMD\_WAIT, 445  
INPUT\_PINDIR\_INPUT, 445

CreateFile  
Loader module functions, 349

CreateFileLinear  
Loader module functions, 349

CreateFileNonLinear  
Loader module functions, 350

cstdlib API, 357  
Random, 357  
SignedRandom, 357

DAC\_MODE\_DCOUT  
NBCCCommon.h, 748  
SuperPro analog output mode constants, 117

DAC\_MODE\_SINEWAVE  
NBCCCommon.h, 749  
SuperPro analog output mode constants, 117

DATA\_MODE\_GPS  
Data mode constants, 509  
NBCCCommon.h, 749

DATA\_MODE\_MASK  
Data mode constants, 509  
NBCCCommon.h, 749

DATA\_MODE\_NXT  
Data mode constants, 509  
NBCCCommon.h, 749

DATA\_MODE\_RAW  
Data mode constants, 509  
NBCCCommon.h, 749

DATA\_MODE\_UPDATE  
Data mode constants, 509  
NBCCCommon.h, 749

DEGREES\_PER\_RADIAN  
NBCCCommon.h, 749

DGPS\_REG\_DISTANCE  
Dexter Industries GPS sensor constants, 656  
NBCCCommon.h, 749

DGPS\_REG\_HEADING  
Dexter Industries GPS sensor constants, 656  
NBCCCommon.h, 750

DGPS\_REG\_LASTANGLE  
Dexter Industries GPS sensor constants, 656  
NBCCCommon.h, 750

DGPS\_REG\_LATITUDE  
Dexter Industries GPS sensor constants, 656  
NBCCCommon.h, 750

DGPS\_REG\_LONGITUDE  
Dexter Industries GPS sensor constants, 656  
NBCCCommon.h, 750

DGPS\_REG\_STATUS  
Dexter Industries GPS sensor constants, 656  
NBCCCommon.h, 750

DGPS\_REG\_TIME  
Dexter Industries GPS sensor constants, 657  
NBCCommon.h, 750

DGPS\_REG\_VELOCITY  
Dexter Industries GPS sensor constants, 657  
NBCCommon.h, 750

DGPS\_REG\_WAYANGLE  
Dexter Industries GPS sensor constants, 657  
NBCCommon.h, 750

DI\_ADDR\_ACCL  
Dexter Industries IMU sensor constants, 658  
NBCCommon.h, 750

DI\_ADDR\_GPS  
Dexter Industries GPS sensor constants, 657  
NBCCommon.h, 750

DI\_ADDR\_GYRO  
Dexter Industries IMU sensor constants, 658  
NBCCommon.h, 750

DIACCL\_MODE\_GLVL2  
Dexter Industries IMU Accelerometer mode control register constants, 678  
NBCCommon.h, 751

DIACCL\_MODE\_GLVL4  
Dexter Industries IMU Accelerometer mode control register constants, 678  
NBCCommon.h, 752

DIACCL\_MODE\_GLVL8  
Dexter Industries IMU Accelerometer mode control register constants, 678  
NBCCommon.h, 752

DIACCL\_REG\_CTRL1  
Dexter Industries IMU Accelerometer register constants, 674  
NBCCommon.h, 752

DIACCL\_REG\_CTRL2  
Dexter Industries IMU Accelerometer register constants, 674  
NBCCommon.h, 752

DIACCL\_REG\_I2CADDR  
Dexter Industries IMU Accelerometer register constants, 675  
NBCCommon.h, 752

DIACCL\_REG\_OUTTEMP  
NBCCommon.h, 753

DIACCL\_REG\_STATUS  
Dexter Industries IMU Accelerometer register constants, 675  
NBCCommon.h, 753

DIACCL\_REG\_WHOAMI  
Dexter Industries IMU Accelerometer register constants, 675  
NBCCommon.h, 753

DIACCL\_REG\_X8

Dexter Industries IMU Accelerometer register constants, 675  
NBCCommon.h, 753

DIACCL\_REG\_XHIGH  
Dexter Industries IMU Accelerometer register constants, 675  
NBCCommon.h, 753

DIACCL\_REG\_XLOW  
Dexter Industries IMU Accelerometer register constants, 676  
NBCCommon.h, 753

DIACCL\_REG\_Y8  
Dexter Industries IMU Accelerometer register constants, 676  
NBCCommon.h, 753

DIACCL\_REG\_YHIGH  
Dexter Industries IMU Accelerometer register constants, 676  
NBCCommon.h, 753

DIACCL\_REG\_YLOW  
Dexter Industries IMU Accelerometer register constants, 676  
NBCCommon.h, 754

DIACCL\_REG\_Z8  
Dexter Industries IMU Accelerometer register constants, 676  
NBCCommon.h, 754

DIACCL\_REG\_ZHIGH  
Dexter Industries IMU Accelerometer register constants, 676  
NBCCommon.h, 754

DIACCL\_REG\_ZLOW  
Dexter Industries IMU Accelerometer register constants, 676  
NBCCommon.h, 754

DIGI\_PIN0  
NBCCommon.h, 754  
SuperPro digital pin constants, 120

DIGI\_PIN1  
NBCCommon.h, 754  
SuperPro digital pin constants, 120

DIGI\_PIN2  
NBCCommon.h, 754  
SuperPro digital pin constants, 120

DIGI\_PIN3  
NBCCommon.h, 755  
SuperPro digital pin constants, 120

DIGI\_PIN4  
NBCCommon.h, 755  
SuperPro digital pin constants, 120

DIGI\_PIN5  
NBCCommon.h, 755  
SuperPro digital pin constants, 120

DIGI\_PIN6

NBCCCommon.h, 755  
SuperPro digital pin constants, 120

DIGI\_PIN7  
NBCCCommon.h, 755  
SuperPro digital pin constants, 120

DIGYRO\_CTRL1\_NORMAL  
NBCCCommon.h, 755

DIGYRO\_REG\_CTRL1  
Dexter Industries IMU Gyro register constants, 660  
NBCCCommon.h, 759

DIGYRO\_REG\_CTRL2  
Dexter Industries IMU Gyro register constants, 660  
NBCCCommon.h, 759

DIGYRO\_REG\_CTRL3  
Dexter Industries IMU Gyro register constants, 661  
NBCCCommon.h, 759

DIGYRO\_REG\_CTRL4  
Dexter Industries IMU Gyro register constants, 661  
NBCCCommon.h, 759

DIGYRO\_REG\_CTRL5  
Dexter Industries IMU Gyro register constants, 661  
NBCCCommon.h, 759

DIGYRO\_REG\_FIFOSRC  
NBCCCommon.h, 759

DIGYRO\_REG\_OUTTEMP  
NBCCCommon.h, 760

DIGYRO\_REG\_STATUS  
Dexter Industries IMU Gyro register constants, 662  
NBCCCommon.h, 760

DIGYRO\_REG\_WHOAMI  
Dexter Industries IMU Gyro register constants, 662  
NBCCCommon.h, 760

DIGYRO\_REG\_XHIGH  
Dexter Industries IMU Gyro register constants, 662  
NBCCCommon.h, 760

DIGYRO\_REG\_XLOW  
Dexter Industries IMU Gyro register constants, 662  
NBCCCommon.h, 760

DIGYRO\_REG\_YHIGH  
Dexter Industries IMU Gyro register constants, 662  
NBCCCommon.h, 761

DIGYRO\_REG\_YLOW  
Dexter Industries IMU Gyro register constants, 662  
NBCCCommon.h, 761

DIGYRO\_REG\_ZHIGH  
Dexter Industries IMU Gyro register constants, 662  
NBCCCommon.h, 761

DIGYRO\_REG\_ZLOW  
Dexter Industries IMU Gyro register constants, 662  
NBCCCommon.h, 761

DISPLAY\_BUSY  
Display flags, 494  
NBCCCommon.h, 762

DISPLAY\_CHAR

DisplayExecuteFunction constants, 487  
NBCCCommon.h, 762

DISPLAY\_ERASE\_ALL  
DisplayExecuteFunction constants, 487  
NBCCCommon.h, 762

DISPLAY\_ERASE\_LINE  
DisplayExecuteFunction constants, 487  
NBCCCommon.h, 762

DISPLAY\_FRAME  
DisplayExecuteFunction constants, 487  
NBCCCommon.h, 762

DISPLAY\_HEIGHT  
Display module constants, 483  
NBCCCommon.h, 762

DISPLAY\_MENUICONS\_Y  
Display module constants, 483

DISPLAY\_ON  
Display flags, 494  
NBCCCommon.h, 763

DISPLAY\_PIXEL  
DisplayExecuteFunction constants, 487  
NBCCCommon.h, 763

DISPLAY\_POPUP  
Display flags, 494  
NBCCCommon.h, 763

DISPLAY\_REFRESH  
Display flags, 494  
NBCCCommon.h, 763

DISPLAY\_WIDTH  
Display module constants, 483  
NBCCCommon.h, 763

DIST\_CMD\_CUSTOM  
MindSensors DIST-Nx constants, 607  
NBCCCommon.h, 765

DIST\_CMD\_GP2D12  
MindSensors DIST-Nx constants, 607  
NBCCCommon.h, 765

DIST\_CMD\_GP2D120  
MindSensors DIST-Nx constants, 607  
NBCCCommon.h, 765

DIST\_CMD\_GP2YA02  
MindSensors DIST-Nx constants, 607  
NBCCCommon.h, 765

DIST\_CMD\_GP2YA21  
MindSensors DIST-Nx constants, 607  
NBCCCommon.h, 765

DIST\_REG\_DIST  
MindSensors DIST-Nx constants, 607  
NBCCCommon.h, 765

DIST\_REG\_DIST1  
MindSensors DIST-Nx constants, 607  
NBCCCommon.h, 765

DIST\_REG\_DIST\_MAX  
MindSensors DIST-Nx constants, 607  
NBCCCommon.h, 765

DIST\_REG\_DIST\_MIN  
NBCCCommon.h, 765

DIST\_REG\_VOLT  
MindSensors DIST-Nx constants, 608  
NBCCCommon.h, 765

DIST\_REG\_VOLT1  
MindSensors DIST-Nx constants, 608  
NBCCCommon.h, 766

DISTNxGP2D12  
MindSensors API Functions, 134

DISTNxGP2D120  
MindSensors API Functions, 134

DISTNxGP2YA02  
MindSensors API Functions, 135

DISTNxGP2YA21  
MindSensors API Functions, 135

DRAW\_OPT\_CLEAR  
Drawing option constants, 490  
NBCCCommon.h, 766

DRAW\_OPT\_CLEAR\_EOL  
Drawing option constants, 490

DRAW\_OPT\_FONT\_WRAP  
Font drawing option constants, 493

DRAW\_OPT\_INVERT  
Drawing option constants, 490  
NBCCCommon.h, 767

DRAW\_OPT\_NORMAL  
Drawing option constants, 490  
NBCCCommon.h, 767

Data mode constants, 509  
DATA\_MODE\_GPS, 509  
DATA\_MODE\_MASK, 509  
DATA\_MODE\_NXT, 509  
DATA\_MODE\_RAW, 509  
DATA\_MODE\_UPDATE, 509

Data type limits, 687  
CHAR\_BIT, 687  
CHAR\_MAX, 687  
CHAR\_MIN, 687  
INT\_MAX, 687  
INT\_MIN, 687  
LONG\_MAX, 687  
LONG\_MIN, 687  
RAND\_MAX, 688  
SCHAR\_MAX, 688  
SCHAR\_MIN, 688  
SHRT\_MAX, 688  
SHRT\_MIN, 688  
UCHAR\_MAX, 688  
UINT\_MAX, 688  
ULONG\_MAX, 688  
USHRT\_MAX, 688

DatalogGetTimes  
NBCCCommon.h, 749

System Call function constants, 367

DatalogWrite  
NBCCCommon.h, 749  
System Call function constants, 367

DeleteFile  
Loader module functions, 350

Device status constants, 527  
BT\_DEVICE\_AWAY, 527  
BT\_DEVICE\_EMPTY, 527  
BT\_DEVICE\_KNOWN, 527  
BT\_DEVICE\_NAME, 527  
BT\_DEVICE\_UNKNOWN, 527

Dexter Industries API Functions, 184  
ReadSensorDIAccl, 185  
ReadSensorDIAccl8, 185  
ReadSensorDIAccl8Raw, 186  
ReadSensorDIAcclDrift, 186  
ReadSensorDIAcclRaw, 186  
ReadSensorDIAcclStatus, 186  
ReadSensorDIGPSDistanceToWaypoint, 187  
ReadSensorDIGPSHeading, 187  
ReadSensorDIGPSHeadingToWaypoint, 187  
ReadSensorDIGPSLatitude, 187  
ReadSensorDIGPSLongitude, 187  
ReadSensorDIGPSRelativeHeading, 188  
ReadSensorDIGPSStatus, 188  
ReadSensorDIGPSTime, 188  
ReadSensorDIGPSVelocity, 188  
ReadSensorDIGyro, 188  
ReadSensorDIGyroRaw, 189  
ReadSensorDIGyroStatus, 189  
ReadSensorDIGyroTemperature, 189  
SetSensorDIAccl, 189  
SetSensorDIAcclDrift, 190  
SetSensorDIAcclEx, 190  
SetSensorDIGPSWaypoint, 190  
SetSensorDIGyro, 190  
SetSensorDIGyroEx, 191

Dexter Industries device constants, 655

Dexter Industries GPS sensor constants, 656  
DGPS\_REG\_DISTANCE, 656  
DGPS\_REG\_HEADING, 656  
DGPS\_REG\_LATITUDE, 656  
DGPS\_REG\_STATUS, 656  
DGPS\_REG\_TIME, 657  
DGPS\_REG\_VELOCITY, 657  
DGPS\_REG\_WAYANGLE, 657  
DI\_ADDR\_DGPS, 657

Dexter Industries IMU Accelerometer control register 1 constants, 680

Dexter Industries IMU Accelerometer control register 2 constants, 682

Dexter Industries IMU Accelerometer interrupt latch reset register constants, 679

- Dexter Industries IMU Accelerometer mode control register constants, [678](#)
- Dexter Industries IMU Accelerometer register constants, [674](#)
- DIACCL\_REG\_CTRL1, [674](#)
  - DIACCL\_REG\_CTRL2, [674](#)
  - DIACCL\_REG\_X8, [675](#)
  - DIACCL\_REG\_XHIGH, [675](#)
  - DIACCL\_REG\_XLOW, [676](#)
  - DIACCL\_REG\_Y8, [676](#)
  - DIACCL\_REG\_YHIGH, [676](#)
  - DIACCL\_REG\_YLOW, [676](#)
  - DIACCL\_REG\_Z8, [676](#)
  - DIACCL\_REG\_ZHIGH, [676](#)
  - DIACCL\_REG\_ZLOW, [676](#)
- Dexter Industries IMU Accelerometer status register constants, [677](#)
- Dexter Industries IMU Gyro control register 1 constants, [664](#)
- Dexter Industries IMU Gyro control register 2 constants, [666](#)
- Dexter Industries IMU Gyro control register 3 constants, [668](#)
- Dexter Industries IMU Gyro control register 4 constants, [669](#)
- Dexter Industries IMU Gyro control register 5 constants, [670](#)
- Dexter Industries IMU Gyro FIFO control register onstants, [672](#)
- Dexter Industries IMU Gyro register constants, [660](#)
- DIGYRO\_REG\_CTRL1, [660](#)
  - DIGYRO\_REG\_CTRL2, [660](#)
  - DIGYRO\_REG\_CTRL3, [661](#)
  - DIGYRO\_REG\_CTRL4, [661](#)
  - DIGYRO\_REG\_CTRL5, [661](#)
  - DIGYRO\_REG\_XHIGH, [662](#)
  - DIGYRO\_REG\_XLOW, [662](#)
  - DIGYRO\_REG\_YHIGH, [662](#)
  - DIGYRO\_REG\_YLOW, [662](#)
  - DIGYRO\_REG\_ZHIGH, [662](#)
  - DIGYRO\_REG\_ZLOW, [662](#)
- Dexter Industries IMU Gyro status register constants, [673](#)
- Dexter Industries IMU sensor constants, [658](#)
- DI\_ADDR\_ACCL, [658](#)
  - DI\_ADDR\_GYRO, [658](#)
- Direct Command functions, [327](#)
- RemoteDatalogRead, [328](#)
  - RemoteDatalogSetTimes, [329](#)
  - RemoteGetBatteryLevel, [329](#)
  - RemoteGetConnectionCount, [329](#)
  - RemoteGetConnectionName, [329](#)
  - RemoteGetContactCount, [329](#)
  - RemoteGetContactName, [330](#)
  - RemoteGetCurrentProgramName, [330](#)
- RemoteGetInputValues, [330](#)
- RemoteGetOutputState, [331](#)
- RemoteGetProperty, [331](#)
- RemoteKeepAlive, [331](#)
- RemoteLowspeedGetStatus, [331](#)
- RemoteLowspeedRead, [332](#)
- RemoteLowspeedWrite, [332](#)
- RemoteMessageRead, [332](#)
- RemoteMessageWrite, [332](#)
- RemotePlaySoundFile, [333](#)
- RemotePlayTone, [333](#)
- RemoteResetMotorPosition, [333](#)
- RemoteResetScaledValue, [333](#)
- RemoteResetTachoCount, [334](#)
- RemoteSetInputModule, [334](#)
- RemoteSetOutputState, [334](#)
- RemoteSetProperty, [335](#)
- RemoteStartProgram, [335](#)
- RemoteStopProgram, [335](#)
- RemoteStopSound, [335](#)
- Display contrast constants, [495](#)
- Display flags, [494](#)
- DISPLAY\_BUSY, [494](#)
  - DISPLAY\_ON, [494](#)
  - DISPLAY\_POPUP, [494](#)
  - DISPLAY\_REFRESH, [494](#)
- Display module, [79](#)
- Display module constants, [482](#)
- BITMAP\_1, [483](#)
  - BITMAP\_2, [483](#)
  - BITMAP\_3, [483](#)
  - BITMAP\_4, [483](#)
  - BITMAPS, [483](#)
  - DISPLAY\_HEIGHT, [483](#)
  - DISPLAY\_MENUICONS\_Y, [483](#)
  - DISPLAY\_WIDTH, [483](#)
  - FRAME\_SELECT, [484](#)
  - MENUICON\_CENTER, [484](#)
  - MENUICON\_LEFT, [484](#)
  - MENUICON\_RIGHT, [484](#)
  - MENUICONS, [484](#)
  - MENUTEXT, [484](#)
  - SCREEN\_BACKGROUND, [484](#)
  - SCREEN\_LARGE, [484](#)
  - SCREEN\_MODE\_CLEAR, [484](#)
  - SCREEN\_MODE\_RESTORE, [484](#)
  - SCREEN\_SMALL, [484](#)
  - SCREENS, [484](#)
  - SPECIALS, [485](#)
  - STATUSICON\_BATTERY, [485](#)
  - STATUSICON\_BLUETOOTH, [485](#)
  - STATUSICON\_USB, [485](#)
  - STATUSICON\_VM, [485](#)
  - STATUSICONS, [485](#)

STATUSTEXT, 485  
STEPICON\_1, 485  
STEPICON\_2, 485  
STEPICON\_3, 485  
STEPICON\_4, 485  
STEPICON\_5, 485  
STEPICONS, 485  
STEPLINE, 485  
TOPLINE, 486  
Display module functions, 250  
  CircleOut, 252  
  CircleOutEx, 252  
  ClearLine, 252  
  ClearScreen, 252  
  EllipseOut, 253  
  EllipseOutEx, 253  
  FontNumOut, 253  
  FontNumOutEx, 254  
  FontTextOut, 254  
  FontTextOutEx, 255  
  GetDisplayContrast, 255  
  GetDisplayDisplay, 255  
  GetDisplayEraseMask, 255  
  GetDisplayFlags, 256  
  GetDisplayFont, 256  
  GetDisplayNormal, 256  
  GetDisplayPopup, 256  
  GetDisplayTextLinesCenterFlags, 257  
  GetDisplayUpdateMask, 257  
  GraphicArrayOut, 257  
  GraphicArrayOutEx, 257  
  GraphicOut, 258  
  GraphicOutEx, 258  
  LineOut, 258  
  LineOutEx, 259  
  NumOut, 259  
  NumOutEx, 259  
  PointOut, 260  
  PointOutEx, 260  
  PolyOut, 260  
  PolyOutEx, 261  
  RectOut, 261  
  RectOutEx, 261  
  SetDisplayContrast, 262  
  SetDisplayDisplay, 262  
  SetDisplayEraseMask, 262  
  SetDisplayFlags, 262  
  SetDisplayFont, 262  
  SetDisplayNormal, 263  
  SetDisplayPopup, 263  
  SetDisplayTextLinesCenterFlags, 263  
  SetDisplayUpdateMask, 263  
  TextOut, 263  
  TextOutEx, 264  
Display module IOMAP offsets, 498  
  DisplayOffsetContrast, 498  
  DisplayOffsetDisplay, 498  
  DisplayOffsetEraseMask, 498  
  DisplayOffsetFlags, 498  
  DisplayOffsetNormal, 498  
  DisplayOffsetPBitmaps, 498  
  DisplayOffsetPFont, 499  
  DisplayOffsetPFunc, 499  
  DisplayOffsetPMenulIcons, 499  
  DisplayOffsetPMenuText, 499  
  DisplayOffsetPScreens, 499  
  DisplayOffsetPStatusIcons, 499  
  DisplayOffsetPStatusText, 499  
  DisplayOffsetPStepIcons, 499  
  DisplayOffsetPTextLines, 499  
  DisplayOffsetPopup, 499  
  DisplayOffsetStatusIcons, 499  
  DisplayOffsetStepIcons, 499  
  DisplayOffsetTextLinesCenterFlags, 499  
  DisplayOffsetUpdateMask, 499  
DisplayExecuteFunction  
  NBCCommon.h, 763  
  System Call function constants, 367  
DisplayExecuteFunction constants, 487  
  DISPLAY\_CHAR, 487  
  DISPLAY\_ERASE\_ALL, 487  
  DISPLAY\_FRAME, 487  
  DISPLAY\_PIXEL, 487  
DisplayModuleID  
  NBCCommon.h, 763  
  NXT firmware module IDs, 201  
DisplayModuleName  
  NBCCommon.h, 763  
  NXT firmware module names, 199  
DisplayOffsetContrast  
  Display module IOMAP offsets, 498  
  NBCCommon.h, 763  
DisplayOffsetDisplay  
  Display module IOMAP offsets, 498  
  NBCCommon.h, 763  
DisplayOffsetEraseMask  
  Display module IOMAP offsets, 498  
  NBCCommon.h, 763  
DisplayOffsetFlags  
  Display module IOMAP offsets, 498  
  NBCCommon.h, 763  
DisplayOffsetNormal  
  Display module IOMAP offsets, 498  
  NBCCommon.h, 764  
DisplayOffsetPBitmaps  
  Display module IOMAP offsets, 498  
  NBCCommon.h, 764  
DisplayOffsetPFont

Display module IOMAP offsets, 499  
NBCCCommon.h, 764

DisplayOffsetPFunc  
    Display module IOMAP offsets, 499  
    NBCCCommon.h, 764

DisplayOffsetPMenulicons  
    Display module IOMAP offsets, 499  
    NBCCCommon.h, 764

DisplayOffsetPMenuText  
    Display module IOMAP offsets, 499  
    NBCCCommon.h, 764

DisplayOffsetPScreens  
    Display module IOMAP offsets, 499  
    NBCCCommon.h, 764

DisplayOffsetPStatusIcons  
    Display module IOMAP offsets, 499  
    NBCCCommon.h, 764

DisplayOffsetPStatusText  
    Display module IOMAP offsets, 499  
    NBCCCommon.h, 764

DisplayOffsetPStepIcons  
    Display module IOMAP offsets, 499  
    NBCCCommon.h, 764

DisplayOffsetPTextLines  
    Display module IOMAP offsets, 499  
    NBCCCommon.h, 764

DisplayOffsetPopup  
    Display module IOMAP offsets, 499  
    NBCCCommon.h, 764

DisplayOffsetStatusIcons  
    Display module IOMAP offsets, 499  
    NBCCCommon.h, 764

DisplayOffsetStepIcons  
    Display module IOMAP offsets, 499  
    NBCCCommon.h, 764

DisplayOffsetTextLinesCenterFlags  
    Display module IOMAP offsets, 499  
    NBCCCommon.h, 765

DisplayOffsetUpdateMask  
    Display module IOMAP offsets, 499  
    NBCCCommon.h, 765

DrawCircle  
    NBCCCommon.h, 767  
    System Call function constants, 367

DrawEllipse  
    NBCCCommon.h, 768  
    System Call function constants, 368

DrawFont  
    NBCCCommon.h, 768  
    System Call function constants, 368

DrawGraphic  
    NBCCCommon.h, 768  
    System Call function constants, 368

DrawGraphicArray

NBCCCommon.h, 768  
System Call function constants, 368

DrawLine  
    NBCCCommon.h, 768  
    System Call function constants, 368

DrawPoint  
    NBCCCommon.h, 768  
    System Call function constants, 368

DrawPolygon  
    NBCCCommon.h, 768  
    System Call function constants, 368

DrawRect  
    NBCCCommon.h, 768  
    System Call function constants, 368

DrawText  
    NBCCCommon.h, 768  
    System Call function constants, 368

Drawing option constants, 489  
    DRAW\_OPT\_CLEAR, 490  
    DRAW\_OPT\_CLEAR\_EOL, 490  
    DRAW\_OPT\_INVERT, 490  
    DRAW\_OPT\_NORMAL, 490

E-Meter sensor constants, 480  
    EMETER\_REG\_AIN, 480  
    EMETER\_REG\_AOUT, 480  
    EMETER\_REG\_JOULES, 480  
    EMETER\_REG\_VIN, 480  
    EMETER\_REG\_VOUT, 480  
    EMETER\_REG\_WIN, 480  
    EMETER\_REG\_WOUT, 480

EMETER\_REG\_AIN  
    E-Meter sensor constants, 480  
    NBCCCommon.h, 768

EMETER\_REG\_AOUT  
    E-Meter sensor constants, 480  
    NBCCCommon.h, 768

EMETER\_REG\_JOULES  
    E-Meter sensor constants, 480  
    NBCCCommon.h, 768

EMETER\_REG\_VIN  
    E-Meter sensor constants, 480  
    NBCCCommon.h, 768

EMETER\_REG\_VOUT  
    E-Meter sensor constants, 480  
    NBCCCommon.h, 768

EMETER\_REG\_WIN  
    E-Meter sensor constants, 480  
    NBCCCommon.h, 769

EMETER\_REG\_WOUT  
    E-Meter sensor constants, 480  
    NBCCCommon.h, 769

EOF  
    Loader module constants, 393

NBCCCommon.h, 769  
EQ  
Comparison Constants, 61  
ERR\_ARG  
Fatal errors, 382  
NBCCCommon.h, 769  
ERR\_BAD\_POOL\_SIZE  
Fatal errors, 382  
NBCCCommon.h, 769  
ERR\_BAD\_PTR  
Fatal errors, 382  
NBCCCommon.h, 769  
ERR\_CLUMP\_COUNT  
Fatal errors, 382  
NBCCCommon.h, 769  
ERR\_COMM\_BUS\_ERR  
Communications specific errors, 385  
NBCCCommon.h, 769  
ERR\_DEFAULT\_OFFSETS  
Fatal errors, 382  
ERR\_FILE  
Fatal errors, 382  
NBCCCommon.h, 769  
ERR\_INSANE\_OFFSET  
Fatal errors, 382  
NBCCCommon.h, 769  
ERR\_INSTR  
Fatal errors, 383  
NBCCCommon.h, 770  
ERR\_INVALID\_FIELD  
General errors, 384  
NBCCCommon.h, 770  
ERR\_INVALID\_PORT  
General errors, 384  
NBCCCommon.h, 770  
ERR\_INVALID\_QUEUE  
General errors, 384  
NBCCCommon.h, 770  
ERR\_INVALID\_SIZE  
General errors, 384  
NBCCCommon.h, 770  
ERR\_LOADER\_ERR  
Fatal errors, 383  
NBCCCommon.h, 770  
ERR\_MEM  
Fatal errors, 383  
NBCCCommon.h, 770  
ERR\_MEMMGR\_FAIL  
Fatal errors, 383  
NBCCCommon.h, 770  
ERR\_NO\_ACTIVE\_CLUMP  
Fatal errors, 383  
ERR\_NO\_CODE  
Fatal errors, 383  
NBCCCommon.h, 770  
ERR\_NO\_PROG  
General errors, 384  
NBCCCommon.h, 770  
ERR\_NON\_FATAL  
Fatal errors, 383  
NBCCCommon.h, 770  
ERR\_RC\_BAD\_PACKET  
NBCCCommon.h, 770  
Remote control (direct commands) errors, 386  
ERR\_RC\_FAILED  
NBCCCommon.h, 770  
Remote control (direct commands) errors, 386  
ERR\_RC\_ILLEGAL\_VAL  
Remote control (direct commands) errors, 386  
ERR\_RC\_UNKNOWN\_CMD  
Remote control (direct commands) errors, 386  
ERR\_SPOTCHECK\_FAIL  
Fatal errors, 383  
NBCCCommon.h, 771  
ERR\_VER  
Fatal errors, 383  
NBCCCommon.h, 771  
EllipseOut  
Display module functions, 253  
EllipseOutEx  
Display module functions, 253  
FALSE  
Miscellaneous NBC/NXC constants, 203  
NBCCCommon.h, 771  
FRAME\_SELECT  
Display module constants, 484  
NBCCCommon.h, 772  
FREQUENCY\_MAX  
NBCCCommon.h, 772  
Sound module miscellaneous constants, 408  
FREQUENCY\_MIN  
NBCCCommon.h, 772  
Sound module miscellaneous constants, 408  
Fatal errors, 382  
ERR\_ARG, 382  
ERR\_BAD\_POOL\_SIZE, 382  
ERR\_BAD\_PTR, 382  
ERR\_CLUMP\_COUNT, 382  
ERR\_DEFAULT\_OFFSETS, 382  
ERR\_FILE, 382  
ERR\_INSANE\_OFFSET, 382  
ERR\_INSTR, 383  
ERR\_LOADER\_ERR, 383  
ERR\_MEM, 383  
ERR\_MEMMGR\_FAIL, 383  
ERR\_NO\_ACTIVE\_CLUMP, 383  
ERR\_NO\_CODE, 383

ERR\_NON\_FATAL, 383  
ERR\_SPOTCHECK\_FAIL, 383  
ERR\_VER, 383  
FileClose  
    NBCCCommon.h, 771  
    System Call function constants, 368  
FileDelete  
    NBCCCommon.h, 771  
    System Call function constants, 368  
FileFindFirst  
    NBCCCommon.h, 771  
    System Call function constants, 368  
FileFindNext  
    NBCCCommon.h, 771  
    System Call function constants, 368  
FileOpenAppend  
    NBCCCommon.h, 771  
    System Call function constants, 368  
FileOpenRead  
    NBCCCommon.h, 771  
    System Call function constants, 369  
FileOpenReadLinear  
    NBCCCommon.h, 771  
    System Call function constants, 369  
FileOpenWrite  
    NBCCCommon.h, 771  
    System Call function constants, 369  
FileOpenWriteLinear  
    NBCCCommon.h, 771  
    System Call function constants, 369  
FileOpenWriteNonLinear  
    NBCCCommon.h, 772  
    System Call function constants, 369  
FileRead  
    NBCCCommon.h, 772  
    System Call function constants, 369  
FileRename  
    NBCCCommon.h, 772  
    System Call function constants, 369  
FileResize  
    NBCCCommon.h, 772  
    System Call function constants, 369  
FileResolveHandle  
    NBCCCommon.h, 772  
    System Call function constants, 369  
FileSeek  
    NBCCCommon.h, 772  
    System Call function constants, 369  
FileTell  
    NBCCCommon.h, 772  
    System Call function constants, 369  
FileWrite  
    NBCCCommon.h, 772  
    System Call function constants, 369  
FindFirstFile  
    Loader module functions, 350  
FindNextFile  
    Loader module functions, 351  
Float  
    Output module functions, 214  
Font drawing option constants, 492  
    DRAW\_OPT\_FONT\_WRAP, 493  
FontNumOut  
    Display module functions, 253  
FontNumOutEx  
    Display module functions, 254  
FontTextOut  
    Display module functions, 254  
FontTextOutEx  
    Display module functions, 255  
ForceOff  
    Ui module functions, 290  
GL\_CAMERA\_DEPTH  
    Graphics library settings, 692  
    NBCCCommon.h, 772  
GL\_CIRCLE  
    Graphics library begin modes, 689  
    NBCCCommon.h, 772  
GL\_CIRCLE\_SIZE  
    Graphics library settings, 692  
    NBCCCommon.h, 773  
GL\_CULL\_BACK  
    Graphics library cull mode, 693  
    NBCCCommon.h, 773  
GL\_CULL\_FRONT  
    Graphics library cull mode, 693  
    NBCCCommon.h, 773  
GL\_CULL\_MODE  
    Graphics library settings, 692  
    NBCCCommon.h, 773  
GL\_CULL\_NONE  
    Graphics library cull mode, 693  
    NBCCCommon.h, 773  
GL\_LINE  
    Graphics library begin modes, 689  
    NBCCCommon.h, 773  
GL\_POINT  
    Graphics library begin modes, 689  
    NBCCCommon.h, 773  
GL\_POLYGON  
    Graphics library begin modes, 689  
    NBCCCommon.h, 773  
GL\_ROTATE\_X  
    Graphics library actions, 690  
    NBCCCommon.h, 773  
GL\_ROTATE\_Y  
    Graphics library actions, 690

NBCCCommon.h, 773  
GL\_ROTATE\_Z  
    Graphics library actions, 690  
    NBCCCommon.h, 773  
GL\_SCALE\_X  
    Graphics library actions, 690  
    NBCCCommon.h, 773  
GL\_SCALE\_Y  
    Graphics library actions, 690  
    NBCCCommon.h, 773  
GL\_SCALE\_Z  
    Graphics library actions, 690  
    NBCCCommon.h, 773  
GL\_TRANSLATE\_X  
    Graphics library actions, 690  
    NBCCCommon.h, 774  
GL\_TRANSLATE\_Y  
    Graphics library actions, 690  
    NBCCCommon.h, 774  
GL\_TRANSLATE\_Z  
    Graphics library actions, 690  
    NBCCCommon.h, 774  
GL\_ZOOM\_FACTOR  
    Graphics library settings, 692  
    NBCCCommon.h, 774  
GT  
    Comparison Constants, 61  
GTEQ  
    Comparison Constants, 61  
General errors, 384  
    ERR\_INVALID\_FIELD, 384  
    ERR\_INVALID\_PORT, 384  
    ERR\_INVALID\_QUEUE, 384  
    ERR\_INVALID\_SIZE, 384  
    ERR\_NO\_PROG, 384  
GetAbortFlag  
    Ui module functions, 290  
GetBTConnectionAddress  
    Comm module functions, 302  
GetBTConnectionClass  
    Comm module functions, 303  
GetBTConnectionHandleNum  
    Comm module functions, 303  
GetBTConnectionLinkQuality  
    Comm module functions, 303  
GetBTConnectionName  
    Comm module functions, 303  
GetBTConnectionPinCode  
    Comm module functions, 303  
GetBTConnectionStreamStatus  
    Comm module functions, 304  
GetBTDataMode  
    Comm module functions, 304  
GetBTDeviceAddress  
    Comm module functions, 304  
GetBTDeviceClass  
    Comm module functions, 304  
GetBTDeviceCount  
    Comm module functions, 305  
GetBTDeviceName  
    Comm module functions, 305  
GetBTDeviceNameCount  
    Comm module functions, 305  
GetBTDeviceStatus  
    Comm module functions, 305  
GetBTInputBuffer  
    Comm module functions, 306  
GetBTInputBufferInPtr  
    Comm module functions, 306  
GetBTInputBufferOutPtr  
    Comm module functions, 306  
GetBTOutputBuffer  
    Comm module functions, 306  
GetBTOutputBufferInPtr  
    Comm module functions, 307  
GetBTOutputBufferOutPtr  
    Comm module functions, 307  
GetBatteryLevel  
    Ui module functions, 290  
GetBatteryState  
    Ui module functions, 291  
GetBluetoothState  
    Ui module functions, 291  
GetBrickDataAddress  
    Comm module functions, 301  
GetBrickDataBluecoreVersion  
    Comm module functions, 301  
GetBrickDataBtHardwareStatus  
    Comm module functions, 301  
GetBrickDataBtStateStatus  
    Comm module functions, 302  
GetBrickDataName  
    Comm module functions, 302  
GetBrickDataTimeoutValue  
    Comm module functions, 302  
GetButtonLongPressCount  
    Button module functions, 285  
GetButtonLongReleaseCount  
    Button module functions, 286  
GetButtonModuleValue  
    Command module functions, 273  
GetButtonPressCount  
    Button module functions, 286  
GetButtonReleaseCount  
    Button module functions, 286  
GetButtonShortReleaseCount  
    Button module functions, 286  
GetButtonState

Button module functions, 286  
GetCommModuleBytes  
    Command module functions, 274  
GetCommModuleValue  
    Command module functions, 274  
GetCommandFlags  
    Ui module functions, 291  
GetCommandModuleBytes  
    Command module functions, 273  
GetCommandModuleValue  
    Command module functions, 273  
GetDisplayContrast  
    Display module functions, 255  
GetDisplayDisplay  
    Display module functions, 255  
GetDisplayEraseMask  
    Display module functions, 255  
GetDisplayFlags  
    Display module functions, 256  
GetDisplayFont  
    Display module functions, 256  
GetDisplayModuleBytes  
    Command module functions, 274  
GetDisplayModuleValue  
    Command module functions, 274  
GetDisplayNormal  
    Display module functions, 256  
GetDisplayPopup  
    Display module functions, 256  
GetDisplayTextLinesCenterFlags  
    Display module functions, 257  
GetDisplayUpdateMask  
    Display module functions, 257  
GetFirstTick  
    Command module functions, 275  
GetFreeMemory  
    Loader module functions, 351  
GetHSAddress  
    Comm module functions, 307  
GetHSDaDataMode  
    Comm module functions, 308  
GetHSFlags  
    Comm module functions, 308  
GetHSInputBuffer  
    Comm module functions, 308  
GetHSInputBufferInPtr  
    Comm module functions, 308  
GetHSInputBufferOutPtr  
    Comm module functions, 309  
GetHSMode  
    Comm module functions, 309  
GetHSOutputBuffer  
    Comm module functions, 309  
GetHSOutputBufferInPtr

    Comm module functions, 310  
GetHSOutputBufferOutPtr  
    Comm module functions, 310  
GetHSSpeed  
    Comm module functions, 310  
GetHSState  
    Comm module functions, 310  
GetIOMapBytes  
    Command module functions, 275  
GetIOMapBytesByID  
    Command module functions, 275  
GetIOMapValue  
    Command module functions, 276  
GetIOMapValueByID  
    Command module functions, 276  
GetInColorADRaw  
    Input module functions, 228  
GetInColorBoolean  
    Input module functions, 228  
GetInColorCallLimits  
    Input module functions, 229  
GetInColorCalibration  
    Input module functions, 228  
GetInColorCalibrationState  
    Input module functions, 229  
GetInColorSensorRaw  
    Input module functions, 229  
GetInColorSensorValue  
    Input module functions, 230  
GetInCustomActiveStatus  
    Input module functions, 230  
GetInCustomPercentFullScale  
    Input module functions, 230  
GetInCustomZeroOffset  
    Input module functions, 230  
GetInDigiPinsDirection  
    Input module functions, 230  
GetInDigiPinsOutputLevel  
    Input module functions, 231  
GetInDigiPinsStatus  
    Input module functions, 231  
GetInSensorBoolean  
    Input module functions, 231  
GetInputModuleValue  
    Command module functions, 275  
GetLSChannelState  
    Low level LowSpeed module functions, 246  
GetLSErrorType  
    Low level LowSpeed module functions, 247  
GetLSInputBuffer  
    Low level LowSpeed module functions, 247  
GetLSInputBufferBytesToRx  
    Low level LowSpeed module functions, 247  
GetLSInputBufferInPtr

Low level LowSpeed module functions, 247  
GetLSInputBufferOutPtr  
    Low level LowSpeed module functions, 247  
GetLSMode  
    Low level LowSpeed module functions, 248  
GetLSNoRestartOnRead  
    Low level LowSpeed module functions, 248  
GetLSSOutputBuffer  
    Low level LowSpeed module functions, 248  
GetLSSOutputBufferBytesToRx  
    Low level LowSpeed module functions, 248  
GetLSSOutputBufferInPtr  
    Low level LowSpeed module functions, 248  
GetLSSpeed  
    Low level LowSpeed module functions, 249  
GetLSSState  
    Low level LowSpeed module functions, 249  
GetLastResponseInfo  
    Command module functions, 276  
GetLoaderModuleValue  
    Command module functions, 277  
GetLowSpeedModuleBytes  
    Command module functions, 277  
GetLowSpeedModuleValue  
    Command module functions, 277  
GetMemoryInfo  
    Command module functions, 277  
GetOnBrickProgramPointer  
    Ui module functions, 291  
GetOutPwnFreq  
    Output module functions, 214  
GetOutRegulationOptions  
    Output module functions, 215  
GetOutRegulationTime  
    Output module functions, 215  
GetOutputModuleValue  
    Command module functions, 278  
GetRechargeableBattery  
    Ui module functions, 291  
GetSleepTimeout  
    Ui module functions, 291  
GetSleepTimer  
    Ui module functions, 292  
GetSoundDuration  
    Sound module functions, 266  
GetSoundFrequency  
    Sound module functions, 266  
GetSoundMode  
    Sound module functions, 266  
GetSoundModuleValue  
    Command module functions, 278  
GetSoundSampleRate  
    Sound module functions, 266  
GetSoundState  
    Sound module functions, 267  
GetSoundVolume  
    Sound module functions, 267  
GetStartTick  
    NBBCommon.h, 772  
    System Call function constants, 369  
GetUIButton  
    Ui module functions, 292  
GetUIModuleValue  
    Command module functions, 278  
GetUIState  
    Ui module functions, 292  
GetUSBInputBuffer  
    Comm module functions, 311  
GetUSBInputBufferInPtr  
    Comm module functions, 311  
GetUSBInputBufferOutPtr  
    Comm module functions, 311  
GetUSBOOutputBuffer  
    Comm module functions, 311  
GetUSBOOutputBufferInPtr  
    Comm module functions, 312  
GetUSBOOutputBufferOutPtr  
    Comm module functions, 312  
GetUSBPollBuffer  
    Comm module functions, 312  
GetUSBPollBufferInPtr  
    Comm module functions, 312  
GetUSBPollBufferOutPtr  
    Comm module functions, 313  
GetUSBState  
    Comm module functions, 313  
GetUsbState  
    Ui module functions, 292  
GetVMRunState  
    Ui module functions, 292  
GetVolume  
    Ui module functions, 292  
glAddToAngleX  
    A simple 3D graphics library, 207  
glAddToAngleY  
    A simple 3D graphics library, 207  
glAddToAngleZ  
    A simple 3D graphics library, 207  
glAddVertex  
    A simple 3D graphics library, 207  
glBegin  
    A simple 3D graphics library, 208  
glBeginObject  
    A simple 3D graphics library, 208  
glBeginRender  
    A simple 3D graphics library, 208

glBox  
    A simple 3D graphics library, [208](#)

glCallObject  
    A simple 3D graphics library, [208](#)

glCos32768  
    A simple 3D graphics library, [209](#)

glCube  
    A simple 3D graphics library, [209](#)

glEnd  
    A simple 3D graphics library, [209](#)

glEndObject  
    A simple 3D graphics library, [209](#)

glFinishRender  
    A simple 3D graphics library, [209](#)

glInit  
    A simple 3D graphics library, [209](#)

glObjectAction  
    A simple 3D graphics library, [209](#)

glPyramid  
    A simple 3D graphics library, [210](#)

glSet  
    A simple 3D graphics library, [210](#)

glSetAngleX  
    A simple 3D graphics library, [210](#)

glSetAngleY  
    A simple 3D graphics library, [210](#)

glSetAngleZ  
    A simple 3D graphics library, [210](#)

glSin32768  
    A simple 3D graphics library, [211](#)

GraphicArrayOut  
    Display module functions, [257](#)

GraphicArrayOutEx  
    Display module functions, [257](#)

GraphicOut  
    Display module functions, [258](#)

GraphicOutEx  
    Display module functions, [258](#)

Graphics library actions, [690](#)

    GL\_ROTATE\_X, [690](#)  
    GL\_ROTATE\_Y, [690](#)  
    GL\_ROTATE\_Z, [690](#)  
    GL\_SCALE\_X, [690](#)  
    GL\_SCALE\_Y, [690](#)  
    GL\_SCALE\_Z, [690](#)  
    GL\_TRANSLATE\_X, [690](#)  
    GL\_TRANSLATE\_Y, [690](#)  
    GL\_TRANSLATE\_Z, [690](#)

Graphics library begin modes, [689](#)

    GL\_CIRCLE, [689](#)  
    GL\_LINE, [689](#)  
    GL\_POINT, [689](#)  
    GL\_POLYGON, [689](#)

Graphics library cull mode, [693](#)

    GL\_CULL\_BACK, [693](#)  
    GL\_CULL\_FRONT, [693](#)  
    GL\_CULL\_NONE, [693](#)

Graphics library settings, [692](#)

    GL\_CAMERA\_DEPTH, [692](#)  
    GL\_CIRCLE\_SIZE, [692](#)  
    GL\_CULL\_MODE, [692](#)  
    GL\_ZOOM\_FACTOR, [692](#)

HS\_ADDRESS\_1  
    Hi-speed port address constants, [525](#)  
    NBCCCommon.h, [774](#)

HS\_ADDRESS\_2  
    Hi-speed port address constants, [525](#)  
    NBCCCommon.h, [774](#)

HS\_ADDRESS\_3  
    Hi-speed port address constants, [525](#)  
    NBCCCommon.h, [774](#)

HS\_ADDRESS\_4  
    Hi-speed port address constants, [525](#)  
    NBCCCommon.h, [774](#)

HS\_ADDRESS\_5  
    Hi-speed port address constants, [525](#)  
    NBCCCommon.h, [774](#)

HS\_ADDRESS\_6  
    Hi-speed port address constants, [525](#)  
    NBCCCommon.h, [774](#)

HS\_ADDRESS\_7  
    Hi-speed port address constants, [525](#)  
    NBCCCommon.h, [774](#)

HS\_ADDRESS\_8  
    Hi-speed port address constants, [525](#)  
    NBCCCommon.h, [774](#)

HS\_ADDRESS\_ALL  
    Hi-speed port address constants, [526](#)  
    NBCCCommon.h, [774](#)

HS\_BAUD\_115200  
    Hi-speed port baud rate constants, [518](#)  
    NBCCCommon.h, [774](#)

HS\_BAUD\_1200  
    Hi-speed port baud rate constants, [518](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_14400  
    Hi-speed port baud rate constants, [518](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_19200  
    Hi-speed port baud rate constants, [518](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_230400  
    Hi-speed port baud rate constants, [518](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_2400  
    Hi-speed port baud rate constants, [518](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_28800  
    Hi-speed port baud rate constants, [519](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_3600  
    Hi-speed port baud rate constants, [519](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_38400  
    Hi-speed port baud rate constants, [519](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_460800  
    Hi-speed port baud rate constants, [519](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_4800  
    Hi-speed port baud rate constants, [519](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_57600  
    Hi-speed port baud rate constants, [519](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_7200  
    Hi-speed port baud rate constants, [519](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_76800  
    Hi-speed port baud rate constants, [519](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_921600  
    Hi-speed port baud rate constants, [519](#)  
    NBCCCommon.h, [775](#)

HS\_BAUD\_9600  
    Hi-speed port baud rate constants, [519](#)  
    NBCCCommon.h, [776](#)

HS\_BAUD\_DEFAULT  
    Hi-speed port baud rate constants, [519](#)  
    NBCCCommon.h, [776](#)

HS\_BYTES\_REMAINING  
    Hi-speed port state constants, [516](#)  
    NBCCCommon.h, [776](#)

HS\_CMD\_READY  
    Comm module status code constants, [531](#)  
    NBCCCommon.h, [776](#)

HS\_CTRL\_EXIT  
    Hi-speed port SysCommHSControl constants, [517](#)  
    NBCCCommon.h, [776](#)

HS\_CTRL\_INIT  
    Hi-speed port SysCommHSControl constants, [517](#)  
    NBCCCommon.h, [776](#)

HS\_CTRL\_UART  
    Hi-speed port SysCommHSControl constants, [517](#)  
    NBCCCommon.h, [776](#)

HS\_DEFAULT  
    Hi-speed port state constants, [516](#)  
    NBCCCommon.h, [776](#)

HS\_DISABLE  
    Hi-speed port state constants, [516](#)  
    NBCCCommon.h, [776](#)

HS\_ENABLE  
    Hi-speed port state constants, [516](#)  
    NBCCCommon.h, [776](#)

HS\_INIT\_RECEIVER  
    Hi-speed port state constants, [516](#)  
    NBCCCommon.h, [776](#)

HS\_INITIALISE  
    Hi-speed port state constants, [516](#)  
    NBCCCommon.h, [776](#)

HS\_MODE\_10\_STOP  
    Hi-speed port stop bits constants, [522](#)  
    NBCCCommon.h, [776](#)

HS\_MODE\_15\_STOP  
    Hi-speed port stop bits constants, [522](#)  
    NBCCCommon.h, [776](#)

HS\_MODE\_20\_STOP  
    Hi-speed port stop bits constants, [522](#)  
    NBCCCommon.h, [777](#)

HS\_MODE\_5\_DATA  
    Hi-speed port data bits constants, [521](#)  
    NBCCCommon.h, [777](#)

HS\_MODE\_6\_DATA  
    Hi-speed port data bits constants, [521](#)  
    NBCCCommon.h, [777](#)

HS\_MODE\_7\_DATA  
    Hi-speed port data bits constants, [521](#)  
    NBCCCommon.h, [777](#)

HS\_MODE\_7E1  
    Hi-speed port combined UART constants, [524](#)  
    NBCCCommon.h, [777](#)

HS\_MODE\_8\_DATA  
    Hi-speed port data bits constants, [521](#)  
    NBCCCommon.h, [777](#)

HS\_MODE\_8N1  
    Hi-speed port combined UART constants, [524](#)  
    NBCCCommon.h, [777](#)

HS\_MODE\_DEFAULT  
    Hi-speed port UART mode constants, [520](#)  
    NBCCCommon.h, [777](#)

HS\_MODE\_E\_PARITY  
    Hi-speed port parity constants, [523](#)  
    NBCCCommon.h, [777](#)

HS\_MODE\_M\_PARITY  
    Hi-speed port parity constants, [523](#)  
    NBCCCommon.h, [777](#)

HS\_MODE\_MASK  
    Hi-speed port UART mode constants, [520](#)  
    NBCCCommon.h, [777](#)

HS\_MODE\_N\_PARITY  
    Hi-speed port parity constants, [523](#)  
    NBCCCommon.h, [777](#)

HS\_MODE\_O\_PARITY  
    Hi-speed port parity constants, [523](#)  
    NBCCCommon.h, [777](#)

HS\_MODE\_S\_PARITY  
    Hi-speed port parity constants, [523](#)  
    NBCCommon.h, [779](#)

HS\_MODE\_UART\_RS232  
    Hi-speed port UART mode constants, [520](#)  
    NBCCommon.h, [778](#)

HS\_MODE\_UART\_RS485  
    Hi-speed port UART mode constants, [520](#)  
    NBCCommon.h, [778](#)

HS\_SEND\_DATA  
    Hi-speed port state constants, [516](#)  
    NBCCommon.h, [778](#)

HS\_UART\_MASK  
    Hi-speed port UART mode constants, [520](#)  
    NBCCommon.h, [778](#)

HS\_UPDATE  
    Hi-speed port flags constants, [515](#)  
    NBCCommon.h, [778](#)

HT\_ADDR\_ACCEL  
    HiTechnic device constants, [582](#)  
    NBCCommon.h, [778](#)

HT\_ADDR\_ANGLE  
    HiTechnic device constants, [582](#)  
    NBCCommon.h, [778](#)

HT\_ADDR\_BAROMETRIC  
    HiTechnic device constants, [583](#)  
    NBCCommon.h, [778](#)

HT\_ADDR\_COLOR  
    HiTechnic device constants, [583](#)  
    NBCCommon.h, [778](#)

HT\_ADDR\_COLOR2  
    HiTechnic device constants, [583](#)  
    NBCCommon.h, [778](#)

HT\_ADDR\_COMPASS  
    HiTechnic device constants, [583](#)  
    NBCCommon.h, [778](#)

HT\_ADDR\_IRLINK  
    HiTechnic device constants, [583](#)  
    NBCCommon.h, [778](#)

HT\_ADDR\_IRRECEIVER  
    HiTechnic device constants, [583](#)  
    NBCCommon.h, [778](#)

HT\_ADDR\_IRSEEKER  
    HiTechnic device constants, [583](#)  
    NBCCommon.h, [778](#)

HT\_ADDR\_IRSEEKER2  
    HiTechnic device constants, [583](#)  
    NBCCommon.h, [779](#)

HT\_ADDR\_PIR  
    HiTechnic device constants, [583](#)  
    NBCCommon.h, [779](#)

HT\_ADDR\_PROTOBOARD  
    HiTechnic device constants, [583](#)  
    NBCCommon.h, [779](#)

HT\_ADDR\_SUPERPRO  
    HiTechnic device constants, [583](#)  
    NBCCommon.h, [779](#)

HT\_CH1\_A  
    HiTechnic IRReceiver constants, [586](#)  
    NBCCommon.h, [779](#)

HT\_CH1\_B  
    HiTechnic IRReceiver constants, [586](#)  
    NBCCommon.h, [779](#)

HT\_CH2\_A  
    HiTechnic IRReceiver constants, [586](#)  
    NBCCommon.h, [779](#)

HT\_CH2\_B  
    HiTechnic IRReceiver constants, [586](#)  
    NBCCommon.h, [779](#)

HT\_CH3\_A  
    HiTechnic IRReceiver constants, [586](#)  
    NBCCommon.h, [779](#)

HT\_CH3\_B  
    HiTechnic IRReceiver constants, [586](#)  
    NBCCommon.h, [779](#)

HT\_CH4\_A  
    HiTechnic IRReceiver constants, [586](#)  
    NBCCommon.h, [779](#)

HT\_CH4\_B  
    HiTechnic IRReceiver constants, [586](#)  
    NBCCommon.h, [779](#)

HT\_CMD\_COLOR2\_50HZ  
    HiTechnic Color2 constants, [587](#)  
    NBCCommon.h, [779](#)

HT\_CMD\_COLOR2\_60HZ  
    HiTechnic Color2 constants, [587](#)  
    NBCCommon.h, [779](#)

HT\_CMD\_COLOR2\_BLCAL  
    HiTechnic Color2 constants, [587](#)

HT\_CMD\_COLOR2\_FAR  
    HiTechnic Color2 constants, [587](#)  
    NBCCommon.h, [780](#)

HT\_CMD\_COLOR2\_NEAR  
    HiTechnic Color2 constants, [587](#)  
    NBCCommon.h, [780](#)

HT\_CMD\_COLOR2\_RAW  
    HiTechnic Color2 constants, [588](#)  
    NBCCommon.h, [780](#)

HT\_CMD\_COLOR2\_WBCAL  
    HiTechnic Color2 constants, [588](#)

HTANGLE\_MODE\_RESET  
    HiTechnic Angle sensor constants, [589](#)  
    NBCCommon.h, [780](#)

HTANGLE\_REG\_ACDIR  
    HiTechnic Angle sensor constants, [589](#)  
    NBCCommon.h, [780](#)

HTANGLE\_REG\_DC01  
    HiTechnic Angle sensor constants, [589](#)

NBCCCommon.h, 780  
HTANGLE\_REG\_DC02  
    HiTechnic Angle sensor constants, 589  
    NBCCCommon.h, 781  
HTANGLE\_REG\_DC03  
    HiTechnic Angle sensor constants, 589  
    NBCCCommon.h, 781  
HTANGLE\_REG\_DC04  
    HiTechnic Angle sensor constants, 589  
    NBCCCommon.h, 781  
HTANGLE\_REG\_DC05  
    HiTechnic Angle sensor constants, 590  
    NBCCCommon.h, 781  
HTANGLE\_REG\_DCAVG  
    HiTechnic Angle sensor constants, 590  
    NBCCCommon.h, 781  
HTANGLE\_REG\_DCDIR  
    HiTechnic Angle sensor constants, 590  
    NBCCCommon.h, 781  
HTANGLE\_REG\_MODE  
    HiTechnic Angle sensor constants, 590  
    NBCCCommon.h, 781  
HTBAR\_REG\_COMMAND  
    HiTechnic Barometric sensor constants, 591  
    NBCCCommon.h, 781  
HTBAR\_REG\_PRESSURE  
    HiTechnic Barometric sensor constants, 591  
    NBCCCommon.h, 781  
HTIR2\_MODE\_1200  
    HiTechnic IRSeeker2 constants, 584  
    NBCCCommon.h, 781  
HTIR2\_MODE\_600  
    HiTechnic IRSeeker2 constants, 584  
    NBCCCommon.h, 781  
HTIR2\_REG\_AC01  
    HiTechnic IRSeeker2 constants, 584  
    NBCCCommon.h, 781  
HTIR2\_REG\_AC02  
    HiTechnic IRSeeker2 constants, 584  
    NBCCCommon.h, 782  
HTIR2\_REG\_AC03  
    HiTechnic IRSeeker2 constants, 584  
    NBCCCommon.h, 782  
HTIR2\_REG\_AC04  
    HiTechnic IRSeeker2 constants, 584  
    NBCCCommon.h, 782  
HTIR2\_REG\_AC05  
    HiTechnic IRSeeker2 constants, 584  
    NBCCCommon.h, 782  
HTIR2\_REG\_ACDIR  
    HiTechnic IRSeeker2 constants, 585  
    NBCCCommon.h, 782  
HTIR2\_REG\_DC01  
    HiTechnic IRSeeker2 constants, 585  
    NBCCCommon.h, 782  
HTIR2\_REG\_DC02  
    HiTechnic IRSeeker2 constants, 585  
    NBCCCommon.h, 782  
HTIR2\_REG\_DC03  
    HiTechnic IRSeeker2 constants, 585  
    NBCCCommon.h, 782  
HTIR2\_REG\_DC04  
    HiTechnic IRSeeker2 constants, 585  
    NBCCCommon.h, 782  
HTIR2\_REG\_DC05  
    HiTechnic IRSeeker2 constants, 585  
    NBCCCommon.h, 782  
HTIR2\_REG\_DCAVG  
    HiTechnic IRSeeker2 constants, 585  
    NBCCCommon.h, 782  
HTIR2\_REG\_DCDIR  
    HiTechnic IRSeeker2 constants, 585  
    NBCCCommon.h, 782  
HTIR2\_REG\_MODE  
    HiTechnic IRSeeker2 constants, 585  
    NBCCCommon.h, 782  
HTIRTrain  
    HiTechnic API Functions, 87  
HTPFCComboDirect  
    HiTechnic API Functions, 87  
HTPFCComboPWM  
    HiTechnic API Functions, 87  
HTPFRawOutput  
    HiTechnic API Functions, 87  
HTPFRRepeat  
    HiTechnic API Functions, 88  
HTPFSingleOutputCST  
    HiTechnic API Functions, 88  
HTPFSingleOutputPWM  
    HiTechnic API Functions, 88  
HTPFSinglePin  
    HiTechnic API Functions, 89  
HTPFTrain  
    HiTechnic API Functions, 89  
HTPIR\_REG\_DEADBAND  
    HiTechnic PIR sensor constants, 603  
    NBCCCommon.h, 782  
HTPIR\_REG\_READING  
    HiTechnic PIR sensor constants, 603  
    NBCCCommon.h, 783  
HTPROTO\_A0  
    HiTechnic Prototype board analog input constants,  
        594  
    NBCCCommon.h, 783  
HTPROTO\_A1  
    HiTechnic Prototype board analog input constants,  
        594  
    NBCCCommon.h, 783

HTPROTO\_A2  
    HiTechnic Prototype board analog input constants, 594  
    NBCCommon.h, 783

HTPROTO\_A3  
    HiTechnic Prototype board analog input constants, 594  
    NBCCommon.h, 783

HTPROTO\_A4  
    HiTechnic Prototype board analog input constants, 594  
    NBCCommon.h, 783

HTPROTO\_REG\_A0  
    HiTechnic Prototype board constants, 592  
    NBCCommon.h, 783

HTPROTO\_REG\_A1  
    HiTechnic Prototype board constants, 592  
    NBCCommon.h, 783

HTPROTO\_REG\_A2  
    HiTechnic Prototype board constants, 592  
    NBCCommon.h, 783

HTPROTO\_REG\_A3  
    HiTechnic Prototype board constants, 592  
    NBCCommon.h, 783

HTPROTO\_REG\_A4  
    HiTechnic Prototype board constants, 592  
    NBCCommon.h, 783

HTPROTO\_REG\_DCTRL  
    HiTechnic Prototype board constants, 592  
    NBCCommon.h, 783

HTPROTO\_REG\_DIN  
    HiTechnic Prototype board constants, 592  
    NBCCommon.h, 783

HTPROTO\_REG\_DOUT  
    HiTechnic Prototype board constants, 593  
    NBCCommon.h, 783

HTPROTO\_REG\_SRATE  
    HiTechnic Prototype board constants, 593  
    NBCCommon.h, 784

HTPowerFunctionCommand  
    HiTechnic API Functions, 89

HTRCXAddToDatalog  
    HiTechnic API Functions, 90

HTRCXBatteryLevel  
    HiTechnic API Functions, 90

HTRCXClearAllEvents  
    HiTechnic API Functions, 90

HTRCXClearCounter  
    HiTechnic API Functions, 90

HTRCXClearMsg  
    HiTechnic API Functions, 90

HTRCXClearSensor  
    HiTechnic API Functions, 90

HTRCXClearSound

HiTechnic API Functions, 91

HTRCXClearTimer  
    HiTechnic API Functions, 91

HTRCXCreateDatalog  
    HiTechnic API Functions, 91

HTRCXDecCounter  
    HiTechnic API Functions, 91

HTRCXDeleteSub  
    HiTechnic API Functions, 91

HTRCXDeleteSubs  
    HiTechnic API Functions, 91

HTRCXDeleteTask  
    HiTechnic API Functions, 92

HTRCXDeleteTasks  
    HiTechnic API Functions, 92

HTRCXDisableOutput  
    HiTechnic API Functions, 92

HTRCXEnableOutput  
    HiTechnic API Functions, 92

HTRCXEvent  
    HiTechnic API Functions, 92

HTRCXFloat  
    HiTechnic API Functions, 92

HTRCXFwd  
    HiTechnic API Functions, 93

HTRCXIncCounter  
    HiTechnic API Functions, 93

HTRCXInvertOutput  
    HiTechnic API Functions, 93

HTRCXMuteSound  
    HiTechnic API Functions, 93

HTRCXObvertOutput  
    HiTechnic API Functions, 93

HTRCXOff  
    HiTechnic API Functions, 93

HTRCXOn  
    HiTechnic API Functions, 94

HTRCXOnFor  
    HiTechnic API Functions, 94

HTRCXOnFwd  
    HiTechnic API Functions, 94

HTRCXOnRev  
    HiTechnic API Functions, 94

HTRCXPBTurnOff  
    HiTechnic API Functions, 94

HTRCXPing  
    HiTechnic API Functions, 94

HTRCXPlaySound  
    HiTechnic API Functions, 94

HTRCXPlayTone  
    HiTechnic API Functions, 95

HTRCXPlayToneVar  
    HiTechnic API Functions, 95

HTRCXPoll

HiTechnic API Functions, 95  
HTRCXPollMemory  
    HiTechnic API Functions, 95  
HTRCXRemote  
    HiTechnic API Functions, 95  
HTRCXRev  
    HiTechnic API Functions, 96  
HTRCXSelectDisplay  
    HiTechnic API Functions, 96  
HTRCXSelectProgram  
    HiTechnic API Functions, 96  
HTRCXSendSerial  
    HiTechnic API Functions, 96  
HTRCXSetDirection  
    HiTechnic API Functions, 96  
HTRCXSetEvent  
    HiTechnic API Functions, 97  
HTRCXSetGlobalDirection  
    HiTechnic API Functions, 97  
HTRCXSetGlobalOutput  
    HiTechnic API Functions, 97  
HTRCXSetIRLinkPort  
    HiTechnic API Functions, 97  
HTRCXSetMaxPower  
    HiTechnic API Functions, 97  
HTRCXSetMessage  
    HiTechnic API Functions, 98  
HTRCXSetOutput  
    HiTechnic API Functions, 98  
HTRCXSetPower  
    HiTechnic API Functions, 98  
HTRCXSetPriority  
    HiTechnic API Functions, 98  
HTRCXSetSensorMode  
    HiTechnic API Functions, 98  
HTRCXSetSensorType  
    HiTechnic API Functions, 99  
HTRCXSetSleepTime  
    HiTechnic API Functions, 99  
HTRCXSetTxPower  
    HiTechnic API Functions, 99  
HTRCXSetWatch  
    HiTechnic API Functions, 99  
HTRCXStartTask  
    HiTechnic API Functions, 99  
HTRCXStopAllTasks  
    HiTechnic API Functions, 99  
HTRCXStopTask  
    HiTechnic API Functions, 100  
HTRCXToggle  
    HiTechnic API Functions, 100  
HTRCXUnmuteSound  
    HiTechnic API Functions, 100  
HTSPRO\_A0  
    HiTechnic SuperPro analog input index constants,  
        601  
    NBCCommon.h, 784  
HTSPRO\_A1  
    HiTechnic SuperPro analog input index constants,  
        601  
    NBCCommon.h, 784  
HTSPRO\_A2  
    HiTechnic SuperPro analog input index constants,  
        601  
    NBCCommon.h, 784  
HTSPRO\_A3  
    HiTechnic SuperPro analog input index constants,  
        601  
    NBCCommon.h, 784  
HTSPRO\_DAC0  
    HiTechnic SuperPro analog output index constants,  
        602  
    NBCCommon.h, 784  
HTSPRO\_DAC1  
    HiTechnic SuperPro analog output index constants,  
        602  
    NBCCommon.h, 784  
HTSPRO\_REG\_A0  
    HiTechnic SuperPro constants, 596  
    NBCCommon.h, 784  
HTSPRO\_REG\_A1  
    HiTechnic SuperPro constants, 596  
    NBCCommon.h, 784  
HTSPRO\_REG\_A2  
    HiTechnic SuperPro constants, 596  
    NBCCommon.h, 784  
HTSPRO\_REG\_A3  
    HiTechnic SuperPro constants, 596  
    NBCCommon.h, 784  
HTSPRO\_REG\_CTRL  
    HiTechnic SuperPro constants, 596  
    NBCCommon.h, 784  
HTSPRO\_REG\_DCTRL  
    HiTechnic SuperPro constants, 597  
    NBCCommon.h, 785  
HTSPRO\_REG\_DIN  
    HiTechnic SuperPro constants, 597  
    NBCCommon.h, 785  
HTSPRO\_REG\_DLDATA  
    HiTechnic SuperPro constants, 597  
    NBCCommon.h, 785  
HTSPRO\_REG\_DOUT  
    HiTechnic SuperPro constants, 597  
    NBCCommon.h, 785  
HTSPRO\_REG\_LED  
    HiTechnic SuperPro constants, 597  
    NBCCommon.h, 785  
HTSPRO\_REG\_STROBE

- HiTechnic SuperPro constants, 600
- NBCCCommon.h, 788
- HTScoutCalibrateSensor
  - HiTechnic API Functions, 100
- HTScoutMuteSound
  - HiTechnic API Functions, 100
- HTScoutSelectSounds
  - HiTechnic API Functions, 100
- HTScoutSendVLL
  - HiTechnic API Functions, 100
- HTScoutSetEventFeedback
  - HiTechnic API Functions, 101
- HTScoutSetLight
  - HiTechnic API Functions, 101
- HTScoutSetScoutMode
  - HiTechnic API Functions, 101
- HTScoutSetSensorClickTime
  - HiTechnic API Functions, 101
- HTScoutSetSensorHysteresis
  - HiTechnic API Functions, 101
- HTScoutSetSensorLowerLimit
  - HiTechnic API Functions, 101
- HTScoutSetSensorUpperLimit
  - HiTechnic API Functions, 102
- HTScoutUnmuteSound
  - HiTechnic API Functions, 102
- Hi-speed port address constants, 525
  - HS\_ADDRESS\_1, 525
  - HS\_ADDRESS\_2, 525
  - HS\_ADDRESS\_3, 525
  - HS\_ADDRESS\_4, 525
  - HS\_ADDRESS\_5, 525
  - HS\_ADDRESS\_6, 525
  - HS\_ADDRESS\_7, 525
  - HS\_ADDRESS\_8, 525
  - HS\_ADDRESS\_ALL, 526
- Hi-speed port baud rate constants, 518
  - HS\_BAUD\_115200, 518
  - HS\_BAUD\_1200, 518
  - HS\_BAUD\_14400, 518
  - HS\_BAUD\_19200, 518
  - HS\_BAUD\_230400, 518
  - HS\_BAUD\_2400, 518
  - HS\_BAUD\_28800, 519
  - HS\_BAUD\_3600, 519
  - HS\_BAUD\_38400, 519
  - HS\_BAUD\_460800, 519
  - HS\_BAUD\_4800, 519
  - HS\_BAUD\_57600, 519
  - HS\_BAUD\_7200, 519
  - HS\_BAUD\_76800, 519
  - HS\_BAUD\_921600, 519
  - HS\_BAUD\_9600, 519
  - HS\_BAUD\_DEFAULT, 519
- Hi-speed port combined UART constants, 524
  - HS\_MODE\_7E1, 524
  - HS\_MODE\_8N1, 524
- Hi-speed port constants, 514
- Hi-speed port data bits constants, 521
  - HS\_MODE\_5\_DATA, 521
  - HS\_MODE\_6\_DATA, 521
  - HS\_MODE\_7\_DATA, 521
  - HS\_MODE\_8\_DATA, 521
- Hi-speed port flags constants, 515
  - HS\_UPDATE, 515
- Hi-speed port parity constants, 523
  - HS\_MODE\_E\_PARITY, 523
  - HS\_MODE\_M\_PARITY, 523
  - HS\_MODE\_N\_PARITY, 523
  - HS\_MODE\_O\_PARITY, 523
  - HS\_MODE\_S\_PARITY, 523
- Hi-speed port state constants, 516
  - HS\_BYTES\_REMAINING, 516
  - HS\_DEFAULT, 516
  - HS\_DISABLE, 516
  - HS\_ENABLE, 516
  - HS\_INIT\_RECEIVER, 516
  - HS\_INITIALISE, 516
  - HS\_SEND\_DATA, 516
- Hi-speed port stop bits constants, 522
  - HS\_MODE\_10\_STOP, 522
  - HS\_MODE\_15\_STOP, 522
  - HS\_MODE\_20\_STOP, 522
- Hi-speed port SysCommHSControl constants, 517
  - HS\_CTRL\_EXIT, 517
  - HS\_CTRL\_INIT, 517
  - HS\_CTRL\_UART, 517
- Hi-speed port UART mode constants, 520
  - HS\_MODE\_DEFAULT, 520
  - HS\_MODE\_MASK, 520
  - HS\_MODE\_UART\_RS232, 520
  - HS\_MODE\_UART\_RS485, 520
  - HS\_UART\_MASK, 520
- HiTechnic API Functions, 80
  - HTIRTrain, 87
  - HTPFCComboDirect, 87
  - HTPFCComboPWM, 87
  - HTPFRawOutput, 87
  - HTPFRRepeat, 88
  - HTPFSingleOutputCST, 88
  - HTPFSingleOutputPWM, 88
  - HTPFSinglePin, 89
  - HTPFTTrain, 89
  - HTPowerFunctionCommand, 89
  - HTRCXAddToDatalog, 90
  - HTRCXBatteryLevel, 90
  - HTRCXClearAllEvents, 90
  - HTRCXClearCounter, 90

HTRCXClearMsg, 90  
HTRCXClearSensor, 90  
HTRCXClearSound, 91  
HTRCXClearTimer, 91  
HTRCXCreateDatalog, 91  
HTRCXDecCounter, 91  
HTRCXDeleteSub, 91  
HTRCXDeleteSubs, 91  
HTRCXDeleteTask, 92  
HTRCXDeleteTasks, 92  
HTRCXDisableOutput, 92  
HTRCXEnableOutput, 92  
HTRCXEvent, 92  
HTRCXFloat, 92  
HTRCXFwd, 93  
HTRCXIncCounter, 93  
HTRCXInvertOutput, 93  
HTRCXMuteSound, 93  
HTRCXObvertOutput, 93  
HTRCXOff, 93  
HTRCXOn, 94  
HTRCXOnFor, 94  
HTRCXOnFwd, 94  
HTRCXOnRev, 94  
HTRCXPBTurnOff, 94  
HTRCXPing, 94  
HTRCXPlaySound, 94  
HTRCXPlayTone, 95  
HTRCXPlayToneVar, 95  
HTRCXPoll, 95  
HTRCXPollMemory, 95  
HTRCXRemote, 95  
HTRCXRev, 96  
HTRCXSelectDisplay, 96  
HTRCXSelectProgram, 96  
HTRCXSendSerial, 96  
HTRCXSetDirection, 96  
HTRCXSetEvent, 97  
HTRCXSetGlobalDirection, 97  
HTRCXSetGlobalOutput, 97  
HTRCXSetIRLinkPort, 97  
HTRCXSetMaxPower, 97  
HTRCXSetMessage, 98  
HTRCXSetOutput, 98  
HTRCXSetPower, 98  
HTRCXSetPriority, 98  
HTRCXSetSensorMode, 98  
HTRCXSetSensorType, 99  
HTRCXSetSleepTime, 99  
HTRCXSetTxPower, 99  
HTRCXSetWatch, 99  
HTRCXStartTask, 99  
HTRCXStopAllTasks, 99  
HTRCXStopTask, 100  
HTRCXToggle, 100  
HTRCXUnmuteSound, 100  
HTScoutCalibrateSensor, 100  
HTScoutMuteSound, 100  
HTScoutSelectSounds, 100  
HTScoutSendVLL, 100  
HTScoutSetEventFeedback, 101  
HTScoutSetLight, 101  
HTScoutSetScoutMode, 101  
HTScoutSetSensorClickTime, 101  
HTScoutSetSensorHysteresis, 101  
HTScoutSetSensorLowerLimit, 101  
HTScoutSetSensorUpperLimit, 102  
HTScoutUnmuteSound, 102  
ReadSensorHTAccel, 102  
ReadSensorHTAngle, 102  
ReadSensorHTBarometric, 103  
ReadSensorHTColor, 103  
ReadSensorHTColor2Active, 103  
ReadSensorHTColorNum, 103  
ReadSensorHTCompass, 104  
ReadSensorHTEOPD, 104  
ReadSensorHTForce, 104  
ReadSensorHTGyro, 104  
ReadSensorHTIRReceiver, 105  
ReadSensorHTIRReceiverEx, 105  
ReadSensorHTIRSeeker, 105  
ReadSensorHTIRSeeker2AC, 105  
ReadSensorHTIRSeeker2Addr, 106  
ReadSensorHTIRSeeker2DC, 106  
ReadSensorHTIRSeekerDir, 106  
ReadSensorHTMagnet, 107  
ReadSensorHTNormalizedColor, 107  
ReadSensorHTNormalizedColor2Active, 107  
ReadSensorHTPIR, 107  
ReadSensorHTProtoAllAnalog, 108  
ReadSensorHTProtoAnalog, 108  
ReadSensorHTProtoDigital, 108  
ReadSensorHTProtoDigitalControl, 108  
ReadSensorHTRawColor, 109  
ReadSensorHTRawColor2, 109  
ReadSensorHTSuperProAllAnalog, 109  
ReadSensorHTSuperProAnalog, 110  
ReadSensorHTSuperProAnalogOut, 110  
ReadSensorHTSuperProDigital, 110  
ReadSensorHTSuperProDigitalControl, 110  
ReadSensorHTSuperProLED, 111  
ReadSensorHTSuperProProgramControl, 111  
ReadSensorHTSuperProStrobe, 111  
ReadSensorHTTouchMultiplexer, 111  
ResetHTBarometricCalibration, 112  
ResetSensorHTAngle, 112  
SetHTBarometricCalibration, 112  
SetHTColor2Mode, 112

- SetHTIRSeeker2Mode, [113](#)  
SetSensorHTEOPD, [113](#)  
SetSensorHTForce, [113](#)  
SetSensorHTGyro, [113](#)  
SetSensorHTMagnet, [113](#)  
SetSensorHTPIRDeadband, [114](#)  
SetSensorHTProtoDigital, [114](#)  
SetSensorHTProtoDigitalControl, [114](#)  
SetSensorHTSuperProAnalogOut, [114](#)  
SetSensorHTSuperProDigital, [115](#)  
SetSensorHTSuperProDigitalControl, [115](#)  
SetSensorHTSuperProLED, [115](#)  
SetSensorHTSuperProProgramControl, [115](#)  
SetSensorHTSuperProStrobe, [116](#)
- HiTechnic Angle sensor constants, [589](#)  
HTANGLE\_MODE\_RESET, [589](#)  
HTANGLE\_REG\_ACDIR, [589](#)  
HTANGLE\_REG\_DC01, [589](#)  
HTANGLE\_REG\_DC02, [589](#)  
HTANGLE\_REG\_DC03, [589](#)  
HTANGLE\_REG\_DC04, [589](#)  
HTANGLE\_REG\_DC05, [590](#)  
HTANGLE\_REG\_DCAVG, [590](#)  
HTANGLE\_REG\_DCDIR, [590](#)  
HTANGLE\_REG\_MODE, [590](#)
- HiTechnic Barometric sensor constants, [591](#)  
HTBAR\_REG\_COMMAND, [591](#)  
HTBAR\_REG\_PRESSURE, [591](#)
- HiTechnic Color2 constants, [587](#)  
HT\_CMD\_COLOR2\_50HZ, [587](#)  
HT\_CMD\_COLOR2\_60HZ, [587](#)  
HT\_CMD\_COLOR2\_FAR, [587](#)  
HT\_CMD\_COLOR2\_NEAR, [587](#)  
HT\_CMD\_COLOR2\_RAW, [588](#)
- HiTechnic device constants, [582](#)  
HT\_ADDR\_ACCEL, [582](#)  
HT\_ADDR\_ANGLE, [582](#)  
HT\_ADDR\_BAROMETRIC, [583](#)  
HT\_ADDR\_COLOR, [583](#)  
HT\_ADDR\_COLOR2, [583](#)  
HT\_ADDR\_COMPASS, [583](#)  
HT\_ADDR\_IRLINK, [583](#)  
HT\_ADDR\_IRRECEIVER, [583](#)  
HT\_ADDR\_IRSEEKER, [583](#)  
HT\_ADDR\_IRSEEKER2, [583](#)  
HT\_ADDR\_PIR, [583](#)  
HT\_ADDR\_PROTOBOARD, [583](#)  
HT\_ADDR\_SUPERPRO, [583](#)
- HiTechnic IRReceiver constants, [586](#)  
HT\_CH1\_A, [586](#)  
HT\_CH1\_B, [586](#)  
HT\_CH2\_A, [586](#)  
HT\_CH2\_B, [586](#)  
HT\_CH3\_A, [586](#)
- HT\_CH3\_B, [586](#)  
HT\_CH4\_A, [586](#)  
HT\_CH4\_B, [586](#)
- HiTechnic IRSeeker2 constants, [584](#)  
HTIR2\_MODE\_1200, [584](#)  
HTIR2\_MODE\_600, [584](#)  
HTIR2\_REG\_AC01, [584](#)  
HTIR2\_REG\_AC02, [584](#)  
HTIR2\_REG\_AC03, [584](#)  
HTIR2\_REG\_AC04, [584](#)  
HTIR2\_REG\_AC05, [584](#)  
HTIR2\_REG\_ACDIR, [585](#)  
HTIR2\_REG\_DC01, [585](#)  
HTIR2\_REG\_DC02, [585](#)  
HTIR2\_REG\_DC03, [585](#)  
HTIR2\_REG\_DC04, [585](#)  
HTIR2\_REG\_DC05, [585](#)  
HTIR2\_REG\_DCAVG, [585](#)  
HTIR2\_REG\_DCDIR, [585](#)  
HTIR2\_REG\_MODE, [585](#)
- HiTechnic PIR sensor constants, [603](#)
- HiTechnic Prototype board analog input constants, [594](#)  
HTPROTO\_A0, [594](#)  
HTPROTO\_A1, [594](#)  
HTPROTO\_A2, [594](#)  
HTPROTO\_A3, [594](#)  
HTPROTO\_A4, [594](#)
- HiTechnic Prototype board constants, [592](#)  
HTPROTO\_REG\_A0, [592](#)  
HTPROTO\_REG\_A1, [592](#)  
HTPROTO\_REG\_A2, [592](#)  
HTPROTO\_REG\_A3, [592](#)  
HTPROTO\_REG\_A4, [592](#)  
HTPROTO\_REG\_DCTRL, [592](#)  
HTPROTO\_REG\_DIN, [592](#)  
HTPROTO\_REG\_DOUT, [593](#)  
HTPROTO\_REG\_SRATE, [593](#)
- HiTechnic SuperPro analog input index constants, [601](#)  
HTSPRO\_A0, [601](#)  
HTSPRO\_A1, [601](#)  
HTSPRO\_A2, [601](#)  
HTSPRO\_A3, [601](#)
- HiTechnic SuperPro analog output index constants, [602](#)  
HTSPRO\_DAC0, [602](#)  
HTSPRO\_DAC1, [602](#)
- HiTechnic SuperPro constants, [595](#)  
HTSPRO\_REG\_A0, [596](#)  
HTSPRO\_REG\_A1, [596](#)  
HTSPRO\_REG\_A2, [596](#)  
HTSPRO\_REG\_A3, [596](#)  
HTSPRO\_REG\_CTRL, [596](#)  
HTSPRO\_REG\_DCTRL, [597](#)  
HTSPRO\_REG\_DIN, [597](#)  
HTSPRO\_REG\_DLDATA, [597](#)

HTSPRO\_REG\_DOUT, 597  
HTSPRO\_REG\_LED, 597  
HTSPRO\_REG\_STROBE, 600  
HiTechnic/mindsensors Power Function/IR Train constants, 569

I2C option constants, 481  
  I2C\_OPTION\_FAST, 481  
  I2C\_OPTION\_STANDARD, 481

I2C\_ADDR\_DEFAULT  
  NBCCCommon.h, 788  
  Standard I2C constants, 474

I2C\_OPTION\_FAST  
  I2C option constants, 481  
  NBCCCommon.h, 788

I2C\_OPTION\_NORESTART  
  I2C option constants, 481

I2C\_OPTION\_STANDARD  
  I2C option constants, 481  
  NBCCCommon.h, 788

I2C\_REG\_CMD  
  NBCCCommon.h, 788  
  Standard I2C constants, 474

I2C\_REG\_DEVICE\_ID  
  NBCCCommon.h, 788  
  Standard I2C constants, 474

I2C\_REG\_VENDOR\_ID  
  NBCCCommon.h, 788  
  Standard I2C constants, 474

I2C\_REG\_VERSION  
  NBCCCommon.h, 788  
  Standard I2C constants, 474

I2CSendCommand  
  LowSpeed module functions, 239

IN\_1  
  NBC Input port constants, 430  
  NBCCCommon.h, 788

IN\_2  
  NBC Input port constants, 430  
  NBCCCommon.h, 788

IN\_3  
  NBC Input port constants, 430  
  NBCCCommon.h, 788

IN\_4  
  NBC Input port constants, 430  
  NBCCCommon.h, 789

IN\_MODE\_ANGLESTEP  
  NBC sensor mode constants, 434  
  NBCCCommon.h, 789

IN\_MODE\_BOOLEAN  
  NBC sensor mode constants, 434  
  NBCCCommon.h, 789

IN\_MODE\_CELSIUS  
  NBC sensor mode constants, 434

  NBCCCommon.h, 789

  IN\_MODE\_FAHRENHEIT  
    NBC sensor mode constants, 434  
    NBCCCommon.h, 789

  IN\_MODE\_MODEMASK  
    NBC sensor mode constants, 434  
    NBCCCommon.h, 789

  IN\_MODE\_RAW  
    NBC sensor mode constants, 435  
    NBCCCommon.h, 789

  IN\_MODE\_SLOPEMASK  
    NBC sensor mode constants, 435  
    NBCCCommon.h, 789

  IN\_TYPE\_ANGLE  
    NBC sensor type constants, 431  
    NBCCCommon.h, 789

  IN\_TYPE\_COLORBLUE  
    NBC sensor type constants, 431  
    NBCCCommon.h, 789

  IN\_TYPE\_COLOREXIT  
    NBC sensor type constants, 431  
    NBCCCommon.h, 789

  IN\_TYPE\_COLORFULL  
    NBC sensor type constants, 431  
    NBCCCommon.h, 790

  IN\_TYPE\_COLORGREEN  
    NBC sensor type constants, 431  
    NBCCCommon.h, 790

  IN\_TYPE\_COLORNONE  
    NBC sensor type constants, 432  
    NBCCCommon.h, 790

  IN\_TYPE\_COLORRED  
    NBC sensor type constants, 432  
    NBCCCommon.h, 790

  IN\_TYPE\_CUSTOM  
    NBC sensor type constants, 432  
    NBCCCommon.h, 790

  IN\_TYPE\_HISPEED  
    NBC sensor type constants, 432  
    NBCCCommon.h, 790

  IN\_TYPE\_LOWSPEED  
    NBC sensor type constants, 432  
    NBCCCommon.h, 790

  IN\_TYPE\_NO\_SENSOR  
    NBC sensor type constants, 432  
    NBCCCommon.h, 790

  IN\_TYPE\_REFLECTION  
    NBC sensor type constants, 432  
    NBCCCommon.h, 790

  IN\_TYPE\_SOUND\_DB  
    NBC sensor type constants, 432  
    NBCCCommon.h, 790

  IN\_TYPE\_SOUND\_DBA  
    NBC sensor type constants, 432

NBCCCommon.h, 790  
IN\_TYPE\_SWITCH  
  NBC sensor type constants, 432  
  NBCCCommon.h, 791  
INPUT\_BLACKCOLOR  
  Color values, 439  
  NBCCCommon.h, 791  
INPUT\_BLANK  
  Color sensor array indices, 438  
  NBCCCommon.h, 791  
INPUT\_BLUE  
  Color sensor array indices, 438  
  NBCCCommon.h, 791  
INPUT\_BLUECOLOR  
  Color values, 439  
  NBCCCommon.h, 791  
INPUT\_CAL\_POINT\_0  
  Color calibration constants, 441  
  NBCCCommon.h, 791  
INPUT\_CAL\_POINT\_1  
  Color calibration constants, 441  
  NBCCCommon.h, 791  
INPUT\_CAL\_POINT\_2  
  Color calibration constants, 441  
  NBCCCommon.h, 791  
INPUT\_CUSTOM9V  
  Input module constants, 64  
  NBCCCommon.h, 791  
INPUT\_CUSTOMACTIVE  
  Input module constants, 64  
  NBCCCommon.h, 791  
INPUT\_CUSTOMINACTIVE  
  Input module constants, 64  
INPUT\_DIGIO  
  Input port digital pin constants, 437  
  NBCCCommon.h, 791  
INPUT\_DIGI1  
  Input port digital pin constants, 437  
  NBCCCommon.h, 791  
INPUT\_GREEN  
  Color sensor array indices, 438  
  NBCCCommon.h, 792  
INPUT\_GREENCOLOR  
  Color values, 439  
  NBCCCommon.h, 792  
INPUT\_INVALID\_DATA  
  Input module constants, 65  
  NBCCCommon.h, 792  
INPUT\_NO\_OF\_COLORS  
  Color sensor array indices, 438  
INPUT\_NO\_OF\_POINTS  
  Color calibration constants, 441  
INPUT\_PINCMD\_CLEAR  
  Constants to use with the Input module's Pin function,  
  445  
  NBCCCommon.h, 792  
INPUT\_PINCMD\_DIR  
  Constants to use with the Input module's Pin function,  
  445  
  NBCCCommon.h, 792  
INPUT\_PINCMD\_MASK  
  Constants to use with the Input module's Pin function,  
  445  
  NBCCCommon.h, 792  
INPUT\_PINCMD\_READ  
  Constants to use with the Input module's Pin function,  
  445  
  NBCCCommon.h, 792  
INPUT\_PINCMD\_SET  
  Constants to use with the Input module's Pin function,  
  445  
  NBCCCommon.h, 792  
INPUT\_PINCMD\_WAIT  
  Constants to use with the Input module's Pin function,  
  445  
  NBCCCommon.h, 792  
INPUT\_PINDIR\_INPUT  
  Constants to use with the Input module's Pin function,  
  445  
  NBCCCommon.h, 792  
INPUT\_RED  
  Color sensor array indices, 438  
  NBCCCommon.h, 792  
INPUT\_REDCOLOR  
  Color values, 439  
  NBCCCommon.h, 793  
INPUT\_RESETCAL  
  Color calibration state constants, 440  
  NBCCCommon.h, 793  
INPUT\_RUNNINGCAL  
  Color calibration state constants, 440  
  NBCCCommon.h, 793  
INPUT\_SENSORCAL  
  Color calibration state constants, 440  
  NBCCCommon.h, 793  
INPUT\_SENSOROFF  
  Color calibration state constants, 440  
  NBCCCommon.h, 793  
INPUT\_STARTCAL  
  Color calibration state constants, 440  
  NBCCCommon.h, 793  
INPUT\_WHITECOLOR  
  Color values, 439  
  NBCCCommon.h, 793  
INPUT\_YELLOWCOLOR  
  Color values, 439  
  NBCCCommon.h, 793

INT\_MAX  
    Data type limits, 687  
    NBCCCommon.h, 795

INT\_MIN  
    Data type limits, 687  
    NBCCCommon.h, 795

INTF\_BTOFF  
    Comm module interface function constants, 528  
    NBCCCommon.h, 795

INTF\_BTTON  
    Comm module interface function constants, 528  
    NBCCCommon.h, 795

INTF\_CONNECT  
    Comm module interface function constants, 528  
    NBCCCommon.h, 795

INTF\_CONNECTBYNAME  
    Comm module interface function constants, 528  
    NBCCCommon.h, 795

INTF\_CONNECTREQ  
    Comm module interface function constants, 528  
    NBCCCommon.h, 795

INTF\_DISCONNECT  
    Comm module interface function constants, 529  
    NBCCCommon.h, 795

INTF\_DISCONNECTALL  
    Comm module interface function constants, 529  
    NBCCCommon.h, 795

INTF\_EXTREAD  
    Comm module interface function constants, 529  
    NBCCCommon.h, 795

INTF\_FACTORYRESET  
    Comm module interface function constants, 529  
    NBCCCommon.h, 796

INTF\_OPENSTREAM  
    Comm module interface function constants, 529  
    NBCCCommon.h, 796

INTF\_PINREQ  
    Comm module interface function constants, 529  
    NBCCCommon.h, 796

INTF\_REMOVEDEVICE  
    Comm module interface function constants, 529  
    NBCCCommon.h, 796

INTF\_SEARCH  
    Comm module interface function constants, 529  
    NBCCCommon.h, 796

INTF\_SENDDATA  
    Comm module interface function constants, 529  
    NBCCCommon.h, 796

INTF\_SENDFILE  
    Comm module interface function constants, 529  
    NBCCCommon.h, 796

INTF\_SETBTNAME  
    Comm module interface function constants, 529  
    NBCCCommon.h, 796

INTF\_SETCMDMODE  
    Comm module interface function constants, 529  
    NBCCCommon.h, 796

INTF\_STOPSEARCH  
    Comm module interface function constants, 529  
    NBCCCommon.h, 796

INTF\_VISIBILITY  
    Comm module interface function constants, 529  
    NBCCCommon.h, 796

IOCTRL\_BOOT  
    NBCCCommon.h, 796  
    PowerOn constants, 391

IOCTRL\_POWERDOWN  
    NBCCCommon.h, 796  
    PowerOn constants, 391

IOCctrl module, 74

IOCctrl module constants, 390

IOCctrl module functions, 347  
    PowerDown, 347  
    RebootInFirmwareMode, 347

IOCctrl module IOMAP offsets, 392

IOCctrlOffsetPowerOn, 392

IOCctrlModuleID  
    NBCCCommon.h, 797  
    NXT firmware module IDs, 201

IOCctrlModuleName  
    NBCCCommon.h, 797  
    NXT firmware module names, 199

IOCctrlOffsetPowerOn  
    IOCctrl module IOMAP offsets, 392  
    NBCCCommon.h, 797

IOMapRead  
    NBCCCommon.h, 797  
    System Call function constants, 370

IOMapReadByID  
    NBCCCommon.h, 797  
    System Call function constants, 370

IOMapWrite  
    NBCCCommon.h, 797  
    System Call function constants, 370

IOMapWriteByID  
    NBCCCommon.h, 797  
    System Call function constants, 370

IR Train channel constants, 574  
    TRAIN\_CHANNEL\_1, 574  
    TRAIN\_CHANNEL\_2, 574  
    TRAIN\_CHANNEL\_3, 574  
    TRAIN\_CHANNEL\_ALL, 574

Input field constants, 436  
    InputModeField, 436  
    InvalidDataField, 436  
    NormalizedValueField, 436  
    RawValueField, 436  
    ScaledValueField, 436

TypeField, 436  
Input module, 63  
Input module constants, 64  
  INPUT\_CUSTOM9V, 64  
  INPUT\_CUSTOMACTIVE, 64  
  INPUT\_CUSTOMINACTIVE, 64  
  INPUT\_INVALID\_DATA, 65  
Input module functions, 226  
  ClearSensor, 228  
  GetInColorADRaw, 228  
  GetInColorBoolean, 228  
  GetInColorCallLimits, 229  
  GetInColorCalibration, 228  
  GetInColorCalibrationState, 229  
  GetInColorSensorRaw, 229  
  GetInColorSensorValue, 230  
  GetInCustomActiveStatus, 230  
  GetInCustomPercentFullScale, 230  
  GetInCustomZeroOffset, 230  
  GetInDigiPinsDirection, 230  
  GetInDigiPinsOutputLevel, 231  
  GetInDigiPinsStatus, 231  
  GetInSensorBoolean, 231  
  ReadSensor, 231  
  ReadSensorColorEx, 231  
  ReadSensorColorRaw, 232  
  ResetSensor, 232  
  SetInCustomActiveStatus, 232  
  SetInCustomPercentFullScale, 233  
  SetInCustomZeroOffset, 233  
  SetInDigiPinsDirection, 233  
  SetInDigiPinsOutputLevel, 233  
  SetInDigiPinsStatus, 233  
  SetInSensorBoolean, 234  
  SetSensorColorBlue, 234  
  SetSensorColorFull, 234  
  SetSensorColorGreen, 234  
  SetSensorColorNone, 235  
  SetSensorColorRed, 235  
  SetSensorEMeter, 235  
  SetSensorLight, 235  
  SetSensorLowspeed, 235  
  SetSensorMode, 236  
  SetSensorSound, 236  
  SetSensorTemperature, 236  
  SetSensorTouch, 236  
  SetSensorType, 237  
  SetSensorUltrasonic, 237  
Input module IOMAP offsets, 442  
  InputOffsetADRaw, 442  
  InputOffsetColorADRaw, 442  
  InputOffsetColorBoolean, 442  
  InputOffsetColorCallLimits, 442  
  InputOffsetColorCalibration, 442  
  InputOffsetColorCalibrationState, 442  
  InputOffsetColorSensorRaw, 443  
  InputOffsetColorSensorValue, 443  
  InputOffsetCustomActiveStatus, 443  
  InputOffsetCustomPctFullScale, 443  
  InputOffsetCustomZeroOffset, 443  
  InputOffsetDigiPinsDir, 443  
  InputOffsetDigiPinsIn, 443  
  InputOffsetDigiPinsOut, 443  
  InputOffsetInvalidData, 443  
  InputOffsetSensorBoolean, 443  
  InputOffsetSensorMode, 443  
  InputOffsetSensorRaw, 443  
  InputOffsetSensorType, 443  
  InputOffsetSensorValue, 443  
Input port digital pin constants, 437  
  INPUT\_DIGI0, 437  
  INPUT\_DIGI1, 437  
InputModeField  
  Input field constants, 436  
  NBCCommon.h, 793  
InputModuleID  
  NBCCommon.h, 793  
  NXT firmware module IDs, 201  
InputModuleName  
  NBCCommon.h, 793  
  NXT firmware module names, 199  
InputOffsetADRaw  
  Input module IOMAP offsets, 442  
  NBCCommon.h, 793  
InputOffsetColorADRaw  
  Input module IOMAP offsets, 442  
  NBCCommon.h, 793  
InputOffsetColorBoolean  
  Input module IOMAP offsets, 442  
  NBCCommon.h, 793  
InputOffsetColorCallLimits  
  Input module IOMAP offsets, 442  
  NBCCommon.h, 794  
InputOffsetColorCalibration  
  Input module IOMAP offsets, 442  
  NBCCommon.h, 794  
InputOffsetColorCalibrationState  
  Input module IOMAP offsets, 442  
  NBCCommon.h, 794  
InputOffsetColorSensorRaw  
  Input module IOMAP offsets, 443  
  NBCCommon.h, 794  
InputOffsetColorSensorValue  
  Input module IOMAP offsets, 443  
  NBCCommon.h, 794  
InputOffsetCustomActiveStatus  
  Input module IOMAP offsets, 443  
  NBCCommon.h, 794

InputOffsetCustomPctFullScale  
    Input module IOMAP offsets, 443  
    NBCCCommon.h, 794

InputOffsetCustomZeroOffset  
    Input module IOMAP offsets, 443  
    NBCCCommon.h, 794

InputOffsetDigiPinsDir  
    Input module IOMAP offsets, 443  
    NBCCCommon.h, 794

InputOffsetDigiPinsIn  
    Input module IOMAP offsets, 443  
    NBCCCommon.h, 794

InputOffsetDigiPinsOut  
    Input module IOMAP offsets, 443  
    NBCCCommon.h, 794

InputOffsetInvalidData  
    Input module IOMAP offsets, 443  
    NBCCCommon.h, 794

InputOffsetSensorBoolean  
    Input module IOMAP offsets, 443  
    NBCCCommon.h, 794

InputOffsetSensorMode  
    Input module IOMAP offsets, 443  
    NBCCCommon.h, 794

InputOffsetSensorRaw  
    Input module IOMAP offsets, 443  
    NBCCCommon.h, 795

InputOffsetSensorType  
    Input module IOMAP offsets, 443  
    NBCCCommon.h, 795

InputOffsetSensorValue  
    Input module IOMAP offsets, 443  
    NBCCCommon.h, 795

InputPinFunction  
    NBCCCommon.h, 795  
    System Call function constants, 369

InvalidDataField  
    Input field constants, 436  
    NBCCCommon.h, 796

JOY\_BTN\_01  
    Joystick message constants, 505  
    NBCCCommon.h, 797

JOY\_BTN\_02  
    Joystick message constants, 505  
    NBCCCommon.h, 797

JOY\_BTN\_03  
    Joystick message constants, 505  
    NBCCCommon.h, 797

JOY\_BTN\_04  
    Joystick message constants, 505  
    NBCCCommon.h, 797

JOY\_BTN\_05  
    Joystick message constants, 505

            NBCCCommon.h, 797

JOY\_BTN\_06  
    Joystick message constants, 505

            NBCCCommon.h, 797

JOY\_BTN\_07  
    Joystick message constants, 505

            NBCCCommon.h, 797

JOY\_BTN\_08  
    Joystick message constants, 505

            NBCCCommon.h, 798

JOY\_BTN\_09  
    Joystick message constants, 505

            NBCCCommon.h, 798

JOY\_BTN\_10  
    Joystick message constants, 505

            NBCCCommon.h, 798

JOY\_BTN\_11  
    Joystick message constants, 505

            NBCCCommon.h, 798

JOY\_BTN\_12  
    Joystick message constants, 505

            NBCCCommon.h, 798

JOY\_BTN\_13  
    Joystick message constants, 505

            NBCCCommon.h, 798

JOY\_BTN\_14  
    Joystick message constants, 505

            NBCCCommon.h, 798

JOY\_BTN\_15  
    Joystick message constants, 505

            NBCCCommon.h, 798

JOY\_BTN\_16  
    Joystick message constants, 506

            NBCCCommon.h, 798

JOY\_BTN\_17  
    Joystick message constants, 506

            NBCCCommon.h, 798

JOY\_BTN\_18  
    Joystick message constants, 506

            NBCCCommon.h, 798

JOY\_BTN\_19  
    Joystick message constants, 506

            NBCCCommon.h, 798

JOY\_BTN\_20  
    Joystick message constants, 506

            NBCCCommon.h, 798

JOY\_BTN\_21  
    Joystick message constants, 506

            NBCCCommon.h, 798

JOY\_BTN\_22  
    Joystick message constants, 506

            NBCCCommon.h, 799

JOY\_BTN\_23  
    Joystick message constants, 506

NBCCommon.h, 799	NBCCommon.h, 800
JOY_BTN_24	Joystick message constants, 506 NBCCommon.h, 799
JOY_BTN_25	Joystick message constants, 506 NBCCommon.h, 799
JOY_BTN_26	Joystick message constants, 506 NBCCommon.h, 799
JOY_BTN_27	Joystick message constants, 506 NBCCommon.h, 799
JOY_BTN_28	Joystick message constants, 506 NBCCommon.h, 799
JOY_BTN_29	Joystick message constants, 506 NBCCommon.h, 799
JOY_BTN_30	Joystick message constants, 507 NBCCommon.h, 799
JOY_BTN_31	Joystick message constants, 507 NBCCommon.h, 799
JOY_BTN_32	Joystick message constants, 507 NBCCommon.h, 799
JOY_POV_BACKWARD	Joystick message constants, 507 NBCCommon.h, 799
JOY_POV_BOTLEFT	Joystick message constants, 507 NBCCommon.h, 799
JOY_POV_BOTRIGHT	Joystick message constants, 507 NBCCommon.h, 799
JOY_POV_CENTERED	Joystick message constants, 507 NBCCommon.h, 800
JOY_POV_FORWARD	Joystick message constants, 507 NBCCommon.h, 800
JOY_POV_LEFT	Joystick message constants, 507 NBCCommon.h, 800
JOY_POV_RIGHT	Joystick message constants, 507 NBCCommon.h, 800
JOY_POV_TOPLEFT	Joystick message constants, 507 NBCCommon.h, 800
JOY_POV_TOPRIGHT	Joystick message constants, 507 NBCCommon.h, 800
KeepAlive	NBCCommon.h, 800 System Call function constants, 370
LCD_LINE1	Line number constants, 372 NBCCommon.h, 800
LCD_LINE2	Line number constants, 372 NBCCommon.h, 800
LCD_LINE3	Line number constants, 372 NBCCommon.h, 800

- Line number constants, 372  
NBCCCommon.h, 800
- LCD\_LINE4  
Line number constants, 372  
NBCCCommon.h, 800
- LCD\_LINE5  
Line number constants, 372  
NBCCCommon.h, 800
- LCD\_LINE6  
Line number constants, 372  
NBCCCommon.h, 800
- LCD\_LINE7  
Line number constants, 372  
NBCCCommon.h, 800
- LCD\_LINE8  
Line number constants, 372  
NBCCCommon.h, 801
- LDR\_BTBUSY  
Loader module error codes, 395  
NBCCCommon.h, 801
- LDR\_BTCONNECTFAIL  
Loader module error codes, 395  
NBCCCommon.h, 801
- LDR\_BTTIMEOUT  
Loader module error codes, 396  
NBCCCommon.h, 801
- LDR\_CMD\_BOOTCMD  
Loader module function constants, 399  
NBCCCommon.h, 801
- LDR\_CMD\_BTGETADR  
Loader module function constants, 400  
NBCCCommon.h, 801
- LDR\_CMD\_CLOSE  
Loader module function constants, 400  
NBCCCommon.h, 801
- LDR\_CMD\_DELETE  
Loader module function constants, 400  
NBCCCommon.h, 801
- LDR\_CMD\_DEVICEINFO  
Loader module function constants, 400  
NBCCCommon.h, 801
- LDR\_CMD\_FINDFIRST  
Loader module function constants, 400  
NBCCCommon.h, 802
- LDR\_CMD\_FINDNEXT  
Loader module function constants, 400  
NBCCCommon.h, 802
- LDR\_CMD\_IOMAPREAD  
Loader module function constants, 400  
NBCCCommon.h, 802
- LDR\_CMD\_IOMAPWRITE  
Loader module function constants, 400  
NBCCCommon.h, 802
- LDR\_CMD\_OPENREAD  
Loader module function constants, 401  
NBCCCommon.h, 802
- LDR\_CMD\_OPENWRITE  
Loader module function constants, 401  
NBCCCommon.h, 802
- LDR\_CMD\_POLLCMD  
Loader module function constants, 401  
NBCCCommon.h, 802
- LDR\_CMD\_POLLCMDLEN  
Loader module function constants, 401  
NBCCCommon.h, 802
- LDR\_CMD\_READ  
Loader module function constants, 401  
NBCCCommon.h, 803
- LDR\_CMD\_RENAMEFILE  
Loader module function constants, 401  
NBCCCommon.h, 803
- LDR\_CMD\_SEEKFROMEND  
Loader module function constants, 401
- LDR\_CMD\_VERSIONS  
Loader module function constants, 402  
NBCCCommon.h, 803
- LDR\_CMD\_WRITE  
Loader module function constants, 402  
NBCCCommon.h, 803
- LDR\_ENDOFFILE  
Loader module error codes, 396  
NBCCCommon.h, 803
- LDR\_EOFEXPECTED  
Loader module error codes, 396  
NBCCCommon.h, 803
- LDR\_FILEEXISTS  
Loader module error codes, 396  
NBCCCommon.h, 803
- LDR\_FILEISBUSY  
Loader module error codes, 396  
NBCCCommon.h, 803
- LDR\_FILEISFULL  
Loader module error codes, 396  
NBCCCommon.h, 803
- LDR\_FILENOFOUND  
Loader module error codes, 396  
NBCCCommon.h, 804
- LDR\_FILETX\_TIMEOUT  
Loader module error codes, 396  
NBCCCommon.h, 804
- LDR\_ILLEGALFILENAME  
Loader module error codes, 396  
NBCCCommon.h, 804
- LDR\_ILLEGALHANDLE  
Loader module error codes, 397  
NBCCCommon.h, 804
- LDR\_INPROGRESS  
Loader module error codes, 397

NBCCCommon.h, 804  
LDR\_INVALIDSEEK  
  Loader module error codes, 397  
  NBCCCommon.h, 804  
LDR\_MODULENOTFOUND  
  Loader module error codes, 397  
  NBCCCommon.h, 804  
LDR\_NOLINEARSPACE  
  Loader module error codes, 397  
  NBCCCommon.h, 804  
LDR\_NOMOREFILES  
  Loader module error codes, 397  
  NBCCCommon.h, 804  
LDR\_NOMOREHANDLES  
  Loader module error codes, 397  
  NBCCCommon.h, 805  
LDR\_NOSPACE  
  Loader module error codes, 397  
  NBCCCommon.h, 805  
LDR\_NOTLINEARFILE  
  Loader module error codes, 397  
  NBCCCommon.h, 805  
LDR\_NOWRITEBUFFERS  
  Loader module error codes, 397  
  NBCCCommon.h, 805  
LDR\_OUTOFBOUNDARY  
  Loader module error codes, 397  
  NBCCCommon.h, 805  
LDR\_REQPIN  
  Loader module error codes, 397  
  NBCCCommon.h, 805  
LDR\_SUCCESS  
  Loader module error codes, 397  
  NBCCCommon.h, 805  
LDR\_UNDEFINEDERROR  
  Loader module error codes, 397  
  NBCCCommon.h, 805  
LED\_BLUE  
  NBCCCommon.h, 805  
  SuperPro LED control constants, 119  
LED\_NONE  
  NBCCCommon.h, 805  
  SuperPro LED control constants, 119  
LED\_RED  
  NBCCCommon.h, 805  
  SuperPro LED control constants, 119  
LEGO I2C address constants, 475  
  LEGO\_ADDR\_TEMP, 475  
  LEGO\_ADDR\_US, 475  
LEGO temperature sensor constants, 478  
  TEMP\_FQ\_1, 478  
  TEMP\_FQ\_2, 478  
  TEMP\_FQ\_4, 478  
  TEMP\_FQ\_6, 478  
TEMP\_OS\_ONESHOT, 478  
TEMP\_POL\_HIGH, 478  
TEMP\_POL\_LOW, 479  
TEMP\_REG\_CONFIG, 479  
TEMP\_REG\_TEMP, 479  
TEMP\_REG\_THIGH, 479  
TEMP\_REG\_TLOW, 479  
TEMP\_RES\_10BIT, 479  
TEMP\_RES\_11BIT, 479  
TEMP\_RES\_12BIT, 479  
TEMP\_RES\_9BIT, 479  
TEMP\_SD\_SHUTDOWN, 479  
LEGO\_ADDR\_EMETER  
  LEGO I2C address constants, 475  
  NBCCCommon.h, 805  
LEGO\_ADDR\_TEMP  
  LEGO I2C address constants, 475  
  NBCCCommon.h, 805  
LEGO\_ADDR\_US  
  LEGO I2C address constants, 475  
  NBCCCommon.h, 805  
LONG\_MAX  
  Data type limits, 687  
  NBCCCommon.h, 806  
LONG\_MIN  
  Data type limits, 687  
  NBCCCommon.h, 806  
LOWSPEED\_DONE  
  LSChannelState constants, 467  
  NBCCCommon.h, 806  
LOWSPEED\_ERROR  
  LSChannelState constants, 467  
  NBCCCommon.h, 806  
LOWSPEED\_IDLE  
  LSChannelState constants, 467  
  NBCCCommon.h, 806  
LOWSPEED\_INIT  
  LSChannelState constants, 467  
  NBCCCommon.h, 807  
LOWSPEED\_NO\_ERROR  
  LSErrorType constants, 469  
  NBCCCommon.h, 807  
LOWSPEED RECEIVING  
  LSMode constants, 468  
  NBCCCommon.h, 807  
LOWSPEED\_RX\_ERROR  
  LSErrorType constants, 469  
  NBCCCommon.h, 807  
LOWSPEED\_TX\_ERROR  
  LSErrorType constants, 469  
  NBCCCommon.h, 807  
LR\_COULD\_NOT\_SAVE  
  Comm module status code constants, 531  
  NBCCCommon.h, 808

LR\_ENTRY\_REMOVED  
    Comm module status code constants, 531  
    NBCCCommon.h, 808

LR\_STORE\_IS\_FULL  
    Comm module status code constants, 531  
    NBCCCommon.h, 808

LR\_SUCCESS  
    Comm module status code constants, 531  
    NBCCCommon.h, 808

LR\_UNKNOWN\_ADDR  
    Comm module status code constants, 531  
    NBCCCommon.h, 808

LSChannelState constants, 467  
    LOWSPEED\_DONE, 467  
    LOWSPEED\_ERROR, 467  
    LOWSPEED\_IDLE, 467  
    LOWSPEED\_INIT, 467

LSErrorType constants, 469  
    LOWSPEED\_NO\_ERROR, 469  
    LOWSPEED\_RX\_ERROR, 469  
    LOWSPEED\_TX\_ERROR, 469

LSMode constants, 468  
    LOWSPEED RECEIVING, 468

LSNoRestartOnRead constants, 472

LSREAD\_RESTART\_ALL  
    NBCCCommon.h, 809

LSState constants, 466

LT  
    Comparison Constants, 61

LTEQ  
    Comparison Constants, 61

Line number constants, 372  
    LCD\_LINE1, 372  
    LCD\_LINE2, 372  
    LCD\_LINE3, 372  
    LCD\_LINE4, 372  
    LCD\_LINE5, 372  
    LCD\_LINE6, 372  
    LCD\_LINE7, 372  
    LCD\_LINE8, 372

LineOut  
    Display module functions, 258

LineOutEx  
    Display module functions, 259

ListFiles  
    NBCCCommon.h, 806  
    System Call function constants, 370

Loader module, 75

Loader module constants, 393  
    EOF, 393  
    NULL, 393

Loader module error codes, 395  
    LDR\_BTBUSY, 395  
    LDR\_BTCONNECTFAIL, 395

    LDR\_BTTIMEOUT, 396  
    LDR\_ENDOFFILE, 396  
    LDR\_EOFEXPECTED, 396  
    LDR\_FILEEXISTS, 396  
    LDR\_FILEISBUSY, 396  
    LDR\_FILEISFULL, 396  
    LDR\_FILENOFOUND, 396  
    LDR\_FILETX\_TIMEOUT, 396  
    LDR\_ILLEGALFILENAME, 396  
    LDR\_ILLEGALHANDLE, 397  
    LDR\_INPROGRESS, 397  
    LDR\_INVALIDSEEK, 397  
    LDR\_MODULENOTFOUND, 397  
    LDR\_NOLINEARSPACE, 397  
    LDR\_NOMOREFILES, 397  
    LDR\_NOMOREHANDLES, 397  
    LDR\_NOSPACE, 397  
    LDR\_NOTLINEARFILE, 397  
    LDR\_NOWRITEBUFFERS, 397  
    LDR\_OUTOFBOUNDARY, 397  
    LDR\_REQPIN, 397  
    LDR\_SUCCESS, 397  
    LDR\_UNDEFINEDERROR, 397

Loader module function constants, 399  
    LDR\_CMD\_BOOTCMD, 399  
    LDR\_CMD\_BTGETADDR, 400  
    LDR\_CMD\_CLOSE, 400  
    LDR\_CMD\_DELETE, 400  
    LDR\_CMD\_DEVICEINFO, 400  
    LDR\_CMD\_FINDFIRST, 400  
    LDR\_CMD\_FINDNEXT, 400  
    LDR\_CMD\_IOMAPREAD, 400  
    LDR\_CMD\_IOMAPWRITE, 400  
    LDR\_CMD\_OPENREAD, 401  
    LDR\_CMD\_OPENWRITE, 401  
    LDR\_CMD\_POLLCMD, 401  
    LDR\_CMD\_POLLCMDLEN, 401  
    LDR\_CMD\_READ, 401  
    LDR\_CMD\_RENAMEFILE, 401  
    LDR\_CMD\_SEEKFROMEND, 401  
    LDR\_CMD VERSIONS, 402  
    LDR\_CMD\_WRITE, 402

Loader module functions, 348  
    CloseFile, 349  
    CreateFile, 349  
    CreateFileLinear, 349  
    CreateFileNonLinear, 350  
    DeleteFile, 350  
    FindFirstFile, 350  
    FindNextFile, 351  
    GetFreeMemory, 351  
    OpenFileAppend, 351  
    OpenFileRead, 351  
    OpenFileReadLinear, 352

Read, 352  
ReadBytes, 352  
ReadLn, 353  
ReadLnString, 353  
RenameFile, 353  
ResizeFile, 353  
ResolveHandle, 354  
SizeOf, 354  
Write, 354  
WriteBytes, 354  
WriteBytesEx, 355  
WriteLn, 355  
WriteLnString, 355  
WriteString, 355  
Loader module IOMAP offsets, 394  
  LoaderOffsetFreeUserFlash, 394  
  LoaderOffsetPFunc, 394  
LoaderExecuteFunction  
  NBCCommon.h, 806  
  System Call function constants, 370  
LoaderModuleID  
  NBCCommon.h, 806  
  NXT firmware module IDs, 201  
LoaderModuleName  
  NBCCommon.h, 806  
  NXT firmware module names, 199  
LoaderOffsetFreeUserFlash  
  Loader module IOMAP offsets, 394  
  NBCCommon.h, 806  
LoaderOffsetPFunc  
  Loader module IOMAP offsets, 394  
  NBCCommon.h, 806  
Low level LowSpeed module functions, 246  
  GetLSChannelState, 246  
  GetLSErrorType, 247  
  GetLSInputBuffer, 247  
  GetLSInputBufferBytesToRx, 247  
  GetLSInputBufferInPtr, 247  
  GetLSInputBufferOutPtr, 247  
  GetLSMode, 248  
  GetLSNoRestartOnRead, 248  
  GetLSOutputBuffer, 248  
  GetLSOutputBufferBytesToRx, 248  
  GetLSOutputBufferInPtr, 248  
  GetLSOutputBufferOutPtr, 249  
  GetLSSpeed, 249  
  GetLSState, 249  
Low Speed module, 78  
Low speed module IOMAP offsets, 470  
  LowSpeedOffsetChannelState, 470  
  LowSpeedOffsetErrorType, 470  
  LowSpeedOffsetInBufBuf, 470  
  LowSpeedOffsetInBufBytesToRx, 470  
  LowSpeedOffsetInBufInPtr, 470  
  LowSpeedOffsetInBufOutPtr, 470  
  LowSpeedOffsetMode, 470  
  LowSpeedOffsetNoRestartOnRead, 471  
  LowSpeedOffsetOutBufBuf, 471  
  LowSpeedOffsetOutBufBytesToRx, 471  
  LowSpeedOffsetOutBufInPtr, 471  
  LowSpeedOffsetOutBufOutPtr, 471  
  LowSpeedOffsetSpeed, 471  
  LowSpeedOffsetState, 471  
LowSpeed module constants, 465  
LowSpeed module functions, 238  
  ConfigureTemperatureSensor, 239  
  I2CSendCommand, 239  
  LowspeedBytesReady, 239  
  LowspeedCheckStatus, 240  
  LowspeedRead, 240  
  LowspeedStatus, 241  
  LowspeedWrite, 241  
  ReadI2CBytes, 241  
  ReadI2CDeviceId, 242  
  ReadI2CDeviceInfo, 242  
  ReadI2CRegister, 242  
  ReadI2CVendorId, 243  
  ReadI2CVersion, 243  
  ReadSensorEMeter, 243  
  ReadSensorTemperature, 244  
  ReadSensorUS, 244  
  ReadSensorUSEx, 244  
  SetI2COptions, 244  
  WriteI2CRegister, 245  
LowSpeedModuleID  
  NBCCommon.h, 807  
  NXT firmware module IDs, 201  
LowSpeedModuleName  
  NBCCommon.h, 807  
  NXT firmware module names, 199  
LowSpeedOffsetChannelState  
  Low speed module IOMAP offsets, 470  
  NBCCommon.h, 807  
LowSpeedOffsetErrorType  
  Low speed module IOMAP offsets, 470  
  NBCCommon.h, 807  
LowSpeedOffsetInBufBuf  
  Low speed module IOMAP offsets, 470  
  NBCCommon.h, 807  
LowSpeedOffsetInBufBytesToRx  
  Low speed module IOMAP offsets, 470  
  NBCCommon.h, 807  
LowSpeedOffsetInBufInPtr  
  Low speed module IOMAP offsets, 470  
  NBCCommon.h, 807  
LowSpeedOffsetInBufOutPtr  
  Low speed module IOMAP offsets, 470  
  NBCCommon.h, 808

LowSpeedOffsetMode  
    Low speed module IOMAP offsets, 470  
    NBCCCommon.h, 808

LowSpeedOffsetNoRestartOnRead  
    Low speed module IOMAP offsets, 471  
    NBCCCommon.h, 808

LowSpeedOffsetOutBufBuf  
    Low speed module IOMAP offsets, 471  
    NBCCCommon.h, 808

LowSpeedOffsetOutBufBytesToRx  
    Low speed module IOMAP offsets, 471  
    NBCCCommon.h, 808

LowSpeedOffsetOutBufInPtr  
    Low speed module IOMAP offsets, 471  
    NBCCCommon.h, 808

LowSpeedOffsetOutBufOutPtr  
    Low speed module IOMAP offsets, 471  
    NBCCCommon.h, 808

LowSpeedOffsetSpeed  
    Low speed module IOMAP offsets, 471  
    NBCCCommon.h, 808

LowSpeedOffsetState  
    Low speed module IOMAP offsets, 471  
    NBCCCommon.h, 808

LowspeedBytesReady  
    LowSpeed module functions, 239

LowspeedCheckStatus  
    LowSpeed module functions, 240

LowspeedRead  
    LowSpeed module functions, 240

LowspeedStatus  
    LowSpeed module functions, 241

LowspeedWrite  
    LowSpeed module functions, 241

MAILBOX1  
    Mailbox constants, 379  
    NBCCCommon.h, 809

MAILBOX10  
    Mailbox constants, 379  
    NBCCCommon.h, 809

MAILBOX2  
    Mailbox constants, 379  
    NBCCCommon.h, 809

MAILBOX3  
    Mailbox constants, 379  
    NBCCCommon.h, 809

MAILBOX4  
    Mailbox constants, 379  
    NBCCCommon.h, 809

MAILBOX5  
    Mailbox constants, 379  
    NBCCCommon.h, 809

MAILBOX6  
    Mailbox constants, 379  
    NBCCCommon.h, 809

MAILBOX7  
    Mailbox constants, 380  
    NBCCCommon.h, 810

MAILBOX8  
    Mailbox constants, 380  
    NBCCCommon.h, 810

MAILBOX9  
    Mailbox constants, 380  
    NBCCCommon.h, 810

MAX\_BT\_MSG\_SIZE  
    Miscellaneous Comm module constants, 502  
    NBCCCommon.h, 810

MENUICON\_CENTER  
    Display module constants, 484  
    NBCCCommon.h, 810

MENUICON\_LEFT  
    Display module constants, 484  
    NBCCCommon.h, 810

MENUICON\_RIGHT  
    Display module constants, 484  
    NBCCCommon.h, 810

MENUICONS  
    Display module constants, 484  
    NBCCCommon.h, 810

MENUTEXT  
    Display module constants, 484  
    NBCCCommon.h, 811

MI\_ADDR\_XG1300L  
    Microinfinity CruizCore XG1300L sensor constants, 684  
    NBCCCommon.h, 811

MIN\_1  
    NBCCCommon.h, 811  
    Time constants, 375

MS\_1  
    NBCCCommon.h, 811  
    Time constants, 375

MS\_10  
    NBCCCommon.h, 811  
    Time constants, 375

MS\_100  
    NBCCCommon.h, 811  
    Time constants, 375

MS\_150  
    NBCCCommon.h, 811  
    Time constants, 375

MS\_2  
    NBCCCommon.h, 811  
    Time constants, 375

MS\_20  
    NBCCCommon.h, 811  
    Time constants, 375

MS\_200  
    NBCCCommon.h, 811  
    Time constants, 375

MS\_250  
    NBCCCommon.h, 811  
    Time constants, 375

MS\_3  
    NBCCCommon.h, 811  
    Time constants, 375

MS\_30  
    NBCCCommon.h, 812  
    Time constants, 375

MS\_300  
    NBCCCommon.h, 812  
    Time constants, 376

MS\_350  
    NBCCCommon.h, 812  
    Time constants, 376

MS\_4  
    NBCCCommon.h, 812  
    Time constants, 376

MS\_40  
    NBCCCommon.h, 812  
    Time constants, 376

MS\_400  
    NBCCCommon.h, 812  
    Time constants, 376

MS\_450  
    NBCCCommon.h, 812  
    Time constants, 376

MS\_5  
    NBCCCommon.h, 812  
    Time constants, 376

MS\_50  
    NBCCCommon.h, 812  
    Time constants, 376

MS\_500  
    NBCCCommon.h, 812  
    Time constants, 376

MS\_6  
    NBCCCommon.h, 812  
    Time constants, 376

MS\_60  
    NBCCCommon.h, 812  
    Time constants, 376

MS\_600  
    NBCCCommon.h, 812  
    Time constants, 376

MS\_7  
    NBCCCommon.h, 812  
    Time constants, 376

MS\_70  
    NBCCCommon.h, 813  
    Time constants, 376

MS\_700  
    NBCCCommon.h, 813  
    Time constants, 377

MS\_8  
    NBCCCommon.h, 813  
    Time constants, 377

MS\_80  
    NBCCCommon.h, 813  
    Time constants, 377

MS\_800  
    NBCCCommon.h, 813  
    Time constants, 377

MS\_9  
    NBCCCommon.h, 813  
    Time constants, 377

MS\_90  
    NBCCCommon.h, 813  
    Time constants, 377

MS\_900  
    NBCCCommon.h, 813  
    Time constants, 377

MS\_ADDR\_ACCLNX  
    MindSensors device constants, 605  
    NBCCCommon.h, 813

MS\_ADDR\_CMPSNX  
    MindSensors device constants, 605  
    NBCCCommon.h, 813

MS\_ADDR\_DISTNX  
    MindSensors device constants, 605  
    NBCCCommon.h, 813

MS\_ADDR\_IVSENS  
    MindSensors device constants, 605  
    NBCCCommon.h, 813

MS\_ADDR\_LINELDR  
    MindSensors device constants, 605  
    NBCCCommon.h, 813

MS\_ADDR\_MTRMUX  
    MindSensors device constants, 605  
    NBCCCommon.h, 813

MS\_ADDR\_NRLINK  
    MindSensors device constants, 605  
    NBCCCommon.h, 814

MS\_ADDR\_NUMERICPAD  
    MindSensors device constants, 605  
    NBCCCommon.h, 814

MS\_ADDR\_NXTCAM  
    MindSensors device constants, 605  
    NBCCCommon.h, 814

MS\_ADDR\_NXTHID  
    MindSensors device constants, 605  
    NBCCCommon.h, 814

MS\_ADDR\_NXTMMX  
    MindSensors device constants, 605  
    NBCCCommon.h, 814

MS\_ADDR\_NXTSERVO  
    MindSensors device constants, [606](#)  
    NBCCommon.h, [814](#)

MS\_ADDR\_PFMATE  
    MindSensors device constants, [606](#)  
    NBCCommon.h, [814](#)

MS\_ADDR\_PSPNX  
    MindSensors device constants, [606](#)  
    NBCCommon.h, [814](#)

MS\_ADDR\_RTCLOCK  
    MindSensors device constants, [606](#)  
    NBCCommon.h, [814](#)

MS\_ADDR\_RXMUX  
    MindSensors device constants, [606](#)  
    NBCCommon.h, [814](#)

MS\_ADDR\_TOUCHPANEL  
    MindSensors device constants, [606](#)  
    NBCCommon.h, [814](#)

MS\_CMD\_ADPA\_OFF  
    MindSensors device constants, [606](#)  
    NBCCommon.h, [814](#)

MS\_CMD\_ADPA\_ON  
    MindSensors device constants, [606](#)  
    NBCCommon.h, [814](#)

MS\_CMD\_DEENERGIZED  
    MindSensors device constants, [606](#)  
    NBCCommon.h, [815](#)

MS\_CMD\_ENERGIZED  
    MindSensors device constants, [606](#)  
    NBCCommon.h, [815](#)

MSADPAOff  
    MindSensors API Functions, [135](#)

MSADPAOn  
    MindSensors API Functions, [135](#)

MSDeenergize  
    MindSensors API Functions, [136](#)

MSEnergize  
    MindSensors API Functions, [136](#)

MSIRTrain  
    MindSensors API Functions, [136](#)

MSPFComboDirect  
    MindSensors API Functions, [136](#)

MSPFComboPWM  
    MindSensors API Functions, [137](#)

MSPFRawOutput  
    MindSensors API Functions, [137](#)

MSPFRepeat  
    MindSensors API Functions, [138](#)

MSPFSingleOutputCST  
    MindSensors API Functions, [138](#)

MSPFSingleOutputPWM  
    MindSensors API Functions, [138](#)

MSPFSinglePin  
    MindSensors API Functions, [139](#)

MSPFTrain  
    MindSensors API Functions, [139](#)

MSRCXAbsVar  
    MindSensors API Functions, [139](#)

MSRCXAddToDatalog  
    MindSensors API Functions, [139](#)

MSRCXAndVar  
    MindSensors API Functions, [140](#)

MSRCXBatteryLevel  
    MindSensors API Functions, [140](#)

MSRCXBoot  
    MindSensors API Functions, [140](#)

MSRCXCalibrateEvent  
    MindSensors API Functions, [140](#)

MSRCXClearAllEvents  
    MindSensors API Functions, [140](#)

MSRCXClearCounter  
    MindSensors API Functions, [140](#)

MSRCXClearMsg  
    MindSensors API Functions, [141](#)

MSRCXClearSensor  
    MindSensors API Functions, [141](#)

MSRCXClearSound  
    MindSensors API Functions, [141](#)

MSRCXClearTimer  
    MindSensors API Functions, [141](#)

MSRCXCreateDatalog  
    MindSensors API Functions, [141](#)

MSRCXDecCounter  
    MindSensors API Functions, [141](#)

MSRCXDeleteSub  
    MindSensors API Functions, [141](#)

MSRCXDeleteSubs  
    MindSensors API Functions, [142](#)

MSRCXDeleteTask  
    MindSensors API Functions, [142](#)

MSRCXDeleteTasks  
    MindSensors API Functions, [142](#)

MSRCXDisableOutput  
    MindSensors API Functions, [142](#)

MSRCXDivVar  
    MindSensors API Functions, [142](#)

MSRCXEnableOutput  
    MindSensors API Functions, [142](#)

MSRCXEvent  
    MindSensors API Functions, [143](#)

MSRCXFloat  
    MindSensors API Functions, [143](#)

MSRCXFwd  
    MindSensors API Functions, [143](#)

MSRCXIIncCounter  
    MindSensors API Functions, [143](#)

MSRCXIInvertOutput  
    MindSensors API Functions, [143](#)

MSRCXMulVar  
    MindSensors API Functions, 143

MSRCXMutateSound  
    MindSensors API Functions, 144

MSRCXObvertOutput  
    MindSensors API Functions, 144

MSRCXOff  
    MindSensors API Functions, 144

MSRCXOn  
    MindSensors API Functions, 144

MSRCXOnFor  
    MindSensors API Functions, 144

MSRCXOnFwd  
    MindSensors API Functions, 145

MSRCXOnRev  
    MindSensors API Functions, 145

MSRCXOrVar  
    MindSensors API Functions, 145

MSRCXPBTurnOff  
    MindSensors API Functions, 145

MSRCXPing  
    MindSensors API Functions, 145

MSRCXPlaySound  
    MindSensors API Functions, 145

MSRCXPlayTone  
    MindSensors API Functions, 145

MSRCXPlayToneVar  
    MindSensors API Functions, 146

MSRCXPoll  
    MindSensors API Functions, 146

MSRCXPollIMemory  
    MindSensors API Functions, 146

MSRCXRemote  
    MindSensors API Functions, 146

MSRCXReset  
    MindSensors API Functions, 146

MSRCXRev  
    MindSensors API Functions, 146

MSRCXSelectDisplay  
    MindSensors API Functions, 147

MSRCXSelectProgram  
    MindSensors API Functions, 147

MSRCXSendSerial  
    MindSensors API Functions, 147

MSRCXSet  
    MindSensors API Functions, 147

MSRCXSetDirection  
    MindSensors API Functions, 147

MSRCXSetEvent  
    MindSensors API Functions, 148

MSRCXSetGlobalDirection  
    MindSensors API Functions, 148

MSRCXSetGlobalOutput  
    MindSensors API Functions, 148

MSRCXSetMaxPower  
    MindSensors API Functions, 148

MSRCXSetMessage  
    MindSensors API Functions, 148

MSRCXSetNRLinkPort  
    MindSensors API Functions, 149

MSRCXSetOutput  
    MindSensors API Functions, 149

MSRCXSetPower  
    MindSensors API Functions, 149

MSRCXSetPriority  
    MindSensors API Functions, 149

MSRCXSetSensorMode  
    MindSensors API Functions, 149

MSRCXSetSensorType  
    MindSensors API Functions, 150

MSRCXSetSleepTime  
    MindSensors API Functions, 150

MSRCXSetTxPower  
    MindSensors API Functions, 150

MSRCXSetUserDisplay  
    MindSensors API Functions, 150

MSRCXSetVar  
    MindSensors API Functions, 150

MSRCXSetWatch  
    MindSensors API Functions, 151

MSRCXSgnVar  
    MindSensors API Functions, 151

MSRCXStartTask  
    MindSensors API Functions, 151

MSRCXStopAllTasks  
    MindSensors API Functions, 151

MSRCXStopTask  
    MindSensors API Functions, 151

MSRCXSubVar  
    MindSensors API Functions, 151

MSRCXSumVar  
    MindSensors API Functions, 152

MSRCXToggle  
    MindSensors API Functions, 152

MSRCXUnlock  
    MindSensors API Functions, 152

MSRCXUnmuteSound  
    MindSensors API Functions, 152

MSReadValue  
    MindSensors API Functions, 152

MSScoutCalibrateSensor  
    MindSensors API Functions, 153

MSScoutMuteSound  
    MindSensors API Functions, 153

MSScoutSelectSounds  
    MindSensors API Functions, 153

MSScoutSendVLL  
    MindSensors API Functions, 153

MSScoutSetCounterLimit  
    MindSensors API Functions, 153

MSScoutSetEventFeedback  
    MindSensors API Functions, 153

MSScoutSetLight  
    MindSensors API Functions, 154

MSScoutSetScoutMode  
    MindSensors API Functions, 154

MSScoutSetScoutRules  
    MindSensors API Functions, 154

MSScoutSetSensorClickTime  
    MindSensors API Functions, 154

MSScoutSetSensorHysteresis  
    MindSensors API Functions, 154

MSScoutSetSensorLowerLimit  
    MindSensors API Functions, 155

MSScoutSetSensorUpperLimit  
    MindSensors API Functions, 155

MSScoutSetTimerLimit  
    MindSensors API Functions, 155

MSScoutUnmuteSound  
    MindSensors API Functions, 155

Mailbox constants, 379  
    MAILBOX1, 379  
    MAILBOX10, 379  
    MAILBOX2, 379  
    MAILBOX3, 379  
    MAILBOX4, 379  
    MAILBOX5, 379  
    MAILBOX6, 379  
    MAILBOX7, 380  
    MAILBOX8, 380  
    MAILBOX9, 380

MaxAccelerationField  
    NBCCommon.h, 810  
    Output field constants, 458

MaxSpeedField  
    NBCCommon.h, 810  
    Output field constants, 458

MemoryManager  
    NBCCommon.h, 810  
    System Call function constants, 370

MessageRead  
    NBCCommon.h, 811  
    System Call function constants, 370

MessageWrite  
    NBCCommon.h, 811  
    System Call function constants, 370

Microinfinity API Functions, 192  
    ReadSensorMIXG1300L, 192  
    ReadSensorMIXG1300LScale, 192  
    ResetMIXG1300L, 193  
    SetSensorMIXG1300LScale, 193

Microinfinity CruzCore XG1300L, 686  
    XG1300L\_SCALE\_2G, 686  
    XG1300L\_SCALE\_4G, 686  
    XG1300L\_SCALE\_8G, 686

Microinfinity CruzCore XG1300L sensor constants, 684  
    MI\_ADDR\_XG1300L, 684  
    XG1300L\_REG\_2G, 684  
    XG1300L\_REG\_4G, 684  
    XG1300L\_REG\_8G, 684  
    XG1300L\_REG\_ANGLE, 684  
    XG1300L\_REG\_RESET, 684  
    XG1300L\_REG\_XAXIS, 685  
    XG1300L\_REG\_YAXIS, 685  
    XG1300L\_REG\_ZAXIS, 685

Microinfinity device constants, 683

MindSensors ACCL-Nx constants, 615

MindSensors ACCL-Nx sensitivity level constants, 618

MindSensors API Functions, 122  
    ACCLNxCalibrateX, 133  
    ACCLNxCalibrateXEnd, 133  
    ACCLNxCalibrateY, 133  
    ACCLNxCalibrateYEnd, 133  
    ACCLNxCalibrateZ, 133  
    ACCLNxCalibrateZEnd, 134  
    ACCLNxResetCalibration, 134  
    DISTNxGP2D12, 134  
    DISTNxGP2D120, 134  
    DISTNxGP2YA02, 135  
    DISTNxGP2YA21, 135  
    MSADPAOff, 135  
    MSADPAOn, 135  
    MSDenergize, 136  
    MSEnergize, 136  
    MSIRTrain, 136  
    MSPFComboDirect, 136  
    MSPFComboPWM, 137  
    MSPFRawOutput, 137  
    MSPFRepeat, 138  
    MSPFSingleOutputCST, 138  
    MSPFSingleOutputPWM, 138  
    MSPFSinglePin, 139  
    MSPFTrain, 139  
    MSRCXAbsVar, 139  
    MSRCXAddToDatalog, 139  
    MSRCXAndVar, 140  
    MSRCXBatteryLevel, 140  
    MSRCXBoot, 140  
    MSRCXCalibrateEvent, 140  
    MSRCXClearAllEvents, 140  
    MSRCXClearCounter, 140  
    MSRCXClearMsg, 141  
    MSRCXClearSensor, 141  
    MSRCXClearSound, 141  
    MSRCXClearTimer, 141  
    MSRCXCreateDatalog, 141

MSRCXDecCounter, 141  
MSRCXDeleteSub, 141  
MSRCXDeleteSubs, 142  
MSRCXDeleteTask, 142  
MSRCXDeleteTasks, 142  
MSRCXDisableOutput, 142  
MSRCXDivVar, 142  
MSRCXEnableOutput, 142  
MSRCXEvent, 143  
MSRCXFloat, 143  
MSRCXFwd, 143  
MSRCXIIncCounter, 143  
MSRCXIInvertOutput, 143  
MSRCXMulVar, 143  
MSRCXMuteSound, 144  
MSRCXObvertOutput, 144  
MSRCXOff, 144  
MSRCXOn, 144  
MSRCXOnFor, 144  
MSRCXOnFwd, 145  
MSRCXOnRev, 145  
MSRCXOrVar, 145  
MSRCXPBTurnOff, 145  
MSRCXPing, 145  
MSRCXPlaySound, 145  
MSRCXPlayTone, 145  
MSRCXPlayToneVar, 146  
MSRCXPoll, 146  
MSRCXPollMemory, 146  
MSRCXRemote, 146  
MSRCXReset, 146  
MSRCXRev, 146  
MSRCXSelectDisplay, 147  
MSRCXSelectProgram, 147  
MSRCXSendSerial, 147  
MSRCXSet, 147  
MSRCXSetDirection, 147  
MSRCXSetEvent, 148  
MSRCXSetGlobalDirection, 148  
MSRCXSetGlobalOutput, 148  
MSRCXSetMaxPower, 148  
MSRCXSetMessage, 148  
MSRCXSetNRLinkPort, 149  
MSRCXSetOutput, 149  
MSRCXSetPower, 149  
MSRCXSetPriority, 149  
MSRCXSetSensorMode, 149  
MSRCXSetSensorType, 150  
MSRCXSetSleepTime, 150  
MSRCXSetTxPower, 150  
MSRCXSetUserDisplay, 150  
MSRCXSetVar, 150  
MSRCXSetWatch, 151  
MSRCXSgnVar, 151  
MSRCXStartTask, 151  
MSRCXStopAllTasks, 151  
MSRCXStopTask, 151  
MSRCXSubVar, 151  
MSRCXSumVar, 152  
MSRCXToggle, 152  
MSRCXUnlock, 152  
MSRCXUnmuteSound, 152  
MSReadValue, 152  
MSScoutCalibrateSensor, 153  
MSScoutMuteSound, 153  
MSScoutSelectSounds, 153  
MSScoutSendVLL, 153  
MSScoutSetCounterLimit, 153  
MSScoutSetEventFeedback, 153  
MSScoutSetLight, 154  
MSScoutSetScoutMode, 154  
MSScoutSetScoutRules, 154  
MSScoutSetSensorClickTime, 154  
MSScoutSetSensorHysteresis, 154  
MSScoutSetSensorLowerLimit, 155  
MSScoutSetSensorUpperLimit, 155  
MSScoutSetTimerLimit, 155  
MSScoutUnmuteSound, 155  
NRLLink2400, 155  
NRLLink4800, 155  
NRLLinkFlush, 156  
NRLLinkIRLong, 156  
NRLLinkIRShort, 156  
NRLLinkSetPF, 156  
NRLLinkSetRCX, 157  
NRLLinkSetTrain, 157  
NRLLinkTxRaw, 157  
NXTHIDAsciiMode, 157  
NXTHIDDirectMode, 158  
NXTHIDLoadCharacter, 158  
NXTHIDTransmit, 158  
NXTLineLeaderCalibrateBlack, 158  
NXTLineLeaderCalibrateWhite, 159  
NXTLineLeaderInvert, 159  
NXTLineLeaderPowerDown, 159  
NXTLineLeaderPowerUp, 160  
NXTLineLeaderReset, 160  
NXTLineLeaderSnapshot, 160  
NXTPowerMeterResetCounters, 160  
NXTServoEditMacro, 161  
NXTServoGotoMacroAddress, 161  
NXTServoHaltMacro, 161  
NXTServoInit, 161  
NXTServoPauseMacro, 162  
NXTServoQuitEdit, 162  
NXTServoReset, 162  
NXTServoResumeMacro, 163  
PFMateSend, 163

PFMateSendRaw, 163  
PSPNxAnalog, 164  
PSPNxDigital, 164  
ReadACCLNxSensitivity, 164  
ReadACCLNxXOffset, 164  
ReadACCLNxXRange, 164  
ReadACCLNxYOffset, 165  
ReadACCLNxYRange, 165  
ReadACCLNxZOffset, 165  
ReadACCLNxZRange, 166  
ReadDISTNxDistance, 166  
ReadDISTNxMaxDistance, 166  
ReadDISTNxMinDistance, 166  
ReadDISTNxModuleType, 167  
ReadDISTNxNumPoints, 167  
ReadDISTNxVoltage, 167  
ReadNRLinkBytes, 167  
ReadNRLinkStatus, 168  
ReadNXTLineLeaderAverage, 168  
ReadNXTLineLeaderResult, 168  
ReadNXTLineLeaderSteering, 169  
ReadNXTPowerMeterCapacityUsed, 169  
ReadNXTPowerMeterElapsedTime, 169  
ReadNXTPowerMeterErrorCount, 169  
ReadNXTPowerMeterMaxCurrent, 170  
ReadNXTPowerMeterMaxVoltage, 170  
ReadNXTPowerMeterMinCurrent, 170  
ReadNXTPowerMeterMinVoltage, 171  
ReadNXTPowerMeterPresentCurrent, 171  
ReadNXTPowerMeterPresentPower, 171  
ReadNXTPowerMeterPresentVoltage, 171  
ReadNXTPowerMeterTotalPowerConsumed, 172  
ReadNXTServoBatteryVoltage, 172  
ReadNXTServoPosition, 172  
ReadNXTServoSpeed, 173  
ReadSensorMSAccel, 173  
ReadSensorMSCompass, 173  
ReadSensorMSDROD, 173  
ReadSensorMSPlayStation, 174  
ReadSensorMSPressure, 174  
ReadSensorMSPressureRaw, 174  
ReadSensorMSRTClock, 174  
ReadSensorMSTilt, 175  
ReadSensorNXTSumoEyes, 175  
RunNRLinkMacro, 175  
SetACCLNxSensitivity, 176  
SetNXTLineLeaderKdFactor, 176  
SetNXTLineLeaderKdValue, 176  
SetNXTLineLeaderKiFactor, 176  
SetNXTLineLeaderKiValue, 177  
SetNXTLineLeaderKpFactor, 177  
SetNXTLineLeaderKpValue, 177  
SetNXTLineLeaderSetpoint, 178  
SetNXTServoPosition, 178  
SetNXTServoQuickPosition, 178  
SetNXTServoSpeed, 179  
SetSensorMSDRODActive, 179  
SetSensorMSDRODInactive, 179  
SetSensorMSPressure, 179  
SetSensorMSTouchMux, 179  
SetSensorNXTSumoEyesLong, 180  
SetSensorNXTSumoEyesShort, 180  
WriteNRLinkBytes, 180  
MindSensors DIST-Nx constants, 607  
DIST\_CMD\_GP2D12, 607  
DIST\_CMD\_GP2D120, 607  
DIST\_CMD\_GP2YA02, 607  
DIST\_CMD\_GP2YA21, 607  
DIST\_REG\_DIST, 607  
DIST\_REG\_DIST1, 607  
DIST\_REG\_VOLT, 608  
DIST\_REG\_VOLT1, 608  
MindSensors device constants, 604  
MS\_ADDR\_ACCLNX, 605  
MS\_ADDR\_CMPSNX, 605  
MS\_ADDR\_DISTNX, 605  
MS\_ADDR\_IVSENS, 605  
MS\_ADDR\_LINEldr, 605  
MS\_ADDR\_MTRMUX, 605  
MS\_ADDR\_NRLINK, 605  
MS\_ADDR\_NUMERICPAD, 605  
MS\_ADDR\_NXTCAM, 605  
MS\_ADDR\_NXTHID, 605  
MS\_ADDR\_NXTMMX, 605  
MS\_ADDR\_NXTSERVO, 606  
MS\_ADDR\_PFMATE, 606  
MS\_ADDR\_PSPNX, 606  
MS\_ADDR\_RTCLOCK, 606  
MS\_ADDR\_RXMUX, 606  
MS\_ADDR\_TOUCHPANEL, 606  
MS\_CMD\_ADPA\_OFF, 606  
MS\_CMD\_ADPA\_ON, 606  
MS\_CMD\_DEENERGIZED, 606  
MS\_CMD\_ENERGIZED, 606  
MindSensors nRLink constants, 613  
NRLINK\_CMD\_2400, 613  
NRLINK\_CMD\_4800, 613  
NRLINK\_CMD\_FLUSH, 613  
NRLINK\_REG\_BYTES, 614  
NRLINK\_REG\_DATA, 614  
MindSensors NXTHID commands, 636  
MindSensors NXTHID constants, 632  
MindSensors NXTHID modifier keys, 634  
MindSensors NXTHID registers, 633  
MindSensors NXTLineLeader commands, 645  
    NXTLL\_CMD\_USA, 645  
MindSensors NXTLineLeader constants, 642  
MindSensors NXTLineLeader registers, 643

NXTLL\_REG\_CMD, 643  
MindSensors NXTNumericPad constants, 647  
MindSensors NXTNumericPad registers, 648  
MindSensors NXTPowerMeter commands, 640  
MindSensors NXTPowerMeter constants, 637  
MindSensors NXTPowerMeter registers, 638  
    NXTPM\_REG\_CMD, 638  
    NXTPM\_REG\_GAIN, 638  
    NXTPM\_REG\_TIME, 639  
MindSensors NXTServo commands, 630  
MindSensors NXTServo constants, 623  
MindSensors NXTServo position constants, 627  
MindSensors NXTServo quick position constants, 628  
MindSensors NXTServo registers, 624  
MindSensors NXTServo servo numbers, 629  
    NXTSERVO\_SERVO\_1, 629  
    NXTSERVO\_SERVO\_2, 629  
    NXTSERVO\_SERVO\_3, 629  
    NXTSERVO\_SERVO\_4, 629  
    NXTSERVO\_SERVO\_5, 629  
    NXTSERVO\_SERVO\_6, 629  
    NXTSERVO\_SERVO\_7, 629  
    NXTSERVO\_SERVO\_8, 629  
MindSensors NXTSumoEyes constants, 641  
MindSensors NXTTouchPanel commands, 651  
    NXTPP\_CMD\_USA, 651  
MindSensors NXTTouchPanel constants, 649  
MindSensors NXTTouchPanel registers, 650  
    NXTPP\_REG\_CMD, 650  
MindSensors PFMate constants, 619  
    PFMATE\_CMD\_GO, 619  
    PFMATE\_CMD\_RAW, 619  
    PFMATE\_REG\_CMD, 620  
MindSensors PSP-Nx button set 1 constants, 611  
    PSP\_BTNSET1\_DOWN, 611  
    PSP\_BTNSET1\_L3, 611  
    PSP\_BTNSET1\_LEFT, 611  
    PSP\_BTNSET1\_R3, 611  
    PSP\_BTNSET1\_UP, 611  
MindSensors PSP-Nx button set 2 constants, 612  
    PSP\_BTNSET2\_L1, 612  
    PSP\_BTNSET2\_L2, 612  
    PSP\_BTNSET2\_R1, 612  
    PSP\_BTNSET2\_R2, 612  
MindSensors PSP-Nx constants, 609  
    PSP\_CMD\_ANALOG, 609  
    PSP\_CMD\_DIGITAL, 609  
    PSP\_REG\_BTNSET1, 609  
    PSP\_REG\_BTNSET2, 609  
    PSP\_REG\_XLEFT, 609  
    PSP\_REG\_XRIGHT, 609  
    PSP\_REG\_YLEFT, 609  
    PSP\_REG\_YRIGHT, 610  
Miscellaneous Comm module constants, 502  
BT\_CMD\_BYTE, 502  
MAX\_BT\_MSG\_SIZE, 502  
SIZE\_OF\_BDADDR, 502  
SIZE\_OF\_BRICK\_NAME, 502  
SIZE\_OF\_BT\_NAME, 503  
SIZE\_OF\_BT\_PINCODE, 503  
SIZE\_OF\_BTBUF, 503  
SIZE\_OF\_HSBUF, 503  
SIZE\_OF\_USBBUF, 503  
SIZE\_OF\_USBDATA, 503  
Miscellaneous NBC/NXC constants, 203  
    FALSE, 203  
    NA, 203  
    PI, 203  
    TRUE, 203  
NA  
    Miscellaneous NBC/NXC constants, 203  
    NBCCCommon.h, 815  
NBC Input port constants, 430  
    IN\_1, 430  
    IN\_2, 430  
    IN\_3, 430  
    IN\_4, 430  
NBC sensor mode constants, 434  
    IN\_MODE\_ANGLESTEP, 434  
    IN\_MODE\_BOOLEAN, 434  
    IN\_MODE\_CELSIUS, 434  
    IN\_MODE\_MODEMASK, 434  
    IN\_MODE\_RAW, 435  
    IN\_MODE\_SLOPEMASK, 435  
NBC sensor type constants, 431  
    IN\_TYPE\_ANGLE, 431  
    IN\_TYPE\_COLORBLUE, 431  
    IN\_TYPE\_COLOREXIT, 431  
    IN\_TYPE\_COLORFULL, 431  
    IN\_TYPE\_COLORNONE, 432  
    IN\_TYPE\_COLORRED, 432  
    IN\_TYPE\_CUSTOM, 432  
    IN\_TYPE\_HISPEED, 432  
    IN\_TYPE\_LOWSPEED, 432  
    IN\_TYPE\_SOUND\_DB, 432  
    IN\_TYPE\_SWITCH, 432  
NBCCAPIDocs.h, 694  
NBCCCommon.h, 694  
    ACCL\_CMD\_X\_CAL, 735  
    ACCL\_CMD\_Y\_CAL, 735  
    ACCL\_CMD\_Z\_CAL, 735  
    ACCL\_REG\_X\_TILT, 736  
    ACCL\_REG\_Y\_TILT, 736  
    ACCL\_REG\_Z\_TILT, 736  
    ActualSpeedField, 737  
    BITMAP\_1, 737  
    BITMAP\_2, 737

BITMAP\_3, 737  
BITMAP\_4, 737  
BITMAPS, 737  
BREAKOUT\_REQ, 737  
BT\_ARM\_CMD\_MODE, 737  
BT\_ARM\_OFF, 738  
BT\_CMD\_BYTE, 738  
BT\_CMD\_READY, 738  
BT\_DEVICE\_AWAY, 738  
BT\_DEVICE\_EMPTY, 738  
BT\_DEVICE\_KNOWN, 738  
BT\_DEVICE\_NAME, 739  
BT\_DISABLE, 739  
BT\_ENABLE, 739  
BTN1, 739  
BTN2, 739  
BTN3, 739  
BTN4, 739  
BTNCENTER, 739  
BTNEXT, 739  
BTNLEFT, 739  
BTNRIGHT, 739  
BTNSTATE\_NONE, 740  
BlockTachoCountField, 737  
ButtonModuleID, 740  
ButtonModuleName, 740  
ButtonOffsetLongPressCnt, 740  
ButtonOffsetLongRelCnt, 740  
ButtonOffsetPressedCnt, 740  
ButtonOffsetRelCnt, 740  
ButtonOffsetShortRelCnt, 740  
ButtonOffsetState, 740  
CHAR\_BIT, 740  
CHAR\_MAX, 740  
CHAR\_MIN, 741  
CLUMP\_DONE, 741  
CLUMP\_SUSPEND, 741  
CONN\_BT0, 747  
CONN\_BT1, 747  
CONN\_BT2, 747  
CONN\_BT3, 747  
CONN\_HS4, 747  
CONN\_HS\_1, 747  
CONN\_HS\_2, 747  
CONN\_HS\_3, 748  
CONN\_HS\_4, 748  
CONN\_HS\_5, 748  
CONN\_HS\_6, 748  
CONN\_HS\_7, 748  
CONN\_HS\_8, 748  
CONN\_HS\_ALL, 748  
CT\_ADDR\_RFID, 748  
CT\_REG\_DATA, 748  
CT\_REG\_MODE, 748  
CT\_REG\_STATUS, 748  
ColorSensorRead, 741  
CommBTCheckStatus, 742  
CommBTConnection, 742  
CommBTOnOff, 742  
CommBTRead, 742  
CommBTWrite, 743  
CommExecuteFunction, 743  
CommHSCheckStatus, 743  
CommHSControl, 743  
CommHSRead, 743  
CommHSWrite, 743  
CommLSCheckStatus, 743  
CommLSRead, 743  
CommLSWrite, 743  
CommLSWriteEx, 743  
CommModuleID, 743  
CommModuleName, 743  
CommOffsetBrickDataBdAddr, 743  
CommOffsetBrickDataBluecoreVersion, 743  
CommOffsetBrickDataBtHwStatus, 744  
CommOffsetBrickDataBtStateStatus, 744  
CommOffsetBrickDataName, 744  
CommOffsetBrickDataTimeOutValue, 744  
CommOffsetBtConnectTableBdAddr, 744  
CommOffsetBtConnectTableClassOfDevice, 744  
CommOffsetBtConnectTableHandleNr, 744  
CommOffsetBtConnectTableLinkQuality, 744  
CommOffsetBtConnectTableName, 744  
CommOffsetBtConnectTablePinCode, 744  
CommOffsetBtConnectTableStreamStatus, 744  
CommOffsetBtDataMode, 744  
CommOffsetBtDeviceCnt, 744  
CommOffsetBtDeviceNameCnt, 744  
CommOffsetBtDeviceTableBdAddr, 745  
CommOffsetBtDeviceTableClassOfDevice, 745  
CommOffsetBtDeviceTableDeviceStatus, 745  
CommOffsetBtDeviceTableName, 745  
CommOffsetBtInBufBuf, 745  
CommOffsetBtInBufInPtr, 745  
CommOffsetBtInBufOutPtr, 745  
CommOffsetBtOutBufBuf, 745  
CommOffsetBtOutBuflnPtr, 745  
CommOffsetBtOutBufOutPtr, 745  
CommOffsetHsAddress, 745  
CommOffsetHsDataMode, 745  
CommOffsetHsFlags, 745  
CommOffsetHsInBufBuf, 745  
CommOffsetHsInBuflnPtr, 746  
CommOffsetHsInBufOutPtr, 746  
CommOffsetHsMode, 746  
CommOffsetHsOutBufBuf, 746  
CommOffsetHsOutBuflnPtr, 746  
CommOffsetHsOutBufOutPtr, 746

CommOffsetHsSpeed, 746  
CommOffsetHsState, 746  
CommOffsetPFunc, 746  
CommOffsetPFuncTwo, 746  
CommOffsetUsbInBufBuf, 746  
CommOffsetUsbInBufInPtr, 746  
CommOffsetUsbInBufOutPtr, 746  
CommOffsetUsbOutBufBuf, 746  
CommOffsetUsbOutBufInPtr, 747  
CommOffsetUsbOutBufOutPtr, 747  
CommOffsetUsbPollBufBuf, 747  
CommOffsetUsbPollBufInPtr, 747  
CommOffsetUsbPollBufOutPtr, 747  
CommOffsetUsbState, 747  
CommandModuleID, 741  
CommandModuleName, 741  
CommandOffsetActivateFlag, 741  
CommandOffsetAwake, 741  
CommandOffsetDeactivateFlag, 741  
CommandOffsetFileName, 742  
CommandOffsetFormatString, 742  
CommandOffsetMemoryPool, 742  
CommandOffsetOffsetDS, 742  
CommandOffsetOffsetDVA, 742  
CommandOffsetPRCHandler, 742  
CommandOffsetProgStatus, 742  
CommandOffsetSyncTick, 742  
CommandOffsetSyncTime, 742  
CommandOffsetTick, 742  
ComputeCalibValue, 747  
DAC\_MODE\_DCOUT, 748  
DATA\_MODE\_GPS, 749  
DATA\_MODE\_MASK, 749  
DATA\_MODE\_NXT, 749  
DATA\_MODE\_RAW, 749  
DATA\_MODE\_UPDATE, 749  
DGPS\_REG\_HEADING, 750  
DGPS\_REG\_STATUS, 750  
DGPS\_REG\_TIME, 750  
DI\_ADDR\_ACCL, 750  
DI\_ADDR\_DGPS, 750  
DI\_ADDR\_GYRO, 750  
DIACCL\_MODE\_GLVL2, 751  
DIACCL\_MODE\_GLVL4, 752  
DIACCL\_MODE\_GLVL8, 752  
DIACCL\_REG\_CTRL1, 752  
DIACCL\_REG\_CTRL2, 752  
DIACCL\_REG\_X8, 753  
DIACCL\_REG\_XHIGH, 753  
DIACCL\_REG\_XLOW, 753  
DIACCL\_REG\_Y8, 753  
DIACCL\_REG\_YHIGH, 753  
DIACCL\_REG\_YLOW, 754  
DIACCL\_REG\_Z8, 754  
DIACCL\_REG\_ZHIGH, 754  
DIACCL\_REG\_ZLOW, 754  
DIGI\_PIN0, 754  
DIGI\_PIN1, 754  
DIGI\_PIN2, 754  
DIGI\_PIN3, 755  
DIGI\_PIN4, 755  
DIGI\_PIN5, 755  
DIGI\_PIN6, 755  
DIGI\_PIN7, 755  
DIGYRO\_REG\_CTRL1, 759  
DIGYRO\_REG\_CTRL2, 759  
DIGYRO\_REG\_CTRL3, 759  
DIGYRO\_REG\_CTRL4, 759  
DIGYRO\_REG\_CTRL5, 759  
DIGYRO\_REG\_XHIGH, 760  
DIGYRO\_REG\_XLOW, 760  
DIGYRO\_REG\_YHIGH, 761  
DIGYRO\_REG\_YLOW, 761  
DIGYRO\_REG\_ZHIGH, 761  
DIGYRO\_REG\_ZLOW, 761  
DISPLAY\_BUSY, 762  
DISPLAY\_CHAR, 762  
DISPLAY\_FRAME, 762  
DISPLAY\_HEIGHT, 762  
DISPLAY\_ON, 763  
DISPLAY\_PIXEL, 763  
DISPLAY\_POPUP, 763  
DISPLAY\_REFRESH, 763  
DISPLAY\_WIDTH, 763  
DIST\_CMD\_CUSTOM, 765  
DIST\_CMD\_GP2D12, 765  
DIST\_CMD\_GP2D120, 765  
DIST\_CMD\_GP2YA02, 765  
DIST\_CMD\_GP2YA21, 765  
DIST\_REG\_DIST, 765  
DIST\_REG\_DIST1, 765  
DIST\_REG\_VOLT, 765  
DIST\_REG\_VOLT1, 766  
DRAW\_OPT\_CLEAR, 766  
DRAW\_OPT\_INVERT, 767  
DRAW\_OPT\_NORMAL, 767  
DatalogGetTimes, 749  
DatalogWrite, 749  
DisplayExecuteFunction, 763  
DisplayModuleID, 763  
DisplayModuleName, 763  
DisplayOffsetContrast, 763  
DisplayOffsetDisplay, 763  
DisplayOffsetEraseMask, 763  
DisplayOffsetFlags, 763  
DisplayOffsetNormal, 764  
DisplayOffsetPBitmaps, 764  
DisplayOffsetPFont, 764

DisplayOffsetPFunc, 764  
DisplayOffsetPMenulcons, 764  
DisplayOffsetPMenuText, 764  
DisplayOffsetPScreens, 764  
DisplayOffsetPStatusIcons, 764  
DisplayOffsetPStatusText, 764  
DisplayOffsetPStepIcons, 764  
DisplayOffsetPTextLines, 764  
DisplayOffsetPopup, 764  
DisplayOffsetStatusIcons, 764  
DisplayOffsetStepIcons, 764  
DisplayOffsetTextLinesCenterFlags, 765  
DisplayOffsetUpdateMask, 765  
DrawCircle, 767  
DrawEllipse, 768  
DrawFont, 768  
DrawGraphic, 768  
DrawGraphicArray, 768  
DrawLine, 768  
DrawPoint, 768  
DrawPolygon, 768  
DrawRect, 768  
DrawText, 768  
EMETER\_REG\_AIN, 768  
EMETER\_REG\_AOUT, 768  
EMETER\_REG\_VIN, 768  
EMETER\_REG\_VOUT, 768  
EMETER\_REG\_WIN, 769  
EMETER\_REG\_WOUT, 769  
EOF, 769  
ERR\_ARG, 769  
ERR\_BAD\_PTR, 769  
ERR\_CLUMP\_COUNT, 769  
ERR\_FILE, 769  
ERR\_INSTR, 770  
ERR\_INVALID\_PORT, 770  
ERR\_INVALID\_SIZE, 770  
ERR\_LOADER\_ERR, 770  
ERR\_MEM, 770  
ERR\_MEMMGR\_FAIL, 770  
ERR\_NO\_CODE, 770  
ERR\_NO\_PROG, 770  
ERR\_NON\_FATAL, 770  
ERR\_RC\_FAILED, 770  
ERR\_VER, 771  
FALSE, 771  
FRAME\_SELECT, 772  
FREQUENCY\_MAX, 772  
FREQUENCY\_MIN, 772  
FileClose, 771  
FileDelete, 771  
FileFindFirst, 771  
FileFindNext, 771  
FileOpenAppend, 771  
FileOpenRead, 771  
FileOpenReadLinear, 771  
FileOpenWrite, 771  
FileOpenWriteLinear, 771  
FileOpenWriteNonLinear, 772  
FileRead, 772  
FileRename, 772  
FileResize, 772  
FileResolveHandle, 772  
FileSeek, 772  
FileTell, 772  
FileWrite, 772  
GL\_CAMERA\_DEPTH, 772  
GL\_CIRCLE, 772  
GL\_CIRCLE\_SIZE, 773  
GL\_CULL\_BACK, 773  
GL\_CULL\_FRONT, 773  
GL\_CULL\_MODE, 773  
GL\_CULL\_NONE, 773  
GL\_LINE, 773  
GL\_POINT, 773  
GL\_POLYGON, 773  
GL\_ROTATE\_X, 773  
GL\_ROTATE\_Y, 773  
GL\_ROTATE\_Z, 773  
GL\_SCALE\_X, 773  
GL\_SCALE\_Y, 773  
GL\_SCALE\_Z, 773  
GL\_TRANSLATE\_X, 774  
GL\_TRANSLATE\_Y, 774  
GL\_TRANSLATE\_Z, 774  
GL\_ZOOM\_FACTOR, 774  
GetStartTick, 772  
HS\_ADDRESS\_1, 774  
HS\_ADDRESS\_2, 774  
HS\_ADDRESS\_3, 774  
HS\_ADDRESS\_4, 774  
HS\_ADDRESS\_5, 774  
HS\_ADDRESS\_6, 774  
HS\_ADDRESS\_7, 774  
HS\_ADDRESS\_8, 774  
HS\_ADDRESS\_ALL, 774  
HS\_BAUD\_115200, 774  
HS\_BAUD\_1200, 775  
HS\_BAUD\_14400, 775  
HS\_BAUD\_19200, 775  
HS\_BAUD\_230400, 775  
HS\_BAUD\_2400, 775  
HS\_BAUD\_28800, 775  
HS\_BAUD\_3600, 775  
HS\_BAUD\_38400, 775  
HS\_BAUD\_460800, 775  
HS\_BAUD\_4800, 775  
HS\_BAUD\_57600, 775

HS\_BAUD\_7200, 775  
HS\_BAUD\_76800, 775  
HS\_BAUD\_921600, 775  
HS\_BAUD\_9600, 776  
HS\_BAUD\_DEFAULT, 776  
HS\_CMD\_READY, 776  
HS\_CTRL\_EXIT, 776  
HS\_CTRL\_INIT, 776  
HS\_CTRL\_UART, 776  
HS\_DEFAULT, 776  
HS\_DISABLE, 776  
HS\_ENABLE, 776  
HS\_INIT\_RECEIVER, 776  
HS\_INITIALISE, 776  
HS\_MODE\_10\_STOP, 776  
HS\_MODE\_15\_STOP, 776  
HS\_MODE\_20\_STOP, 777  
HS\_MODE\_5\_DATA, 777  
HS\_MODE\_6\_DATA, 777  
HS\_MODE\_7\_DATA, 777  
HS\_MODE\_7E1, 777  
HS\_MODE\_8\_DATA, 777  
HS\_MODE\_8N1, 777  
HS\_MODE\_DEFAULT, 777  
HS\_MODE\_MASK, 777  
HS\_MODE\_UART\_RS232, 778  
HS\_MODE\_UART\_RS485, 778  
HS\_SEND\_DATA, 778  
HS\_UART\_MASK, 778  
HS\_UPDATE, 778  
HT\_ADDR\_ACCEL, 778  
HT\_ADDR\_ANGLE, 778  
HT\_ADDR\_COLOR, 778  
HT\_ADDR\_COLOR2, 778  
HT\_ADDR\_COMPASS, 778  
HT\_ADDR\_IRLINK, 778  
HT\_ADDR\_IRSEEKER, 778  
HT\_ADDR\_IRSEEKER2, 779  
HT\_ADDR\_PIR, 779  
HT\_ADDR\_SUPERPRO, 779  
HT\_CH1\_A, 779  
HT\_CH1\_B, 779  
HT\_CH2\_A, 779  
HT\_CH2\_B, 779  
HT\_CH3\_A, 779  
HT\_CH3\_B, 779  
HT\_CH4\_A, 779  
HT\_CH4\_B, 779  
HT\_CMD\_COLOR2\_50HZ, 779  
HT\_CMD\_COLOR2\_60HZ, 779  
HTANGLE\_REG\_DC01, 780  
HTANGLE\_REG\_DC02, 781  
HTANGLE\_REG\_DC03, 781  
HTANGLE\_REG\_DC04, 781  
HTANGLE\_REG\_DC05, 781  
HTANGLE\_REG\_MODE, 781  
HTIR2\_MODE\_1200, 781  
HTIR2\_MODE\_600, 781  
HTIR2\_REG\_AC01, 781  
HTIR2\_REG\_AC02, 782  
HTIR2\_REG\_AC03, 782  
HTIR2\_REG\_AC04, 782  
HTIR2\_REG\_AC05, 782  
HTIR2\_REG\_ACDIR, 782  
HTIR2\_REG\_DC01, 782  
HTIR2\_REG\_DC02, 782  
HTIR2\_REG\_DC03, 782  
HTIR2\_REG\_DC04, 782  
HTIR2\_REG\_DC05, 782  
HTIR2\_REG\_DCAVG, 782  
HTIR2\_REG\_DCDIR, 782  
HTIR2\_REG\_MODE, 782  
HTPROTO\_A0, 783  
HTPROTO\_A1, 783  
HTPROTO\_A2, 783  
HTPROTO\_A3, 783  
HTPROTO\_A4, 783  
HTPROTO\_REG\_A0, 783  
HTPROTO\_REG\_A1, 783  
HTPROTO\_REG\_A2, 783  
HTPROTO\_REG\_A3, 783  
HTPROTO\_REG\_A4, 783  
HTPROTO\_REG\_DIN, 783  
HTPROTO\_REG\_DOUT, 783  
HTSPRO\_A0, 784  
HTSPRO\_A1, 784  
HTSPRO\_A2, 784  
HTSPRO\_A3, 784  
HTSPRO\_DAC0, 784  
HTSPRO\_DAC1, 784  
HTSPRO\_REG\_A0, 784  
HTSPRO\_REG\_A1, 784  
HTSPRO\_REG\_A2, 784  
HTSPRO\_REG\_A3, 784  
HTSPRO\_REG\_CTRL, 784  
HTSPRO\_REG\_DCTRL, 785  
HTSPRO\_REG\_DIN, 785  
HTSPRO\_REG\_DOUT, 785  
HTSPRO\_REG\_LED, 785  
I2C\_ADDR\_DEFAULT, 788  
I2C\_OPTION\_FAST, 788  
I2C\_REG\_CMD, 788  
I2C\_REG\_VERSION, 788  
IN\_1, 788  
IN\_2, 788  
IN\_3, 788  
IN\_4, 789  
IN\_MODE\_BOOLEAN, 789

IN\_MODE\_CELSIUS, 789  
IN\_MODE\_MODEMASK, 789  
IN\_MODE\_RAW, 789  
IN\_TYPE\_ANGLE, 789  
IN\_TYPE\_COLORRED, 790  
IN\_TYPE\_CUSTOM, 790  
IN\_TYPE\_HISPEED, 790  
IN\_TYPE\_LOWSPEED, 790  
IN\_TYPE\_SWITCH, 791  
INPUT\_BLACKCOLOR, 791  
INPUT\_BLANK, 791  
INPUT\_BLUE, 791  
INPUT\_BLUECOLOR, 791  
INPUT\_CUSTOM9V, 791  
INPUT\_DIGI0, 791  
INPUT\_DIGI1, 791  
INPUT\_GREEN, 792  
INPUT\_GREENCOLOR, 792  
INPUT\_PINCMD\_DIR, 792  
INPUT\_PINCMD\_SET, 792  
INPUT\_RED, 792  
INPUT\_REDCOLOR, 793  
INPUT\_RESETCAL, 793  
INPUT\_RUNNINGCAL, 793  
INPUT\_SENSORCAL, 793  
INPUT\_SENSOROFF, 793  
INPUT\_STARTCAL, 793  
INPUT\_WHITECOLOR, 793  
INPUT\_YELLOWCOLOR, 793  
INT\_MAX, 795  
INT\_MIN, 795  
INTF\_BTOFF, 795  
INTF\_BTNON, 795  
INTF\_CONNECT, 795  
INTF\_CONNECTREQ, 795  
INTF\_DISCONNECT, 795  
INTF\_EXTREAD, 795  
INTF\_FACTORYRESET, 796  
INTF\_OPENSTREAM, 796  
INTF\_PINREQ, 796  
INTF\_REMOVEDevice, 796  
INTF\_SEARCH, 796  
INTF\_SENDDATA, 796  
INTF\_SENDFILE, 796  
INTF\_SETBTNAME, 796  
INTF\_SETCMDMODE, 796  
INTF\_STOPSEARCH, 796  
INTF\_VISIBILITY, 796  
IOCTRL\_BOOT, 796  
IOCTRL\_POWERDOWN, 796  
IOCtrlModuleID, 797  
IOCtrlModuleName, 797  
IOCtrlOffsetPowerOn, 797  
IOMapRead, 797  
IOMapReadByID, 797  
IOMapWrite, 797  
IOMapWriteByID, 797  
InputModeField, 793  
InputModuleID, 793  
InputModuleName, 793  
InputOffsetADRaw, 793  
InputOffsetColorADRaw, 793  
InputOffsetColorBoolean, 793  
InputOffsetColorCallLimits, 794  
InputOffsetColorCalibration, 794  
InputOffsetColorCalibrationState, 794  
InputOffsetColorSensorRaw, 794  
InputOffsetColorSensorValue, 794  
InputOffsetCustomActiveStatus, 794  
InputOffsetCustomPctFullScale, 794  
InputOffsetCustomZeroOffset, 794  
InputOffsetDigiPinsDir, 794  
InputOffsetDigiPinsIn, 794  
InputOffsetDigiPinsOut, 794  
InputOffsetInvalidData, 794  
InputOffsetSensorBoolean, 794  
InputOffsetSensorMode, 794  
InputOffsetSensorRaw, 795  
InputOffsetSensorType, 795  
InputOffsetSensorValue, 795  
InputPinFunction, 795  
InvalidDataField, 796  
JOY\_BTN\_01, 797  
JOY\_BTN\_02, 797  
JOY\_BTN\_03, 797  
JOY\_BTN\_04, 797  
JOY\_BTN\_05, 797  
JOY\_BTN\_06, 797  
JOY\_BTN\_07, 797  
JOY\_BTN\_08, 798  
JOY\_BTN\_09, 798  
JOY\_BTN\_10, 798  
JOY\_BTN\_11, 798  
JOY\_BTN\_12, 798  
JOY\_BTN\_13, 798  
JOY\_BTN\_14, 798  
JOY\_BTN\_15, 798  
JOY\_BTN\_16, 798  
JOY\_BTN\_17, 798  
JOY\_BTN\_18, 798  
JOY\_BTN\_19, 798  
JOY\_BTN\_20, 798  
JOY\_BTN\_21, 798  
JOY\_BTN\_22, 799  
JOY\_BTN\_23, 799  
JOY\_BTN\_24, 799  
JOY\_BTN\_25, 799  
JOY\_BTN\_26, 799

JOY\_BTN\_27, 799  
JOY\_BTN\_28, 799  
JOY\_BTN\_29, 799  
JOY\_BTN\_30, 799  
JOY\_BTN\_31, 799  
JOY\_BTN\_32, 799  
JOY\_POV\_BACKWARD, 799  
JOY\_POV\_BOTLEFT, 799  
JOY\_POV\_BOTRIGHT, 799  
JOY\_POV\_CENTERED, 800  
JOY\_POV\_FORWARD, 800  
JOY\_POV\_LEFT, 800  
JOY\_POV\_RIGHT, 800  
JOY\_POV\_TOPLEFT, 800  
JOY\_POV\_TOPRIGHT, 800  
KeepAlive, 800  
LCD\_LINE1, 800  
LCD\_LINE2, 800  
LCD\_LINE3, 800  
LCD\_LINE4, 800  
LCD\_LINE5, 800  
LCD\_LINE6, 800  
LCD\_LINE7, 800  
LCD\_LINE8, 801  
LDR\_BTBUSY, 801  
LDR\_BTCONNECTFAIL, 801  
LDR\_BTTIMEOUT, 801  
LDR\_CMD\_BOOTCMD, 801  
LDR\_CMD\_BTGETADR, 801  
LDR\_CMD\_CLOSE, 801  
LDR\_CMD\_DELETE, 801  
LDR\_CMD\_FINDNEXT, 802  
LDR\_CMD\_OPENREAD, 802  
LDR\_CMD\_POLLCMD, 802  
LDR\_CMD\_READ, 803  
LDR\_CMD\_VERSIONS, 803  
LDR\_CMD\_WRITE, 803  
LDR\_ENDOFFILE, 803  
LDR\_EOFEXPECTED, 803  
LDR\_FILEEXIST, 803  
LDR\_FILEISBUSY, 803  
LDR\_FILEISFULL, 803  
LDR\_FILENOFOUND, 804  
LDR\_ILLEGALHANDLE, 804  
LDR\_INPROGRESS, 804  
LDR\_INVALIDSEEK, 804  
LDR\_NOLINEARSPACE, 804  
LDR\_NOMOREFILES, 804  
LDR\_NOMOREHANDLES, 805  
LDR\_NOSPACE, 805  
LDR\_NOTLINEARFILE, 805  
LDR\_OUTOFBOUNDARY, 805  
LDR\_REQPIN, 805  
LDR\_SUCCESS, 805  
LED\_BLUE, 805  
LED\_NONE, 805  
LED\_RED, 805  
LEGO\_ADDR\_EMETER, 805  
LEGO\_ADDR\_TEMP, 805  
LEGO\_ADDR\_US, 805  
LONG\_MAX, 806  
LONG\_MIN, 806  
LOWSPEED\_DONE, 806  
LOWSPEED\_ERROR, 806  
LOWSPEED\_IDLE, 806  
LOWSPEED\_INIT, 807  
LR\_ENTRY\_REMOVED, 808  
LR\_SUCCESS, 808  
LR\_UNKNOWN\_ADDR, 808  
ListFiles, 806  
LoaderExecuteFunction, 806  
LoaderModuleID, 806  
LoaderModuleName, 806  
LoaderOffsetFreeUserFlash, 806  
LoaderOffsetPFunc, 806  
LowSpeedModuleID, 807  
LowSpeedModuleName, 807  
LowSpeedOffsetChannelState, 807  
LowSpeedOffsetErrorType, 807  
LowSpeedOffsetInBufBuf, 807  
LowSpeedOffsetInBufBytesToRx, 807  
LowSpeedOffsetInBufInPtr, 807  
LowSpeedOffsetInBufOutPtr, 808  
LowSpeedOffsetMode, 808  
LowSpeedOffsetNoRestartOnRead, 808  
LowSpeedOffsetOutBufBuf, 808  
LowSpeedOffsetOutBufBytesToRx, 808  
LowSpeedOffsetOutBufInPtr, 808  
LowSpeedOffsetOutBufOutPtr, 808  
LowSpeedOffsetSpeed, 808  
LowSpeedOffsetState, 808  
MAILBOX1, 809  
MAILBOX10, 809  
MAILBOX2, 809  
MAILBOX3, 809  
MAILBOX4, 809  
MAILBOX5, 809  
MAILBOX6, 809  
MAILBOX7, 810  
MAILBOX8, 810  
MAILBOX9, 810  
MAX\_BT\_MSG\_SIZE, 810  
MENUICON\_CENTER, 810  
MENUICON\_LEFT, 810  
MENUICON\_RIGHT, 810  
MENUICONS, 810  
MENUTEXT, 811  
MI\_ADDR\_XG1300L, 811

MIN\_1, 811  
MS\_1, 811  
MS\_10, 811  
MS\_100, 811  
MS\_150, 811  
MS\_2, 811  
MS\_20, 811  
MS\_200, 811  
MS\_250, 811  
MS\_3, 811  
MS\_30, 812  
MS\_300, 812  
MS\_350, 812  
MS\_4, 812  
MS\_40, 812  
MS\_400, 812  
MS\_450, 812  
MS\_5, 812  
MS\_50, 812  
MS\_500, 812  
MS\_6, 812  
MS\_60, 812  
MS\_600, 812  
MS\_7, 812  
MS\_70, 813  
MS\_700, 813  
MS\_8, 813  
MS\_80, 813  
MS\_800, 813  
MS\_9, 813  
MS\_90, 813  
MS\_900, 813  
MS\_ADDR\_ACCLNX, 813  
MS\_ADDR\_CMPSNX, 813  
MS\_ADDR\_DISTNX, 813  
MS\_ADDR\_IVSENS, 813  
MS\_ADDR\_LINELDR, 813  
MS\_ADDR\_MTRMUX, 813  
MS\_ADDR\_NRLINK, 814  
MS\_ADDR\_NXTCAM, 814  
MS\_ADDR\_NXTHID, 814  
MS\_ADDR\_NXTMMX, 814  
MS\_ADDR\_NXTSERVO, 814  
MS\_ADDR\_PFMATE, 814  
MS\_ADDR\_PSPNX, 814  
MS\_ADDR\_RTCLOCK, 814  
MS\_ADDR\_RXMUX, 814  
MS\_CMD\_ADPA\_OFF, 814  
MS\_CMD\_ADPA\_ON, 814  
MS\_CMD\_ENERGIZED, 815  
MaxAccelerationField, 810  
MaxSpeedField, 810  
MemoryManager, 810  
MessageRead, 811  
MessageWrite, 811  
NA, 815  
NO\_ERR, 815  
NO\_OF\_BTNS, 815  
NOTE\_EIGHT, 815  
NOTE\_HALF, 815  
NOTE\_QUARTER, 815  
NOTE\_SIXTEEN, 815  
NOTE\_WHOLE, 815  
NRLINK\_CMD\_2400, 815  
NRLINK\_CMD\_4800, 815  
NRLINK\_CMD\_FLUSH, 815  
NRLINK\_REG\_BYTES, 816  
NRLINK\_REG\_DATA, 816  
NULL, 816  
NXTHID\_CMD\_ASCII, 816  
NXTHID\_MOD\_NONE, 817  
NXTHID\_REG\_CMD, 817  
NXTHID\_REG\_DATA, 817  
NXTLL\_CMD\_BLACK, 817  
NXTLL\_CMD\_INVERT, 818  
NXTLL\_CMD\_RESET, 818  
NXTLL\_CMD\_USA, 818  
NXTLL\_CMD\_WHITE, 818  
NXTLL\_REG\_CMD, 818  
NXTLL\_REG\_RESULT, 819  
NXTPM\_CMD\_RESET, 819  
NXTPM\_REG\_CMD, 820  
NXTPM\_REG\_GAIN, 820  
NXTPM\_REG\_POWER, 820  
NXTPM\_REG\_TIME, 820  
NXTSE\_ZONE\_FRONT, 820  
NXTSE\_ZONE\_LEFT, 821  
NXTSE\_ZONE\_NONE, 821  
NXTSE\_ZONE\_RIGHT, 821  
NXTSERVO\_POS\_MAX, 822  
NXTSERVO\_POS\_MIN, 822  
NXTSERVO\_REG\_CMD, 822  
NXTSERVO\_SERVO\_1, 824  
NXTSERVO\_SERVO\_2, 824  
NXTSERVO\_SERVO\_3, 824  
NXTSERVO\_SERVO\_4, 824  
NXTSERVO\_SERVO\_5, 824  
NXTSERVO\_SERVO\_6, 824  
NXTSERVO\_SERVO\_7, 824  
NXTSERVO\_SERVO\_8, 824  
NXTTP\_CMD\_USA, 824  
NXTTP\_REG\_CMD, 825  
NormalizedValueField, 815  
OPARR\_MAX, 825  
OPARR\_MEAN, 825  
OPARR\_MIN, 825  
OPARR\_SORT, 825  
OPARR\_STD, 825

OPARR\_SUM, 825  
OPARR\_SUMSQR, 825  
OPARR\_TOLOWER, 825  
OPARR\_TOUPPER, 825  
OUT\_A, 825  
OUT\_AB, 825  
OUT\_ABC, 825  
OUT\_AC, 825  
OUT\_B, 826  
OUT\_BC, 826  
OUT\_C, 826  
OUT\_MODE\_BRAKE, 826  
OUT\_MODE\_COAST, 826  
OUT\_MODE\_MOTORON, 826  
OUT\_REGMODE\_IDLE, 826  
OUT\_REGMODE\_POS, 826  
OUT\_REGMODE\_SYNC, 826  
OutputModeField, 827  
OutputModuleID, 827  
OutputModuleName, 827  
OutputOffsetActualSpeed, 827  
OutputOffsetBlockTachoCount, 827  
OutputOffsetFlags, 827  
OutputOffsetMaxAccel, 827  
OutputOffsetMaxSpeed, 828  
OutputOffsetMode, 828  
OutputOffsetMotorRPM, 828  
OutputOffsetOptions, 828  
OutputOffsetOverloaded, 828  
OutputOffsetRegDParameter, 828  
OutputOffsetRegIParameter, 828  
OutputOffsetRegMode, 828  
OutputOffsetRegPParameter, 828  
OutputOffsetRegulationOptions, 828  
OutputOffsetRegulationTime, 828  
OutputOffsetRotationCount, 828  
OutputOffsetRunState, 828  
OutputOffsetSpeed, 828  
OutputOffsetSyncTurnParameter, 829  
OutputOffsetTachoCount, 829  
OutputOffsetTachoLimit, 829  
OutputOptionsField, 829  
OverloadField, 829  
PF\_CHANNEL\_1, 829  
PF\_CHANNEL\_2, 829  
PF\_CHANNEL\_3, 829  
PF\_CHANNEL\_4, 829  
PF\_CMD\_BRAKE, 829  
PF\_CMD\_FLOAT, 830  
PF\_CMD\_FWD, 830  
PF\_CMD\_REV, 830  
PF\_CMD\_STOP, 830  
PF\_CST\_FULL\_FWD, 830  
PF\_CST\_FULL\_REV, 830  
PF\_CST\_SET1\_SET2, 830  
PF\_FUNC\_CLEAR, 830  
PF\_FUNC\_NOCHANGE, 831  
PF\_FUNC\_SET, 831  
PF\_FUNC\_TOGGLE, 831  
PF\_MODE\_TRAIN, 831  
PF\_OUT\_A, 831  
PF\_OUT\_B, 831  
PF\_PIN\_C1, 831  
PF\_PIN\_C2, 831  
PF\_PWM\_BRAKE, 832  
PF\_PWM\_FLOAT, 832  
PF\_PWM\_FWD1, 832  
PF\_PWM\_FWD2, 832  
PF\_PWM\_FWD3, 832  
PF\_PWM\_FWD4, 832  
PF\_PWM\_FWD5, 832  
PF\_PWM\_FWD6, 832  
PF\_PWM\_FWD7, 832  
PF\_PWM\_REV1, 832  
PF\_PWM\_REV2, 832  
PF\_PWM\_REV3, 832  
PF\_PWM\_REV4, 832  
PF\_PWM\_REV5, 832  
PF\_PWM\_REV6, 833  
PF\_PWM\_REV7, 833  
PFMATE\_CHANNEL\_1, 833  
PFMATE\_CHANNEL\_2, 833  
PFMATE\_CHANNEL\_3, 833  
PFMATE\_CHANNEL\_4, 833  
PFMATE\_CMD\_GO, 833  
PFMATE\_CMD\_RAW, 833  
PFMATE MOTORS\_A, 833  
PFMATE MOTORS\_B, 833  
PFMATE\_REG\_CMD, 834  
PI, 834  
PID\_0, 834  
PID\_1, 834  
PID\_2, 834  
PID\_3, 834  
PID\_4, 834  
PID\_5, 834  
PID\_6, 834  
PID\_7, 834  
POOL\_MAX\_SIZE, 834  
PROG\_ABORT, 835  
PROG\_ERROR, 835  
PROG\_IDLE, 835  
PROG\_OK, 835  
PROG\_RESET, 835  
PROG\_RUNNING, 835  
PSP\_BTNSET1\_DOWN, 835  
PSP\_BTNSET1\_L3, 835  
PSP\_BTNSET1\_LEFT, 835

PSP\_BTNSET1\_R3, 835  
PSP\_BTNSET1\_RIGHT, 835  
PSP\_BTNSET1\_START, 836  
PSP\_BTNSET1\_UP, 836  
PSP\_BTNSET2\_CROSS, 836  
PSP\_BTNSET2\_L1, 836  
PSP\_BTNSET2\_L2, 836  
PSP\_BTNSET2\_R1, 836  
PSP\_BTNSET2\_R2, 836  
PSP\_CMD\_ANALOG, 836  
PSP\_CMD\_DIGITAL, 836  
PSP\_REG\_BTNSET1, 836  
PSP\_REG\_BTNSET2, 836  
PSP\_REG\_XLEFT, 837  
PSP\_REG\_XRIGHT, 837  
PSP\_REG\_YLEFT, 837  
PSP\_REG\_YRIGHT, 837  
PowerField, 835  
RAND\_MAX, 837  
RC\_PROP\_BTTONOFF, 837  
RCX\_AbsVarOp, 837  
RCX\_AndVarOp, 838  
RCX\_AutoOffOp, 838  
RCX\_BatteryLevelOp, 838  
RCX\_BatteryLevelSrc, 838  
RCX\_BootModeOp, 838  
RCX\_CalibrateEventOp, 838  
RCX\_ClearAllEventsOp, 838  
RCX\_ClearCounterOp, 838  
RCX\_ClearMsgOp, 838  
RCX\_ClearSensorOp, 838  
RCX\_ClearSoundOp, 838  
RCX\_ClearTimerOp, 838  
RCX\_ClickCounterSrc, 838  
RCX\_ConstantSrc, 838  
RCX\_CounterSrc, 839  
RCX\_DatalogOp, 839  
RCX\_DatalogRawDirectSrc, 839  
RCX\_DatalogRawIndirectSrc, 839  
RCX\_DatalogSrcDirectSrc, 839  
RCX\_DatalogSrcIndirectSrc, 839  
RCX\_DatalogValueDirectSrc, 839  
RCX\_DatalogValueIndirectSrc, 839  
RCX\_DecCounterOp, 839  
RCX\_DeleteSubOp, 839  
RCX\_DeleteSubsOp, 839  
RCX\_DeleteTaskOp, 839  
RCX\_DeleteTasksOp, 839  
RCX\_DirectEventOp, 839  
RCX\_DisplayOp, 840  
RCX\_DivVarOp, 840  
RCX\_DurationSrc, 840  
RCX\_EventStateSrc, 840  
RCX\_FirmwareVersionSrc, 840  
RCX\_GOutputDirOp, 840  
RCX\_GOutputModeOp, 840  
RCX\_GOutputPowerOp, 840  
RCX\_GlobalMotorStatusSrc, 840  
RCX\_HysteresisSrc, 840  
RCX\_IRModeOp, 841  
RCX\_IncCounterOp, 840  
RCX\_IndirectVarSrc, 840  
RCX\_InputBooleanSrc, 840  
RCX\_InputModeOp, 840  
RCX\_InputModeSrc, 841  
RCX\_InputRawSrc, 841  
RCX\_InputTypeOp, 841  
RCX\_InputTypeSrc, 841  
RCX\_InputValueSrc, 841  
RCX\_LSBlinkTimeOp, 841  
RCX\_LSCalibrateOp, 841  
RCX\_LSHysteresisOp, 841  
RCX\_LSLowerThreshOp, 841  
RCX\_LSSupperThreshOp, 841  
RCX\_LightOp, 841  
RCX\_LowerThresholdSrc, 841  
RCX\_MessageOp, 841  
RCX\_MessageSrc, 842  
RCX\_MulVarOp, 842  
RCX\_MuteSoundOp, 842  
RCX\_OUT\_A, 842  
RCX\_OUT\_AB, 842  
RCX\_OUT\_ABC, 842  
RCX\_OUT\_AC, 842  
RCX\_OUT\_B, 842  
RCX\_OUT\_BC, 842  
RCX\_OUT\_C, 842  
RCX\_OUT\_FLOAT, 842  
RCX\_OUT\_FULL, 842  
RCX\_OUT\_FWD, 843  
RCX\_OUT\_HALF, 843  
RCX\_OUT\_LOW, 843  
RCX\_OUT\_OFF, 843  
RCX\_OUT\_ON, 843  
RCX\_OUT\_REV, 843  
RCX\_OUT\_TOGGLE, 843  
RCX\_OnOffFloatOp, 842  
RCX\_OrVarOp, 842  
RCX\_OutputDirOp, 843  
RCX\_OutputPowerOp, 843  
RCX\_OutputStatusSrc, 843  
RCX\_PBTurnOffOp, 843  
RCX\_PingOp, 843  
RCX\_PlaySoundOp, 843  
RCX\_PlayToneOp, 843  
RCX\_PlayToneVarOp, 844  
RCX\_PollMemoryOp, 844  
RCX\_PollOp, 844

RCX\_ProgramSlotSrc, 844  
RCX\_RandomSrc, 844  
RCX\_RemoteKeysReleased, 844  
RCX\_RemoteOp, 844  
RCX\_RemoteOutABackward, 844  
RCX\_RemoteOutAForward, 844  
RCX\_RemoteOutBBackward, 844  
RCX\_RemoteOutBForward, 844  
RCX\_RemoteOutCBackward, 844  
RCX\_RemoteOutCForward, 844  
RCX\_RemotePBMessagel, 844  
RCX\_RemotePBMessage2, 845  
RCX\_RemotePBMessagel, 845  
RCX\_RemotePlayASound, 845  
RCX\_RemoteSelProgram1, 845  
RCX\_RemoteSelProgram2, 845  
RCX\_RemoteSelProgram3, 845  
RCX\_RemoteSelProgram4, 845  
RCX\_RemoteSelProgram5, 845  
RCX\_RemoteStopOutOff, 845  
RCX\_ScoutCounterLimitSrc, 845  
RCX\_ScoutEventFBSrc, 845  
RCX\_ScoutLightParamsSrc, 845  
RCX\_ScoutOp, 845  
RCX\_ScoutRulesOp, 845  
RCX\_ScoutRulesSrc, 846  
RCX\_ScoutTimerLimitSrc, 846  
RCX\_SelectProgramOp, 846  
RCX\_SendUARTDataOp, 846  
RCX\_SetCounterOp, 846  
RCX\_SetDatalogOp, 846  
RCX\_SetEventOp, 846  
RCX\_SetFeedbackOp, 846  
RCX\_SetPriorityOp, 846  
RCX\_SetSourceValueOp, 846  
RCX\_SetTimerLimitOp, 846  
RCX\_SetVarOp, 846  
RCX\_SetWatchOp, 846  
RCX\_SgnVarOp, 846  
RCX\_SoundOp, 847  
RCX\_StartTaskOp, 847  
RCX\_StopAllTasksOp, 847  
RCX\_StopTaskOp, 847  
RCX\_SubVarOp, 847  
RCX\_SumVarOp, 847  
RCX\_TaskEventsSrc, 847  
RCX\_TenMSTimerSrc, 847  
RCX\_TimerSrc, 847  
RCX\_UARTSetupSrc, 847  
RCX\_UnlockFirmOp, 847  
RCX\_UnlockOp, 847  
RCX\_UnmuteSoundOp, 847  
RCX\_UploadDatalogOp, 847  
RCX\_UpperThresholdSrc, 848  
RCX\_VLLOp, 848  
RCX\_VariableSrc, 848  
RCX\_ViewSourceValOp, 848  
RCX\_WatchSrc, 848  
RESET\_ALL, 849  
RESET\_COUNT, 849  
RESET\_NONE, 849  
RFID\_MODE\_SINGLE, 849  
RFID\_MODE\_STOP, 849  
RICArg, 849  
RICImgPoint, 850  
RICImgRect, 850  
RICMapArg, 850  
RICMapElement, 850  
RICMapFunction, 850  
RICOpcircle, 850  
RICOpcopybits, 851  
RICOpDescription, 851  
RICOpEllipse, 851  
RICOpLine, 851  
RICOpNumBox, 851  
RICOpPixel, 852  
RICOpPolygon, 852  
RICOpRect, 852  
RICOpSprite, 852  
RICOpVarMap, 853  
RICPolygonPoints, 853  
RICSpriteData, 853  
ROTATE\_QUEUE, 853  
RandomEx, 837  
RandomNumber, 837  
RawValueField, 837  
ReadButton, 848  
ReadLastResponse, 848  
ReadSemData, 848  
RegDValueField, 848  
RegIValueField, 848  
RegModeField, 848  
RegPValueField, 849  
RotationCountField, 853  
RunStateField, 853  
SAMPLERATE\_MAX, 854  
SAMPLERATE\_MIN, 854  
SCHAR\_MAX, 854  
SCHAR\_MIN, 854  
SCOUT\_FXR\_ALARM, 854  
SCOUT\_FXR\_BUG, 854  
SCOUT\_FXR\_NONE, 854  
SCOUT\_FXR\_RANDOM, 854  
SCOUT\_LIGHT\_OFF, 854  
SCOUT\_LIGHT\_ON, 854  
SCOUT\_LR\_AVOID, 854  
SCOUT\_LR\_IGNORE, 854  
SCOUT\_MODE\_POWER, 855

SCOUT\_MR\_FORWARD, 855  
SCOUT\_MR\_LOOP\_A, 855  
SCOUT\_MR\_LOOP\_B, 855  
SCOUT\_MR\_ZIGZAG, 855  
SCOUT SNDSET\_BUG, 856  
SCOUT\_SOUND\_NONE, 857  
SCOUT\_TGS\_LONG, 858  
SCOUT\_TGS\_MEDIUM, 858  
SCOUT\_TGS\_SHORT, 858  
SCOUT\_TR\_AVOID, 858  
SCOUT\_TR\_IGNORE, 858  
SCOUT\_TR\_REVERSE, 858  
SCREEN\_BACKGROUND, 858  
SCREEN\_LARGE, 858  
SCREEN\_SMALL, 859  
SCREENS, 859  
SEC\_1, 859  
SEC\_10, 859  
SEC\_15, 859  
SEC\_2, 859  
SEC\_20, 859  
SEC\_3, 859  
SEC\_30, 859  
SEC\_4, 859  
SEC\_5, 859  
SEC\_6, 859  
SEC\_7, 859  
SEC\_8, 859  
SEC\_9, 860  
SHRT\_MAX, 860  
SHRT\_MIN, 860  
SIZE\_OF\_BDADDR, 860  
SIZE\_OF\_BT\_NAME, 860  
SIZE\_OF\_BTBUF, 860  
SIZE\_OF\_HSBUF, 860  
SIZE\_OF\_USBBUF, 861  
SIZE\_OF\_USBDATA, 861  
SOUND\_CLICK, 861  
SOUND\_DOWN, 861  
SOUND\_FAST\_UP, 861  
SOUND\_FLAGS\_IDLE, 861  
SOUND\_LOW\_BEEP, 861  
SOUND\_MODE\_LOOP, 861  
SOUND\_MODE\_ONCE, 861  
SOUND\_MODE\_TONE, 861  
SOUND\_STATE\_FILE, 861  
SOUND\_STATE\_IDLE, 862  
SOUND\_STATE\_STOP, 862  
SOUND\_STATE\_TONE, 862  
SOUND\_UP, 862  
SPECIALS, 863  
STATUSICON\_USB, 863  
STATUSICON\_VM, 863  
STATUSICONS, 863  
STATUSTEXT, 863  
STEPICON\_1, 863  
STEPICON\_2, 864  
STEPICON\_3, 864  
STEPICON\_4, 864  
STEPICON\_5, 864  
STEPICONS, 864  
STEPLINE, 864  
STOP\_REQ, 864  
STROBE\_READ, 864  
STROBE\_S0, 864  
STROBE\_S1, 864  
STROBE\_S2, 864  
STROBE\_S3, 864  
STROBE\_WRITE, 864  
ScaledValueField, 854  
SetScreenMode, 860  
SetSleepTimeoutVal, 860  
SoundGetState, 862  
SoundModuleID, 862  
SoundModuleName, 862  
SoundOffsetDuration, 862  
SoundOffsetFlags, 862  
SoundOffsetFreq, 862  
SoundOffsetMode, 862  
SoundOffsetSampleRate, 862  
SoundOffsetSoundFilename, 862  
SoundOffsetState, 862  
SoundOffsetVolume, 863  
SoundPlayFile, 863  
SoundPlayTone, 863  
SoundSetState, 863  
TEMP\_FQ\_1, 865  
TEMP\_FQ\_2, 865  
TEMP\_FQ\_4, 865  
TEMP\_FQ\_6, 865  
TEMP\_OS\_ONESHOT, 865  
TEMP\_POL\_HIGH, 865  
TEMP\_POL\_LOW, 865  
TEMP\_REG\_CONFIG, 865  
TEMP\_REG\_TEMP, 865  
TEMP\_REG\_THIGH, 865  
TEMP\_REG\_TLOW, 865  
TEMP\_RES\_10BIT, 866  
TEMP\_RES\_11BIT, 866  
TEMP\_RES\_12BIT, 866  
TEMP\_RES\_9BIT, 866  
TEMP\_SD\_SHUTDOWN, 866  
TEXTLINE\_1, 866  
TEXTLINE\_2, 866  
TEXTLINE\_3, 866  
TEXTLINE\_4, 866  
TEXTLINE\_5, 866  
TEXTLINE\_6, 866

TEXTLINE\_7, 867  
TEXTLINE\_8, 867  
TEXTLINES, 867  
TIMES\_UP, 867  
TONE\_A3, 867  
TONE\_A4, 867  
TONE\_A5, 867  
TONE\_A6, 867  
TONE\_A7, 867  
TONE\_AS3, 867  
TONE\_AS4, 867  
TONE\_AS5, 867  
TONE\_AS6, 867  
TONE\_AS7, 867  
TONE\_B3, 868  
TONE\_B4, 868  
TONE\_B5, 868  
TONE\_B6, 868  
TONE\_B7, 868  
TONE\_C3, 868  
TONE\_C4, 868  
TONE\_C5, 868  
TONE\_C6, 868  
TONE\_C7, 868  
TONE\_CS3, 868  
TONE\_CS4, 868  
TONE\_CS5, 868  
TONE\_CS6, 868  
TONE\_CS7, 869  
TONE\_D3, 869  
TONE\_D4, 869  
TONE\_D5, 869  
TONE\_D6, 869  
TONE\_D7, 869  
TONE\_DS3, 869  
TONE\_DS4, 869  
TONE\_DS5, 869  
TONE\_DS6, 869  
TONE\_DS7, 869  
TONE\_E3, 869  
TONE\_E4, 869  
TONE\_E5, 869  
TONE\_E6, 870  
TONE\_E7, 870  
TONE\_F3, 870  
TONE\_F4, 870  
TONE\_F5, 870  
TONE\_F6, 870  
TONE\_F7, 870  
TONE\_FS3, 870  
TONE\_FS4, 870  
TONE\_FS5, 870  
TONE\_FS6, 870  
TONE\_FS7, 870  
TONE\_G3, 870  
TONE\_G4, 870  
TONE\_G5, 871  
TONE\_G6, 871  
TONE\_G7, 871  
TONE\_GS3, 871  
TONE\_GS4, 871  
TONE\_GS5, 871  
TONE\_GS6, 871  
TONE\_GS7, 871  
TOPLINE, 871  
TRAIN\_CHANNEL\_1, 871  
TRAIN\_CHANNEL\_2, 871  
TRAIN\_CHANNEL\_3, 871  
TRAIN\_FUNC\_STOP, 872  
TRUE, 872  
TachoCountField, 864  
TachoLimitField, 865  
TurnRatioField, 872  
TypeField, 872  
UCHAR\_MAX, 872  
UF\_UPDATE\_MODE, 872  
UF\_UPDATE\_SPEED, 873  
UI\_BT\_STATE\_OFF, 873  
UI\_BUTTON\_ENTER, 873  
UI\_BUTTON\_EXIT, 873  
UI\_BUTTON\_LEFT, 873  
UI\_BUTTON\_NONE, 873  
UI\_BUTTON\_RIGHT, 873  
UI\_FLAGS\_BUSY, 874  
UI\_FLAGS\_UPDATE, 874  
UI\_VM\_IDLE, 875  
UI\_VM\_RESET1, 875  
UI\_VM\_RESET2, 875  
UI\_VM\_RUN\_FREE, 876  
UI\_VM\_RUN\_PAUSE, 876  
UIModuleID, 876  
UIModuleName, 876  
UINT\_MAX, 876  
UIOffsetAbortFlag, 876  
UIOffsetBatteryState, 876  
UIOffsetBatteryVoltage, 876  
UIOffsetBluetoothState, 876  
UIOffsetButton, 876  
UIOffsetError, 876  
UIOffsetFlags, 876  
UIOffsetForceOff, 876  
UIOffsetLMSfilename, 877  
UIOffsetOBPPointer, 877  
UIOffsetPMenu, 877  
UIOffsetRechargeable, 877  
UIOffsetRunState, 877  
UIOffsetSleepTimeout, 877  
UIOffsetSleepTimer, 877

UIOffsetState, 877  
UIOffsetUsbState, 877  
UIOffsetVolume, 877  
ULONG\_MAX, 877  
US\_CMD\_OFF, 878  
US\_CMD\_WARMRESET, 878  
USB\_CMD\_READY, 878  
USHRT\_MAX, 879  
UpdateCalibCacheInfo, 877  
UpdateFlagsField, 877  
VT\_A1\_FLOAT, 879  
VT\_A1\_SBYTE, 879  
VT\_A1\_SLONG, 879  
VT\_A1\_STRUCT, 879  
VT\_A1\_SWORD, 879  
VT\_A1\_UBYTE, 879  
VT\_A1 ULONG, 879  
VT\_A1\_UWORD, 879  
VT\_A2\_FLOAT, 879  
VT\_A2\_SBYTE, 879  
VT\_A2\_SLONG, 879  
VT\_A2\_STRUCT, 879  
VT\_A2\_SWORD, 879  
VT\_A2\_UBYTE, 880  
VT\_A2 ULONG, 880  
VT\_A2\_UWORD, 880  
VT\_ARRAY\_MASK, 880  
VT\_FLOAT, 880  
VT\_MUTEX, 880  
VT\_SBYTE, 880  
VT\_SLONG, 880  
VT\_STRUCT, 880  
VT\_SWORD, 880  
VT\_UBYTE, 880  
VT ULONG, 880  
VT\_UWORD, 880  
WriteSemData, 880  
XG1300L\_REG\_2G, 881  
XG1300L\_REG\_4G, 881  
XG1300L\_REG\_8G, 881  
XG1300L\_REG\_ANGLE, 881  
XG1300L\_REG\_RESET, 881  
XG1300L\_REG\_TURNRATE, 881  
XG1300L\_REG\_XAXIS, 881  
XG1300L\_REG\_YAXIS, 881  
XG1300L\_REG\_ZAXIS, 881  
XG1300L\_SCALE\_2G, 881  
XG1300L\_SCALE\_4G, 881  
XG1300L\_SCALE\_8G, 881  
  
NEQ  
Comparison Constants, 61  
  
NO\_ERR  
Command module constants, 70  
NBCCommon.h, 815  
  
NO\_OF\_BTNS  
Button name constants, 417  
NBCCommon.h, 815  
  
NOTE\_EIGHT  
NBCCommon.h, 815  
Time constants, 377  
  
NOTE\_HALF  
NBCCommon.h, 815  
Time constants, 377  
  
NOTE\_QUARTER  
NBCCommon.h, 815  
Time constants, 377  
  
NOTE\_SIXTEEN  
NBCCommon.h, 815  
Time constants, 377  
  
NOTE\_WHOLE  
NBCCommon.h, 815  
Time constants, 377  
  
NRLINK\_CMD\_2400  
MindSensors nRLink constants, 613  
NBCCommon.h, 815  
  
NRLINK\_CMD\_4800  
MindSensors nRLink constants, 613  
NBCCommon.h, 815  
  
NRLINK\_CMD\_FLUSH  
MindSensors nRLink constants, 613  
NBCCommon.h, 815  
  
NRLINK\_CMD\_SET\_PF  
MindSensors nRLink constants, 613  
NBCCommon.h, 816  
  
NRLINK\_CMD\_TX\_RAW  
MindSensors nRLink constants, 614  
NBCCommon.h, 816  
  
NRLINK\_REG\_BYTES  
MindSensors nRLink constants, 614  
NBCCommon.h, 816  
  
NRLINK\_REG\_DATA  
MindSensors nRLink constants, 614  
NBCCommon.h, 816  
  
NRLINK\_REG\_EEPROM  
MindSensors nRLink constants, 614  
NBCCommon.h, 816  
  
NRLink2400  
MindSensors API Functions, 155  
  
NRLink4800  
MindSensors API Functions, 155  
  
NRLinkFlush  
MindSensors API Functions, 156  
  
NRLinkIRLong  
MindSensors API Functions, 156  
  
NRLinkIRShort  
MindSensors API Functions, 156  
  
NRLinkSetPF  
MindSensors API Functions, 156

NRLinkSetRCX  
    MindSensors API Functions, 157

NRLinkSetTrain  
    MindSensors API Functions, 157

NRLinkTxRaw  
    MindSensors API Functions, 157

NULL  
    Loader module constants, 393  
    NBCCCommon.h, 816

NXT firmware module IDs, 201  
    ButtonModuleID, 201  
    CommModuleID, 201  
    CommandModuleID, 201  
    DisplayModuleID, 201  
    IOCtrlModuleID, 201  
    InputModuleID, 201  
    LoaderModuleID, 201  
    LowSpeedModuleID, 201  
    OutputModuleID, 202  
    SoundModuleID, 202  
    UIModuleID, 202

NXT firmware module names, 199  
    ButtonModuleName, 199  
    CommModuleName, 199  
    CommandModuleName, 199  
    DisplayModuleName, 199  
    IOCtrlModuleName, 199  
    InputModuleName, 199  
    LoaderModuleName, 199  
    LowSpeedModuleName, 199  
    OutputModuleName, 200  
    SoundModuleName, 200  
    UIModuleName, 200

NXT Firmware Modules, 62

NXTHID\_CMD\_ASCII  
    MindSensors NXTHID commands, 636  
    NBCCCommon.h, 816

NXTHID\_CMD\_DIRECT  
    NBCCCommon.h, 816

NXTHID\_MOD\_NONE  
    MindSensors NXTHID modifier keys, 634  
    NBCCCommon.h, 817

NXTHID\_REG\_CMD  
    MindSensors NXTHID registers, 633  
    NBCCCommon.h, 817

NXTHID\_REG\_DATA  
    MindSensors NXTHID registers, 633  
    NBCCCommon.h, 817

NXTHIDAsciiMode  
    MindSensors API Functions, 157

NXTHIDDirectMode  
    MindSensors API Functions, 158

NXTHIDLoadCharacter  
    MindSensors API Functions, 158

NXTHIDTransmit  
    MindSensors API Functions, 158

NXTLL\_CMD\_BLACK  
    MindSensors NXTLineLeader commands, 645  
    NBCCCommon.h, 817

NXTLL\_CMD\_EUROPEAN  
    NBCCCommon.h, 817

NXTLL\_CMD\_INVERT  
    MindSensors NXTLineLeader commands, 645  
    NBCCCommon.h, 818

NXTLL\_CMD\_POWERUP  
    NBCCCommon.h, 818

NXTLL\_CMD\_RESET  
    MindSensors NXTLineLeader commands, 645  
    NBCCCommon.h, 818

NXTLL\_CMD\_SNAPSHOT  
    NBCCCommon.h, 818

NXTLL\_CMD\_USA  
    MindSensors NXTLineLeader commands, 645  
    NBCCCommon.h, 818

NXTLL\_CMD\_WHITE  
    MindSensors NXTLineLeader commands, 646  
    NBCCCommon.h, 818

NXTLL\_REG\_AVERAGE  
    NBCCCommon.h, 818

NXTLL\_REG\_CMD  
    MindSensors NXTLineLeader registers, 643  
    NBCCCommon.h, 818

NXTLL\_REG\_RESULT  
    MindSensors NXTLineLeader registers, 644  
    NBCCCommon.h, 819

NXTLL\_REG\_SETPOINT  
    NBCCCommon.h, 819

NXTLL\_REG\_STEERING  
    NBCCCommon.h, 819

NXTLineLeaderCalibrateBlack  
    MindSensors API Functions, 158

NXTLineLeaderCalibrateWhite  
    MindSensors API Functions, 159

NXTLineLeaderInvert  
    MindSensors API Functions, 159

NXTLineLeaderPowerDown  
    MindSensors API Functions, 159

NXTLineLeaderPowerUp  
    MindSensors API Functions, 160

NXTLineLeaderReset  
    MindSensors API Functions, 160

NXTLineLeaderSnapshot  
    MindSensors API Functions, 160

NXTNP\_REG\_BUTTONS  
    NBCCCommon.h, 819

NXTPM\_CMD\_RESET  
    MindSensors NXTPowerMeter commands, 640  
    NBCCCommon.h, 819

NXTPM\_REG\_CAPACITY  
NBCCCommon.h, 819

NXTPM\_REG\_CMD  
MindSensors NXTPowerMeter registers, 638  
NBCCCommon.h, 820

NXTPM\_REG\_CURRENT  
NBCCCommon.h, 820

NXTPM\_REG\_GAIN  
MindSensors NXTPowerMeter registers, 638  
NBCCCommon.h, 820

NXTPM\_REG\_POWER  
MindSensors NXTPowerMeter registers, 639  
NBCCCommon.h, 820

NXTPM\_REG\_TIME  
MindSensors NXTPowerMeter registers, 639  
NBCCCommon.h, 820

NXTPM\_REG\_USERGAIN  
NBCCCommon.h, 820

NXTPM\_REG\_VOLTAGE  
NBCCCommon.h, 820

NXTPowerMeterResetCounters  
MindSensors API Functions, 160

NXTSE\_ZONE\_FRONT  
MindSensors NXTSumoEyes constants, 641  
NBCCCommon.h, 820

NXTSE\_ZONE\_LEFT  
MindSensors NXTSumoEyes constants, 641  
NBCCCommon.h, 821

NXTSE\_ZONE\_NONE  
MindSensors NXTSumoEyes constants, 641  
NBCCCommon.h, 821

NXTSE\_ZONE\_RIGHT  
MindSensors NXTSumoEyes constants, 641  
NBCCCommon.h, 821

NXTSERVO\_CMD\_EDIT1  
MindSensors NXTServo commands, 630  
NBCCCommon.h, 821

NXTSERVO\_CMD\_EDIT2  
MindSensors NXTServo commands, 630  
NBCCCommon.h, 821

NXTSERVO\_CMD\_GOTO  
MindSensors NXTServo commands, 630  
NBCCCommon.h, 821

NXTSERVO\_CMD\_HALT  
MindSensors NXTServo commands, 630  
NBCCCommon.h, 821

NXTSERVO\_CMD\_INIT  
MindSensors NXTServo commands, 630  
NBCCCommon.h, 821

NXTSERVO\_CMD\_PAUSE  
NBCCCommon.h, 821

NXTSERVO\_CMD\_RESET  
NBCCCommon.h, 821

NXTSERVO\_POS\_MAX  
MindSensors NXTServo position constants, 627  
NBCCCommon.h, 822

NXTSERVO\_POS\_MIN  
MindSensors NXTServo position constants, 627  
NBCCCommon.h, 822

NXTSERVO\_QPOS\_MAX  
NBCCCommon.h, 822

NXTSERVO\_QPOS\_MIN  
NBCCCommon.h, 822

NXTSERVO\_REG\_CMD  
MindSensors NXTServo registers, 625  
NBCCCommon.h, 822

NXTSERVO\_SERVO\_1  
MindSensors NXTServo servo numbers, 629  
NBCCCommon.h, 824

NXTSERVO\_SERVO\_2  
MindSensors NXTServo servo numbers, 629  
NBCCCommon.h, 824

NXTSERVO\_SERVO\_3  
MindSensors NXTServo servo numbers, 629  
NBCCCommon.h, 824

NXTSERVO\_SERVO\_4  
MindSensors NXTServo servo numbers, 629  
NBCCCommon.h, 824

NXTSERVO\_SERVO\_5  
MindSensors NXTServo servo numbers, 629  
NBCCCommon.h, 824

NXTSERVO\_SERVO\_6  
MindSensors NXTServo servo numbers, 629  
NBCCCommon.h, 824

NXTSERVO\_SERVO\_7  
MindSensors NXTServo servo numbers, 629  
NBCCCommon.h, 824

NXTSERVO\_SERVO\_8  
MindSensors NXTServo servo numbers, 629  
NBCCCommon.h, 824

NXTServoEditMacro  
MindSensors API Functions, 161

NXTServoGotoMacroAddress  
MindSensors API Functions, 161

NXTServoHaltMacro  
MindSensors API Functions, 161

NXTServoInit  
MindSensors API Functions, 161

NXTServoPauseMacro  
MindSensors API Functions, 162

NXTServoQuitEdit  
MindSensors API Functions, 162

NXTServoReset  
MindSensors API Functions, 162

NXTServoResumeMacro  
MindSensors API Functions, 163

NXTTP\_CMD\_USA  
MindSensors NXTTouchPanel commands, 651

NBCCCommon.h, 824  
NXTP\_REG\_CMD  
MindSensors NXTTouchPanel registers, 650  
NBCCCommon.h, 825  
NormalizedValueField  
Input field constants, 436  
NBCCCommon.h, 815  
NumOut  
Display module functions, 259  
NumOutEx  
Display module functions, 259  
  
OPARR\_MAX  
Array operation constants, 363  
NBCCCommon.h, 825  
OPARR\_MEAN  
Array operation constants, 363  
NBCCCommon.h, 825  
OPARR\_MIN  
Array operation constants, 363  
NBCCCommon.h, 825  
OPARR\_SORT  
Array operation constants, 363  
NBCCCommon.h, 825  
OPARR\_STD  
Array operation constants, 363  
NBCCCommon.h, 825  
OPARR\_SUM  
Array operation constants, 363  
NBCCCommon.h, 825  
OPARR\_SUMSQR  
Array operation constants, 363  
NBCCCommon.h, 825  
OPARR\_TOLOWER  
Array operation constants, 363  
NBCCCommon.h, 825  
OPARR\_TOUPPER  
Array operation constants, 363  
NBCCCommon.h, 825  
OUT\_A  
NBCCCommon.h, 825  
Output port constants, 446  
OUT\_AB  
NBCCCommon.h, 825  
Output port constants, 446  
OUT\_ABC  
NBCCCommon.h, 825  
Output port constants, 446  
OUT\_AC  
NBCCCommon.h, 825  
Output port constants, 446  
OUT\_B  
NBCCCommon.h, 826  
Output port constants, 446  
  
OUT\_BC  
NBCCCommon.h, 826  
Output port constants, 446  
OUT\_C  
NBCCCommon.h, 826  
Output port constants, 446  
OUT\_MODE\_BRAKE  
NBCCCommon.h, 826  
Output port mode constants, 452  
OUT\_MODE\_COAST  
NBCCCommon.h, 826  
Output port mode constants, 452  
OUT\_MODE\_MOTORON  
NBCCCommon.h, 826  
Output port mode constants, 452  
OUT\_MODE\_REGMETHOD  
NBCCCommon.h, 826  
Output port mode constants, 452  
OUT\_MODE\_REGULATED  
NBCCCommon.h, 826  
Output port mode constants, 452  
OUT\_REGMODE\_IDLE  
NBCCCommon.h, 826  
Output port regulation mode constants, 456  
OUT\_REGMODE\_POS  
NBCCCommon.h, 826  
Output port regulation mode constants, 456  
OUT\_REGMODE\_SPEED  
NBCCCommon.h, 826  
Output port regulation mode constants, 456  
OUT\_REGMODE\_SYNC  
NBCCCommon.h, 826  
Output port regulation mode constants, 456  
OUT\_RUNSTATE\_HOLD  
NBCCCommon.h, 827  
Output port run state constants, 455  
OUT\_RUNSTATE\_IDLE  
NBCCCommon.h, 827  
Output port run state constants, 455  
OUT\_RUNSTATE\_RAMPUP  
Output port run state constants, 455  
Off  
Output module functions, 215  
OffEx  
Output module functions, 215  
OnFwd  
Output module functions, 215  
OnFwdEx  
Output module functions, 216  
OnFwdExPID  
Output module functions, 216  
OnFwdReg  
Output module functions, 216  
OnFwdRegEx

Output module functions, 217  
OnFwdRegExPID  
    Output module functions, 217  
OnFwdRegPID  
    Output module functions, 217  
OnFwdSync  
    Output module functions, 218  
OnFwdSyncEx  
    Output module functions, 218  
OnFwdSyncExPID  
    Output module functions, 218  
OnFwdSyncPID  
    Output module functions, 218  
OnRev  
    Output module functions, 219  
OnRevEx  
    Output module functions, 219  
OnRevExPID  
    Output module functions, 219  
OnRevReg  
    Output module functions, 219  
OnRevRegEx  
    Output module functions, 220  
OnRevRegExPID  
    Output module functions, 220  
OnRevRegPID  
    Output module functions, 220  
OnRevSync  
    Output module functions, 221  
OnRevSyncEx  
    Output module functions, 221  
OnRevSyncExPID  
    Output module functions, 221  
OnRevSyncPID  
    Output module functions, 222  
OpenFileAppend  
    Loader module functions, 351  
OpenFileRead  
    Loader module functions, 351  
OpenFileReadLinear  
    Loader module functions, 352  
Output field constants, 457  
    ActualSpeedField, 458  
    BlockTachoCountField, 458  
    MaxAccelerationField, 458  
    MaxSpeedField, 458  
    OutputModeField, 458  
    OutputOptionsField, 458  
    OverloadField, 459  
    PowerField, 459  
    RegDValueField, 459  
    RegIValueField, 459  
    RegModeField, 459  
    RegPValueField, 459  
RotationCountField, 460  
RunStateField, 460  
TachoCountField, 460  
TachoLimitField, 460  
TurnRatioField, 460  
UpdateFlagsField, 461  
Output module, 67  
Output module constants, 68  
Output module functions, 212  
    Coast, 214  
    CoastEx, 214  
    Float, 214  
    GetOutPwnFreq, 214  
    GetOutRegulationOptions, 215  
    GetOutRegulationTime, 215  
    Off, 215  
    OffEx, 215  
    OnFwd, 215  
    OnFwdEx, 216  
    OnFwdExPID, 216  
    OnFwdReg, 216  
    OnFwdRegEx, 217  
    OnFwdRegExPID, 217  
    OnFwdRegPID, 217  
    OnFwdSync, 218  
    OnFwdSyncEx, 218  
    OnFwdSyncExPID, 218  
    OnFwdSyncPID, 218  
    OnRev, 219  
    OnRevEx, 219  
    OnRevExPID, 219  
    OnRevReg, 219  
    OnRevRegEx, 220  
    OnRevRegExPID, 220  
    OnRevRegPID, 220  
    OnRevSync, 221  
    OnRevSyncEx, 221  
    OnRevSyncExPID, 221  
    OnRevSyncPID, 222  
    ResetAllTachoCounts, 222  
    ResetBlockTachoCount, 222  
    ResetRotationCount, 222  
    ResetTachoCount, 223  
    RotateMotor, 223  
    RotateMotorEx, 223  
    RotateMotorExPID, 223  
    RotateMotorPID, 224  
    SetOutPwnFreq, 224  
    SetOutRegulationOptions, 224  
    SetOutRegulationTime, 225  
Output module IOMAP offsets, 462  
    OutputOffsetActualSpeed, 462  
    OutputOffsetBlockTachoCount, 462  
    OutputOffsetFlags, 462

OutputOffsetMaxAccel, 462  
OutputOffsetMaxSpeed, 462  
OutputOffsetMode, 463  
OutputOffsetMotorRPM, 463  
OutputOffsetOptions, 463  
OutputOffsetOverloaded, 463  
OutputOffsetRegDParameter, 463  
OutputOffsetRegIParameter, 463  
OutputOffsetRegMode, 463  
OutputOffsetRegPPParameter, 463  
OutputOffsetRegulationOptions, 463  
OutputOffsetRegulationTime, 463  
OutputOffsetRotationCount, 463  
OutputOffsetRunState, 463  
OutputOffsetSpeed, 463  
OutputOffsetSyncTurnParameter, 463  
OutputOffsetTachoCount, 464  
OutputOffsetTachoLimit, 464  
Output port constants, 446  
    OUT\_A, 446  
    OUT\_AB, 446  
    OUT\_ABC, 446  
    OUT\_AC, 446  
    OUT\_B, 446  
    OUT\_BC, 446  
    OUT\_C, 446  
Output port mode constants, 452  
    OUT\_MODE BRAKE, 452  
    OUT\_MODE COAST, 452  
    OUT\_MODE MOTORON, 452  
    OUT\_MODE REGMETHOD, 452  
    OUT\_MODE REGULATED, 452  
Output port option constants, 453  
Output port regulation mode constants, 456  
    OUT\_REGMODE IDLE, 456  
    OUT\_REGMODE POS, 456  
    OUT\_REGMODE SPEED, 456  
    OUT\_REGMODE SYNC, 456  
Output port run state constants, 455  
    OUT\_RUNSTATE HOLD, 455  
    OUT\_RUNSTATE IDLE, 455  
    OUT\_RUNSTATE RAMPUP, 455  
Output port update flag constants, 449  
    UF\_PENDING\_UPDATES, 449  
    UF\_UPDATE\_MODE, 449  
    UF\_UPDATE\_SPEED, 449  
Output regulation option constants, 454  
OutputModeField  
    NBCCommon.h, 827  
    Output field constants, 458  
OutputModuleID  
    NBCCommon.h, 827  
    NXT firmware module IDs, 202  
OutputModuleName  
    NBCCommon.h, 827  
    Output module IOMAP offsets, 462  
OutputOffsetActualSpeed  
    NBCCommon.h, 827  
    Output module IOMAP offsets, 462  
OutputOffsetBlockTachoCount  
    NBCCommon.h, 827  
    Output module IOMAP offsets, 462  
OutputOffsetFlags  
    NBCCommon.h, 827  
    Output module IOMAP offsets, 462  
OutputOffsetMaxAccel  
    NBCCommon.h, 827  
    Output module IOMAP offsets, 462  
OutputOffsetMaxSpeed  
    NBCCommon.h, 828  
    Output module IOMAP offsets, 462  
OutputOffsetMode  
    NBCCommon.h, 828  
    Output module IOMAP offsets, 463  
OutputOffsetMotorRPM  
    NBCCommon.h, 828  
    Output module IOMAP offsets, 463  
OutputOffsetOptions  
    NBCCommon.h, 828  
    Output module IOMAP offsets, 463  
OutputOffsetOverloaded  
    NBCCommon.h, 828  
    Output module IOMAP offsets, 463  
OutputOffsetRegDParameter  
    NBCCommon.h, 828  
    Output module IOMAP offsets, 463  
OutputOffsetRegIParameter  
    NBCCommon.h, 828  
    Output module IOMAP offsets, 463  
OutputOffsetRegMode  
    NBCCommon.h, 828  
    Output module IOMAP offsets, 463  
OutputOffsetRegPPParameter  
    NBCCommon.h, 828  
    Output module IOMAP offsets, 463  
OutputOffsetRegulationOptions  
    NBCCommon.h, 828  
    Output module IOMAP offsets, 463  
OutputOffsetRegulationTime  
    NBCCommon.h, 828  
    Output module IOMAP offsets, 463  
OutputOffsetRotationCount  
    NBCCommon.h, 828  
    Output module IOMAP offsets, 463  
OutputOffsetRunState  
    NBCCommon.h, 828  
    Output module IOMAP offsets, 463  
OutputOffsetSpeed  
    NBCCommon.h, 827  
    Output module IOMAP offsets, 462

NBCCCommon.h, 828  
Output module IOMAP offsets, 463

OutputOffsetSyncTurnParameter  
    NBCCCommon.h, 829  
    Output module IOMAP offsets, 463

OutputOffsetTachoCount  
    NBCCCommon.h, 829  
    Output module IOMAP offsets, 464

OutputOffsetTachoLimit  
    NBCCCommon.h, 829  
    Output module IOMAP offsets, 464

OutputOptionsField  
    NBCCCommon.h, 829  
    Output field constants, 458

OverloadField  
    NBCCCommon.h, 829  
    Output field constants, 459

PF/IR Train function constants, 573  
    TRAIN\_FUNC\_STOP, 573

PF\_CHANNEL\_1  
    NBCCCommon.h, 829  
    Power Function channel constants, 571

PF\_CHANNEL\_2  
    NBCCCommon.h, 829  
    Power Function channel constants, 571

PF\_CHANNEL\_3  
    NBCCCommon.h, 829  
    Power Function channel constants, 571

PF\_CHANNEL\_4  
    NBCCCommon.h, 829  
    Power Function channel constants, 571

PF\_CMD\_BRAKE  
    NBCCCommon.h, 829  
    Power Function command constants, 570

PF\_CMD\_FLOAT  
    NBCCCommon.h, 830  
    Power Function command constants, 570

PF\_CMD\_FWD  
    NBCCCommon.h, 830  
    Power Function command constants, 570

PF\_CMD\_REV  
    NBCCCommon.h, 830  
    Power Function command constants, 570

PF\_CMD\_STOP  
    NBCCCommon.h, 830  
    Power Function command constants, 570

PF\_CST\_CLEAR1\_SET2  
    NBCCCommon.h, 830  
    Power Function CST options constants, 578

PF\_CST\_FULL\_FWD  
    NBCCCommon.h, 830  
    Power Function CST options constants, 578

PF\_CST\_FULL\_REV  
    NBCCCommon.h, 830  
    Power Function CST options constants, 578

NBCCCommon.h, 830  
Power Function CST options constants, 578

PF\_CST\_SET1\_CLEAR2  
    NBCCCommon.h, 830  
    Power Function CST options constants, 578

PF\_CST\_SET1\_SET2  
    NBCCCommon.h, 830  
    Power Function CST options constants, 578

PF\_CST\_TOGGLE\_DIR  
    NBCCCommon.h, 830  
    Power Function CST options constants, 578

PF\_FUNC\_CLEAR  
    NBCCCommon.h, 830  
    Power Function single pin function constants, 577

PF\_FUNC\_NOCHANGE  
    NBCCCommon.h, 831  
    Power Function single pin function constants, 577

PF\_FUNC\_SET  
    NBCCCommon.h, 831  
    Power Function single pin function constants, 577

PF\_FUNC\_TOGGLE  
    NBCCCommon.h, 831  
    Power Function single pin function constants, 577

PF\_MODE\_COMBO\_PWM  
    NBCCCommon.h, 831  
    Power Function mode constants, 572

PF\_MODE\_TRAIN  
    NBCCCommon.h, 831  
    Power Function mode constants, 572

PF\_OUT\_A  
    NBCCCommon.h, 831  
    Power Function output constants, 575

PF\_OUT\_B  
    NBCCCommon.h, 831  
    Power Function output constants, 575

PF\_PIN\_C1  
    NBCCCommon.h, 831  
    Power Function pin constants, 576

PF\_PIN\_C2  
    NBCCCommon.h, 831  
    Power Function pin constants, 576

PF\_PWM\_BRAKE  
    NBCCCommon.h, 832  
    Power Function PWM option constants, 580

PF\_PWM\_FLOAT  
    NBCCCommon.h, 832  
    Power Function PWM option constants, 580

PF\_PWM\_FWD1  
    NBCCCommon.h, 832  
    Power Function PWM option constants, 580

PF\_PWM\_FWD2  
    NBCCCommon.h, 832  
    Power Function PWM option constants, 580

PF\_PWM\_FWD3  
    NBCCCommon.h, 832  
    Power Function PWM option constants, 580

NBCCCommon.h, 832  
Power Function PWM option constants, 580

PF\_PWM\_FWD4  
NBCCCommon.h, 832  
Power Function PWM option constants, 580

PF\_PWM\_FWD5  
NBCCCommon.h, 832  
Power Function PWM option constants, 580

PF\_PWM\_FWD6  
NBCCCommon.h, 832  
Power Function PWM option constants, 581

PF\_PWM\_FWD7  
NBCCCommon.h, 832  
Power Function PWM option constants, 581

PF\_PWM\_REV1  
NBCCCommon.h, 832  
Power Function PWM option constants, 581

PF\_PWM\_REV2  
NBCCCommon.h, 832  
Power Function PWM option constants, 581

PF\_PWM\_REV3  
NBCCCommon.h, 832  
Power Function PWM option constants, 581

PF\_PWM\_REV4  
NBCCCommon.h, 832  
Power Function PWM option constants, 581

PF\_PWM\_REV5  
NBCCCommon.h, 832  
Power Function PWM option constants, 581

PF\_PWM\_REV6  
NBCCCommon.h, 833  
Power Function PWM option constants, 581

PF\_PWM\_REV7  
NBCCCommon.h, 833  
Power Function PWM option constants, 581

PFMATE\_CHANNEL\_1  
NBCCCommon.h, 833  
PFMate channel constants, 622

PFMATE\_CHANNEL\_2  
NBCCCommon.h, 833  
PFMate channel constants, 622

PFMATE\_CHANNEL\_3  
NBCCCommon.h, 833  
PFMate channel constants, 622

PFMATE\_CHANNEL\_4  
NBCCCommon.h, 833  
PFMate channel constants, 622

PFMATE\_CMD\_GO  
MindSensors PFMate constants, 619  
NBCCCommon.h, 833

PFMATE\_CMD\_RAW  
MindSensors PFMate constants, 619  
NBCCCommon.h, 833

PFMATE\_MOTORS\_A  
NBCCCommon.h, 833  
PFMate motor constants, 621

PFMATE\_MOTORS\_B  
NBCCCommon.h, 833  
PFMate motor constants, 621

PFMATE\_MOTORS\_BOTH  
NBCCCommon.h, 833  
PFMate motor constants, 621

PFMATE\_REG\_A\_CMD  
MindSensors PFMate constants, 619  
NBCCCommon.h, 833

PFMATE\_REG\_B\_CMD  
MindSensors PFMate constants, 619  
NBCCCommon.h, 833

PFMATE\_REG\_CHANNEL  
MindSensors PFMate constants, 619  
NBCCCommon.h, 834

PFMATE\_REG\_CMD  
MindSensors PFMate constants, 620  
NBCCCommon.h, 834

PFMATE\_REG MOTORS  
MindSensors PFMate constants, 620  
NBCCCommon.h, 834

PFMate channel constants, 622  
PFMATE\_CHANNEL\_1, 622  
PFMATE\_CHANNEL\_2, 622  
PFMATE\_CHANNEL\_3, 622  
PFMATE\_CHANNEL\_4, 622

PFMate motor constants, 621  
PFMATE\_MOTORS\_A, 621  
PFMATE\_MOTORS\_B, 621

PFMateSend  
MindSensors API Functions, 163

PFMateSendRaw  
MindSensors API Functions, 163

PI  
Miscellaneous NBC/NXC constants, 203  
NBCCCommon.h, 834

PID constants, 447  
PID\_0, 447  
PID\_1, 447  
PID\_2, 447  
PID\_3, 447  
PID\_4, 447  
PID\_5, 447  
PID\_6, 447  
PID\_7, 447

PID\_0  
NBCCCommon.h, 834  
PID constants, 447

PID\_1  
NBCCCommon.h, 834  
PID constants, 447

PID\_2

NBCCCommon.h, 834  
PID constants, 447

PID\_3  
    NBCCCommon.h, 834  
    PID constants, 447

PID\_4  
    NBCCCommon.h, 834  
    PID constants, 447

PID\_5  
    NBCCCommon.h, 834  
    PID constants, 447

PID\_6  
    NBCCCommon.h, 834  
    PID constants, 447

PID\_7  
    NBCCCommon.h, 834  
    PID constants, 447

POOL\_MAX\_SIZE  
    Command module constants, 70  
    NBCCCommon.h, 834

PROG\_ABORT  
    NBCCCommon.h, 835  
    Program status constants, 387

PROG\_ERROR  
    NBCCCommon.h, 835  
    Program status constants, 387

PROG\_IDLE  
    NBCCCommon.h, 835  
    Program status constants, 387

PROG\_OK  
    NBCCCommon.h, 835  
    Program status constants, 387

PROG\_RESET  
    NBCCCommon.h, 835  
    Program status constants, 387

PROG\_RUNNING  
    NBCCCommon.h, 835  
    Program status constants, 387

PSP\_BTNSET1\_DOWN  
    MindSensors PSP-Nx button set 1 constants, 611  
    NBCCCommon.h, 835

PSP\_BTNSET1\_L3  
    MindSensors PSP-Nx button set 1 constants, 611  
    NBCCCommon.h, 835

PSP\_BTNSET1\_LEFT  
    MindSensors PSP-Nx button set 1 constants, 611  
    NBCCCommon.h, 835

PSP\_BTNSET1\_R3  
    MindSensors PSP-Nx button set 1 constants, 611  
    NBCCCommon.h, 835

PSP\_BTNSET1\_RIGHT  
    MindSensors PSP-Nx button set 1 constants, 611  
    NBCCCommon.h, 835

PSP\_BTNSET1\_SELECT  
    MindSensors PSP-Nx button set 1 constants, 611  
    NBCCCommon.h, 835

PSP\_BTNSET1\_START  
    MindSensors PSP-Nx button set 1 constants, 611  
    NBCCCommon.h, 835

PSP\_BTNSET1\_UP  
    MindSensors PSP-Nx button set 1 constants, 611  
    NBCCCommon.h, 836

PSP\_BTNSET2\_CIRCLE  
    MindSensors PSP-Nx button set 2 constants, 612  
    NBCCCommon.h, 836

PSP\_BTNSET2\_CROSS  
    MindSensors PSP-Nx button set 2 constants, 612  
    NBCCCommon.h, 836

PSP\_BTNSET2\_L1  
    MindSensors PSP-Nx button set 2 constants, 612  
    NBCCCommon.h, 836

PSP\_BTNSET2\_L2  
    MindSensors PSP-Nx button set 2 constants, 612  
    NBCCCommon.h, 836

PSP\_BTNSET2\_R1  
    MindSensors PSP-Nx button set 2 constants, 612  
    NBCCCommon.h, 836

PSP\_BTNSET2\_R2  
    MindSensors PSP-Nx button set 2 constants, 612  
    NBCCCommon.h, 836

PSP\_BTNSET2\_SQUARE  
    MindSensors PSP-Nx button set 2 constants, 612  
    NBCCCommon.h, 836

PSP\_CMD\_ANALOG  
    MindSensors PSP-Nx constants, 609  
    NBCCCommon.h, 836

PSP\_CMD\_DIGITAL  
    MindSensors PSP-Nx constants, 609  
    NBCCCommon.h, 836

PSP\_REG\_BTNSET1  
    MindSensors PSP-Nx constants, 609  
    NBCCCommon.h, 836

PSP\_REG\_BTNSET2  
    MindSensors PSP-Nx constants, 609  
    NBCCCommon.h, 836

PSP\_REG\_XLEFT  
    MindSensors PSP-Nx constants, 609  
    NBCCCommon.h, 837

PSP\_REG\_XRIGHT  
    MindSensors PSP-Nx constants, 609  
    NBCCCommon.h, 837

PSP\_REG\_YLEFT  
    MindSensors PSP-Nx constants, 609  
    NBCCCommon.h, 837

PSP\_REG\_YRIGHT  
    MindSensors PSP-Nx constants, 610  
    NBCCCommon.h, 837

PSPNxAnalog

MindSensors API Functions, [164](#)  
PSPNxDigital  
    MindSensors API Functions, [164](#)  
PlayFile  
    Sound module functions, [267](#)  
PlayFileEx  
    Sound module functions, [267](#)  
PlayTone  
    Sound module functions, [268](#)  
PlayToneEx  
    Sound module functions, [268](#)  
PointOut  
    Display module functions, [260](#)  
PointOutEx  
    Display module functions, [260](#)  
PolyOut  
    Display module functions, [260](#)  
PolyOutEx  
    Display module functions, [261](#)  
Power Function CST options constants, [578](#)  
    PF\_CST\_CLEAR1\_SET2, [578](#)  
    PF\_CST\_FULL\_FWD, [578](#)  
    PF\_CST\_FULL\_REV, [578](#)  
    PF\_CST\_SET1\_CLEAR2, [578](#)  
    PF\_CST\_SET1\_SET2, [578](#)  
Power Function channel constants, [571](#)  
    PF\_CHANNEL\_1, [571](#)  
    PF\_CHANNEL\_2, [571](#)  
    PF\_CHANNEL\_3, [571](#)  
    PF\_CHANNEL\_4, [571](#)  
Power Function command constants, [570](#)  
    PF\_CMD\_BRAKE, [570](#)  
    PF\_CMD\_FLOAT, [570](#)  
    PF\_CMD\_FWD, [570](#)  
    PF\_CMD\_REV, [570](#)  
    PF\_CMD\_STOP, [570](#)  
Power Function mode constants, [572](#)  
    PF\_MODE\_COMBO\_PWM, [572](#)  
    PF\_MODE\_TRAIN, [572](#)  
Power Function output constants, [575](#)  
    PF\_OUT\_A, [575](#)  
    PF\_OUT\_B, [575](#)  
Power Function PWM option constants, [580](#)  
    PF\_PWM\_BRAKE, [580](#)  
    PF\_PWM\_FLOAT, [580](#)  
    PF\_PWM\_FWD1, [580](#)  
    PF\_PWM\_FWD2, [580](#)  
    PF\_PWM\_FWD3, [580](#)  
    PF\_PWM\_FWD4, [580](#)  
    PF\_PWM\_FWD5, [580](#)  
    PF\_PWM\_FWD6, [581](#)  
    PF\_PWM\_FWD7, [581](#)  
    PF\_PWM\_REV1, [581](#)  
    PF\_PWM\_REV2, [581](#)  
    PF\_PWM\_REV3, [581](#)  
    PF\_PWM\_REV4, [581](#)  
    PF\_PWM\_REV5, [581](#)  
    PF\_PWM\_REV6, [581](#)  
    PF\_PWM\_REV7, [581](#)  
Power Function pin constants, [576](#)  
    PF\_PIN\_C1, [576](#)  
    PF\_PIN\_C2, [576](#)  
Power Function single pin function constants, [577](#)  
    PF\_FUNC\_CLEAR, [577](#)  
    PF\_FUNC\_NOCHANGE, [577](#)  
    PF\_FUNC\_SET, [577](#)  
    PF\_FUNC\_TOGGLE, [577](#)  
PowerDown  
    IOCtrl module functions, [347](#)  
PowerField  
    NBCCommon.h, [835](#)  
    Output field constants, [459](#)  
PowerOn constants, [391](#)  
    IOCTRL\_BOOT, [391](#)  
    IOCTRL\_POWERDOWN, [391](#)  
Program status constants, [387](#)  
    PROG\_ABORT, [387](#)  
    PROG\_ERROR, [387](#)  
    PROG\_IDLE, [387](#)  
    PROG\_OK, [387](#)  
    PROG\_RESET, [387](#)  
    PROG\_RUNNING, [387](#)  
Property constants, [359](#)  
    RC\_PROP\_BTTONOFF, [359](#)  
    RC\_PROP\_DEBUGGING, [359](#)  
    RC\_PROP\_SOUND\_LEVEL, [359](#)  
RADIANS\_PER\_DEGREE  
    NBCCommon.h, [837](#)  
RAND\_MAX  
    Data type limits, [688](#)  
    NBCCommon.h, [837](#)  
RC\_PROP\_BTTONOFF  
    NBCCommon.h, [837](#)  
    Property constants, [359](#)  
RC\_PROP\_DEBUGGING  
    NBCCommon.h, [837](#)  
    Property constants, [359](#)  
RC\_PROP\_SOUND\_LEVEL  
    Property constants, [359](#)  
RCX and Scout opcode constants, [561](#)  
    RCX\_AbsVarOp, [562](#)  
    RCX\_AndVarOp, [562](#)  
    RCX\_AutoOffOp, [562](#)  
    RCX\_BatteryLevelOp, [562](#)  
    RCX\_BootModeOp, [563](#)  
    RCX\_CalibrateEventOp, [563](#)  
    RCX\_ClearAllEventsOp, [563](#)

RCX\_ClearCounterOp, 563  
RCX\_ClearMsgOp, 563  
RCX\_ClearSensorOp, 563  
RCX\_ClearSoundOp, 563  
RCX\_ClearTimerOp, 563  
RCX\_DatalogOp, 563  
RCX\_DecCounterOp, 563  
RCX\_DeleteSubOp, 563  
RCX\_DeleteSubsOp, 563  
RCX\_DeleteTaskOp, 563  
RCX\_DeleteTasksOp, 563  
RCX\_DirectEventOp, 564  
RCX\_DisplayOp, 564  
RCX\_DivVarOp, 564  
RCX\_GOutputDirOp, 564  
RCX\_GOutputModeOp, 564  
RCX\_GOutputPowerOp, 564  
RCX\_IRModeOp, 564  
RCX\_IncCounterOp, 564  
RCX\_InputModeOp, 564  
RCX\_InputTypeOp, 564  
RCX\_LSBlinkTimeOp, 564  
RCX\_LSCalibrateOp, 564  
RCX\_LSHysteresisOp, 564  
RCX\_LSLowerThreshOp, 565  
RCX\_LSUppерThreshOp, 565  
RCX\_LightOp, 564  
RCX\_MessageOp, 565  
RCX\_MulVarOp, 565  
RCX\_MuteSoundOp, 565  
RCX\_OnOffFloatOp, 565  
RCX\_OrVarOp, 565  
RCX\_OutputDirOp, 565  
RCX\_OutputPowerOp, 565  
RCX\_PBTurnOffOp, 565  
RCX\_PingOp, 565  
RCX\_PlaySoundOp, 565  
RCX\_PlayToneOp, 565  
RCX\_PlayToneVarOp, 565  
RCX\_PollMemoryOp, 566  
RCX\_PollOp, 566  
RCX\_RemoteOp, 566  
RCX\_ScoutOp, 566  
RCX\_ScoutRulesOp, 566  
RCX\_SelectProgramOp, 566  
RCX\_SendUARTDataOp, 566  
RCX\_SetCounterOp, 566  
RCX\_SetDatalogOp, 566  
RCX\_SetEventOp, 566  
RCX\_SetFeedbackOp, 566  
RCX\_SetPriorityOp, 566  
RCX\_SetSourceValueOp, 566  
RCX\_SetTimerLimitOp, 566  
RCX\_SetVarOp, 567  
RCX\_SetWatchOp, 567  
RCX\_SgnVarOp, 567  
RCX\_SoundOp, 567  
RCX\_StartTaskOp, 567  
RCX\_StopAllTasksOp, 567  
RCX\_StopTaskOp, 567  
RCX\_SubVarOp, 567  
RCX\_SumVarOp, 567  
RCX\_UnlockFirmOp, 567  
RCX\_UnlockOp, 567  
RCX\_UnmuteSoundOp, 567  
RCX\_UploadDatalogOp, 567  
RCX\_VLLOp, 568  
RCX\_ViewSourceValOp, 567  
RCX and Scout sound constants, 544  
SOUND\_CLICK, 544  
SOUND\_DOUBLE\_BEEP, 544  
SOUND\_DOWN, 544  
SOUND\_FAST\_UP, 544  
SOUND\_LOW\_BEEP, 544  
SOUND\_UP, 544  
RCX and Scout source constants, 557  
RCX\_BatteryLevelSrc, 558  
RCX\_ClickCounterSrc, 558  
RCX\_ConstantSrc, 558  
RCX\_CounterSrc, 558  
RCX\_DatalogRawDirectSrc, 558  
RCX\_DatalogRawIndirectSrc, 558  
RCX\_DatalogSrcDirectSrc, 558  
RCX\_DatalogSrcIndirectSrc, 558  
RCX\_DatalogValueDirectSrc, 558  
RCX\_DatalogValueIndirectSrc, 558  
RCX\_DurationSrc, 558  
RCX\_EventStateSrc, 558  
RCX\_FirmwareVersionSrc, 558  
RCX\_GlobalMotorStatusSrc, 558  
RCX\_HysteresisSrc, 558  
RCX\_IndirectVarSrc, 559  
RCX\_InputBooleanSrc, 559  
RCX\_InputModeSrc, 559  
RCX\_InputRawSrc, 559  
RCX\_InputTypeSrc, 559  
RCX\_InputValueSrc, 559  
RCX\_LowerThresholdSrc, 559  
RCX\_MessageSrc, 559  
RCX\_OutputStatusSrc, 559  
RCX\_ProgramSlotSrc, 559  
RCX\_RandomSrc, 559  
RCX\_ScoutCounterLimitSrc, 559  
RCX\_ScoutEventFBSrc, 559  
RCX\_ScoutLightParamsSrc, 559  
RCX\_ScoutRulesSrc, 560  
RCX\_ScoutTimerLimitSrc, 560  
RCX\_TaskEventsSrc, 560

RCX\_TenMSTimerSrc, 560  
RCX\_TimerSrc, 560  
RCX\_UARTSetupSrc, 560  
RCX\_UpperThresholdSrc, 560  
RCX\_VariableSrc, 560  
RCX\_WatchSrc, 560  
RCX constants, 537  
RCX IR remote constants, 542  
    RCX\_RemoteKeysReleased, 542  
    RCX\_RemoteOutABackward, 542  
    RCX\_RemoteOutAForward, 542  
    RCX\_RemoteOutBBackward, 542  
    RCX\_RemoteOutBForward, 542  
    RCX\_RemoteOutCBackward, 542  
    RCX\_RemoteOutCForward, 542  
    RCX\_RemotePBMessag1, 543  
    RCX\_RemotePBMessag2, 543  
    RCX\_RemotePBMessag3, 543  
    RCX\_RemotePlayASound, 543  
    RCX\_RemoteSelProgram1, 543  
    RCX\_RemoteSelProgram2, 543  
    RCX\_RemoteSelProgram3, 543  
    RCX\_RemoteSelProgram4, 543  
    RCX\_RemoteSelProgram5, 543  
    RCX\_RemoteStopOutOff, 543  
RCX output constants, 538  
    RCX\_OUT\_A, 538  
    RCX\_OUT\_AB, 538  
    RCX\_OUT\_ABC, 538  
    RCX\_OUT\_AC, 538  
    RCX\_OUT\_B, 538  
    RCX\_OUT\_BC, 538  
    RCX\_OUT\_C, 538  
RCX output direction constants, 540  
    RCX\_OUT\_FWD, 540  
    RCX\_OUT\_REV, 540  
    RCX\_OUT\_TOGGLE, 540  
RCX output mode constants, 539  
    RCX\_OUT\_FLOAT, 539  
    RCX\_OUT\_OFF, 539  
    RCX\_OUT\_ON, 539  
RCX output power constants, 541  
    RCX\_OUT\_FULL, 541  
    RCX\_OUT\_HALF, 541  
    RCX\_OUT\_LOW, 541  
RCX\_AbsVarOp  
    NBCCommon.h, 837  
    RCX and Scout opcode constants, 562  
RCX\_AndVarOp  
    NBCCommon.h, 838  
    RCX and Scout opcode constants, 562  
RCX\_AutoOffOp  
    NBCCommon.h, 838  
    RCX and Scout opcode constants, 562  
RCX\_BatteryLevelOp  
    NBCCommon.h, 838  
    RCX and Scout opcode constants, 562  
RCX\_BatteryLevelSrc  
    NBCCommon.h, 838  
    RCX and Scout source constants, 558  
RCX\_BootModeOp  
    NBCCommon.h, 838  
    RCX and Scout opcode constants, 563  
RCX\_CalibrateEventOp  
    NBCCommon.h, 838  
    RCX and Scout opcode constants, 563  
RCX\_ClearAllEventsOp  
    NBCCommon.h, 838  
    RCX and Scout opcode constants, 563  
RCX\_ClearCounterOp  
    NBCCommon.h, 838  
    RCX and Scout opcode constants, 563  
RCX\_ClearMsgOp  
    NBCCommon.h, 838  
    RCX and Scout opcode constants, 563  
RCX\_ClearSensorOp  
    NBCCommon.h, 838  
    RCX and Scout opcode constants, 563  
RCX\_ClearSoundOp  
    NBCCommon.h, 838  
    RCX and Scout opcode constants, 563  
RCX\_ClearTimerOp  
    NBCCommon.h, 838  
    RCX and Scout opcode constants, 563  
RCX\_ClickCounterSrc  
    NBCCommon.h, 838  
    RCX and Scout source constants, 558  
RCX\_ConstantSrc  
    NBCCommon.h, 838  
    RCX and Scout source constants, 558  
RCX\_CounterSrc  
    NBCCommon.h, 839  
    RCX and Scout source constants, 558  
RCX\_DatalogOp  
    NBCCommon.h, 839  
    RCX and Scout opcode constants, 563  
RCX\_DatalogRawDirectSrc  
    NBCCommon.h, 839  
    RCX and Scout source constants, 558  
RCX\_DatalogRawIndirectSrc  
    NBCCommon.h, 839  
    RCX and Scout source constants, 558  
RCX\_DatalogSrcDirectSrc  
    NBCCommon.h, 839  
    RCX and Scout source constants, 558  
RCX\_DatalogSrcIndirectSrc  
    NBCCommon.h, 839  
    RCX and Scout source constants, 558

RCX\_DatalogValueDirectSrc  
  NBCCCommon.h, 839  
  RCX and Scout source constants, 558

RCX\_DatalogValueIndirectSrc  
  NBCCCommon.h, 839  
  RCX and Scout source constants, 558

RCX\_DecCounterOp  
  NBCCCommon.h, 839  
  RCX and Scout opcode constants, 563

RCX\_DeleteSubOp  
  NBCCCommon.h, 839  
  RCX and Scout opcode constants, 563

RCX\_DeleteSubsOp  
  NBCCCommon.h, 839  
  RCX and Scout opcode constants, 563

RCX\_DeleteTaskOp  
  NBCCCommon.h, 839  
  RCX and Scout opcode constants, 563

RCX\_DeleteTasksOp  
  NBCCCommon.h, 839  
  RCX and Scout opcode constants, 563

RCX\_DirectEventOp  
  NBCCCommon.h, 839  
  RCX and Scout opcode constants, 564

RCX\_DisplayOp  
  NBCCCommon.h, 840  
  RCX and Scout opcode constants, 564

RCX\_DivVarOp  
  NBCCCommon.h, 840  
  RCX and Scout opcode constants, 564

RCX\_DurationSrc  
  NBCCCommon.h, 840  
  RCX and Scout source constants, 558

RCX\_EventStateSrc  
  NBCCCommon.h, 840  
  RCX and Scout source constants, 558

RCX\_FirmwareVersionSrc  
  NBCCCommon.h, 840  
  RCX and Scout source constants, 558

RCX\_GOutputDirOp  
  NBCCCommon.h, 840  
  RCX and Scout opcode constants, 564

RCX\_GOutputModeOp  
  NBCCCommon.h, 840  
  RCX and Scout opcode constants, 564

RCX\_GOutputPowerOp  
  NBCCCommon.h, 840  
  RCX and Scout opcode constants, 564

RCX\_GlobalMotorStatusSrc  
  NBCCCommon.h, 840  
  RCX and Scout source constants, 558

RCX\_HysteresisSrc  
  NBCCCommon.h, 840  
  RCX and Scout source constants, 558

RCX\_IRModeOp  
  NBCCCommon.h, 841  
  RCX and Scout opcode constants, 564

RCX\_IncCounterOp  
  NBCCCommon.h, 840  
  RCX and Scout opcode constants, 564

RCX\_IndirectVarSrc  
  NBCCCommon.h, 840  
  RCX and Scout source constants, 559

RCX\_InputBooleanSrc  
  NBCCCommon.h, 840  
  RCX and Scout source constants, 559

RCX\_InputModeOp  
  NBCCCommon.h, 840  
  RCX and Scout opcode constants, 564

RCX\_InputModeSrc  
  NBCCCommon.h, 841  
  RCX and Scout source constants, 559

RCX\_InputRawSrc  
  NBCCCommon.h, 841  
  RCX and Scout source constants, 559

RCX\_InputTypeOp  
  NBCCCommon.h, 841  
  RCX and Scout opcode constants, 564

RCX\_InputTypeSrc  
  NBCCCommon.h, 841  
  RCX and Scout source constants, 559

RCX\_InputValueSrc  
  NBCCCommon.h, 841  
  RCX and Scout source constants, 559

RCX\_LSBlinkTimeOp  
  NBCCCommon.h, 841  
  RCX and Scout opcode constants, 564

RCX\_LSCalibrateOp  
  NBCCCommon.h, 841  
  RCX and Scout opcode constants, 564

RCX\_LSHysteresisOp  
  NBCCCommon.h, 841  
  RCX and Scout opcode constants, 564

RCX\_LSLowerThreshOp  
  NBCCCommon.h, 841  
  RCX and Scout opcode constants, 565

RCX\_LSSupperThreshOp  
  NBCCCommon.h, 841  
  RCX and Scout opcode constants, 565

RCX\_LightOp  
  NBCCCommon.h, 841  
  RCX and Scout opcode constants, 564

RCX\_LowerThresholdSrc  
  NBCCCommon.h, 841  
  RCX and Scout source constants, 559

RCX\_MessageOp  
  NBCCCommon.h, 841  
  RCX and Scout opcode constants, 565

RCX\_MessageSrc  
NBCCCommon.h, 842  
RCX and Scout source constants, 559

RCX\_MulVarOp  
NBCCCommon.h, 842  
RCX and Scout opcode constants, 565

RCX\_MuteSoundOp  
NBCCCommon.h, 842  
RCX and Scout opcode constants, 565

RCX\_OUT\_A  
NBCCCommon.h, 842  
RCX output constants, 538

RCX\_OUT\_AB  
NBCCCommon.h, 842  
RCX output constants, 538

RCX\_OUT\_ABC  
NBCCCommon.h, 842  
RCX output constants, 538

RCX\_OUT\_AC  
NBCCCommon.h, 842  
RCX output constants, 538

RCX\_OUT\_B  
NBCCCommon.h, 842  
RCX output constants, 538

RCX\_OUT\_BC  
NBCCCommon.h, 842  
RCX output constants, 538

RCX\_OUT\_C  
NBCCCommon.h, 842  
RCX output constants, 538

RCX\_OUT\_FLOAT  
NBCCCommon.h, 842  
RCX output mode constants, 539

RCX\_OUT\_FULL  
NBCCCommon.h, 842  
RCX output power constants, 541

RCX\_OUT\_FWD  
NBCCCommon.h, 843  
RCX output direction constants, 540

RCX\_OUT\_HALF  
NBCCCommon.h, 843  
RCX output power constants, 541

RCX\_OUT\_LOW  
NBCCCommon.h, 843  
RCX output power constants, 541

RCX\_OUT\_OFF  
NBCCCommon.h, 843  
RCX output mode constants, 539

RCX\_OUT\_ON  
NBCCCommon.h, 843  
RCX output mode constants, 539

RCX\_OUT\_REV  
NBCCCommon.h, 843  
RCX output direction constants, 540

RCX\_OUT\_TOGGLE  
NBCCCommon.h, 843  
RCX output direction constants, 540

RCX\_OnOffFloatOp  
NBCCCommon.h, 842  
RCX and Scout opcode constants, 565

RCX\_OrVarOp  
NBCCCommon.h, 842  
RCX and Scout opcode constants, 565

RCX\_OutputDirOp  
NBCCCommon.h, 843  
RCX and Scout opcode constants, 565

RCX\_OutputPowerOp  
NBCCCommon.h, 843  
RCX and Scout opcode constants, 565

RCX\_OutputStatusSrc  
NBCCCommon.h, 843  
RCX and Scout source constants, 559

RCX\_PBTurnOffOp  
NBCCCommon.h, 843  
RCX and Scout opcode constants, 565

RCX\_PingOp  
NBCCCommon.h, 843  
RCX and Scout opcode constants, 565

RCX\_PlaySoundOp  
NBCCCommon.h, 843  
RCX and Scout opcode constants, 565

RCX\_PlayToneOp  
NBCCCommon.h, 843  
RCX and Scout opcode constants, 565

RCX\_PlayToneVarOp  
NBCCCommon.h, 844  
RCX and Scout opcode constants, 565

RCX\_PollMemoryOp  
NBCCCommon.h, 844  
RCX and Scout opcode constants, 566

RCX\_PollOp  
NBCCCommon.h, 844  
RCX and Scout opcode constants, 566

RCX\_ProgramSlotSrc  
NBCCCommon.h, 844  
RCX and Scout source constants, 559

RCX\_RandomSrc  
NBCCCommon.h, 844  
RCX and Scout source constants, 559

RCX\_RemoteKeysReleased  
NBCCCommon.h, 844  
RCX IR remote constants, 542

RCX\_RemoteOp  
NBCCCommon.h, 844  
RCX and Scout opcode constants, 566

RCX\_RemoteOutABackward  
NBCCCommon.h, 844  
RCX IR remote constants, 542

RCX\_RemoteOutAForward  
NBCCCommon.h, 844  
RCX IR remote constants, 542

RCX\_RemoteOutBBackward  
NBCCCommon.h, 844  
RCX IR remote constants, 542

RCX\_RemoteOutBForward  
NBCCCommon.h, 844  
RCX IR remote constants, 542

RCX\_RemoteOutCBackward  
NBCCCommon.h, 844  
RCX IR remote constants, 542

RCX\_RemoteOutCForward  
NBCCCommon.h, 844  
RCX IR remote constants, 542

RCX\_RemotePBMessagel  
NBCCCommon.h, 844  
RCX IR remote constants, 543

RCX\_RemotePBMessage2  
NBCCCommon.h, 845  
RCX IR remote constants, 543

RCX\_RemotePBMessage3  
NBCCCommon.h, 845  
RCX IR remote constants, 543

RCX\_RemotePlayASound  
NBCCCommon.h, 845  
RCX IR remote constants, 543

RCX\_RemoteSelProgram1  
NBCCCommon.h, 845  
RCX IR remote constants, 543

RCX\_RemoteSelProgram2  
NBCCCommon.h, 845  
RCX IR remote constants, 543

RCX\_RemoteSelProgram3  
NBCCCommon.h, 845  
RCX IR remote constants, 543

RCX\_RemoteSelProgram4  
NBCCCommon.h, 845  
RCX IR remote constants, 543

RCX\_RemoteSelProgram5  
NBCCCommon.h, 845  
RCX IR remote constants, 543

RCX\_RemoteStopOutOff  
NBCCCommon.h, 845  
RCX IR remote constants, 543

RCX\_ScoutCounterLimitSrc  
NBCCCommon.h, 845  
RCX and Scout source constants, 559

RCX\_ScoutEventFBSrc  
NBCCCommon.h, 845  
RCX and Scout source constants, 559

RCX\_ScoutLightParamsSrc  
NBCCCommon.h, 845  
RCX and Scout source constants, 559

RCX\_ScoutOp  
NBCCCommon.h, 845  
RCX and Scout opcode constants, 566

RCX\_ScoutRulesOp  
NBCCCommon.h, 845  
RCX and Scout opcode constants, 566

RCX\_ScoutRulesSrc  
NBCCCommon.h, 846  
RCX and Scout source constants, 560

RCX\_ScoutTimerLimitSrc  
NBCCCommon.h, 846  
RCX and Scout source constants, 560

RCX\_SelectProgramOp  
NBCCCommon.h, 846  
RCX and Scout opcode constants, 566

RCX\_SendUARTDataOp  
NBCCCommon.h, 846  
RCX and Scout opcode constants, 566

RCX\_SetCounterOp  
NBCCCommon.h, 846  
RCX and Scout opcode constants, 566

RCX\_SetDatalogOp  
NBCCCommon.h, 846  
RCX and Scout opcode constants, 566

RCX\_SetEventOp  
NBCCCommon.h, 846  
RCX and Scout opcode constants, 566

RCX\_SetFeedbackOp  
NBCCCommon.h, 846  
RCX and Scout opcode constants, 566

RCX\_SetPriorityOp  
NBCCCommon.h, 846  
RCX and Scout opcode constants, 566

RCX\_SetSourceValueOp  
NBCCCommon.h, 846  
RCX and Scout opcode constants, 566

RCX\_SetTimerLimitOp  
NBCCCommon.h, 846  
RCX and Scout opcode constants, 566

RCX\_SetVarOp  
NBCCCommon.h, 846  
RCX and Scout opcode constants, 567

RCX\_SetWatchOp  
NBCCCommon.h, 846  
RCX and Scout opcode constants, 567

RCX\_SgnVarOp  
NBCCCommon.h, 846  
RCX and Scout opcode constants, 567

RCX\_SoundOp  
NBCCCommon.h, 847  
RCX and Scout opcode constants, 567

RCX\_StartTaskOp  
NBCCCommon.h, 847  
RCX and Scout opcode constants, 567

RCX\_StopAllTasksOp  
NBCCCommon.h, 847  
RCX and Scout opcode constants, 567

RCX\_StopTaskOp  
NBCCCommon.h, 847  
RCX and Scout opcode constants, 567

RCX\_SubVarOp  
NBCCCommon.h, 847  
RCX and Scout opcode constants, 567

RCX\_SumVarOp  
NBCCCommon.h, 847  
RCX and Scout opcode constants, 567

RCX\_TaskEventsSrc  
NBCCCommon.h, 847  
RCX and Scout source constants, 560

RCX\_TenMSTimerSrc  
NBCCCommon.h, 847  
RCX and Scout source constants, 560

RCX\_TimerSrc  
NBCCCommon.h, 847  
RCX and Scout source constants, 560

RCX\_UARTSetupSrc  
NBCCCommon.h, 847  
RCX and Scout source constants, 560

RCX\_UnlockFirmOp  
NBCCCommon.h, 847  
RCX and Scout opcode constants, 567

RCX\_UnlockOp  
NBCCCommon.h, 847  
RCX and Scout opcode constants, 567

RCX\_UnmuteSoundOp  
NBCCCommon.h, 847  
RCX and Scout opcode constants, 567

RCX\_UploadDatalogOp  
NBCCCommon.h, 847  
RCX and Scout opcode constants, 567

RCX\_UpperThresholdSrc  
NBCCCommon.h, 848  
RCX and Scout source constants, 560

RCX\_VLLOp  
NBCCCommon.h, 848  
RCX and Scout opcode constants, 568

RCX\_VariableSrc  
NBCCCommon.h, 848  
RCX and Scout source constants, 560

RCX\_ViewSourceValOp  
NBCCCommon.h, 848  
RCX and Scout opcode constants, 567

RCX\_WatchSrc  
NBCCCommon.h, 848  
RCX and Scout source constants, 560

RESET\_ALL  
NBCCCommon.h, 849  
Tachometer counter reset flags, 451

RESET\_BLOCK\_COUNT  
NBCCCommon.h, 849  
Tachometer counter reset flags, 451

RESET\_BLOCKANDTACHO  
NBCCCommon.h, 849  
Tachometer counter reset flags, 451

RESET\_COUNT  
NBCCCommon.h, 849  
Tachometer counter reset flags, 451

RESET\_NONE  
NBCCCommon.h, 849  
Tachometer counter reset flags, 451

RFID\_MODE\_SINGLE  
Codatex RFID sensor modes, 654  
NBCCCommon.h, 849

RFID\_MODE\_STOP  
Codatex RFID sensor modes, 654  
NBCCCommon.h, 849

RFIDInit  
Codatex API Functions, 181

RFIDMode  
Codatex API Functions, 181

RFIDRead  
Codatex API Functions, 182

RFIDReadContinuous  
Codatex API Functions, 182

RFIDReadSingle  
Codatex API Functions, 182

RFIDStatus  
Codatex API Functions, 182

RFIDStop  
Codatex API Functions, 183

RIC Macro Wrappers, 194

- RICArg, 195
- RICImgPoint, 195
- RICImgRect, 195
- RICMapArg, 195
- RICMapElement, 195
- RICMapFunction, 196
- RICOpcircle, 196
- RICOpcopybits, 196
- RICOpdescription, 196
- RICOpellipse, 196
- RICOpline, 197
- RICOpnumbox, 197
- RICOppixel, 197
- RICOppolygon, 197
- RICOprrect, 198
- RICOpsprite, 198
- RICOpvarmap, 198
- RICPolygonPoints, 198
- RICSpriteData, 198

RICArg  
NBCCCommon.h, 849

RIC Macro Wrappers, 195  
RICImgPoint  
  NBCCCommon.h, 850  
  RIC Macro Wrappers, 195  
RICImgRect  
  NBCCCommon.h, 850  
  RIC Macro Wrappers, 195  
RICMapArg  
  NBCCCommon.h, 850  
  RIC Macro Wrappers, 195  
RICMapElement  
  NBCCCommon.h, 850  
  RIC Macro Wrappers, 195  
RICMapFunction  
  NBCCCommon.h, 850  
  RIC Macro Wrappers, 196  
RICOOpCircle  
  NBCCCommon.h, 850  
  RIC Macro Wrappers, 196  
RICOOpCopyBits  
  NBCCCommon.h, 851  
  RIC Macro Wrappers, 196  
RICOOpDescription  
  NBCCCommon.h, 851  
  RIC Macro Wrappers, 196  
RICOOpEllipse  
  NBCCCommon.h, 851  
  RIC Macro Wrappers, 196  
RICOOpLine  
  NBCCCommon.h, 851  
  RIC Macro Wrappers, 197  
RICOOpNumBox  
  NBCCCommon.h, 851  
  RIC Macro Wrappers, 197  
RICOOpPixel  
  NBCCCommon.h, 852  
  RIC Macro Wrappers, 197  
RICOOpPolygon  
  NBCCCommon.h, 852  
  RIC Macro Wrappers, 197  
RICOOpRect  
  NBCCCommon.h, 852  
  RIC Macro Wrappers, 198  
RICOOpSprite  
  NBCCCommon.h, 852  
  RIC Macro Wrappers, 198  
RICOOpVarMap  
  NBCCCommon.h, 853  
  RIC Macro Wrappers, 198  
RICPolygonPoints  
  NBCCCommon.h, 853  
  RIC Macro Wrappers, 198  
RICSpriteData  
  NBCCCommon.h, 853  
RIC Macro Wrappers, 198  
ROTATE\_QUEUE  
  NBCCCommon.h, 853  
  VM state constants, 381  
RS485Control  
  Comm module functions, 315  
RS485Disable  
  Comm module functions, 316  
RS485Enable  
  Comm module functions, 316  
RS485Initialize  
  Comm module functions, 316  
RS485Read  
  Comm module functions, 317  
RS485ReadEx  
  Comm module functions, 317  
RS485Status  
  Comm module functions, 317  
RS485Uart  
  Comm module functions, 317  
RS485Write  
  Comm module functions, 318  
Random  
  cstdlib API, 357  
RandomEx  
  NBCCCommon.h, 837  
  System Call function constants, 370  
RandomNumber  
  NBCCCommon.h, 837  
  System Call function constants, 370  
RawValueField  
  Input field constants, 436  
  NBCCCommon.h, 837  
Read  
  Loader module functions, 352  
ReadACCLNxSensitivity  
  MindSensors API Functions, 164  
ReadACCLNxXOffset  
  MindSensors API Functions, 164  
ReadACCLNxXRange  
  MindSensors API Functions, 164  
ReadACCLNxYOffset  
  MindSensors API Functions, 165  
ReadACCLNxYRange  
  MindSensors API Functions, 165  
ReadACCLNxZOffset  
  MindSensors API Functions, 165  
ReadACCLNxZRange  
  MindSensors API Functions, 166  
ReadButton  
  NBCCCommon.h, 848  
  System Call function constants, 370  
ReadButtonEx  
  Button module functions, 286

ReadBytes  
    Loader module functions, [352](#)

ReadDISTNxDistance  
    MindSensors API Functions, [166](#)

ReadDISTNxMaxDistance  
    MindSensors API Functions, [166](#)

ReadDISTNxMinDistance  
    MindSensors API Functions, [166](#)

ReadDISTNxModuleType  
    MindSensors API Functions, [167](#)

ReadDISTNxNumPoints  
    MindSensors API Functions, [167](#)

ReadDISTNxVoltage  
    MindSensors API Functions, [167](#)

ReadI2CBytes  
    LowSpeed module functions, [241](#)

ReadI2CDeviceId  
    LowSpeed module functions, [242](#)

ReadI2CDeviceInfo  
    LowSpeed module functions, [242](#)

ReadI2CRegister  
    LowSpeed module functions, [242](#)

ReadI2CVendorId  
    LowSpeed module functions, [243](#)

ReadI2CVersion  
    LowSpeed module functions, [243](#)

ReadLastResponse  
    NBCCommon.h, [848](#)  
    System Call function constants, [370](#)

ReadLn  
    Loader module functions, [353](#)

ReadLnString  
    Loader module functions, [353](#)

ReadNRLinkBytes  
    MindSensors API Functions, [167](#)

ReadNRLinkStatus  
    MindSensors API Functions, [168](#)

ReadNXTLineLeaderAverage  
    MindSensors API Functions, [168](#)

ReadNXTLineLeaderResult  
    MindSensors API Functions, [168](#)

ReadNXTLineLeaderSteering  
    MindSensors API Functions, [169](#)

ReadNXTPowerMeterCapacityUsed  
    MindSensors API Functions, [169](#)

ReadNXTPowerMeterElapsedTime  
    MindSensors API Functions, [169](#)

ReadNXTPowerMeterErrorCount  
    MindSensors API Functions, [169](#)

ReadNXTPowerMeterMaxCurrent  
    MindSensors API Functions, [170](#)

ReadNXTPowerMeterMaxVoltage  
    MindSensors API Functions, [170](#)

ReadNXTPowerMeterMinCurrent  
    MindSensors API Functions, [170](#)

ReadNXTPowerMeterMinVoltage  
    MindSensors API Functions, [171](#)

ReadNXTPowerMeterPresentCurrent  
    MindSensors API Functions, [171](#)

ReadNXTPowerMeterPresentPower  
    MindSensors API Functions, [171](#)

ReadNXTPowerMeterPresentVoltage  
    MindSensors API Functions, [171](#)

ReadNXTPowerMeterTotalPowerConsumed  
    MindSensors API Functions, [172](#)

ReadNXTServoBatteryVoltage  
    MindSensors API Functions, [172](#)

ReadNXTServoPosition  
    MindSensors API Functions, [172](#)

ReadNXTServoSpeed  
    MindSensors API Functions, [173](#)

ReadSemData  
    NBCCommon.h, [848](#)  
    System Call function constants, [371](#)

ReadSensor  
    Input module functions, [231](#)

ReadSensorColorEx  
    Input module functions, [231](#)

ReadSensorColorRaw  
    Input module functions, [232](#)

ReadSensorDIAccl  
    Dexter Industries API Functions, [185](#)

ReadSensorDIAccl8  
    Dexter Industries API Functions, [185](#)

ReadSensorDIAccl8Raw  
    Dexter Industries API Functions, [186](#)

ReadSensorDIAcclDrift  
    Dexter Industries API Functions, [186](#)

ReadSensorDIAcclRaw  
    Dexter Industries API Functions, [186](#)

ReadSensorDIAcclStatus  
    Dexter Industries API Functions, [186](#)

ReadSensorDIGPSDistanceToWaypoint  
    Dexter Industries API Functions, [187](#)

ReadSensorDIGPSHeading  
    Dexter Industries API Functions, [187](#)

ReadSensorDIGPSHeadingToWaypoint  
    Dexter Industries API Functions, [187](#)

ReadSensorDIGPSLatitude  
    Dexter Industries API Functions, [187](#)

ReadSensorDIGPSLongitude  
    Dexter Industries API Functions, [187](#)

ReadSensorDIGPSRelativeHeading  
    Dexter Industries API Functions, [188](#)

ReadSensorDIGPSStatus  
    Dexter Industries API Functions, [188](#)

ReadSensorDIGPSTime  
    Dexter Industries API Functions, [188](#)

- ReadSensorDIGPSVelocity  
    Dexter Industries API Functions, 188
- ReadSensorDIGyro  
    Dexter Industries API Functions, 188
- ReadSensorDIGyroRaw  
    Dexter Industries API Functions, 189
- ReadSensorDIGyroStatus  
    Dexter Industries API Functions, 189
- ReadSensorDIGyroTemperature  
    Dexter Industries API Functions, 189
- ReadSensorEMeter  
    LowSpeed module functions, 243
- ReadSensorHTAccel  
    HiTechnic API Functions, 102
- ReadSensorHTAngle  
    HiTechnic API Functions, 102
- ReadSensorHTBarometric  
    HiTechnic API Functions, 103
- ReadSensorHTColor  
    HiTechnic API Functions, 103
- ReadSensorHTColor2Active  
    HiTechnic API Functions, 103
- ReadSensorHTColorNum  
    HiTechnic API Functions, 103
- ReadSensorHTCompass  
    HiTechnic API Functions, 104
- ReadSensorHTEOPD  
    HiTechnic API Functions, 104
- ReadSensorHTForce  
    HiTechnic API Functions, 104
- ReadSensorHTGyro  
    HiTechnic API Functions, 104
- ReadSensorHTIRReceiver  
    HiTechnic API Functions, 105
- ReadSensorHTIRReceiverEx  
    HiTechnic API Functions, 105
- ReadSensorHTIRSeeker  
    HiTechnic API Functions, 105
- ReadSensorHTIRSeeker2AC  
    HiTechnic API Functions, 105
- ReadSensorHTIRSeeker2Addr  
    HiTechnic API Functions, 106
- ReadSensorHTIRSeeker2DC  
    HiTechnic API Functions, 106
- ReadSensorHTIRSeekerDir  
    HiTechnic API Functions, 106
- ReadSensorHTMagnet  
    HiTechnic API Functions, 107
- ReadSensorHTNormalizedColor  
    HiTechnic API Functions, 107
- ReadSensorHTNormalizedColor2Active  
    HiTechnic API Functions, 107
- ReadSensorHTPIR  
    HiTechnic API Functions, 107
- ReadSensorHTProtoAllAnalog  
    HiTechnic API Functions, 108
- ReadSensorHTProtoAnalog  
    HiTechnic API Functions, 108
- ReadSensorHTProtoDigital  
    HiTechnic API Functions, 108
- ReadSensorHTProtoDigitalControl  
    HiTechnic API Functions, 108
- ReadSensorHTRawColor  
    HiTechnic API Functions, 109
- ReadSensorHTRawColor2  
    HiTechnic API Functions, 109
- ReadSensorHTSuperProAllAnalog  
    HiTechnic API Functions, 109
- ReadSensorHTSuperProAnalogs  
    HiTechnic API Functions, 110
- ReadSensorHTSuperProAnalogOut  
    HiTechnic API Functions, 110
- ReadSensorHTSuperProDigital  
    HiTechnic API Functions, 110
- ReadSensorHTSuperProDigitalControl  
    HiTechnic API Functions, 110
- ReadSensorHTSuperProLED  
    HiTechnic API Functions, 111
- ReadSensorHTSuperProProgramControl  
    HiTechnic API Functions, 111
- ReadSensorHTSuperProStrobe  
    HiTechnic API Functions, 111
- ReadSensorHTTouchMultiplexer  
    HiTechnic API Functions, 111
- ReadSensorMIXG1300L  
    Microinfinity API Functions, 192
- ReadSensorMIXG1300LScale  
    Microinfinity API Functions, 192
- ReadSensorMSAccel  
    MindSensors API Functions, 173
- ReadSensorMSCompass  
    MindSensors API Functions, 173
- ReadSensorMSDROD  
    MindSensors API Functions, 173
- ReadSensorMSPlayStation  
    MindSensors API Functions, 174
- ReadSensorMSPressure  
    MindSensors API Functions, 174
- ReadSensorMSPressureRaw  
    MindSensors API Functions, 174
- ReadSensorMSRTClock  
    MindSensors API Functions, 174
- ReadSensorMSTilt  
    MindSensors API Functions, 175
- ReadSensorNXTSumoEyes  
    MindSensors API Functions, 175
- ReadSensorTemperature  
    LowSpeed module functions, 244

ReadSensorUS  
    LowSpeed module functions, 244

ReadSensorUSEx  
    LowSpeed module functions, 244

RebootInFirmwareMode  
    IOCtrl module functions, 347

ReceiveMessage  
    Comm module functions, 313

ReceiveRemoteBool  
    Comm module functions, 314

ReceiveRemoteMessageEx  
    Comm module functions, 314

ReceiveRemoteNumber  
    Comm module functions, 314

ReceiveRemoteString  
    Comm module functions, 314

RectOut  
    Display module functions, 261

RectOutEx  
    Display module functions, 261

RegDValueField  
    NBCCCommon.h, 848  
    Output field constants, 459

RegIValueField  
    NBCCCommon.h, 848  
    Output field constants, 459

RegModeField  
    NBCCCommon.h, 848  
    Output field constants, 459

RegPValueField  
    NBCCCommon.h, 849  
    Output field constants, 459

Remote connection constants, 511  
    CONN\_BT0, 511  
    CONN\_BT1, 511  
    CONN\_BT2, 511  
    CONN\_BT3, 511  
    CONN\_HS4, 511  
    CONN\_HS\_1, 511  
    CONN\_HS\_2, 511  
    CONN\_HS\_3, 512  
    CONN\_HS\_4, 512  
    CONN\_HS\_5, 512  
    CONN\_HS\_6, 512  
    CONN\_HS\_7, 512  
    CONN\_HS\_8, 512  
    CONN\_HS\_ALL, 512

Remote control (direct commands) errors, 386  
    ERR\_RC\_BAD\_PACKET, 386  
    ERR\_RC\_FAILED, 386  
    ERR\_RC\_ILLEGAL\_VAL, 386  
    ERR\_RC\_UNKNOWN\_CMD, 386

RemoteBluetoothFactoryReset  
    System Command functions, 338

RemoteCloseFile  
    System Command functions, 338

RemoteConnectionIdle  
    Comm module functions, 315

RemoteConnectionWrite  
    Comm module functions, 315

RemoteDatalogRead  
    Direct Command functions, 328

RemoteDatalogSetTimes  
    Direct Command functions, 329

RemoteDeleteFile  
    System Command functions, 339

RemoteDeleteUserFlash  
    System Command functions, 339

RemoteFindFirstFile  
    System Command functions, 339

RemoteFindNextFile  
    System Command functions, 340

RemoteGetBatteryLevel  
    Direct Command functions, 329

RemoteGetBluetoothAddress  
    System Command functions, 340

RemoteGetConnectionCount  
    Direct Command functions, 329

RemoteGetConnectionName  
    Direct Command functions, 329

RemoteGetContactCount  
    Direct Command functions, 329

RemoteGetContactName  
    Direct Command functions, 330

RemoteGetCurrentProgramName  
    Direct Command functions, 330

RemoteGetDeviceInfo  
    System Command functions, 340

RemoteGetFirmwareVersion  
    System Command functions, 341

RemoteGetInputValues  
    Direct Command functions, 330

RemoteGetOutputState  
    Direct Command functions, 331

RemoteGetProperty  
    Direct Command functions, 331

RemoteIOMapRead  
    System Command functions, 341

RemoteIOMapWriteBytes  
    System Command functions, 341

RemoteIOMapWriteValue  
    System Command functions, 342

RemoteKeepAlive  
    Direct Command functions, 331

RemoteLowspeedGetStatus  
    Direct Command functions, 331

RemoteLowspeedRead  
    Direct Command functions, 332

RemoteLowspeedWrite  
    Direct Command functions, [332](#)

RemoteMessageRead  
    Direct Command functions, [332](#)

RemoteMessageWrite  
    Direct Command functions, [332](#)

RemoteOpenAppendData  
    System Command functions, [342](#)

RemoteOpenRead  
    System Command functions, [342](#)

RemoteOpenWrite  
    System Command functions, [343](#)

RemoteOpenWriteData  
    System Command functions, [343](#)

RemoteOpenWriteLinear  
    System Command functions, [344](#)

RemotePlaySoundFile  
    Direct Command functions, [333](#)

RemotePlayTone  
    Direct Command functions, [333](#)

RemotePollCommand  
    System Command functions, [344](#)

RemotePollCommandLength  
    System Command functions, [344](#)

RemoteRead  
    System Command functions, [344](#)

RemoteRenameFile  
    System Command functions, [345](#)

RemoteResetMotorPosition  
    Direct Command functions, [333](#)

RemoteResetScaledValue  
    Direct Command functions, [333](#)

RemoteResetTachoCount  
    Direct Command functions, [334](#)

RemoteSetBrickName  
    System Command functions, [345](#)

RemoteSetInputMode  
    Direct Command functions, [334](#)

RemoteSetOutputState  
    Direct Command functions, [334](#)

Remote SetProperty  
    Direct Command functions, [335](#)

RemoteStartProgram  
    Direct Command functions, [335](#)

RemoteStopProgram  
    Direct Command functions, [335](#)

RemoteStopSound  
    Direct Command functions, [335](#)

RemoteWrite  
    System Command functions, [345](#)

RenameFile  
    Loader module functions, [353](#)

ResetAllTachoCounts  
    Output module functions, [222](#)

ResetBlockTachoCount  
    Output module functions, [222](#)

ResetHTBarometricCalibration  
    HiTechnic API Functions, [112](#)

ResetMIXG1300L  
    Microinfinity API Functions, [193](#)

ResetRotationCount  
    Output module functions, [222](#)

ResetSensor  
    Input module functions, [232](#)

ResetSensorHTAngle  
    HiTechnic API Functions, [112](#)

ResetSleepTimer  
    Command module functions, [279](#)

ResetTachoCount  
    Output module functions, [223](#)

ResizeFile  
    Loader module functions, [353](#)

ResolveHandle  
    Loader module functions, [354](#)

RotateMotor  
    Output module functions, [223](#)

RotateMotorEx  
    Output module functions, [223](#)

RotateMotorExPID  
    Output module functions, [223](#)

RotateMotorPID  
    Output module functions, [224](#)

RotationCountField  
    NBCCommon.h, [853](#)  
    Output field constants, [460](#)

RunNRLinkMacro  
    MindSensors API Functions, [175](#)

RunStateField  
    NBCCommon.h, [853](#)  
    Output field constants, [460](#)

SAMPLERATE\_DEFAULT  
    NBCCommon.h, [853](#)  
    Sound module miscellaneous constants, [408](#)

SAMPLERATE\_MAX  
    NBCCommon.h, [854](#)  
    Sound module miscellaneous constants, [408](#)

SAMPLERATE\_MIN  
    NBCCommon.h, [854](#)  
    Sound module miscellaneous constants, [408](#)

SCHAR\_MAX  
    Data type limits, [688](#)  
    NBCCommon.h, [854](#)

SCHAR\_MIN  
    Data type limits, [688](#)  
    NBCCommon.h, [854](#)

SCOUT\_FXR\_ALARM  
    NBCCommon.h, [854](#)

Scout special effect constants, 556  
SCOUT\_FXR\_BUG  
    NBCCCommon.h, 854  
    Scout special effect constants, 556  
SCOUT\_FXR\_NONE  
    NBCCCommon.h, 854  
    Scout special effect constants, 556  
SCOUT\_FXR\_RANDOM  
    NBCCCommon.h, 854  
    Scout special effect constants, 556  
SCOUT\_FXR\_SCIENCE  
    NBCCCommon.h, 854  
    Scout special effect constants, 556  
SCOUT\_LIGHT\_OFF  
    NBCCCommon.h, 854  
    Scout light constants, 546  
SCOUT\_LIGHT\_ON  
    NBCCCommon.h, 854  
    Scout light constants, 546  
SCOUT\_LR\_AVOID  
    NBCCCommon.h, 854  
    Scout light rule constants, 554  
SCOUT\_LR\_IGNORE  
    NBCCCommon.h, 854  
    Scout light rule constants, 554  
SCOUT\_LR\_OFF\_WHEN  
    NBCCCommon.h, 855  
    Scout light rule constants, 554  
SCOUT\_LR\_SEEK\_DARK  
    Scout light rule constants, 554  
SCOUT\_LR\_WAIT\_FOR  
    NBCCCommon.h, 855  
    Scout light rule constants, 554  
SCOUT\_MODE\_POWER  
    NBCCCommon.h, 855  
    Scout mode constants, 551  
SCOUT\_MR\_FORWARD  
    NBCCCommon.h, 855  
    Scout motion rule constants, 552  
SCOUT\_MR\_LOOP\_A  
    NBCCCommon.h, 855  
    Scout motion rule constants, 552  
SCOUT\_MR\_LOOP\_AB  
    NBCCCommon.h, 855  
    Scout motion rule constants, 552  
SCOUT\_MR\_LOOP\_B  
    NBCCCommon.h, 855  
    Scout motion rule constants, 552  
SCOUT\_MR\_NO\_MOTION  
    Scout motion rule constants, 552  
SCOUT\_MR\_ZIGZAG  
    NBCCCommon.h, 855  
    Scout motion rule constants, 552  
SCOUT\_SNDSET\_ALARM  
    NBCCCommon.h, 856  
    Scout sound set constants, 550  
SCOUT\_SNDSET\_BASIC  
    NBCCCommon.h, 856  
    Scout sound set constants, 550  
SCOUT\_SNDSET\_BUG  
    NBCCCommon.h, 856  
    Scout sound set constants, 550  
SCOUT\_SNDSET\_NONE  
    NBCCCommon.h, 856  
    Scout sound set constants, 550  
SCOUT\_SNDSET\_RANDOM  
    Scout sound set constants, 550  
SCOUT\_SOUND\_1\_BLINK  
    Scout sound constants, 547  
SCOUT\_SOUND\_2\_BLINK  
    Scout sound constants, 547  
SCOUT\_SOUND\_COUNTER1  
    Scout sound constants, 547  
SCOUT\_SOUND\_COUNTER2  
    Scout sound constants, 547  
SCOUT\_SOUND\_ENTERSA  
    Scout sound constants, 548  
SCOUT\_SOUND\_NONE  
    NBCCCommon.h, 857  
    Scout sound constants, 548  
SCOUT\_SOUND\_REMOTE  
    NBCCCommon.h, 857  
    Scout sound constants, 548  
SCOUT\_SOUND\_SPECIAL1  
    Scout sound constants, 548  
SCOUT\_SOUND\_SPECIAL2  
    Scout sound constants, 548  
SCOUT\_SOUND\_SPECIAL3  
    Scout sound constants, 548  
SCOUT\_SOUND\_TIMER1  
    NBCCCommon.h, 857  
    Scout sound constants, 548  
SCOUT\_SOUND\_TIMER2  
    NBCCCommon.h, 857  
    Scout sound constants, 548  
SCOUT\_SOUND\_TIMER3  
    NBCCCommon.h, 857  
    Scout sound constants, 548  
SCOUT\_TGS\_LONG  
    NBCCCommon.h, 858  
    Scout transmit rule constants, 555  
SCOUT\_TGS\_MEDIUM  
    NBCCCommon.h, 858  
    Scout transmit rule constants, 555  
SCOUT\_TGS\_SHORT  
    NBCCCommon.h, 858  
    Scout transmit rule constants, 555  
SCOUT\_TR\_AVOID

NBCCCommon.h, 858  
Scout touch rule constants, 553

SCOUT\_TR\_IGNORE  
    NBCCCommon.h, 858  
        Scout touch rule constants, 553

SCOUT\_TR\_OFF\_WHEN  
    NBCCCommon.h, 858  
        Scout touch rule constants, 553

SCOUT\_TR\_REVERSE  
    NBCCCommon.h, 858  
        Scout touch rule constants, 553

SCOUT\_TR\_WAIT\_FOR  
    NBCCCommon.h, 858  
        Scout touch rule constants, 553

SCREEN\_BACKGROUND  
    Display module constants, 484  
    NBCCCommon.h, 858

SCREEN\_LARGE  
    Display module constants, 484  
    NBCCCommon.h, 858

SCREEN\_MODE\_CLEAR  
    Display module constants, 484  
    NBCCCommon.h, 858

SCREEN\_MODE\_RESTORE  
    Display module constants, 484

SCREEN\_SMALL  
    Display module constants, 484  
    NBCCCommon.h, 859

SCREENS  
    Display module constants, 484  
    NBCCCommon.h, 859

SEC\_1  
    NBCCCommon.h, 859  
    Time constants, 377

SEC\_10  
    NBCCCommon.h, 859  
    Time constants, 377

SEC\_15  
    NBCCCommon.h, 859  
    Time constants, 378

SEC\_2  
    NBCCCommon.h, 859  
    Time constants, 378

SEC\_20  
    NBCCCommon.h, 859  
    Time constants, 378

SEC\_3  
    NBCCCommon.h, 859  
    Time constants, 378

SEC\_30  
    NBCCCommon.h, 859  
    Time constants, 378

SEC\_4  
    NBCCCommon.h, 859

Time constants, 378

SEC\_5  
    NBCCCommon.h, 859  
    Time constants, 378

SEC\_6  
    NBCCCommon.h, 859  
    Time constants, 378

SEC\_7  
    NBCCCommon.h, 859  
    Time constants, 378

SEC\_8  
    NBCCCommon.h, 859  
    Time constants, 378

SEC\_9  
    NBCCCommon.h, 860  
    Time constants, 378

SHRT\_MAX  
    Data type limits, 688  
    NBCCCommon.h, 860

SHRT\_MIN  
    Data type limits, 688  
    NBCCCommon.h, 860

SIZE\_OF\_BDADDR  
    Miscellaneous Comm module constants, 502  
    NBCCCommon.h, 860

SIZE\_OF\_BRICK\_NAME  
    Miscellaneous Comm module constants, 502

SIZE\_OF\_BT\_NAME  
    Miscellaneous Comm module constants, 503  
    NBCCCommon.h, 860

SIZE\_OF\_BT\_PINCODE  
    Miscellaneous Comm module constants, 503

SIZE\_OF\_BTBUF  
    Miscellaneous Comm module constants, 503  
    NBCCCommon.h, 860

SIZE\_OF\_HSBUF  
    Miscellaneous Comm module constants, 503  
    NBCCCommon.h, 860

SIZE\_OF\_USBBUF  
    Miscellaneous Comm module constants, 503  
    NBCCCommon.h, 861

SIZE\_OF\_USBDATA  
    Miscellaneous Comm module constants, 503  
    NBCCCommon.h, 861

SOUND\_CLICK  
    NBCCCommon.h, 861  
    RCX and Scout sound constants, 544

SOUND\_DOUBLE\_BEEP  
    NBCCCommon.h, 861  
    RCX and Scout sound constants, 544

SOUND\_DOWN  
    NBCCCommon.h, 861  
    RCX and Scout sound constants, 544

SOUND\_FAST\_UP

NBCCCommon.h, [861](#)  
RCX and Scout sound constants, [544](#)

SOUND\_FLAGS\_IDLE  
    NBCCCommon.h, [861](#)  
    SoundFlags constants, [404](#)

SOUND\_FLAGS\_RUNNING  
    SoundFlags constants, [404](#)

SOUND\_FLAGS\_UPDATE  
    NBCCCommon.h, [861](#)  
    SoundFlags constants, [404](#)

SOUND\_LOW\_BEEP  
    NBCCCommon.h, [861](#)  
    RCX and Scout sound constants, [544](#)

SOUND\_MODE\_LOOP  
    NBCCCommon.h, [861](#)  
    SoundMode constants, [406](#)

SOUND\_MODE\_ONCE  
    NBCCCommon.h, [861](#)  
    SoundMode constants, [406](#)

SOUND\_MODE\_TONE  
    NBCCCommon.h, [861](#)  
    SoundMode constants, [406](#)

SOUND\_STATE\_FILE  
    NBCCCommon.h, [861](#)  
    SoundState constants, [405](#)

SOUND\_STATE\_IDLE  
    NBCCCommon.h, [862](#)  
    SoundState constants, [405](#)

SOUND\_STATE\_STOP  
    NBCCCommon.h, [862](#)  
    SoundState constants, [405](#)

SOUND\_STATE\_TONE  
    NBCCCommon.h, [862](#)  
    SoundState constants, [405](#)

SOUND\_UP  
    NBCCCommon.h, [862](#)  
    RCX and Scout sound constants, [544](#)

SPECIALS  
    Display module constants, [485](#)  
    NBCCCommon.h, [863](#)

STAT\_COMM\_PENDING  
    Command module constants, [70](#)  
    NBCCCommon.h, [863](#)

STATUSICON\_BATTERY  
    Display module constants, [485](#)  
    NBCCCommon.h, [863](#)

STATUSICON\_BLUETOOTH  
    Display module constants, [485](#)

STATUSICON\_USB  
    Display module constants, [485](#)  
    NBCCCommon.h, [863](#)

STATUSICON\_VM  
    Display module constants, [485](#)  
    NBCCCommon.h, [863](#)

STATUSICONS  
    Display module constants, [485](#)  
    NBCCCommon.h, [863](#)

STATUSTEXT  
    Display module constants, [485](#)  
    NBCCCommon.h, [863](#)

STEPICON\_1  
    Display module constants, [485](#)  
    NBCCCommon.h, [863](#)

STEPICON\_2  
    Display module constants, [485](#)  
    NBCCCommon.h, [864](#)

STEPICON\_3  
    Display module constants, [485](#)  
    NBCCCommon.h, [864](#)

STEPICON\_4  
    Display module constants, [485](#)  
    NBCCCommon.h, [864](#)

STEPICON\_5  
    Display module constants, [485](#)  
    NBCCCommon.h, [864](#)

STEPICONS  
    Display module constants, [485](#)  
    NBCCCommon.h, [864](#)

STEPLINE  
    Display module constants, [485](#)  
    NBCCCommon.h, [864](#)

STOP\_REQ  
    NBCCCommon.h, [864](#)  
    VM state constants, [381](#)

STROBE\_READ  
    NBCCCommon.h, [864](#)  
    SuperPro Strobe control constants, [121](#)

STROBE\_S0  
    NBCCCommon.h, [864](#)  
    SuperPro Strobe control constants, [121](#)

STROBE\_S1  
    NBCCCommon.h, [864](#)  
    SuperPro Strobe control constants, [121](#)

STROBE\_S2  
    NBCCCommon.h, [864](#)  
    SuperPro Strobe control constants, [121](#)

STROBE\_S3  
    NBCCCommon.h, [864](#)  
    SuperPro Strobe control constants, [121](#)

STROBE\_WRITE  
    NBCCCommon.h, [864](#)  
    SuperPro Strobe control constants, [121](#)

ScaledValueField  
    Input field constants, [436](#)  
    NBCCCommon.h, [854](#)

Scout constants, [545](#)  
Scout light constants, [546](#)  
    SCOUT\_LIGHT\_OFF, [546](#)

SCOUT\_LIGHT\_ON, 546  
Scout light rule constants, 554  
SCOUT\_LR\_AVOID, 554  
SCOUT\_LR\_IGNORE, 554  
SCOUT\_LR\_OFF\_WHEN, 554  
SCOUT\_LR\_SEEK\_DARK, 554  
SCOUT\_LR\_WAIT\_FOR, 554  
Scout mode constants, 551  
SCOUT\_MODE\_POWER, 551  
Scout motion rule constants, 552  
SCOUT\_MR\_FORWARD, 552  
SCOUT\_MR\_LOOP\_A, 552  
SCOUT\_MR\_LOOP\_AB, 552  
SCOUT\_MR\_LOOP\_B, 552  
SCOUT\_MR\_NO\_MOTION, 552  
SCOUT\_MR\_ZIGZAG, 552  
Scout sound constants, 547  
SCOUT\_SOUND\_1\_BLINK, 547  
SCOUT\_SOUND\_2\_BLINK, 547  
SCOUT\_SOUND\_COUNTER1, 547  
SCOUT\_SOUND\_COUNTER2, 547  
SCOUT\_SOUND\_ENTERSA, 548  
SCOUT\_SOUND\_NONE, 548  
SCOUT\_SOUND\_REMOTE, 548  
SCOUT\_SOUND\_SPECIAL1, 548  
SCOUT\_SOUND\_SPECIAL2, 548  
SCOUT\_SOUND\_SPECIAL3, 548  
SCOUT\_SOUND\_TIMER1, 548  
SCOUT\_SOUND\_TIMER2, 548  
SCOUT\_SOUND\_TIMER3, 548  
Scout sound set constants, 550  
SCOUT SNDSET\_ALARM, 550  
SCOUT SNDSET\_BASIC, 550  
SCOUT SNDSET\_BUG, 550  
SCOUT SNDSET\_NONE, 550  
SCOUT SNDSET\_RANDOM, 550  
Scout special effect constants, 556  
SCOUT\_FXR\_ALARM, 556  
SCOUT\_FXR\_BUG, 556  
SCOUT\_FXR\_NONE, 556  
SCOUT\_FXR\_RANDOM, 556  
SCOUT\_FXR\_SCIENCE, 556  
Scout touch rule constants, 553  
SCOUT\_TR\_AVOID, 553  
SCOUT\_TR\_IGNORE, 553  
SCOUT\_TR\_OFF\_WHEN, 553  
SCOUT\_TR\_REVERSE, 553  
SCOUT\_TR\_WAIT\_FOR, 553  
Scout transmit rule constants, 555  
SCOUT\_TGS\_LONG, 555  
SCOUT\_TGS\_MEDIUM, 555  
SCOUT\_TGS\_SHORT, 555  
SendMessage  
    Comm module functions, 318  
SendRS485Bool  
    Comm module functions, 320  
SendRS485Number  
    Comm module functions, 320  
SendRS485String  
    Comm module functions, 320  
SendRemoteBool  
    Comm module functions, 318  
SendRemoteNumber  
    Comm module functions, 318  
SendRemoteString  
    Comm module functions, 319  
SendResponseBool  
    Comm module functions, 319  
SendResponseNumber  
    Comm module functions, 319  
SendResponseString  
    Comm module functions, 319  
Sensor types and modes, 66  
SetACCLNxSensitivity  
    MindSensors API Functions, 176  
SetAbortFlag  
    Ui module functions, 293  
SetBTDataMode  
    Comm module functions, 321  
SetBTInputBuffer  
    Comm module functions, 321  
SetBTInputBufferInPtr  
    Comm module functions, 321  
SetBTInputBufferOutPtr  
    Comm module functions, 321  
SetBTOutputBuffer  
    Comm module functions, 321  
SetBTOutputBufferInPtr  
    Comm module functions, 322  
SetBTOutputBufferOutPtr  
    Comm module functions, 322  
SetBatteryState  
    Ui module functions, 293  
SetBluetoothState  
    Ui module functions, 293  
SetButtonLongPressCount  
    Button module functions, 287  
SetButtonLongReleaseCount  
    Button module functions, 287  
SetButtonModuleValue  
    Command module functions, 279  
SetButtonPressCount  
    Button module functions, 287  
SetButtonReleaseCount  
    Button module functions, 287  
SetButtonShortReleaseCount  
    Button module functions, 288  
SetButtonState

Button module functions, 288  
SetCommModuleBytes  
    Command module functions, 279  
SetCommModuleValue  
    Command module functions, 280  
SetCommandFlags  
    Ui module functions, 293  
SetCommandModuleBytes  
    Command module functions, 279  
SetCommandModuleValue  
    Command module functions, 279  
SetDisplayContrast  
    Display module functions, 262  
SetDisplayDisplay  
    Display module functions, 262  
SetDisplayEraseMask  
    Display module functions, 262  
SetDisplayFlags  
    Display module functions, 262  
SetDisplayFont  
    Display module functions, 262  
SetDisplayModuleBytes  
    Command module functions, 280  
SetDisplayModuleValue  
    Command module functions, 280  
SetDisplayNormal  
    Display module functions, 263  
SetDisplayPopup  
    Display module functions, 263  
SetDisplayTextLinesCenterFlags  
    Display module functions, 263  
SetDisplayUpdateMask  
    Display module functions, 263  
SetHSAddress  
    Comm module functions, 322  
SetHSDaDataMode  
    Comm module functions, 322  
SetHSFlags  
    Comm module functions, 322  
SetHSInputBuffer  
    Comm module functions, 323  
SetHSInputBufferInPtr  
    Comm module functions, 323  
SetHSInputBufferOutPtr  
    Comm module functions, 323  
SetHSMode  
    Comm module functions, 323  
SetHSOutputBuffer  
    Comm module functions, 323  
SetHSOutputBufferInPtr  
    Comm module functions, 324  
SetHSOutputBufferOutPtr  
    Comm module functions, 324  
SetHSSpeed  
    Comm module functions, 324  
SetHSState  
    Comm module functions, 324  
SetHTBarometricCalibration  
    HiTechnic API Functions, 112  
SetHTColor2Mode  
    HiTechnic API Functions, 112  
SetHTIRSeeker2Mode  
    HiTechnic API Functions, 113  
SetI2COptions  
    LowSpeed module functions, 244  
SetIOCtlModuleValue  
    Command module functions, 281  
SetIOMapBytes  
    Command module functions, 281  
SetIOMapBytesByID  
    Command module functions, 281  
SetIOMapValue  
    Command module functions, 282  
SetIOMapValueByID  
    Command module functions, 282  
SetInCustomActiveStatus  
    Input module functions, 232  
SetInCustomPercentFullScale  
    Input module functions, 233  
SetInCustomZeroOffset  
    Input module functions, 233  
SetInDigiPinsDirection  
    Input module functions, 233  
SetInDigiPinsOutputLevel  
    Input module functions, 233  
SetInDigiPinsStatus  
    Input module functions, 233  
SetInSensorBoolean  
    Input module functions, 234  
SetInputModuleValue  
    Command module functions, 280  
SetLoaderModuleValue  
    Command module functions, 282  
SetLowSpeedModuleBytes  
    Command module functions, 282  
SetLowSpeedModuleValue  
    Command module functions, 283  
SetNXTLineLeaderKdFactor  
    MindSensors API Functions, 176  
SetNXTLineLeaderKdValue  
    MindSensors API Functions, 176  
SetNXTLineLeaderKiFactor  
    MindSensors API Functions, 176  
SetNXTLineLeaderKiValue  
    MindSensors API Functions, 177  
SetNXTLineLeaderKpFactor  
    MindSensors API Functions, 177  
SetNXTLineLeaderKpValue

- MindSensors API Functions, [177](#)
- SetNXTLineLeaderSetpoint
  - MindSensors API Functions, [178](#)
- SetNXTServoPosition
  - MindSensors API Functions, [178](#)
- SetNXTServoQuickPosition
  - MindSensors API Functions, [178](#)
- SetNXTServoSpeed
  - MindSensors API Functions, [179](#)
- SetOnBrickProgramPointer
  - Ui module functions, [293](#)
- SetOutPwnFreq
  - Output module functions, [224](#)
- SetOutRegulationOptions
  - Output module functions, [224](#)
- SetOutRegulationTime
  - Output module functions, [225](#)
- SetOutputModuleValue
  - Command module functions, [283](#)
- SetScreenMode
  - NBCCommon.h, [860](#)
  - System Call function constants, [371](#)
- SetSensorColorBlue
  - Input module functions, [234](#)
- SetSensorColorFull
  - Input module functions, [234](#)
- SetSensorColorGreen
  - Input module functions, [234](#)
- SetSensorColorNone
  - Input module functions, [235](#)
- SetSensorColorRed
  - Input module functions, [235](#)
- SetSensorDIAccl
  - Dexter Industries API Functions, [189](#)
- SetSensorDIAcclDrift
  - Dexter Industries API Functions, [190](#)
- SetSensorDIAcclEx
  - Dexter Industries API Functions, [190](#)
- SetSensorDIGPSWaypoint
  - Dexter Industries API Functions, [190](#)
- SetSensorDIGyro
  - Dexter Industries API Functions, [190](#)
- SetSensorDIGyroEx
  - Dexter Industries API Functions, [191](#)
- SetSensorEMeter
  - Input module functions, [235](#)
- SetSensorHTEOPD
  - HiTechnic API Functions, [113](#)
- SetSensorHTForce
  - HiTechnic API Functions, [113](#)
- SetSensorHTGyro
  - HiTechnic API Functions, [113](#)
- SetSensorHTMagnet
  - HiTechnic API Functions, [113](#)
- SetSensorHTPIRDeadband
  - HiTechnic API Functions, [114](#)
- SetSensorHTProtoDigital
  - HiTechnic API Functions, [114](#)
- SetSensorHTProtoDigitalControl
  - HiTechnic API Functions, [114](#)
- SetSensorHTSuperProAnalogOut
  - HiTechnic API Functions, [114](#)
- SetSensorHTSuperProDigital
  - HiTechnic API Functions, [115](#)
- SetSensorHTSuperProDigitalControl
  - HiTechnic API Functions, [115](#)
- SetSensorHTSuperProLED
  - HiTechnic API Functions, [115](#)
- SetSensorHTSuperProProgramControl
  - HiTechnic API Functions, [115](#)
- SetSensorHTSuperProStrobe
  - HiTechnic API Functions, [116](#)
- SetSensorLight
  - Input module functions, [235](#)
- SetSensorLowspeed
  - Input module functions, [235](#)
- SetSensorMIXG1300LScale
  - Microinfinity API Functions, [193](#)
- SetSensorMSDRODActive
  - MindSensors API Functions, [179](#)
- SetSensorMSDRODInactive
  - MindSensors API Functions, [179](#)
- SetSensorMSPressure
  - MindSensors API Functions, [179](#)
- SetSensorMSTouchMux
  - MindSensors API Functions, [179](#)
- SetSensorMode
  - Input module functions, [236](#)
- SetSensorNXTSumoEyesLong
  - MindSensors API Functions, [180](#)
- SetSensorNXTSumoEyesShort
  - MindSensors API Functions, [180](#)
- SetSensorSound
  - Input module functions, [236](#)
- SetSensorTemperature
  - Input module functions, [236](#)
- SetSensorTouch
  - Input module functions, [236](#)
- SetSensorType
  - Input module functions, [237](#)
- SetSensorUltrasonic
  - Input module functions, [237](#)
- SetSleepTimeout
  - Ui module functions, [294](#)
- SetSleepTimeoutVal
  - NBCCommon.h, [860](#)
  - System Call function constants, [371](#)
- SetSleepTimer
  -

Ui module functions, 294  
SetSoundDuration  
    Sound module functions, 268  
SetSoundFlags  
    Sound module functions, 268  
SetSoundFrequency  
    Sound module functions, 268  
SetSoundMode  
    Sound module functions, 269  
SetSoundModuleBytes  
    Command module functions, 283  
SetSoundModuleState  
    Sound module functions, 269  
SetSoundModuleValue  
    Command module functions, 283  
SetSoundSampleRate  
    Sound module functions, 269  
SetSoundState  
    Sound module functions, 270  
SetSoundVolume  
    Sound module functions, 270  
SetUIButton  
    Ui module functions, 294  
SetUIModuleValue  
    Command module functions, 284  
SetUIState  
    Ui module functions, 294  
SetUSBInputBuffer  
    Comm module functions, 324  
SetUSBInputBufferInPtr  
    Comm module functions, 325  
SetUSBInputBufferOutPtr  
    Comm module functions, 325  
SetUSBOOutputBuffer  
    Comm module functions, 325  
SetUSBOOutputBufferInPtr  
    Comm module functions, 325  
SetUSBOOutputBufferOutPtr  
    Comm module functions, 325  
SetUSBPollBuffer  
    Comm module functions, 325  
SetUSBPollBufferInPtr  
    Comm module functions, 326  
SetUSBPollBufferOutPtr  
    Comm module functions, 326  
SetUSBState  
    Comm module functions, 326  
SetUsbState  
    Ui module functions, 294  
SetVMRunState  
    Ui module functions, 295  
SetVolume  
    Ui module functions, 295  
SignedRandom  
    cstdlib API, 357  
SizeOf  
    Loader module functions, 354  
Sound module, 76  
Sound module constants, 403  
Sound module functions, 265  
    GetSoundDuration, 266  
    GetSoundFrequency, 266  
    GetSoundMode, 266  
    GetSoundSampleRate, 266  
    GetSoundState, 267  
    GetSoundVolume, 267  
    PlayFile, 267  
    PlayFileEx, 267  
    PlayTone, 268  
    PlayToneEx, 268  
    SetSoundDuration, 268  
    SetSoundFlags, 268  
    SetSoundFrequency, 268  
    SetSoundMode, 269  
    SetSoundModuleState, 269  
    SetSoundSampleRate, 269  
    SetSoundState, 270  
    SetSoundVolume, 270  
Sound module IOMAP offsets, 407  
    SoundOffsetDuration, 407  
    SoundOffsetFlags, 407  
    SoundOffsetFreq, 407  
    SoundOffsetMode, 407  
    SoundOffsetSampleRate, 407  
    SoundOffsetSoundFilename, 407  
    SoundOffsetState, 407  
    SoundOffsetVolume, 407  
Sound module miscellaneous constants, 408  
    FREQUENCY\_MAX, 408  
    FREQUENCY\_MIN, 408  
    SAMPLERATE\_DEFAULT, 408  
    SAMPLERATE\_MAX, 408  
    SAMPLERATE\_MIN, 408  
SoundFlags constants, 404  
    SOUND\_FLAGS\_IDLE, 404  
    SOUND\_FLAGS\_RUNNING, 404  
    SOUND\_FLAGS\_UPDATE, 404  
SoundGetState  
    NBCCommon.h, 862  
    System Call function constants, 371  
SoundMode constants, 406  
    SOUND\_MODE\_LOOP, 406  
    SOUND\_MODE\_ONCE, 406  
    SOUND\_MODE\_TONE, 406  
SoundModuleID  
    NBCCommon.h, 862  
    NXT firmware module IDs, 202  
SoundModuleName

NBCCCommon.h, 862  
NXT firmware module names, 200  
SoundOffsetDuration  
  NBCCCommon.h, 862  
  Sound module IOMAP offsets, 407  
SoundOffsetFlags  
  NBCCCommon.h, 862  
  Sound module IOMAP offsets, 407  
SoundOffsetFreq  
  NBCCCommon.h, 862  
  Sound module IOMAP offsets, 407  
SoundOffsetMode  
  NBCCCommon.h, 862  
  Sound module IOMAP offsets, 407  
SoundOffsetSampleRate  
  NBCCCommon.h, 862  
  Sound module IOMAP offsets, 407  
SoundOffsetSoundFilename  
  NBCCCommon.h, 862  
  Sound module IOMAP offsets, 407  
SoundOffsetState  
  NBCCCommon.h, 862  
  Sound module IOMAP offsets, 407  
SoundOffsetVolume  
  NBCCCommon.h, 863  
  Sound module IOMAP offsets, 407  
SoundPlayFile  
  NBCCCommon.h, 863  
  System Call function constants, 371  
SoundPlayTone  
  NBCCCommon.h, 863  
  System Call function constants, 371  
SoundSetState  
  NBCCCommon.h, 863  
  System Call function constants, 371  
SoundState constants, 405  
  SOUND\_STATE\_FILE, 405  
  SOUND\_STATE\_IDLE, 405  
  SOUND\_STATE\_STOP, 405  
  SOUND\_STATE\_TONE, 405  
Standard I2C constants, 474  
  I2C\_ADDR\_DEFAULT, 474  
  I2C\_REG\_CMD, 474  
  I2C\_REG\_DEVICE\_ID, 474  
  I2C\_REG\_VENDOR\_ID, 474  
  I2C\_REG\_VERSION, 474  
Standard-C API functions, 205  
SuperPro analog output mode constants, 117  
  DAC\_MODE\_DCOUT, 117  
  DAC\_MODE\_SINEWAVE, 117  
SuperPro digital pin constants, 120  
  DIGI\_PIN0, 120  
  DIGI\_PIN1, 120  
  DIGI\_PIN2, 120  
  DIGI\_PIN3, 120  
  DIGI\_PIN4, 120  
  DIGI\_PIN5, 120  
  DIGI\_PIN6, 120  
  DIGI\_PIN7, 120  
SuperPro LED control constants, 119  
  LED\_BLUE, 119  
  LED\_NONE, 119  
  LED\_RED, 119  
SuperPro Strobe control constants, 121  
  STROBE\_READ, 121  
  STROBE\_S0, 121  
  STROBE\_S1, 121  
  STROBE\_S2, 121  
  STROBE\_S3, 121  
  STROBE\_WRITE, 121  
System Call function constants, 365  
  ColorSensorRead, 366  
  CommBTCheckStatus, 366  
  CommBTConnection, 366  
  CommBTOnOff, 366  
  CommBTRead, 366  
  CommBTWrite, 366  
  CommExecuteFunction, 367  
  CommHSCheckStatus, 367  
  CommHSControl, 367  
  CommHSRead, 367  
  CommHSWrite, 367  
  CommLSCheckStatus, 367  
  CommLSRead, 367  
  CommLSWrite, 367  
  CommLSWriteEx, 367  
  ComputeCalibValue, 367  
  DatalogGetTimes, 367  
  DatalogWrite, 367  
  DisplayExecuteFunction, 367  
  DrawCircle, 367  
  DrawEllipse, 368  
  DrawFont, 368  
  DrawGraphic, 368  
  DrawGraphicArray, 368  
  DrawLine, 368  
  DrawPoint, 368  
  DrawPolygon, 368  
  DrawRect, 368  
  DrawText, 368  
  FileClose, 368  
  FileDelete, 368  
  FileFindFirst, 368  
  FileFindNext, 368  
  FileOpenAppend, 368  
  FileOpenRead, 369  
  FileOpenReadLinear, 369  
  FileOpenWrite, 369

FileOpenWriteLinear, 369  
FileOpenWriteNonLinear, 369  
FileRead, 369  
FileRename, 369  
FileResize, 369  
FileResolveHandle, 369  
FileSeek, 369  
FileTell, 369  
FileWrite, 369  
GetStartTick, 369  
IOMapRead, 370  
IOMapReadByID, 370  
IOMapWrite, 370  
IOMapWriteByID, 370  
InputPinFunction, 369  
KeepAlive, 370  
ListFiles, 370  
LoaderExecuteFunction, 370  
MemoryManager, 370  
MessageRead, 370  
MessageWrite, 370  
RandomEx, 370  
RandomNumber, 370  
ReadButton, 370  
ReadLastResponse, 370  
ReadSemData, 371  
SetScreenMode, 371  
SetSleepTimeoutVal, 371  
SoundGetState, 371  
SoundPlayFile, 371  
SoundPlayTone, 371  
SoundSetState, 371  
UpdateCalibCacheInfo, 371  
WriteSemData, 371  
System Command functions, 337  
  RemoteBluetoothFactoryReset, 338  
  RemoteCloseFile, 338  
  RemoteDeleteFile, 339  
  RemoteDeleteUserFlash, 339  
  RemoteFindFirstFile, 339  
  RemoteFindNextFile, 340  
  RemoteGetBluetoothAddress, 340  
  RemoteGetDeviceInfo, 340  
  RemoteGetFirmwareVersion, 341  
  RemoteIOMapRead, 341  
  RemoteIOMapWriteBytes, 341  
  RemoteIOMapWriteValue, 342  
  RemoteOpenAppendData, 342  
  RemoteOpenRead, 342  
  RemoteOpenWrite, 343  
  RemoteOpenWriteData, 343  
  RemoteOpenWriteLinear, 344  
  RemotePollCommand, 344  
  RemotePollCommandLength, 344  
    RemoteRead, 344  
    RemoteRenameFile, 345  
    RemoteSetBrickName, 345  
    RemoteWrite, 345  
  
TEMP\_FQ\_1  
  LEGO temperature sensor constants, 478  
  NBCCommon.h, 865  
  
TEMP\_FQ\_2  
  LEGO temperature sensor constants, 478  
  NBCCommon.h, 865  
  
TEMP\_FQ\_4  
  LEGO temperature sensor constants, 478  
  NBCCommon.h, 865  
  
TEMP\_FQ\_6  
  LEGO temperature sensor constants, 478  
  NBCCommon.h, 865  
  
TEMP\_OS\_ONESHOT  
  LEGO temperature sensor constants, 478  
  NBCCommon.h, 865  
  
TEMP\_POL\_HIGH  
  LEGO temperature sensor constants, 478  
  NBCCommon.h, 865  
  
TEMP\_POL\_LOW  
  LEGO temperature sensor constants, 479  
  NBCCommon.h, 865  
  
TEMP\_REG\_CONFIG  
  LEGO temperature sensor constants, 479  
  NBCCommon.h, 865  
  
TEMP\_REG\_TEMP  
  LEGO temperature sensor constants, 479  
  NBCCommon.h, 865  
  
TEMP\_REG\_THIGH  
  LEGO temperature sensor constants, 479  
  NBCCommon.h, 865  
  
TEMP\_REG\_TLOW  
  LEGO temperature sensor constants, 479  
  NBCCommon.h, 865  
  
TEMP\_RES\_10BIT  
  LEGO temperature sensor constants, 479  
  NBCCommon.h, 866  
  
TEMP\_RES\_11BIT  
  LEGO temperature sensor constants, 479  
  NBCCommon.h, 866  
  
TEMP\_RES\_12BIT  
  LEGO temperature sensor constants, 479  
  NBCCommon.h, 866  
  
TEMP\_RES\_9BIT  
  LEGO temperature sensor constants, 479  
  NBCCommon.h, 866  
  
TEMP\_SD\_CONTINUOUS  
  NBCCommon.h, 866  
  
TEMP\_SD\_SHUTDOWN  
  LEGO temperature sensor constants, 479

NBCCCommon.h, 866  
TEMP\_TM\_COMPARATOR  
  NBCCCommon.h, 866  
TEMP\_TM\_INTERRUPT  
  LEGO temperature sensor constants, 479  
  NBCCCommon.h, 866  
TEXTLINE\_1  
  NBCCCommon.h, 866  
  Text line constants, 496  
TEXTLINE\_2  
  NBCCCommon.h, 866  
  Text line constants, 496  
TEXTLINE\_3  
  NBCCCommon.h, 866  
  Text line constants, 496  
TEXTLINE\_4  
  NBCCCommon.h, 866  
  Text line constants, 496  
TEXTLINE\_5  
  NBCCCommon.h, 866  
  Text line constants, 496  
TEXTLINE\_6  
  NBCCCommon.h, 866  
  Text line constants, 496  
TEXTLINE\_7  
  NBCCCommon.h, 867  
  Text line constants, 496  
TEXTLINE\_8  
  NBCCCommon.h, 867  
  Text line constants, 496  
TEXTLINES  
  NBCCCommon.h, 867  
  Text line constants, 497  
TIMES\_UP  
  NBCCCommon.h, 867  
  VM state constants, 381  
TONE\_A3  
  NBCCCommon.h, 867  
  Tone constants, 410  
TONE\_A4  
  NBCCCommon.h, 867  
  Tone constants, 410  
TONE\_A5  
  NBCCCommon.h, 867  
  Tone constants, 410  
TONE\_A6  
  NBCCCommon.h, 867  
  Tone constants, 410  
TONE\_A7  
  NBCCCommon.h, 867  
  Tone constants, 410  
TONE\_AS3  
  NBCCCommon.h, 867  
  Tone constants, 410  
TONE\_AS4  
  NBCCCommon.h, 867  
  Tone constants, 410  
TONE\_AS5  
  NBCCCommon.h, 867  
  Tone constants, 410  
TONE\_AS6  
  NBCCCommon.h, 867  
  Tone constants, 411  
TONE\_AS7  
  NBCCCommon.h, 867  
  Tone constants, 411  
TONE\_B3  
  NBCCCommon.h, 868  
  Tone constants, 411  
TONE\_B4  
  NBCCCommon.h, 868  
  Tone constants, 411  
TONE\_B5  
  NBCCCommon.h, 868  
  Tone constants, 411  
TONE\_B6  
  NBCCCommon.h, 868  
  Tone constants, 411  
TONE\_B7  
  NBCCCommon.h, 868  
  Tone constants, 411  
TONE\_C3  
  NBCCCommon.h, 868  
  Tone constants, 411  
TONE\_C4  
  NBCCCommon.h, 868  
  Tone constants, 411  
TONE\_C5  
  NBCCCommon.h, 868  
  Tone constants, 411  
TONE\_C6  
  NBCCCommon.h, 868  
  Tone constants, 411  
TONE\_C7  
  NBCCCommon.h, 868  
  Tone constants, 411  
TONE\_CS3  
  NBCCCommon.h, 868  
  Tone constants, 411  
TONE\_CS4  
  NBCCCommon.h, 868  
  Tone constants, 411  
TONE\_CS5  
  NBCCCommon.h, 868  
  Tone constants, 412  
TONE\_CS6  
  NBCCCommon.h, 868  
  Tone constants, 412

- TONE\_CS7  
    NBCCCommon.h, 869  
    Tone constants, 412
- TONE\_D3  
    NBCCCommon.h, 869  
    Tone constants, 412
- TONE\_D4  
    NBCCCommon.h, 869  
    Tone constants, 412
- TONE\_D5  
    NBCCCommon.h, 869  
    Tone constants, 412
- TONE\_D6  
    NBCCCommon.h, 869  
    Tone constants, 412
- TONE\_D7  
    NBCCCommon.h, 869  
    Tone constants, 412
- TONE\_DS3  
    NBCCCommon.h, 869  
    Tone constants, 412
- TONE\_DS4  
    NBCCCommon.h, 869  
    Tone constants, 412
- TONE\_DS5  
    NBCCCommon.h, 869  
    Tone constants, 412
- TONE\_DS6  
    NBCCCommon.h, 869  
    Tone constants, 412
- TONE\_DS7  
    NBCCCommon.h, 869  
    Tone constants, 412
- TONE\_E3  
    NBCCCommon.h, 869  
    Tone constants, 412
- TONE\_E4  
    NBCCCommon.h, 869  
    Tone constants, 413
- TONE\_E5  
    NBCCCommon.h, 869  
    Tone constants, 413
- TONE\_E6  
    NBCCCommon.h, 870  
    Tone constants, 413
- TONE\_E7  
    NBCCCommon.h, 870  
    Tone constants, 413
- TONE\_F3  
    NBCCCommon.h, 870  
    Tone constants, 413
- TONE\_F4  
    NBCCCommon.h, 870  
    Tone constants, 413
- TONE\_F5  
    NBCCCommon.h, 870  
    Tone constants, 413
- TONE\_F6  
    NBCCCommon.h, 870  
    Tone constants, 413
- TONE\_F7  
    NBCCCommon.h, 870  
    Tone constants, 413
- TONE\_FS3  
    NBCCCommon.h, 870  
    Tone constants, 413
- TONE\_FS4  
    NBCCCommon.h, 870  
    Tone constants, 413
- TONE\_FS5  
    NBCCCommon.h, 870  
    Tone constants, 413
- TONE\_FS6  
    NBCCCommon.h, 870  
    Tone constants, 413
- TONE\_FS7  
    NBCCCommon.h, 870  
    Tone constants, 413
- TONE\_G3  
    NBCCCommon.h, 870  
    Tone constants, 414
- TONE\_G4  
    NBCCCommon.h, 870  
    Tone constants, 414
- TONE\_G5  
    NBCCCommon.h, 871  
    Tone constants, 414
- TONE\_G6  
    NBCCCommon.h, 871  
    Tone constants, 414
- TONE\_G7  
    NBCCCommon.h, 871  
    Tone constants, 414
- TONE\_GS3  
    NBCCCommon.h, 871  
    Tone constants, 414
- TONE\_GS4  
    NBCCCommon.h, 871  
    Tone constants, 414
- TONE\_GS5  
    NBCCCommon.h, 871  
    Tone constants, 414
- TONE\_GS6  
    NBCCCommon.h, 871  
    Tone constants, 414
- TONE\_GS7  
    NBCCCommon.h, 871  
    Tone constants, 414

TOPLINE  
    Display module constants, 486  
    NBCCommon.h, 871

TRAIN\_CHANNEL\_1  
    IR Train channel constants, 574  
    NBCCommon.h, 871

TRAIN\_CHANNEL\_2  
    IR Train channel constants, 574  
    NBCCommon.h, 871

TRAIN\_CHANNEL\_3  
    IR Train channel constants, 574  
    NBCCommon.h, 871

TRAIN\_CHANNEL\_ALL  
    IR Train channel constants, 574  
    NBCCommon.h, 871

TRAIN\_FUNC\_STOP  
    NBCCommon.h, 872  
    PF/IR Train function constants, 573

TRUE  
    Miscellaneous NBC/NXC constants, 203  
    NBCCommon.h, 872

TachoCountField  
    NBCCommon.h, 864  
    Output field constants, 460

TachoLimitField  
    NBCCommon.h, 865  
    Output field constants, 460

Tachometer counter reset flags, 451  
    RESET\_ALL, 451  
    RESET\_BLOCK\_COUNT, 451  
    RESET\_BLOCKANDTACHO, 451  
    RESET\_COUNT, 451  
    RESET\_NONE, 451

Text line constants, 496  
    TEXTLINE\_1, 496  
    TEXTLINE\_2, 496  
    TEXTLINE\_3, 496  
    TEXTLINE\_4, 496  
    TEXTLINE\_5, 496  
    TEXTLINE\_6, 496  
    TEXTLINE\_7, 496  
    TEXTLINE\_8, 496  
    TEXTLINES, 497

TextOut  
    Display module functions, 263

TextOutEx  
    Display module functions, 264

Third-party NXT devices, 204

Time constants, 374  
    MIN\_1, 375  
    MS\_1, 375  
    MS\_10, 375  
    MS\_100, 375  
    MS\_150, 375

    MS\_2, 375  
    MS\_20, 375  
    MS\_200, 375  
    MS\_250, 375  
    MS\_3, 375  
    MS\_30, 375  
    MS\_300, 376  
    MS\_350, 376  
    MS\_4, 376  
    MS\_40, 376  
    MS\_400, 376  
    MS\_450, 376  
    MS\_5, 376  
    MS\_50, 376  
    MS\_500, 376  
    MS\_6, 376  
    MS\_60, 376  
    MS\_600, 376  
    MS\_7, 376  
    MS\_70, 376  
    MS\_700, 377  
    MS\_8, 377  
    MS\_80, 377  
    MS\_800, 377  
    MS\_9, 377  
    MS\_90, 377  
    MS\_900, 377  
    NOTE\_EIGHT, 377  
    NOTE\_HALF, 377  
    NOTE\_QUARTER, 377  
    NOTE\_SIXTEEN, 377  
    NOTE\_WHOLE, 377  
    SEC\_1, 377  
    SEC\_10, 377  
    SEC\_15, 378  
    SEC\_2, 378  
    SEC\_20, 378  
    SEC\_3, 378  
    SEC\_30, 378  
    SEC\_4, 378  
    SEC\_5, 378  
    SEC\_6, 378  
    SEC\_7, 378  
    SEC\_8, 378  
    SEC\_9, 378

Tone constants, 409  
    TONE\_A3, 410  
    TONE\_A4, 410  
    TONE\_A5, 410  
    TONE\_A6, 410  
    TONE\_A7, 410  
    TONE\_AS3, 410  
    TONE\_AS4, 410  
    TONE\_AS5, 410

TONE\_AS6, 411  
TONE\_AS7, 411  
TONE\_B3, 411  
TONE\_B4, 411  
TONE\_B5, 411  
TONE\_B6, 411  
TONE\_B7, 411  
TONE\_C3, 411  
TONE\_C4, 411  
TONE\_C5, 411  
TONE\_C6, 411  
TONE\_C7, 411  
TONE\_CS3, 411  
TONE\_CS4, 411  
TONE\_CS5, 412  
TONE\_CS6, 412  
TONE\_CS7, 412  
TONE\_D3, 412  
TONE\_D4, 412  
TONE\_D5, 412  
TONE\_D6, 412  
TONE\_D7, 412  
TONE\_DS3, 412  
TONE\_DS4, 412  
TONE\_DS5, 412  
TONE\_DS6, 412  
TONE\_DS7, 412  
TONE\_E3, 412  
TONE\_E4, 413  
TONE\_E5, 413  
TONE\_E6, 413  
TONE\_E7, 413  
TONE\_F3, 413  
TONE\_F4, 413  
TONE\_F5, 413  
TONE\_F6, 413  
TONE\_F7, 413  
TONE\_FS3, 413  
TONE\_FS4, 413  
TONE\_FS5, 413  
TONE\_FS6, 413  
TONE\_FS7, 413  
TONE\_G3, 414  
TONE\_G4, 414  
TONE\_G5, 414  
TONE\_G6, 414  
TONE\_G7, 414  
TONE\_GS3, 414  
TONE\_GS4, 414  
TONE\_GS5, 414  
TONE\_GS6, 414  
TONE\_GS7, 414  
TurnRatioField  
    NBCCommon.h, 872  
Output field constants, 460  
TypeField  
    Input field constants, 436  
    NBCCommon.h, 872  
UCHAR\_MAX  
    Data type limits, 688  
    NBCCommon.h, 872  
UF\_PENDING\_UPDATES  
    NBCCommon.h, 872  
    Output port update flag constants, 449  
UF\_UPDATE\_MODE  
    NBCCommon.h, 872  
    Output port update flag constants, 449  
UF\_UPDATE\_SPEED  
    NBCCommon.h, 873  
    Output port update flag constants, 449  
UI\_BT\_PIN\_REQUEST  
    BluetoothState constants, 426  
    NBCCommon.h, 873  
UI\_BT\_STATE\_OFF  
    BluetoothState constants, 426  
    NBCCommon.h, 873  
UI\_BUTTON\_ENTER  
    NBCCommon.h, 873  
    UIButton constants, 425  
UI\_BUTTON\_EXIT  
    NBCCommon.h, 873  
    UIButton constants, 425  
UI\_BUTTON\_LEFT  
    NBCCommon.h, 873  
    UIButton constants, 425  
UI\_BUTTON\_NONE  
    NBCCommon.h, 873  
    UIButton constants, 425  
UI\_BUTTON\_RIGHT  
    NBCCommon.h, 873  
    UIButton constants, 425  
UI\_FLAGS\_BUSY  
    CommandFlags constants, 421  
    NBCCommon.h, 874  
UI\_FLAGS\_UPDATE  
    CommandFlags constants, 421  
    NBCCommon.h, 874  
UI\_STATE\_BT\_ERROR  
    NBCCommon.h, 874  
    UIState constants, 423  
UI\_STATE\_DRAW\_MENU  
    UIState constants, 423  
UI\_STATE\_INIT\_MENU  
    UIState constants, 424  
UI\_STATE\_INIT\_WAIT  
    UIState constants, 424  
UI\_STATE\_NEXT\_MENU

UIState constants, [424](#)  
UI\_VM\_IDLE  
    NBCCommon.h, [875](#)  
    VM run state constants, [427](#)  
UI\_VM\_RESET1  
    NBCCommon.h, [875](#)  
    VM run state constants, [427](#)  
UI\_VM\_RESET2  
    NBCCommon.h, [875](#)  
    VM run state constants, [427](#)  
UI\_VM\_RUN\_FREE  
    NBCCommon.h, [876](#)  
    VM run state constants, [427](#)  
UI\_VM\_RUN\_PAUSE  
    NBCCommon.h, [876](#)  
    VM run state constants, [427](#)  
UI\_VM\_RUN\_SINGLE  
    NBCCommon.h, [876](#)  
    VM run state constants, [427](#)  
UIButton constants, [425](#)  
    UI\_BUTTON\_ENTER, [425](#)  
    UI\_BUTTON\_EXIT, [425](#)  
    UI\_BUTTON\_LEFT, [425](#)  
    UI\_BUTTON\_NONE, [425](#)  
    UI\_BUTTON\_RIGHT, [425](#)  
UIModuleID  
    NBCCommon.h, [876](#)  
    NXT firmware module IDs, [202](#)  
UIModuleName  
    NBCCommon.h, [876](#)  
    NXT firmware module names, [200](#)  
UINT\_MAX  
    Data type limits, [688](#)  
    NBCCommon.h, [876](#)  
UIOffsetAbortFlag  
    NBCCommon.h, [876](#)  
    Ui module IOMAP offsets, [428](#)  
UIOffsetBatteryState  
    NBCCommon.h, [876](#)  
    Ui module IOMAP offsets, [428](#)  
UIOffsetBatteryVoltage  
    NBCCommon.h, [876](#)  
    Ui module IOMAP offsets, [428](#)  
UIOffsetBluetoothState  
    NBCCommon.h, [876](#)  
    Ui module IOMAP offsets, [428](#)  
UIOffsetButton  
    NBCCommon.h, [876](#)  
    Ui module IOMAP offsets, [428](#)  
UIOffsetError  
    NBCCommon.h, [876](#)  
    Ui module IOMAP offsets, [428](#)  
UIOffsetFlags  
    NBCCommon.h, [876](#)  
Ui module IOMAP offsets, [429](#)  
UIOffsetForceOff  
    NBCCommon.h, [876](#)  
    Ui module IOMAP offsets, [429](#)  
UIOffsetLMSfilename  
    NBCCommon.h, [877](#)  
    Ui module IOMAP offsets, [429](#)  
UIOffsetOBPPointer  
    NBCCommon.h, [877](#)  
    Ui module IOMAP offsets, [429](#)  
UIOffsetPMenu  
    NBCCommon.h, [877](#)  
    Ui module IOMAP offsets, [429](#)  
UIOffsetRechargeable  
    NBCCommon.h, [877](#)  
    Ui module IOMAP offsets, [429](#)  
UIOffsetRunState  
    NBCCommon.h, [877](#)  
    Ui module IOMAP offsets, [429](#)  
UIOffsetSleepTimeout  
    NBCCommon.h, [877](#)  
    Ui module IOMAP offsets, [429](#)  
UIOffsetSleepTimer  
    NBCCommon.h, [877](#)  
    Ui module IOMAP offsets, [429](#)  
UIOffsetState  
    NBCCommon.h, [877](#)  
    Ui module IOMAP offsets, [429](#)  
UIOffsetUsbState  
    NBCCommon.h, [877](#)  
    Ui module IOMAP offsets, [429](#)  
UIOffsetVolume  
    NBCCommon.h, [877](#)  
    Ui module IOMAP offsets, [429](#)  
UIState constants, [423](#)  
    UI\_STATE\_BT\_ERROR, [423](#)  
ULONG\_MAX  
    Data type limits, [688](#)  
    NBCCommon.h, [877](#)  
US\_CMD\_CONTINUOUS  
    NBCCommon.h, [878](#)  
    Ultrasonic sensor constants, [476](#)  
US\_CMD\_EVENTCAPTURE  
    Ultrasonic sensor constants, [476](#)  
US\_CMD\_OFF  
    NBCCommon.h, [878](#)  
    Ultrasonic sensor constants, [476](#)  
US\_CMD\_SINGLESHT  
    NBCCommon.h, [878](#)  
    Ultrasonic sensor constants, [476](#)  
US\_CMD\_WARMRESET  
    NBCCommon.h, [878](#)  
    Ultrasonic sensor constants, [476](#)  
US\_REG\_ACTUAL\_ZERO

Ultrasonic sensor constants, 476  
US\_REG\_CM\_INTERVAL  
    Ultrasonic sensor constants, 476  
USB\_CMD\_READY  
    Comm module status code constants, 531  
    NBCCommon.h, 878  
USHRT\_MAX  
    Data type limits, 688  
    NBCCommon.h, 879  
Ui module, 77  
Ui module constants, 420  
Ui module functions, 289  
    ForceOff, 290  
    GetAbortFlag, 290  
    GetBatteryLevel, 290  
    GetBatteryState, 291  
    GetBluetoothState, 291  
    GetCommandFlags, 291  
    GetOnBrickProgramPointer, 291  
    GetRechargeableBattery, 291  
    GetSleepTimeout, 291  
    GetSleepTimer, 292  
    GetUIButton, 292  
    GetUIState, 292  
    GetUsbState, 292  
    GetVMRunState, 292  
    GetVolume, 292  
    SetAbortFlag, 293  
    SetBatteryState, 293  
    SetBluetoothState, 293  
    SetCommandFlags, 293  
    SetOnBrickProgramPointer, 293  
    SetSleepTimeout, 294  
    SetSleepTimer, 294  
    SetUIButton, 294  
    SetUIState, 294  
    SetUsbState, 294  
    SetVMRunState, 295  
    SetVolume, 295  
Ui module IOMAP offsets, 428  
    UIOffsetAbortFlag, 428  
    UIOffsetBatteryState, 428  
    UIOffsetBatteryVoltage, 428  
    UIOffsetBluetoothState, 428  
    UIOffsetButton, 428  
    UIOffsetError, 428  
    UIOffsetFlags, 429  
    UIOffsetForceOff, 429  
    UIOffsetLMSfilename, 429  
    UIOffsetOBPPointer, 429  
    UIOffsetPMenu, 429  
    UIOffsetRechargeable, 429  
    UIOffsetRunState, 429  
    UIOffsetSleepTimeout, 429  
    UIOffsetSleepTimer, 429  
    UIOffsetState, 429  
    UIOffsetUsbState, 429  
    UIOffsetVolume, 429  
Ultrasonic sensor constants, 476  
US\_CMD\_CONTINUOUS, 476  
US\_CMD\_EVENTCAPTURE, 476  
US\_CMD\_OFF, 476  
US\_CMD\_SINGLESHT, 476  
US\_CMD\_WARMRESET, 476  
US\_REG\_ACTUAL\_ZERO, 476  
US\_REG\_CM\_INTERVAL, 476  
UpdateCalibCacheInfo  
    NBCCommon.h, 877  
    System Call function constants, 371  
UpdateFlagsField  
    NBCCommon.h, 877  
    Output field constants, 461  
UseRS485  
    Comm module functions, 326  
VM run state constants, 427  
    UI\_VM\_IDLE, 427  
    UI\_VM\_RESET1, 427  
    UI\_VM\_RESET2, 427  
    UI\_VM\_RUN\_FREE, 427  
    UI\_VM\_RUN\_PAUSE, 427  
    UI\_VM\_RUN\_SINGLE, 427  
VM state constants, 381  
    BREAKOUT\_REQ, 381  
    CLUMP\_DONE, 381  
    CLUMP\_SUSPEND, 381  
    ROTATE\_QUEUE, 381  
    STOP\_REQ, 381  
    TIMES\_UP, 381  
VT\_A1\_FLOAT  
    NBCCommon.h, 879  
    Variable type constants, 360  
VT\_A1\_SBYTE  
    NBCCommon.h, 879  
    Variable type constants, 360  
VT\_A1\_SLONG  
    NBCCommon.h, 879  
    Variable type constants, 360  
VT\_A1\_STRUCT  
    NBCCommon.h, 879  
    Variable type constants, 360  
VT\_A1\_SWORD  
    NBCCommon.h, 879  
    Variable type constants, 361  
VT\_A1\_UBYTE  
    NBCCommon.h, 879  
    Variable type constants, 361  
VT\_A1 ULONG

NBCCCommon.h, 879  
Variable type constants, 361

VT\_A1\_UWORD  
NBCCCommon.h, 879  
Variable type constants, 361

VT\_A2\_FLOAT  
NBCCCommon.h, 879  
Variable type constants, 361

VT\_A2\_SBYTE  
NBCCCommon.h, 879  
Variable type constants, 361

VT\_A2\_SLONG  
NBCCCommon.h, 879  
Variable type constants, 361

VT\_A2\_STRUCT  
NBCCCommon.h, 879  
Variable type constants, 361

VT\_A2\_SWORD  
NBCCCommon.h, 879  
Variable type constants, 361

VT\_A2\_UBYTE  
NBCCCommon.h, 880  
Variable type constants, 361

VT\_A2 ULONG  
NBCCCommon.h, 880  
Variable type constants, 361

VT\_A2\_UWORD  
NBCCCommon.h, 880  
Variable type constants, 361

VT\_ARRAY\_MASK  
NBCCCommon.h, 880  
Variable type constants, 361

VT\_FLOAT  
NBCCCommon.h, 880  
Variable type constants, 361

VT\_MUTEX  
NBCCCommon.h, 880  
Variable type constants, 362

VT\_SBYTE  
NBCCCommon.h, 880  
Variable type constants, 362

VT\_SLONG  
NBCCCommon.h, 880  
Variable type constants, 362

VT\_STRUCT  
NBCCCommon.h, 880  
Variable type constants, 362

VT\_SWORD  
NBCCCommon.h, 880  
Variable type constants, 362

VT\_UBYTE  
NBCCCommon.h, 880  
Variable type constants, 362

VT ULONG

NBCCCommon.h, 880  
Variable type constants, 362

VT\_UWORD  
NBCCCommon.h, 880  
Variable type constants, 362

VT\_A1\_FLOAT, 360  
VT\_A1\_SBYTE, 360  
VT\_A1\_SLONG, 360  
VT\_A1\_STRUCT, 360  
VT\_A1\_SWORD, 361  
VT\_A1\_UBYTE, 361  
VT\_A1 ULONG, 361  
VT\_A1\_UWORD, 361  
VT\_A2\_FLOAT, 361  
VT\_A2\_SBYTE, 361  
VT\_A2\_SLONG, 361  
VT\_A2\_STRUCT, 361  
VT\_A2\_SWORD, 361  
VT\_A2\_UBYTE, 361  
VT\_A2 ULONG, 361  
VT\_A2\_UWORD, 361  
VT\_ARRAY\_MASK, 361  
VT\_FLOAT, 361  
VT\_MUTEX, 362  
VT\_SBYTE, 362  
VT\_SLONG, 362  
VT\_STRUCT, 362  
VT\_SWORD, 362  
VT\_UBYTE, 362  
VT ULONG, 362  
VT\_UWORD, 362

Wait  
Command module functions, 284

Write  
Loader module functions, 354

WriteBytes  
Loader module functions, 354

WriteBytesEx  
Loader module functions, 355

Writel2CRegister  
LowSpeed module functions, 245

WriteLn  
Loader module functions, 355

WriteLnString  
Loader module functions, 355

WriteNRLLinkBytes  
MindSensors API Functions, 180

WriteSemData  
NBCCCommon.h, 880  
System Call function constants, 371

WriteString  
Loader module functions, 355

XG1300L\_REG\_2G  
Microinfinity CruzCore XG1300L sensor constants,  
[684](#)  
NBCCommon.h, [881](#)

XG1300L\_REG\_4G  
Microinfinity CruzCore XG1300L sensor constants,  
[684](#)  
NBCCommon.h, [881](#)

XG1300L\_REG\_8G  
Microinfinity CruzCore XG1300L sensor constants,  
[684](#)  
NBCCommon.h, [881](#)

XG1300L\_REG\_ANGLE  
Microinfinity CruzCore XG1300L sensor constants,  
[684](#)  
NBCCommon.h, [881](#)

XG1300L\_REG\_RESET  
Microinfinity CruzCore XG1300L sensor constants,  
[684](#)  
NBCCommon.h, [881](#)

XG1300L\_REG\_TURNRATE  
Microinfinity CruzCore XG1300L sensor constants,  
[684](#)  
NBCCommon.h, [881](#)

XG1300L\_REG\_XAXIS  
Microinfinity CruzCore XG1300L sensor constants,  
[685](#)  
NBCCommon.h, [881](#)

XG1300L\_REG\_YAXIS  
Microinfinity CruzCore XG1300L sensor constants,  
[685](#)  
NBCCommon.h, [881](#)

XG1300L\_REG\_ZAXIS  
Microinfinity CruzCore XG1300L sensor constants,  
[685](#)  
NBCCommon.h, [881](#)

XG1300L\_SCALE\_2G  
Microinfinity CruzCore XG1300L, [686](#)  
NBCCommon.h, [881](#)

XG1300L\_SCALE\_4G  
Microinfinity CruzCore XG1300L, [686](#)  
NBCCommon.h, [881](#)

XG1300L\_SCALE\_8G  
Microinfinity CruzCore XG1300L, [686](#)  
NBCCommon.h, [881](#)