

NBC

Version 1.2.1 r5

Generated by Doxygen 1.6.2

Mon Oct 17 09:52:27 2011

Contents

1 NBC Programmer's Guide	1
2 Introduction	2
3 The NBC Language	2
3.1 Lexical Rules	3
3.1.1 Comments	3
3.1.2 Whitespace	4
3.1.3 Numerical Constants	4
3.1.4 String Constants	4
3.1.5 Identifiers and Keywords	5
3.2 Program Structure	6
3.2.1 Threads	6
3.2.2 Subroutines	7
3.2.3 Macro Functions	9
3.2.4 Data Segments	10
3.3 The Preprocessor	17
3.3.1 #include	18
3.3.2 #define	18
3.3.3 ## (Concatenation)	18
3.3.4 Conditional Compilation	19
3.3.5 #import	20
3.3.6 #download	20
3.4 Compiler Tokens	20
3.5 Expression Evaluator	21
3.6 Statements	22
3.6.1 Assignment Statements	23
3.6.2 Math Statements	24
3.6.3 Logic Statements	34
3.6.4 Bit Manipulation Statements	36

3.6.5	Comparison Statements	38
3.6.6	Control Flow Statements	39
3.6.7	syscall	40
3.6.8	Timing Statements	58
3.6.9	Array Statements	59
3.6.10	String Statements	62
3.6.11	Scheduling Statements	65
3.6.12	Input Statements	68
3.6.13	Output Statements	69
3.6.14	Compile-time Statements	70
4	TXGPacket	73
5	Module Documentation	73
5.1	Comparison Constants	73
5.1.1	Detailed Description	73
5.1.2	Define Documentation	73
5.2	NXT Firmware Modules	74
5.2.1	Detailed Description	75
5.3	Input module	75
5.3.1	Detailed Description	76
5.4	Input module constants	76
5.4.1	Detailed Description	77
5.4.2	Define Documentation	77
5.5	Sensor types and modes	77
5.5.1	Detailed Description	78
5.6	Output module	79
5.6.1	Detailed Description	79
5.7	Output module constants	79
5.7.1	Detailed Description	80
5.8	Command module	80
5.8.1	Detailed Description	81

5.9	Command module constants	81
5.9.1	Detailed Description	82
5.9.2	Define Documentation	82
5.10	Comm module	83
5.10.1	Detailed Description	83
5.11	Button module	84
5.11.1	Detailed Description	84
5.12	IOCctrl module	84
5.12.1	Detailed Description	84
5.13	Loader module	84
5.13.1	Detailed Description	85
5.14	Sound module	85
5.14.1	Detailed Description	85
5.15	Ui module	86
5.15.1	Detailed Description	86
5.16	Low Speed module	86
5.16.1	Detailed Description	87
5.17	Display module	88
5.17.1	Detailed Description	88
5.18	HiTechnic API Functions	88
5.18.1	Detailed Description	101
5.18.2	Define Documentation	101
5.19	SuperPro analog output mode constants	142
5.19.1	Detailed Description	143
5.19.2	Define Documentation	143
5.20	SuperPro LED control constants	144
5.20.1	Detailed Description	144
5.20.2	Define Documentation	144
5.21	SuperPro digital pin constants	145
5.21.1	Detailed Description	145
5.21.2	Define Documentation	145

5.22	SuperPro Strobe control constants	146
5.22.1	Detailed Description	146
5.22.2	Define Documentation	147
5.23	MindSensors API Functions	147
5.23.1	Detailed Description	167
5.23.2	Define Documentation	167
5.24	Codatex API Functions	236
5.24.1	Detailed Description	237
5.24.2	Define Documentation	237
5.25	Dexter Industries API Functions	239
5.25.1	Detailed Description	242
5.25.2	Define Documentation	242
5.26	Microinfinity API Functions	251
5.26.1	Detailed Description	252
5.26.2	Define Documentation	252
5.27	RIC Macro Wrappers	253
5.27.1	Detailed Description	256
5.27.2	Define Documentation	256
5.28	NXT firmware module names	261
5.28.1	Detailed Description	262
5.28.2	Define Documentation	262
5.29	NXT firmware module IDs	263
5.29.1	Detailed Description	264
5.29.2	Define Documentation	264
5.30	Miscellaneous NBC/NXC constants	265
5.30.1	Detailed Description	265
5.30.2	Define Documentation	266
5.31	Third-party NXT devices	266
5.31.1	Detailed Description	267
5.32	Standard-C API functions	267
5.32.1	Detailed Description	267

5.33 A simple 3D graphics library	268
5.33.1 Detailed Description	270
5.33.2 Define Documentation	270
5.34 Output module functions	276
5.34.1 Detailed Description	279
5.34.2 Define Documentation	279
5.35 Input module functions	296
5.35.1 Detailed Description	300
5.35.2 Define Documentation	300
5.36 LowSpeed module functions	313
5.36.1 Detailed Description	315
5.36.2 Define Documentation	315
5.37 Low level LowSpeed module functions	324
5.37.1 Detailed Description	325
5.37.2 Define Documentation	325
5.38 Display module functions	329
5.38.1 Detailed Description	333
5.38.2 Define Documentation	333
5.39 Sound module functions	349
5.39.1 Detailed Description	351
5.39.2 Define Documentation	351
5.40 Command module functions	357
5.40.1 Detailed Description	361
5.40.2 Define Documentation	361
5.41 Button module functions	377
5.41.1 Detailed Description	379
5.41.2 Define Documentation	379
5.42 Ui module functions	382
5.42.1 Detailed Description	384
5.42.2 Define Documentation	385
5.43 Comm module functions	391

5.43.1	Detailed Description	400
5.43.2	Define Documentation	400
5.44	Direct Command functions	433
5.44.1	Detailed Description	436
5.44.2	Define Documentation	436
5.45	System Command functions	447
5.45.1	Detailed Description	449
5.45.2	Define Documentation	450
5.46	IOCtrl module functions	461
5.46.1	Detailed Description	461
5.46.2	Define Documentation	461
5.47	Loader module functions	461
5.47.1	Detailed Description	464
5.47.2	Define Documentation	464
5.48	cstdlib API	473
5.48.1	Detailed Description	474
5.48.2	Define Documentation	474
5.49	cmath API	474
5.49.1	Detailed Description	475
5.49.2	Define Documentation	475
5.50	Property constants	475
5.50.1	Detailed Description	475
5.50.2	Define Documentation	476
5.51	Array operation constants	476
5.51.1	Detailed Description	476
5.51.2	Define Documentation	476
5.52	System Call function constants	477
5.52.1	Detailed Description	479
5.52.2	Define Documentation	479
5.53	Line number constants	486
5.53.1	Detailed Description	487

5.53.2 Define Documentation	487
5.54 Time constants	488
5.54.1 Detailed Description	489
5.54.2 Define Documentation	489
5.55 Mailbox constants	494
5.55.1 Detailed Description	494
5.55.2 Define Documentation	495
5.56 VM state constants	496
5.56.1 Detailed Description	496
5.56.2 Define Documentation	496
5.57 Fatal errors	497
5.57.1 Detailed Description	497
5.57.2 Define Documentation	497
5.58 General errors	499
5.58.1 Detailed Description	499
5.58.2 Define Documentation	500
5.59 Communications specific errors	500
5.59.1 Detailed Description	500
5.59.2 Define Documentation	501
5.60 Remote control (direct commands) errors	501
5.60.1 Detailed Description	501
5.60.2 Define Documentation	501
5.61 Program status constants	502
5.61.1 Detailed Description	502
5.61.2 Define Documentation	502
5.62 Command module IOMAP offsets	503
5.62.1 Detailed Description	503
5.62.2 Define Documentation	504
5.63 IOCtrl module constants	505
5.63.1 Detailed Description	505
5.64 PowerOn constants	505

5.64.1	Detailed Description	506
5.64.2	Define Documentation	506
5.65	IOCtrl module IOMAP offsets	506
5.65.1	Detailed Description	506
5.65.2	Define Documentation	506
5.66	Loader module constants	507
5.66.1	Detailed Description	507
5.66.2	Define Documentation	507
5.67	Loader module IOMAP offsets	507
5.67.1	Detailed Description	508
5.67.2	Define Documentation	508
5.68	Loader module error codes	508
5.68.1	Detailed Description	509
5.68.2	Define Documentation	509
5.69	Loader module function constants	512
5.69.1	Detailed Description	513
5.69.2	Define Documentation	513
5.70	Sound module constants	516
5.70.1	Detailed Description	517
5.71	SoundFlags constants	517
5.71.1	Detailed Description	517
5.71.2	Define Documentation	518
5.72	SoundState constants	518
5.72.1	Detailed Description	518
5.72.2	Define Documentation	518
5.73	SoundMode constants	519
5.73.1	Detailed Description	519
5.73.2	Define Documentation	519
5.74	Sound module IOMAP offsets	520
5.74.1	Detailed Description	520
5.74.2	Define Documentation	520

5.75	Sound module miscellaneous constants	521
5.75.1	Detailed Description	521
5.75.2	Define Documentation	521
5.76	Tone constants	522
5.76.1	Detailed Description	523
5.76.2	Define Documentation	523
5.77	Button module constants	529
5.77.1	Detailed Description	529
5.78	Button name constants	529
5.78.1	Detailed Description	529
5.78.2	Define Documentation	530
5.79	ButtonState constants	531
5.79.1	Detailed Description	531
5.79.2	Define Documentation	531
5.80	Button module IOMAP offsets	532
5.80.1	Detailed Description	532
5.80.2	Define Documentation	532
5.81	Ui module constants	533
5.81.1	Detailed Description	534
5.82	CommandFlags constants	534
5.82.1	Detailed Description	534
5.82.2	Define Documentation	534
5.83	UIState constants	535
5.83.1	Detailed Description	536
5.83.2	Define Documentation	536
5.84	UIButton constants	538
5.84.1	Detailed Description	538
5.84.2	Define Documentation	538
5.85	BluetoothState constants	539
5.85.1	Detailed Description	539
5.85.2	Define Documentation	539

5.86	VM run state constants	540
5.86.1	Detailed Description	540
5.86.2	Define Documentation	540
5.87	Ui module IOMAP offsets	541
5.87.1	Detailed Description	541
5.87.2	Define Documentation	542
5.88	NBC Input port constants	543
5.88.1	Detailed Description	544
5.88.2	Define Documentation	544
5.89	NBC sensor type constants	544
5.89.1	Detailed Description	545
5.89.2	Define Documentation	545
5.90	NBC sensor mode constants	547
5.90.1	Detailed Description	548
5.90.2	Define Documentation	548
5.91	Input field constants	549
5.91.1	Detailed Description	549
5.91.2	Define Documentation	549
5.92	Input port digital pin constants	550
5.92.1	Detailed Description	550
5.92.2	Define Documentation	550
5.93	Color sensor array indices	551
5.93.1	Detailed Description	551
5.93.2	Define Documentation	551
5.94	Color values	552
5.94.1	Detailed Description	552
5.94.2	Define Documentation	552
5.95	Color calibration state constants	553
5.95.1	Detailed Description	553
5.95.2	Define Documentation	553
5.96	Color calibration constants	554

5.96.1 Detailed Description	554
5.96.2 Define Documentation	554
5.97 Input module IOMAP offsets	555
5.97.1 Detailed Description	555
5.97.2 Define Documentation	555
5.98 Constants to use with the Input module's Pin function	557
5.98.1 Detailed Description	558
5.98.2 Define Documentation	558
5.99 Output port constants	559
5.99.1 Detailed Description	559
5.99.2 Define Documentation	559
5.100PID constants	560
5.100.1 Detailed Description	560
5.100.2 Define Documentation	561
5.101Output port update flag constants	562
5.101.1 Detailed Description	562
5.101.2 Define Documentation	562
5.102Tachometer counter reset flags	563
5.102.1 Detailed Description	563
5.102.2 Define Documentation	564
5.103Output port mode constants	564
5.103.1 Detailed Description	565
5.103.2 Define Documentation	565
5.104Output port option constants	565
5.104.1 Detailed Description	566
5.104.2 Define Documentation	566
5.105Output regulation option constants	566
5.105.1 Detailed Description	566
5.105.2 Define Documentation	566
5.106Output port run state constants	566
5.106.1 Detailed Description	567

5.106.2 Define Documentation	567
5.107 Output port regulation mode constants	568
5.107.1 Detailed Description	568
5.107.2 Define Documentation	568
5.108 Output field constants	569
5.108.1 Detailed Description	570
5.108.2 Define Documentation	570
5.109 Output module IOMAP offsets	575
5.109.1 Detailed Description	576
5.109.2 Define Documentation	576
5.110 LowSpeed module constants	578
5.110.1 Detailed Description	579
5.111 LSState constants	579
5.111.1 Detailed Description	579
5.111.2 Define Documentation	580
5.112 LSChannelState constants	580
5.112.1 Detailed Description	581
5.112.2 Define Documentation	581
5.113 LSMode constants	581
5.113.1 Detailed Description	582
5.113.2 Define Documentation	582
5.114 LSErrorType constants	582
5.114.1 Detailed Description	583
5.114.2 Define Documentation	583
5.115 Low speed module IOMAP offsets	583
5.115.1 Detailed Description	584
5.115.2 Define Documentation	584
5.116 LSNoRestartOnRead constants	585
5.116.1 Detailed Description	586
5.116.2 Define Documentation	586
5.117 Standard I2C constants	587

5.117.1 Detailed Description	587
5.117.2 Define Documentation	587
5.118 LEGO I2C address constants	588
5.118.1 Detailed Description	588
5.118.2 Define Documentation	588
5.119 Ultrasonic sensor constants	588
5.119.1 Detailed Description	589
5.119.2 Define Documentation	589
5.120 LEGO temperature sensor constants	590
5.120.1 Detailed Description	591
5.120.2 Define Documentation	591
5.121 E-Meter sensor constants	593
5.121.1 Detailed Description	593
5.121.2 Define Documentation	593
5.122 I2C option constants	594
5.122.1 Detailed Description	594
5.122.2 Define Documentation	595
5.123 Display module constants	595
5.123.1 Detailed Description	597
5.123.2 Define Documentation	597
5.124 DisplayExecuteFunction constants	601
5.124.1 Detailed Description	601
5.124.2 Define Documentation	601
5.125 Drawing option constants	602
5.125.1 Detailed Description	603
5.125.2 Define Documentation	603
5.126 Font drawing option constants	605
5.126.1 Detailed Description	605
5.126.2 Define Documentation	605
5.127 Display flags	606
5.127.1 Detailed Description	607

5.127.2 Define Documentation	607
5.128 Display contrast constants	607
5.128.1 Detailed Description	608
5.128.2 Define Documentation	608
5.129 Text line constants	608
5.129.1 Detailed Description	608
5.129.2 Define Documentation	609
5.130 Display module IOMAP offsets	610
5.130.1 Detailed Description	610
5.130.2 Define Documentation	610
5.131 Comm module constants	612
5.131.1 Detailed Description	613
5.132 Miscellaneous Comm module constants	614
5.132.1 Detailed Description	614
5.132.2 Define Documentation	614
5.133 Bluetooth State constants	616
5.133.1 Detailed Description	616
5.133.2 Define Documentation	616
5.134 Data mode constants	617
5.134.1 Detailed Description	617
5.134.2 Define Documentation	617
5.135 Bluetooth state status constants	618
5.135.1 Detailed Description	618
5.135.2 Define Documentation	618
5.136 Remote connection constants	619
5.136.1 Detailed Description	619
5.136.2 Define Documentation	619
5.137 Bluetooth hardware status constants	621
5.137.1 Detailed Description	621
5.137.2 Define Documentation	621
5.138 Hi-speed port constants	621

5.138.1 Detailed Description	622
5.139 Hi-speed port flags constants	622
5.139.1 Detailed Description	622
5.139.2 Define Documentation	622
5.140 Hi-speed port state constants	623
5.140.1 Detailed Description	623
5.140.2 Define Documentation	623
5.141 Hi-speed port SysCommHSControl constants	624
5.141.1 Detailed Description	624
5.141.2 Define Documentation	624
5.142 Hi-speed port baud rate constants	625
5.142.1 Detailed Description	625
5.142.2 Define Documentation	625
5.143 Hi-speed port UART mode constants	627
5.143.1 Detailed Description	628
5.143.2 Define Documentation	628
5.144 Hi-speed port data bits constants	628
5.144.1 Detailed Description	629
5.144.2 Define Documentation	629
5.145 Hi-speed port stop bits constants	629
5.145.1 Detailed Description	629
5.145.2 Define Documentation	630
5.146 Hi-speed port parity constants	630
5.146.1 Detailed Description	630
5.146.2 Define Documentation	630
5.147 Hi-speed port combined UART constants	631
5.147.1 Detailed Description	631
5.147.2 Define Documentation	631
5.148 Hi-speed port address constants	632
5.148.1 Detailed Description	632
5.148.2 Define Documentation	632

5.149Device status constants	633
5.149.1 Detailed Description	633
5.149.2 Define Documentation	633
5.150Comm module interface function constants	634
5.150.1 Detailed Description	635
5.150.2 Define Documentation	635
5.151Comm module status code constants	637
5.151.1 Detailed Description	637
5.151.2 Define Documentation	637
5.152Comm module IOMAP offsets	638
5.152.1 Detailed Description	639
5.152.2 Define Documentation	640
5.153RCX constants	645
5.153.1 Detailed Description	646
5.154RCX output constants	646
5.154.1 Detailed Description	646
5.154.2 Define Documentation	646
5.155RCX output mode constants	647
5.155.1 Detailed Description	647
5.155.2 Define Documentation	647
5.156RCX output direction constants	648
5.156.1 Detailed Description	648
5.156.2 Define Documentation	648
5.157RCX output power constants	648
5.157.1 Detailed Description	649
5.157.2 Define Documentation	649
5.158RCX IR remote constants	649
5.158.1 Detailed Description	650
5.158.2 Define Documentation	650
5.159RCX and Scout sound constants	651
5.159.1 Detailed Description	652

5.159.2 Define Documentation	652
5.160 Scout constants	653
5.160.1 Detailed Description	653
5.161 Scout light constants	653
5.161.1 Detailed Description	654
5.161.2 Define Documentation	654
5.162 Scout sound constants	654
5.162.1 Detailed Description	655
5.162.2 Define Documentation	655
5.163 Scout sound set constants	657
5.163.1 Detailed Description	657
5.163.2 Define Documentation	658
5.164 Scout mode constants	658
5.164.1 Detailed Description	658
5.164.2 Define Documentation	659
5.165 Scout motion rule constants	659
5.165.1 Detailed Description	659
5.165.2 Define Documentation	659
5.166 Scout touch rule constants	660
5.166.1 Detailed Description	660
5.166.2 Define Documentation	661
5.167 Scout light rule constants	661
5.167.1 Detailed Description	661
5.167.2 Define Documentation	662
5.168 Scout transmit rule constants	662
5.168.1 Detailed Description	662
5.168.2 Define Documentation	663
5.169 Scout special effect constants	663
5.169.1 Detailed Description	663
5.169.2 Define Documentation	663
5.170 RCX and Scout source constants	664

5.170.1 Detailed Description	665
5.170.2 Define Documentation	665
5.171RCX and Scout opcode constants	669
5.171.1 Detailed Description	671
5.171.2 Define Documentation	671
5.172HiTechnic/mindsensors Power Function/IR Train constants	679
5.172.1 Detailed Description	679
5.173Power Function command constants	680
5.173.1 Detailed Description	680
5.173.2 Define Documentation	680
5.174Power Function channel constants	681
5.174.1 Detailed Description	681
5.174.2 Define Documentation	681
5.175Power Function mode constants	681
5.175.1 Detailed Description	682
5.175.2 Define Documentation	682
5.176PF/IR Train function constants	683
5.176.1 Detailed Description	683
5.176.2 Define Documentation	683
5.177IR Train channel constants	683
5.177.1 Detailed Description	684
5.177.2 Define Documentation	684
5.178Power Function output constants	684
5.178.1 Detailed Description	685
5.178.2 Define Documentation	685
5.179Power Function pin constants	685
5.179.1 Detailed Description	685
5.179.2 Define Documentation	685
5.180Power Function single pin function constants	685
5.180.1 Detailed Description	686
5.180.2 Define Documentation	686

5.181 Power Function CST options constants	686
5.181.1 Detailed Description	687
5.181.2 Define Documentation	687
5.182 Power Function PWM option constants	688
5.182.1 Detailed Description	688
5.182.2 Define Documentation	688
5.183 HiTechnic device constants	690
5.183.1 Detailed Description	691
5.183.2 Define Documentation	691
5.184 HiTechnic IRSeeker2 constants	692
5.184.1 Detailed Description	693
5.184.2 Define Documentation	693
5.185 HiTechnic IRReceiver constants	695
5.185.1 Detailed Description	695
5.185.2 Define Documentation	695
5.186 HiTechnic Color2 constants	696
5.186.1 Detailed Description	696
5.186.2 Define Documentation	697
5.187 HiTechnic Angle sensor constants	698
5.187.1 Detailed Description	698
5.187.2 Define Documentation	698
5.188 HiTechnic Barometric sensor constants	700
5.188.1 Detailed Description	700
5.188.2 Define Documentation	700
5.189 HiTechnic Prototype board constants	700
5.189.1 Detailed Description	701
5.189.2 Define Documentation	701
5.190 HiTechnic Prototype board analog input constants	702
5.190.1 Detailed Description	702
5.190.2 Define Documentation	703
5.191 HiTechnic SuperPro constants	703

5.191.1 Detailed Description	705
5.191.2 Define Documentation	705
5.192 HiTechnic SuperPro analog input index constants	710
5.192.1 Detailed Description	711
5.192.2 Define Documentation	711
5.193 HiTechnic SuperPro analog output index constants	711
5.193.1 Detailed Description	712
5.193.2 Define Documentation	712
5.194 MindSensors device constants	712
5.194.1 Detailed Description	713
5.194.2 Define Documentation	713
5.195 MindSensors DIST-Nx constants	715
5.195.1 Detailed Description	716
5.195.2 Define Documentation	716
5.196 MindSensors PSP-Nx constants	717
5.196.1 Detailed Description	718
5.196.2 Define Documentation	718
5.197 MindSensors PSP-Nx button set 1 constants	719
5.197.1 Detailed Description	719
5.197.2 Define Documentation	719
5.198 MindSensors PSP-Nx button set 2 constants	720
5.198.1 Detailed Description	721
5.198.2 Define Documentation	721
5.199 MindSensors nRLink constants	722
5.199.1 Detailed Description	722
5.199.2 Define Documentation	722
5.200 MindSensors ACCL-Nx constants	724
5.200.1 Detailed Description	724
5.200.2 Define Documentation	724
5.201 MindSensors ACCL-Nx sensitivity level constants	726
5.201.1 Detailed Description	727

5.201.2 Define Documentation	727
5.202 MindSensors PFMate constants	727
5.202.1 Detailed Description	728
5.202.2 Define Documentation	728
5.203 PFMate motor constants	729
5.203.1 Detailed Description	729
5.203.2 Define Documentation	729
5.204 PFMate channel constants	730
5.204.1 Detailed Description	730
5.204.2 Define Documentation	730
5.205 MindSensors NXTServo constants	730
5.205.1 Detailed Description	731
5.206 MindSensors NXTServo registers	731
5.206.1 Detailed Description	732
5.206.2 Define Documentation	732
5.207 MindSensors NXTServo position constants	735
5.207.1 Detailed Description	735
5.207.2 Define Documentation	735
5.208 MindSensors NXTServo quick position constants	736
5.208.1 Detailed Description	736
5.208.2 Define Documentation	736
5.209 MindSensors NXTServo servo numbers	736
5.209.1 Detailed Description	737
5.209.2 Define Documentation	737
5.210 MindSensors NXTServo commands	738
5.210.1 Detailed Description	738
5.210.2 Define Documentation	738
5.211 MindSensors NXTHID constants	739
5.211.1 Detailed Description	740
5.212 MindSensors NXTHID registers	740
5.212.1 Detailed Description	740

5.212.2 Define Documentation	740
5.213 MindSensors NXTHID modifier keys	741
5.213.1 Detailed Description	741
5.213.2 Define Documentation	741
5.214 MindSensors NXTHID commands	742
5.214.1 Detailed Description	742
5.214.2 Define Documentation	742
5.215 MindSensors NXTPowerMeter constants	743
5.215.1 Detailed Description	743
5.216 MindSensors NXTPowerMeter registers	743
5.216.1 Detailed Description	744
5.216.2 Define Documentation	744
5.217 MindSensors NXTPowerMeter commands	745
5.217.1 Detailed Description	745
5.217.2 Define Documentation	746
5.218 MindSensors NXTSumoEyes constants	746
5.218.1 Detailed Description	746
5.218.2 Define Documentation	746
5.219 MindSensors NXTLineLeader constants	747
5.219.1 Detailed Description	747
5.220 MindSensors NXTLineLeader registers	747
5.220.1 Detailed Description	748
5.220.2 Define Documentation	748
5.221 MindSensors NXTLineLeader commands	749
5.221.1 Detailed Description	750
5.221.2 Define Documentation	750
5.222 Codatex device constants	751
5.222.1 Detailed Description	751
5.223 Codatex RFID sensor constants	751
5.223.1 Detailed Description	752
5.223.2 Define Documentation	752

5.224Codatex RFID sensor modes	752
5.224.1 Detailed Description	753
5.224.2 Define Documentation	753
5.225Dexter Industries device constants	753
5.225.1 Detailed Description	753
5.226Dexter Industries GPS sensor constants	754
5.226.1 Detailed Description	754
5.226.2 Define Documentation	754
5.227Dexter Industries IMU sensor constants	755
5.227.1 Detailed Description	757
5.227.2 Define Documentation	757
5.228Dexter Industries IMU Gyro register constants	757
5.228.1 Detailed Description	758
5.228.2 Define Documentation	758
5.229Dexter Industries IMU Gyro control register 1 constants	762
5.229.1 Detailed Description	762
5.229.2 Define Documentation	762
5.230Dexter Industries IMU Gyro control register 2 constants	764
5.230.1 Detailed Description	764
5.230.2 Define Documentation	764
5.231Dexter Industries IMU Gyro control register 3 constants	766
5.231.1 Detailed Description	766
5.231.2 Define Documentation	766
5.232Dexter Industries IMU Gyro control register 4 constants	767
5.232.1 Detailed Description	767
5.232.2 Define Documentation	768
5.233Dexter Industries IMU Gyro control register 5 constants	768
5.233.1 Detailed Description	769
5.233.2 Define Documentation	769
5.234Dexter Industries IMU Gyro FIFO control register onstants	770
5.234.1 Detailed Description	770

5.234.2 Define Documentation	770
5.235 Dexter Industries IMU Gyro status register constants	771
5.235.1 Detailed Description	771
5.235.2 Define Documentation	771
5.236 Dexter Industries IMU Accelerometer register constants	772
5.236.1 Detailed Description	773
5.236.2 Define Documentation	773
5.237 Dexter Industries IMU Accelerometer status register constants	776
5.237.1 Detailed Description	776
5.237.2 Define Documentation	777
5.238 Dexter Industries IMU Accelerometer mode control register constants	777
5.238.1 Detailed Description	777
5.238.2 Define Documentation	777
5.239 Dexter Industries IMU Accelerometer interrupt latch reset register constants	778
5.239.1 Detailed Description	778
5.239.2 Define Documentation	779
5.240 Dexter Industries IMU Accelerometer control register 1 constants	779
5.240.1 Detailed Description	779
5.240.2 Define Documentation	779
5.241 Dexter Industries IMU Accelerometer control register 2 constants	780
5.241.1 Detailed Description	781
5.241.2 Define Documentation	781
5.242 Microinfinity device constants	781
5.242.1 Detailed Description	781
5.243 Microinfinity CruizCore XG1300L sensor constants	782
5.243.1 Detailed Description	782
5.243.2 Define Documentation	782
5.244 Microinfinity CruizCore XG1300L	783
5.244.1 Detailed Description	784
5.244.2 Define Documentation	784

5.245Data type limits	784
5.245.1 Detailed Description	785
5.245.2 Define Documentation	785
5.246Graphics library begin modes	786
5.246.1 Detailed Description	787
5.246.2 Define Documentation	787
5.247Graphics library actions	787
5.247.1 Detailed Description	788
5.247.2 Define Documentation	788
5.248Graphics library settings	789
5.248.1 Detailed Description	789
5.248.2 Define Documentation	789
5.249Graphics library cull mode	790
5.249.1 Detailed Description	790
5.249.2 Define Documentation	790
6 File Documentation	790
6.1 NBCAPIDocs.h File Reference	790
6.1.1 Detailed Description	790
6.2 NBCCommon.h File Reference	791
6.2.1 Detailed Description	838
6.2.2 Define Documentation	838

1 NBC Programmer's Guide

October 10, 2011

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 BricxCC 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	arrtostr
jmp	brcmp	brdst	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
compelse	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	addrdf	

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
ComplexStruct 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
ComplexStruct 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
<code>byte, ubyte, db</code>	8 bit unsigned
<code>sbyte</code>	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__</code> , <code>__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__</code> , <code>__ResetJ__</code>	These tokens are replaced with nothing. As a side effect the value of I or J is reset to zero.
<code>__IncI__</code> , <code>__IncJ__</code>	These tokens are replaced with nothing. As a side effect the value of I or J is incremented by one.
<code>__DecI__</code> , <code>__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:

+	addition
-	subtraction
*	multiplication
/	division
\wedge	exponent
%	modulo (remainder)
&	bitwise and
	bitwise or
\sim	bitwise xor
<<	shift left
>>	shift right
()	grouping subexpressions
PI	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 two ways to assign a new value to a variable.

- `mov`
- `set`
- `addror`

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 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, sinhhd, coshd, tanhd, 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
```

3.6.2.21 acosd

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
```

3.6.2.22 atand

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
```

3.6.2.23 atan2d

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
```

3.6.2.24 sinh

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
```

3.6.2.25 cosh

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
```

3.6.2.26 tanh

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
```

3.6.2.27 sinh

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
```

3.6.2.28 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](#)
- [lsr](#)
- [lsl](#)
- [rotr](#)
- [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](#)
- [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 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](#)
- [TCommBTOnOff](#)
- [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 su
```

```

    cceeded.
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.

- [index](#)
- [replace](#)

- [arrsize](#)
- [arrinit](#)
- [arrsubset](#)
- [arrbuild](#)
- [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](#), and [OPARR_SORT](#).

In the case of [OPARR_SORT](#) the output parameter should be an array of the same type as the input array. In all the other cases it 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
```

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](#)
- [start](#)
- [stopthread](#)
- [priority](#)
- [precedes](#)
- [follows](#)
- [acquire](#)
- [release](#)
- [subcall](#)
- [subret](#)
- [call](#)
- [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 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
```

3.6.11.4 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
```

3.6.11.5 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
```

3.6.11.6 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
```

3.6.11.7 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
```

3.6.11.8 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
```

3.6.11.9 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
```

3.6.11.10 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
```

3.6.11.11 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
```

3.6.11.12 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
```

3.6.11.13 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 compend statement. To optionally provide an else clause use the compelse statement between the compif and compend 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
compend
```

3.6.14.6 compelse

The compelse Statement

The compelse statement works together with the compif and compend 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 compend 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
compend
```

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.

Defines

- #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 Define 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

- [Output module](#)

Constants and functions related to the Output module.

- [Input module](#)

Constants and functions related to the Input module.

- [Low Speed module](#)

Constants and functions related to the Low Speed module.

- [Display module](#)

Constants and functions related to the Display module.

- [Sound module](#)

Constants and functions related to the Sound module.

- [Command module](#)

Constants and functions related to the Command module.

- [Button module](#)

Constants and functions related to the Button module.

- [Ui module](#)

Constants and functions related to the Ui module.

- [Comm module](#)

Constants and functions related to the Comm module.

- [IOCtrl module](#)

Constants and functions related to the IOCtrl module.

- [Loader module](#)

Constants and functions related to the Loader module.

- [NXT firmware module names](#)

Constant string names for all the NXT firmware modules.

- [NXT firmware module IDs](#)

Constant numeric IDs for all the NXT firmware modules.

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 functions](#)

Functions for accessing and modifying input module features.

- [Input module constants](#)

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

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

- [NBC Input port constants](#)

Input port constants are used when calling sensor control API functions.

- [Input field constants](#)

Constants for use with `SetInput()` and `GetInput()`.

- [Input port digital pin constants](#)

Constants for use when directly controlling or reading a port's digital pin state.

- [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.

- [Color calibration state constants](#)

Constants for use with the color calibration state function.

- [Color calibration constants](#)

Constants for use with the color calibration functions.

- [Input module IOMAP offsets](#)

Constant offsets into the Input module IOMAP structure.

- [Constants to use with the Input module's Pin function](#)

Constants for use with the Input module's Pin function.

- [Sensor types and modes](#)

Constants that are used for defining sensor types and modes.

Defines

- #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 Define 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 type constants](#)

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

- [NBC sensor mode constants](#)

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

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 functions](#)

Functions for accessing and modifying output module features.

- [Output module constants](#)

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

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 port constants](#)

Output port constants are used when calling motor control API functions.

- [PID constants](#)

PID constants are for adjusting the Proportional, Integral, and Derivative motor controller parameters.

- [Output port update flag constants](#)

Use these constants to specify which motor values need to be updated.

- [Tachometer counter reset flags](#)

Use these constants to specify which of the three tachometer counters should be reset.

- [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 regulation option constants](#)

Use these constants to configure the desired options for position regulation.

- [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 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 field constants](#)

Constants for use with SetOutput() and GetOutput().

- [Output module IOMAP offsets](#)

Constant offsets into the Output module IOMAP structure.

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 functions](#)

Functions for accessing and modifying Command module features.

- [Command module constants](#)

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

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.

- [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.

- [Fatal errors](#)

Constants defining various fatal error conditions.

- [General errors](#)

Constants defining general error conditions.

- [Communications specific errors](#)

Constants defining communication error conditions.

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

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

- [Program status constants](#)

Constants defining various states of the command module virtual machine.

- [Command module IOMAP offsets](#)

Constant offsets into the Command module IOMAP structure.

Defines

- #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 Define 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 functions](#)

Functions for accessing and modifying Comm module features.

- [Comm module constants](#)

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

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 BluetoothWrite 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 MessageWrite 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 functions](#)

Functions for accessing and modifying Button module features.

- [Button module constants](#)

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

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 functions](#)

Functions for accessing and modifying IOCtrl module features.

- [IOCtrl module constants](#)

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

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 functions](#)

Functions for accessing and modifying Loader module features.

- [Loader module constants](#)

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

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 functions](#)

Functions for accessing and modifying sound module features.

- [Sound module constants](#)

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

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 functions](#)

Functions for accessing and modifying Ui module features.

- [Ui module constants](#)

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

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 functions](#)

Functions for accessing and modifying low speed module features.

- [LowSpeed module constants](#)

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

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 functions](#)

Functions for accessing and modifying display module features.

- [Display module constants](#)

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

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.

Defines

- `#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 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)` __-
HTPFComboDirect(_port, _channel, _outa, _outb, _result)
HTPFComboDirect function.
- #define `HTPFComboPWM(_port, _channel, _outa, _outb, _result)` __-
HTPFComboPWM(_port, _channel, _outa, _outb, _result)
HTPFComboPWM function.
- #define `HTPFRawOutput(_port, _nibble0, _nibble1, _nibble2, _result)` __-
HTPFRawOutput(_port, _nibble0, _nibble1, _nibble2, _result)
HTPFRawOutput function.
- #define `HTPFRepeat(_port, _count, _delay, _result)` __-
HTPFRepeatLastCommand(_port, _count, _delay, _result)
HTPFRepeat 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 `HTPFTTrain(_port, _channel, _func, _result)` __HTIRTTrain(_port, _-
channel, _func, TRUE, _result)
HTPFTTrain 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 **HTRCXInvertOutput**(_outputs) __HTRCXSetGlobalDirection(_outputs, RCX_OUT_REV)

HTRCXInvertOutput 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)
HTRCXSetDirection function.
- #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(_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 Sensor HT Color2 Active.

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) __ReadSensorHTProtoAllAnalog(_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.

5.18.1 Detailed Description

Functions for accessing and modifying HiTechnic devices.

5.18.2 Define 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:

_port The sensor port. See [NBC Input port constants](#).
_channel The IR Train channel. See [IR Train channel constants](#).
_func The IR Train function. See [PF/IR Train function constants](#)
_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

**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:

_port The sensor port. See [NBC Input port constants](#).
_channel The Power Function channel. See [Power Function channel constants](#).
_outa The Power Function command for output A. See [Power Function command constants](#).
_outb The Power Function command for output B. See [Power Function command constants](#).
_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

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:

- _port* The sensor port. See [NBC Input port constants](#).
- _channel* The Power Function channel. See [Power Function channel constants](#).
- _outa* The Power Function PWM command for output A. See [Power Function PWM option constants](#).
- _outb* The Power Function PWM command for output B. See [Power Function PWM option constants](#).
- _result* The function call result. [NO_ERR](#) or [Communications specific errors](#).

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:

- _port* The sensor port. See [NBC Input port constants](#).
- _nibble0* The first raw data nibble.
- _nibble1* The second raw data nibble.
- _nibble2* The third raw data nibble.
- _result* The function call result. [NO_ERR](#) or [Communications specific errors](#).

**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:

_port The sensor port. See [NBC Input port constants](#).
_count The number of times to repeat the command.
_delay The number of milliseconds to delay between each repetition.
_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

**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:

_port The sensor port. See [NBC Input port constants](#).
_channel The Power Function channel. See [Power Function channel constants](#).
_out The Power Function output. See [Power Function output constants](#).
_func The Power Function CST function. See [Power Function CST options constants](#).
_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

```
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 iRLLink 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:

_port The sensor port. See [NBC Input port constants](#).
_channel The Power Function channel. See [Power Function channel constants](#).
_out The Power Function output. See [Power Function output constants](#).
_func The Power Function PWM function. See [Power Function PWM option constants](#).
_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

```
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:

_port The sensor port. See [NBC Input port constants](#).
_channel The Power Function channel. See [Power Function channel constants](#).

_out The Power Function output. See [Power Function output constants](#).
_pin The Power Function pin. See [Power Function pin constants](#).
_func The Power Function single pin function. See [Power Function single pin function constants](#).
_cont Control whether the mode is continuous or timeout.
_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

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

HTPFTrain function. Control both outputs on a Power Function receiver set to the specified channel using the HiTechnic iRLink 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:

_port The sensor port. See [NBC Input port constants](#).
_channel The Power Function channel. See [Power Function channel constants](#).
_func The Power Function train function. See [PF/IR Train function constants](#).
_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

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:

_port The sensor port. See [NBC Input port constants](#).
_channel The Power Function channel. See [Power Function channel constants](#).

_outa The Power Function command for output A. See [Power Function command constants](#).

_outb The Power Function command for output B. See [Power Function command constants](#).

_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

5.18.2.11 #define HTRCXAddToDatalog(_src, _value) __- HTRCXAddToDatalog(_src, _value)

HTRCXAddToDatalog function. Send the AddToDatalog command to an RCX.

Parameters:

_src The RCX source. See [RCX and Scout source constants](#).

_value 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:

_result 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:

_counter 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:

_port 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:

_timer The timer to clear.

5.18.2.19 #define HTRCXCreateDatalog(_size) __HTRCXCreateDatalog(_size)

HTRCXCreateDatalog function. Send the CreateDatalog command to an RCX.

Parameters:

_size The new datalog size.

5.18.2.20 #define HTRCXDecCounter(_counter) __HTRCXDecCounter(_-counter)

HTRCXDecCounter function. Send the DecCounter command to an RCX.

Parameters:

_counter The counter to decrement.

5.18.2.21 #define HTRCXDeleteSub(_s) __HTRCXDeleteSub(_s)

HTRCXDeleteSub function. Send the DeleteSub command to an RCX.

Parameters:

_s 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:

_t 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:

_outputs The RCX output(s) to disable. See [RCX output constants](#).

**5.18.2.26 #define HTRCXEnableOutput(_outputs) __-
HTRCXSetGlobalOutput(_outputs, RCX_OUT_ON)**

HTRCXEnableOutput function. Send the EnableOutput command to an RCX.

Parameters:

_outputs The RCX output(s) to enable. See [RCX output constants](#).

5.18.2.27 #define HTRCXEvent(_src, _value) __HTRCXEvent(_src, _value)

HTRCXEvent function. Send the Event command to an RCX.

Parameters:

_src The RCX source. See [RCX and Scout source constants](#).

_value 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:

_outputs The RCX output(s) to float. See [RCX output constants](#).

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:

_outputs The RCX output(s) to set forward. See [RCX output constants](#).

5.18.2.30 #define HTRCXIncCounter(_counter) __HTRCXIncCounter(_-counter)

HTRCXIncCounter function. Send the IncCounter command to an RCX.

Parameters:

_counter The counter to increment.

5.18.2.31 #define HTRCXIvertOutput(_outputs) __-HTRCXSetGlobalDirection(_outputs, RCX_OUT_REV)

HTRCXIvertOutput function. Send the InvertOutput command to an RCX.

Parameters:

_outputs The RCX output(s) to invert. See [RCX output constants](#).

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:

_outputs The RCX output(s) to obvert. See [RCX output constants](#).

**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:

_outputs The RCX output(s) to turn off. See [RCX output constants](#).

**5.18.2.35 #define HTRCXOn(_outputs) __HTRCXSetOutput(_outputs,
RCX_OUT_ON)**

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

Parameters:

_outputs The RCX output(s) to turn on. See [RCX output constants](#).

**5.18.2.36 #define HTRCXOnFor(_outputs, _ms) __HTRCXOnFor(_outputs,
_ms)**

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

Parameters:

_outputs The RCX output(s) to turn on. See [RCX output constants](#).

_ms 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:

_outputs The RCX output(s) to turn on in the forward direction. See [RCX output constants](#).

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:

_outputs The RCX output(s) to turn on in the reverse direction. See [RCX output constants](#).

5.18.2.39 #define HTRECXPBTurnOff() __HTRECXOpNoArgs(RCX_PBTurnOffOp)

HTRECXPBTurnOff 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 HTRECXPlaySound(_snd) __HTRECXPlaySound(_snd)

HTRECXPlaySound function. Send the PlaySound command to an RCX.

Parameters:

_snd The sound number to play.

5.18.2.42 #define HTRECXPlayTone(_freq, _duration) __HTRECXPlayTone(_freq, _duration)

HTRECXPlayTone function. Send the PlayTone command to an RCX.

Parameters:

- _freq* The frequency of the tone to play.
- _duration* The duration of the tone to play.

5.18.2.43 #define HTRECXPlayToneVar(_varnum, _duration) __HTRECXPlayToneVar(_varnum, _duration)

HTRECXPlayToneVar function. Send the PlayToneVar command to an RCX.

Parameters:

- _varnum* The variable containing the tone frequency to play.
- _duration* The duration of the tone to play.

5.18.2.44 #define HTRECXPoll(_src, _value, _result) __HTRECXPoll(_src, _value, _result)

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

Parameters:

- _src* The RCX source. See [RCX and Scout source constants](#).
- _value* The RCX value.
- _result* The value read from the specified port and value.

5.18.2.45 #define HTRECXPollMemory(_memaddress, _result) __HTRECXPollMemory(_memaddress, _result)

HTRECXPollMemory function. Send the PollMemory command to an RCX.

Parameters:

- _memaddress* The RCX memory address.
- _result* 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:

- _cmd* The RCX IR remote command to send. See [RCX IR remote constants](#).

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:

- _outputs* The RCX output(s) to reverse direction. See [RCX output constants](#).

**5.18.2.48 #define HTRCXSelectDisplay(_src, _value) __-
HTRCXSelectDisplay(_src, _value)**

HTRCXSelectDisplay function. Send the SelectDisplay command to an RCX.

Parameters:

- _src* The RCX source. See [RCX and Scout source constants](#).
- _value* The RCX value.

**5.18.2.49 #define HTRCXSelectProgram(_prog) __HTRCXSelectProgram(_-
prog)**

HTRCXSelectProgram function. Send the SelectProgram command to an RCX.

Parameters:

prog 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:

first The first byte address.

count 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:

_outputs The RCX output(s) to set direction. See [RCX output constants](#).

_dir The RCX output direction. See [RCX output direction constants](#).

5.18.2.52 #define HTRCXSetEvent(_evt, _src, _type) __HTRCXSetEvent(_evt, _src, _type)

HTRCXSetEvent function. Send the SetEvent command to an RCX.

Parameters:

_evt The event number to set.

_src The RCX source. See [RCX and Scout source constants](#).

_type The event type.

**5.18.2.53 #define HTRCXSetGlobalDirection(_outputs,
_dir) __HTRCXSetGlobalDirection(_outputs, _dir)**

HTRCXSetGlobalDirection function. Send the SetGlobalDirection command to an RCX.

Parameters:

_outputs The RCX output(s) to set global direction. See [RCX output constants](#).
_dir The RCX output direction. See [RCX output direction constants](#).

**5.18.2.54 #define HTRCXSetGlobalOutput(_outputs,
_mode) __HTRCXSetGlobalOutput(_outputs, _mode)**

HTRCXSetGlobalOutput function. Send the SetGlobalOutput command to an RCX.

Parameters:

_outputs The RCX output(s) to set global mode. See [RCX output constants](#).
_mode The RCX output mode. See [RCX output mode constants](#).

**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:

_port The sensor port. See [NBC Input port constants](#).

**5.18.2.56 #define HTRCXSetMaxPower(_outputs, _pwrsrc,
_pwrval) __HTRCXSetMaxPower(_outputs, _pwrsrc, _pwrval)**

HTRCXSetMaxPower function. Send the SetMaxPower command to an RCX.

Parameters:

- _outputs* The RCX output(s) to set max power. See [RCX output constants](#).
- _pwrsrc* The RCX source. See [RCX and Scout source constants](#).
- _pwrval* The RCX value.

5.18.2.57 #define HTRCXSetMessage(_msg) __HTRCXSetMessage(_msg)

HTRCXSetMessage function. Send the SetMessage command to an RCX.

Parameters:

- _msg* 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:

- _outputs* The RCX output(s) to set mode. See [RCX output constants](#).
- _mode* The RCX output mode. See [RCX output mode constants](#).

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:

- _outputs* The RCX output(s) to set power. See [RCX output constants](#).
- _pwrsrc* The RCX source. See [RCX and Scout source constants](#).
- _pwrval* The RCX value.

5.18.2.60 #define HTRECXSetPriority(_p) __HTRECXSetPriority(_p)

HTRECXSetPriority function. Send the SetPriority command to an RCX.

Parameters:

_p The new task priority.

**5.18.2.61 #define HTRECXSetSensorMode(_port,
_mode) __HTRECXSetSensorMode(_port, _mode)**

HTRECXSetSensorMode function. Send the SetSensorMode command to an RCX.

Parameters:

_port The RCX sensor port.

_mode The RCX sensor mode.

**5.18.2.62 #define HTRECXSetSensorType(_port, _type) __-
HTRECXSetSensorType(_port, _type)**

HTRECXSetSensorType function. Send the SetSensorType command to an RCX.

Parameters:

_port The RCX sensor port.

_type The RCX sensor type.

5.18.2.63 #define HTRECXSetSleepTime(_t) __HTRECXSetSleepTime(_t)

HTRECXSetSleepTime function. Send the SetSleepTime command to an RCX.

Parameters:

_t The new sleep time value.

5.18.2.64 #define HTRCXSetTxPower(_pwr) __HTRCXSetTxPower(_pwr)

HTRCXSetTxPower function. Send the SetTxPower command to an RCX.

Parameters:

_pwr 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:

_hours The new watch time hours value.

_minutes 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:

_t 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:

_t 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:

_outputs The RCX output(s) to toggle. See [RCX output constants](#).

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:

_grp 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:

_src The Scout source. See [RCX and Scout source constants](#).
_value The Scout value.

5.18.2.75 #define HTScoutSetEventFeedback(_src, _value) __HTScoutSetEventFeedback(_src, _value)

HTScoutSetEventFeedback function. Send the SetEventFeedback command to a Scout.

Parameters:

_src The Scout source. See [RCX and Scout source constants](#).
_value The Scout value.

5.18.2.76 #define HTScoutSetLight(_x) __HTScoutSetLight(_x)

HTScoutSetLight function. Send the SetLight command to a Scout.

Parameters:

_x Set the light on or off using this value. See [Scout light constants](#).

5.18.2.77 #define HTScoutSetScoutMode(_mode) __HTScoutSetScoutMode(_mode)

HTScoutSetScoutMode function. Send the SetScoutMode command to a Scout.

Parameters:

_mode Set the scout mode. See [Scout mode constants](#).

**5.18.2.78 #define HTScoutSetSensorClickTime(_src,
_value) __HTScoutSetSensorClickTime(_src, _value)**

HTScoutSetSensorClickTime function. Send the SetSensorClickTime command to a Scout.

Parameters:

_src The Scout source. See [RCX and Scout source constants](#).
_value The Scout value.

**5.18.2.79 #define HTScoutSetSensorHysteresis(_src,
_value) __HTScoutSetSensorHysteresis(_src, _value)**

HTScoutSetSensorHysteresis function. Send the SetSensorHysteresis command to a Scout.

Parameters:

_src The Scout source. See [RCX and Scout source constants](#).
_value The Scout value.

**5.18.2.80 #define HTScoutSetSensorLowerLimit(_src,
_value) __HTScoutSetSensorLowerLimit(_src, _value)**

HTScoutSetSensorLowerLimit function. Send the SetSensorLowerLimit command to a Scout.

Parameters:

_src The Scout source. See [RCX and Scout source constants](#).
_value The Scout value.

**5.18.2.81 #define HTScoutSetSensorUpperLimit(_src,
_value) __HTScoutSetSensorUpperLimit(_src, _value)**

HTScoutSetSensorUpperLimit function. Send the SetSensorUpperLimit command to a Scout.

Parameters:

_src The Scout source. See [RCX and Scout source constants](#).
_value 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:

_port The sensor port. See [NBC Input port constants](#).
_x The output x-axis acceleration.
_y The output y-axis acceleration.
_z The output z-axis acceleration.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_Angle Current angle in degrees (0-359).

_AccAngle Accumulated angle in degrees (-2147483648 to 2147483647).

_RPM rotations per minute (-1000 to 1000).

_result 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:

_port The sensor port. See [NBC Input port constants](#).

_temp Current temperature in 1/10ths of degrees Celcius.

_press Current barometric pressure in 1/1000 inches of mercury.

_result 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:

_port The sensor port. See [NBC Input port constants](#).

_ColorNum The output color number.

_Red The red color value.

_Green The green color value.

_Blue The blue color value.

_result 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:

_port The sensor port. See [NBC Input port constants](#).
_ColorNum The output color number.
_Red The red color value.
_Green The green color value.
_Blue The blue color value.
_White The white color value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_value 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:

_port The sensor port. See [NBC Input port constants](#).
_value 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:

_port The sensor port. See [NBC Input port constants](#).
_val The EOPD sensor reading.

**5.18.2.91 #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:

_p The sensor port. See [NBC Input port constants](#).
_offset The zero offset.
_val The Gyro sensor reading.

**5.18.2.92 #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:

_port The sensor port. See [NBC Input port constants](#).

_pfdata Eight bytes of power function remote IR data.

_result The function call result.

5.18.2.93 #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:

_port The sensor port. See [NBC Input port constants](#).

_reg The power function data offset. See [HiTechnic IRReceiver constants](#).

_pfchar A single byte of power function remote IR data.

_result The function call result.

5.18.2.94 #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:

_port The sensor port. See [NBC Input port constants](#).

_dir The direction.

_s1 The signal strength from sensor 1.

_s3 The signal strength from sensor 3.

_s5 The signal strength from sensor 5.

_s7 The signal strength from sensor 7.

_s9 The signal strength from sensor 9.

_result The function call result.

```
5.18.2.95 #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:

_port The sensor port. See [NBC Input port constants](#).
_dir The direction.
_s1 The signal strength from sensor 1.
_s3 The signal strength from sensor 3.
_s5 The signal strength from sensor 5.
_s7 The signal strength from sensor 7.
_s9 The signal strength from sensor 9.
_result The function call result.

```
5.18.2.96 #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:

_port The sensor port. See [NBC Input port constants](#).
_reg The register address. See [HiTechnic IRSeeker2 constants](#).
_value The IRSeeker2 register value.

```
5.18.2.97 #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:

- _port* The sensor port. See [NBC Input port constants](#).
- _dir* The direction.
- _s1* The signal strength from sensor 1.
- _s3* The signal strength from sensor 3.
- _s5* The signal strength from sensor 5.
- _s7* The signal strength from sensor 7.
- _s9* The signal strength from sensor 9.
- _avg* The average signal strength.
- _result* The function call result.

**5.18.2.98 #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:

- _port* The sensor port. See [NBC Input port constants](#).
- _value* The IRSeeker direction.

**5.18.2.99 #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:

- _p* The sensor port. See [NBC Input port constants](#).

_offset The zero offset.
_val The Magnet sensor reading.

5.18.2.100 #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:

_port The sensor port. See [NBC Input port constants](#).
_ColorIdx The output color index.
_Red The normalized red color value.
_Green The normalized green color value.
_Blue The normalized blue color value.
_result The function call result.

5.18.2.101 #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:

_port The sensor port. See [NBC Input port constants](#).
_ColorIdx The output color index.
_Red The normalized red color value.
_Green The normalized green color value.
_Blue The normalized blue color value.
_result The function call result.

```
5.18.2.102 #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:

_port The sensor port. See [NBC Input port constants](#).
_a0 The A0 analog input value.
_a1 The A1 analog input value.
_a2 The A2 analog input value.
_a3 The A3 analog input value.
_a4 The A4 analog input value.
_result The function call result.

```
5.18.2.103 #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:

_port The sensor port. See [NBC Input port constants](#).
_input The analog input. See [HiTechnic Prototype board analog input constants](#).
_value The analog input value.
_result The function call result.

```
5.18.2.104 #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:

_port The sensor port. See [NBC Input port constants](#).
_value The digital input values. See [SuperPro digital pin constants](#).
_result The function call result.

```
5.18.2.105 #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:

_port The sensor port. See [NBC Input port constants](#).
_out The digital control value. See [SuperPro LED control constants](#).
_result The function call result.

```
5.18.2.106 #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:

_port The sensor port. See [NBC Input port constants](#).
_Red The raw red color value.
_Green The raw green color value.
_Blue The raw blue color value.
_result The function call result.

5.18.2.107 #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:

_port The sensor port. See [NBC Input port constants](#).
_Red The raw red color value.
_Green The raw green color value.
_Blue The raw blue color value.
_White The raw white color value.
_result The function call result.

5.18.2.108 #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:

_port The sensor port. See [NBC Input port constants](#).

- _a0* The A0 analog input value.
- _a1* The A1 analog input value.
- _a2* The A2 analog input value.
- _a3* The A3 analog input value.
- _result* The function call result.

5.18.2.109 #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:

- _port* The sensor port. See [NBC Input port constants](#).
- _input* The analog input. See [HiTechnic SuperPro analog input index constants](#).
- _value* The analog input value.
- _result* The function call result.

5.18.2.110 #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:

- _port* The sensor port. See [NBC Input port constants](#).
- _dac* The analog output index. See [HiTechnic SuperPro analog output index constants](#).
- _mode* The analog output mode. See [SuperPro analog output mode constants](#).
- _freq* The analog output frequency. Between 1 and 8191.

_volt The analog output voltage level. A 10 bit value (0..1023).

_result The function call result.

```
5.18.2.111 #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:

_port The sensor port. See [NBC Input port constants](#).

_value The digital input values. See [SuperPro digital pin constants](#).

_result The function call result.

```
5.18.2.112 #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:

_port The sensor port. See [NBC Input port constants](#).

_out The digital control value. See [SuperPro LED control constants](#).

_result The function call result.

```
5.18.2.113 #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:

- _port* The sensor port. See [NBC Input port constants](#).
- _out* The LED value. See [SuperPro LED control constants](#).
- _result* The function call result.

5.18.2.114 #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:

- _port* The sensor port. See [NBC Input port constants](#).
- _out* The program control value.
- _result* The function call result.

5.18.2.115 #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:

- _port* The sensor port. See [NBC Input port constants](#).
- _out* The strobe value. See [SuperPro Strobe control constants](#).
- _result* The function call result.

5.18.2.116 #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:

- _p* The sensor port. See [NBC Input port constants](#).
- _t1* The value of touch sensor 1.
- _t2* The value of touch sensor 2.
- _t3* The value of touch sensor 3.
- _t4* The value of touch sensor 4.

**5.18.2.117 #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:

- _port* The sensor port. See [NBC Input port constants](#).
- _result* The function call result.

**5.18.2.118 #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:

- _port* The sensor port. See [NBC Input port constants](#).
- _mode* The Angle reset mode. See [HiTechnic Angle sensor constants](#).
- _result* The function call result. [NO_ERR](#) or Communications specific errors.

**5.18.2.119 #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:

_port The sensor port. See [NBC Input port constants](#).

_cal The new pressure calibration value.

_result The function call result.

5.18.2.120 #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:

_port The sensor port. See [NBC Input port constants](#).

_mode The Color2 mode. See [HiTechnic Color2 constants](#).

_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

5.18.2.121 #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:

_port The sensor port. See [NBC Input port constants](#).

_mode The IRSeeker2 mode. See [HiTechnic IRSeeker2 constants](#).

_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

5.18.2.122 #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:

- port* The sensor port. See [NBC Input port constants](#).
- _bStd* Configure in standard or long-range mode.

5.18.2.123 #define SetSensorHTGyro(_port) __SetSensorHTGyro(_port)

Set sensor as HiTechnic Gyro. Configure the sensor on the specified port as a HiTechnic Gyro sensor.

Parameters:

- port* The sensor port. See [NBC Input port constants](#).

5.18.2.124 #define SetSensorHTMagnet(_port) __SetSensorHTGyro(_port)

Set sensor as HiTechnic Magnet. Configure the sensor on the specified port as a HiTechnic Magnet sensor.

Parameters:

- port* The sensor port. See [NBC Input port constants](#).

5.18.2.125 #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:

_port The sensor port. See [NBC Input port constants](#).
_value The digital pin output values. See [SuperPro digital pin constants](#).
_result The function call result.

5.18.2.126 #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:

_port The sensor port. See [NBC Input port constants](#).
_value The digital pin control value. See [SuperPro digital pin constants](#). OR into this value the pins that you want to be output pins. The pins not included in the value will be input pins.
_result The function call result.

5.18.2.127 #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:

_port The sensor port. See [NBC Input port constants](#).
_dac The analog output index. See [HiTechnic SuperPro analog output index constants](#).
_mode The analog output mode. See [SuperPro analog output mode constants](#).
_freq The analog output frequency. Between 1 and 8191.
_volt The analog output voltage level. A 10 bit value (0..1023).
_result The function call result.

```
5.18.2.128 #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:

_port The sensor port. See [NBC Input port constants](#).
_value The digital pin output values. See [SuperPro digital pin constants](#).
_result The function call result.

```
5.18.2.129 #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:

_port The sensor port. See [NBC Input port constants](#).
_value The digital pin control value. See [SuperPro digital pin constants](#). OR into this value the pins that you want to be output pins. The pins not included in the value will be input pins.
_result The function call result.

```
5.18.2.130 #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:

- _port* The sensor port. See [NBC Input port constants](#).
- _value* The LED value. See [SuperPro LED control constants](#).
- _result* The function call result.

5.18.2.131 #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:

- _port* The sensor port. See [NBC Input port constants](#).
- _value* The program control value.
- _result* The function call result.

5.18.2.132 #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:

- _port* The sensor port. See [NBC Input port constants](#).
- _value* The strobe value. See [SuperPro Strobe control constants](#).
- _result* The function call result.

5.19 SuperPro analog output mode constants

Constants for controlling the 2 analog output modes.

Defines

- #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

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 $\sim 3.2\text{mV}$ 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 Define 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_SAWNNEGWAVE 4

Negative going sawtooth output.

5.19.2.4 #define DAC_MODE_SAWPOSWAVE 3

Positive going sawtooth output.

5.19.2.5 #define DAC_MODE_SINEWAVE 1

Sine wave output.

5.19.2.6 #define DAC_MODE_SQUAREWAVE 2

Square wave output.

5.19.2.7 #define DAC_MODE_TRIANGLEWAVE 5

Triangle wave output.

5.20 SuperPro LED control constants

Constants for controlling the 2 onboard LEDs.

Defines

- #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 Define 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.

Defines

- #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 Define 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.

Defines

- #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 Define 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.

Defines

- #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_DIST, 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)`

Powerdown 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)`

Take NXTLineLeader line snapshot.

- #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 power function mode.
- #define `NRLinkSetRCX(_port, _i2caddr, _result)` __I2CSendCmd(_port, _i2caddr, NRLINK_CMD_SET_RCX, _result)
Configure NRLink in RCX mode.

- #define **NRLLinkSetTrain**(_port, _i2caddr, _result) __I2CSendCmd(_port, _i2caddr, NRLINK_CMD_SET_TRAIN, _result)
Configure NRLLink in IR train mode.
- #define **NRLLinkTxRaw**(_port, _i2caddr, _result) __I2CSendCmd(_port, _i2caddr, NRLINK_CMD_TX_RAW, _result)
Configure NRLLink in raw IR transmit mode.
- #define **ReadNRLLinkStatus**(_port, _i2caddr, _value, _result) __ReadNRLLinkStatus(_port, _i2caddr, _value, _result)
Read NRLLink status.
- #define **RunNRLLinkMacro**(_port, _i2caddr, _macro, _result) __RunNRLLinkMacro(_port, _i2caddr, _macro, _result)
Run NRLLink macro.
- #define **WriteNRLLinkBytes**(_port, _i2caddr, _bytes, _result) __WriteNRLLinkBytes(_port, _i2caddr, _bytes, _result)
Write data to NRLLink.
- #define **ReadNRLLinkBytes**(_port, _i2caddr, _bytes, _result) __ReadNRLLinkBytes(_port, _i2caddr, _bytes, _result)
Read data from NRLLink.
- #define **MSIRTrain**(_port, _i2caddr, _channel, _func, _result) __MSIRTrain(_port, _i2caddr, _channel, _func, FALSE, _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, TRUE, _result)

MSPFTrain function.

- #define **MSRCXSetNRLLinkPort**(_port, _i2caddr) __MSRCXSetNRLLink(_port, _i2caddr)

MSRCXSetIRLinkPort 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)
MSRCXRev function.
- #define **MSRCXSelectDisplay(_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*, *_pwrvval*) __MSRCXSetPower(*_outputs*, *_pwrsrc*, *_pwrvval*)
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*)

MSRCX`SgnVar` function.

- #define **MSRCXStartTask**(_t) __MSRCXStartTask(_t)
MSRCX`StartTask` function.
- #define **MSRCXStopAllTasks**() __MSRCXOpNoArgs(RCX_-
StopAllTasksOp)
MSRCX`StopAllTasks` function.
- #define **MSRCXStopTask**(_t) __MSRCXStopTask(_t)
MSRCX`StopTask` function.
- #define **MSRCXSubVar**(_varnum, _src, _value) __MSRCXVarOp(RCX_-
SubVarOp, _varnum, _src, _value)
MSRCX`SubVar` function.
- #define **MSRCXSumVar**(_varnum, _src, _value) __MSRCXVarOp(RCX_-
SumVarOp, _varnum, _src, _value)
MSRCX`SumVar` function.
- #define **MSRCXToggle**(_outputs) __MSRCXSetDirection(_outputs, RCX_-
OUT_TOGGLE)
MSRCX`Toggle` function.
- #define **MSRCXUnlock**() __MSRCXUnlock()
MSRCX`Unlock` function.
- #define **MSRCXUnmuteSound**() __MSRCXOpNoArgs(RCX_-
UnmuteSoundOp)
MSRCX`UnmuteSound` function.
- #define **MSScoutCalibrateSensor**() __MSRCXOpNoArgs(RCX_-
LSCalibrateOp)
MSScout`CalibrateSensor` function.
- #define **MSScoutMuteSound**() __MSScoutMuteSound()
MSScout`MuteSound` function.
- #define **MSScoutSelectSounds**(_grp) __MSScoutSelectSounds(_grp)
MSScout`SelectSounds` function.
- #define **MSScoutSendVLL**(_src, _value) __MSScoutSendVLL(_src, _value)
MSScout`SendVLL` 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 Define 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _channel* The IR Train channel. See [IR Train channel constants](#).
- _func* The IR Train function. See [PF/IR Train function constants](#)
- _result* The function call result. [NO_ERR](#) or [Communications specific errors](#).

```
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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _channel* The Power Function channel. See [Power Function channel constants](#).
- _outa* The Power Function command for output A. See [Power Function command constants](#).
- _outb* The Power Function command for output B. See [Power Function command constants](#).
- _result* The function call result. [NO_ERR](#) or [Communications specific errors](#).

```
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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _channel* The Power Function channel. See [Power Function channel constants](#).
- _outa* The Power Function PWM command for output A. See [Power Function PWM option constants](#).
- _outb* The Power Function PWM command for output B. See [Power Function PWM option constants](#).
- _result* The function call result. [NO_ERR](#) or [Communications specific errors](#).

```
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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_nibble0 The first raw data nibble.
_nibble1 The second raw data nibble.
_nibble2 The third raw data nibble.
_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

```
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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_count The number of times to repeat the command.
_delay The number of milliseconds to delay between each repetition.
_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

```
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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_channel The Power Function channel. See [Power Function channel constants](#).
_out The Power Function output. See [Power Function output constants](#).
_func The Power Function CST function. See [Power Function CST options constants](#).
_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

```
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:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.
_channel The Power Function channel. See [Power Function channel constants](#).
_out The Power Function output. See [Power Function output constants](#).
_func The Power Function PWM function. See [Power Function PWM option constants](#).
_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_channel The Power Function channel. See [Power Function channel constants](#).
_out The Power Function output. See [Power Function output constants](#).
_pin The Power Function pin. See [Power Function pin constants](#).
_func The Power Function single pin function. See [Power Function single pin function constants](#).
_cont Control whether the mode is continuous or timeout.
_result The function call result. [NO_ERR](#) or [Communications specific errors](#).

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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _channel* The Power Function channel. See [Power Function channel constants](#).
- _func* The Power Function train function. See [PF/IR Train function constants](#).
- _result* The function call result. `NO_ERR` or [Communications specific errors](#).

**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:

- _varnum* The variable number to change.
- _src* The RCX source. See [RCX and Scout source constants](#).
- _value* The RCX value.

**5.23.2.26 #define MSRCXAddToDatalog(_src, _value) __-
_MSRCXAddToDatalog(_src, _value)**

MSRCXAddToDatalog function. Send the AddToDatalog command to an RCX.

Parameters:

- _src* The RCX source. See [RCX and Scout source constants](#).
- _value* 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:

_varnum The variable number to change.
_src The RCX source. See [RCX and Scout source constants](#).
_value 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:

_result 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:

_evt The event number.
_low The low threshold.
_hi The high threshold.
_hyst 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:

_counter 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:

_port 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:

_timer The timer to clear.

5.23.2.37 #define MSRCXCreateDatalog(_size) __MSRCXCreateDatalog(_size)

MSRCXCreateDatalog function. Send the CreateDatalog command to an RCX.

Parameters:

_size The new datalog size.

5.23.2.38 #define MSRCXDecCounter(_counter) __MSRCXDecCounter(_- counter)

MSRCXDecCounter function. Send the DecCounter command to an RCX.

Parameters:

_counter The counter to decrement.

5.23.2.39 #define MSRCXDeleteSub(_s) __MSRCXDeleteSub(_s)

MSRCXDeleteSub function. Send the DeleteSub command to an RCX.

Parameters:

_s 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:

_t 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:

_outputs The RCX output(s) to disable. See [RCX output constants](#).

**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:

_varnum The variable number to change.

_src The RCX source. See [RCX and Scout source constants](#).

_value 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:

_outputs The RCX output(s) to enable. See [RCX output constants](#).

5.23.2.46 #define MSRCXEvent(_src, _value) __MSRCXEvent(_src, _value)

MSRCXEvent function. Send the Event command to an RCX.

Parameters:

_src The RCX source. See [RCX and Scout source constants](#).

_value 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:

_outputs The RCX output(s) to float. See [RCX output constants](#).

**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:

_outputs The RCX output(s) to set forward. See [RCX output constants](#).

5.23.2.49 #define MSRCXIncCounter(_counter) __MSRCXIncCounter(_-counter)

MSRCXIncCounter function. Send the IncCounter command to an RCX.

Parameters:

_counter The counter to increment.

5.23.2.50 #define MSRCXIvertOutput(_outputs) __-MSRCXSetGlobalDirection(_outputs, RCX_OUT_REV)

MSRCXIvertOutput function. Send the InvertOutput command to an RCX.

Parameters:

_outputs The RCX output(s) to invert. See [RCX output constants](#).

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:

_varnum The variable number to change.

_src The RCX source. See [RCX and Scout source constants](#).

_value 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:

_outputs The RCX output(s) to obvert. See [RCX output constants](#).

**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:

_outputs The RCX output(s) to turn off. See [RCX output constants](#).

**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:

_outputs The RCX output(s) to turn on. See [RCX output constants](#).

**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:

_outputs The RCX output(s) to turn on. See [RCX output constants](#).

_ms 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:

_outputs The RCX output(s) to turn on in the forward direction. See [RCX output constants](#).

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:

_outputs The RCX output(s) to turn on in the reverse direction. See [RCX output constants](#).

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:

_varnum The variable number to change.
_src The RCX source. See [RCX and Scout source constants](#).
_value 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:

_snd 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:

_freq The frequency of the tone to play.

_duration 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:

_varnum The variable containing the tone frequency to play.

_duration 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:

- _src* The RCX source. See [RCX and Scout source constants](#).
- _value* The RCX value.
- _result* 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:

- _memaddress* The RCX memory address.
- _result* 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:

- _cmd* The RCX IR remote command to send. See [RCX IR remote constants](#).

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:

_outputs The RCX output(s) to reverse direction. See [RCX output constants](#).

**5.23.2.70 #define MSRCXSelectDisplay(_src, _value) __-
MSRCXSelectDisplay(_src, _value)**

MSRCXSelectDisplay function. Send the SelectDisplay command to an RCX.

Parameters:

_src The RCX source. See [RCX and Scout source constants](#).

_value The RCX value.

**5.23.2.71 #define MSRCXSelectProgram(_prog) __MSRCXSelectProgram(_-
prog)**

MSRCXSelectProgram function. Send the SelectProgram command to an RCX.

Parameters:

_prog 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:

_first The first byte address.

_count 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:

_dstsrc The RCX destination source. See [RCX and Scout source constants](#).
_dstval The RCX destination value.
_src The RCX source. See [RCX and Scout source constants](#).
_value 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:

_outputs The RCX output(s) to set direction. See [RCX output constants](#).
_dir The RCX output direction. See [RCX output direction constants](#).

5.23.2.75 #define MSRCXSetEvent(_evt, _src, _type) __MSRCXSetEvent(_evt, _src, _type)

MSRCXSetEvent function. Send the SetEvent command to an RCX.

Parameters:

_evt The event number to set.
_src The RCX source. See [RCX and Scout source constants](#).
_type The event type.

**5.23.2.76 #define MSRCXSetGlobalDirection(_outputs,
_dir) __MSRCXSetGlobalDirection(_outputs, _dir)**

MSRCXSetGlobalDirection function. Send the SetGlobalDirection command to an RCX.

Parameters:

_outputs The RCX output(s) to set global direction. See [RCX output constants](#).
_dir The RCX output direction. See [RCX output direction constants](#).

**5.23.2.77 #define MSRCXSetGlobalOutput(_outputs,
_mode) __MSRCXSetGlobalOutput(_outputs, _mode)**

MSRCXSetGlobalOutput function. Send the SetGlobalOutput command to an RCX.

Parameters:

_outputs The RCX output(s) to set global mode. See [RCX output constants](#).
_mode The RCX output mode. See [RCX output mode constants](#).

**5.23.2.78 #define MSRCXSetMaxPower(_outputs, _pwrsrc,
_pwrval) __MSRCXSetMaxPower(_outputs, _pwrsrc, _pwrval)**

MSRCXSetMaxPower function. Send the SetMaxPower command to an RCX.

Parameters:

_outputs The RCX output(s) to set max power. See [RCX output constants](#).
_pwrsrc The RCX source. See [RCX and Scout source constants](#).
_pwrval The RCX value.

5.23.2.79 #define MSRCXSetMessage(_msg) __MSRCXSetMessage(_msg)

MSRCXSetMessage function. Send the SetMessage command to an RCX.

Parameters:

_msg 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:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr 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:

_outputs The RCX output(s) to set mode. See [RCX output constants](#).

_mode The RCX output mode. See [RCX output mode constants](#).

**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:

_outputs The RCX output(s) to set power. See [RCX output constants](#).

_pwrsrc The RCX source. See [RCX and Scout source constants](#).

_pwrval The RCX value.

5.23.2.83 #define MSRCXSetPriority(_p) __MSRCXSetPriority(_p)

MSRCXSetPriority function. Send the SetPriority command to an RCX.

Parameters:

_p 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:

_port The RCX sensor port.

_mode 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:

_port The RCX sensor port.

_type The RCX sensor type.

5.23.2.86 #define MSRCXSetSleepTime(_t) __MSRCXSetSleepTime(_t)

MSRCXSetSleepTime function. Send the SetSleepTime command to an RCX.

Parameters:

_t The new sleep time value.

5.23.2.87 #define MSRCXSetTxPower(_pwr) __MSRCXSetTxPower(_pwr)

MSRCXSetTxPower function. Send the SetTxPower command to an RCX.

Parameters:

_pwr 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:

_src The RCX source. See [RCX and Scout source constants](#).

_value The RCX value.

_precision 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:

_varnum The variable number to change.

_src The RCX source. See [RCX and Scout source constants](#).

_value The RCX value.

**5.23.2.90 #define MSRCXSetWatch(_hours, _minutes) __MSRCXSetWatch(_-
 hours, _minutes)**

MSRCXSetWatch function. Send the SetWatch command to an RCX.

Parameters:

_hours The new watch time hours value.
_minutes 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:

_varnum The variable number to change.
_src The RCX source. See [RCX and Scout source constants](#).
_value The RCX value.

5.23.2.92 #define MSRCXStartTask(_t) __MSRCXStartTask(_t)

MSRCXStartTask function. Send the StartTask command to an RCX.

Parameters:

_t 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:

_t 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:

_varnum The variable number to change.
_src The RCX source. See [RCX and Scout source constants](#).
_value 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:

_varnum The variable number to change.
_src The RCX source. See [RCX and Scout source constants](#).
_value 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:

_outputs The RCX output(s) to toggle. See [RCX output constants](#).

```
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 mind-sensors 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _reg* The device register to read.
- _bytes* The number of bytes to read. Only 1, 2, or 4 byte values are supported.
- _out* The value read from the device.
- _result* 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:

_grp 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:

_src The Scout source. See [RCX and Scout source constants](#).

_value 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:

_ctr The counter for which to set the limit.

_src The Scout source. See [RCX and Scout source constants](#).

_value The Scout value.

5.23.2.106 #define MSScoutSetEventFeedback(_src, _value) __MSScoutSetEventFeedback(_src, _value)

MSScoutSetEventFeedback function. Send the SetEventFeedback command to a Scout.

Parameters:

_src The Scout source. See [RCX and Scout source constants](#).

_value The Scout value.

5.23.2.107 #define MSScoutSetLight(_x) __MSScoutSetLight(_x)

MSScoutSetLight function. Send the SetLight command to a Scout.

Parameters:

_x Set the light on or off using this value. See [Scout light constants](#).

5.23.2.108 #define MSScoutSetScoutMode(_mode) __MSScoutSetScoutMode(_mode)

MSScoutSetScoutMode function. Send the SetScoutMode command to a Scout.

Parameters:

_mode Set the scout mode. See [Scout mode constants](#).

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:

_m Scout motion rule. See [Scout motion rule constants](#).

_t Scout touch rule. See [Scout touch rule constants](#).

_l Scout light rule. See [Scout light rule constants](#).

_tm Scout transmit rule. See [Scout transmit rule constants](#).

_fx Scout special effects rule. See [Scout special effect constants](#).

5.23.2.110 #define MSScoutSetSensorClickTime(_src, _value) __MSScoutSetSensorClickTime(_src, _value)

MSScoutSetSensorClickTime function. Send the SetSensorClickTime command to a Scout.

Parameters:

_src The Scout source. See [RCX and Scout source constants](#).
_value The Scout value.

**5.23.2.111 #define MSScoutSetSensorHysteresis(_src,
_value) __MSScoutSetSensorHysteresis(_src, _value)**

MSScoutSetSensorHysteresis function. Send the SetSensorHysteresis command to a Scout.

Parameters:

_src The Scout source. See [RCX and Scout source constants](#).
_value The Scout value.

**5.23.2.112 #define MSScoutSetSensorLowerLimit(_src,
_value) __MSScoutSetSensorLowerLimit(_src, _value)**

MSScoutSetSensorLowerLimit function. Send the SetSensorLowerLimit command to a Scout.

Parameters:

_src The Scout source. See [RCX and Scout source constants](#).
_value The Scout value.

**5.23.2.113 #define MSScoutSetSensorUpperLimit(_src,
_value) __MSScoutSetSensorUpperLimit(_src, _value)**

MSScoutSetSensorUpperLimit function. Send the SetSensorUpperLimit command to a Scout.

Parameters:

_src The Scout source. See [RCX and Scout source constants](#).
_value 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:

- _tmr* The timer for which to set a limit.
- _src* The Scout source. See [RCX and Scout source constants](#).
- _value* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _modifier* The key modifier.
- _character* The character.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.

_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.

_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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 [NXTLineLeaderPowerDown](#) command. The port must be configured as a Lowspeed port before using this function.

Parameters:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.

_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _macro* The EEPROM macro address.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _servo* The servo number. See [MindSensors NXTServo servo numbers](#) group.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_channel The power function IR receiver channel. See the [PFMate channel constants](#) group.
_motors The motor(s) to control. See the [PFMate motor constants](#) group.
_cmdA The power function command for motor A. See the [Power Function command constants](#) group.
_spdA The power function speed for motor A.
_cmdB The power function command for motor B. See the [Power Function command constants](#) group.
_spdB The power function speed for motor B.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_channel The power function IR receiver channel. See the [PFMate channel constants](#) group.
_b1 Raw byte 1.
_b2 Raw byte 2.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The sensitivity value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The X offset value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The X range value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The Y offset value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.

_out The Y range value.

_result 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:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.

_out The Z offset value.

_result 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:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.

_out The Z range value.

_result 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _out* The distance value.
- _result* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _out* The maximum distance value.
- _result* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _out* The minimum distance value.
- _result* 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The module type value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The num points value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.

_out The voltage value.

_result 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:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.

_bytes A byte array that will contain the data read from the device on output.

_result 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:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.

_value The mindsensors NRLink status.

_result 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _out* The NXTLineLeader average value.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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 (b1111111). The port must be configured as a Lowspeed port before using this function.

Parameters:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _out* The NXTLineLeader result value.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The NXTLineLeader steering value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The NXTPowerMeter capacity used value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The NXTPowerMeter elapsed time value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The NXTPowerMeter error count value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The NXTPowerMeter maximum current value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The NXTPowerMeter maximum voltage value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The NXTPowerMeter minimum current value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The NXTPowerMeter minimum voltage value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The NXTPowerMeter present current.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The NXTPowerMeter present power value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The NXTPowerMeter present voltage.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The NXTPowerMeter total power consumed value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_out The NXTServo battery voltage.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_servo The servo number. See [MindSensors NXTServo servo numbers](#) group.
_out The specified servo's position value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_servo The servo number. See [MindSensors NXTServo servo numbers](#) group.
_out The specified servo's speed value.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _x* The output x-axis acceleration.
- _y* The output y-axis acceleration.
- _z* The output z-axis acceleration.
- _result* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _value* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _value* 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_b1 The button set 1 values. See [MindSensors PSP-Nx button set 1 constants](#).
_b2 The button set 2 values. See [MindSensors PSP-Nx button set 2 constants](#).
_xleft The left joystick x value.
_yleft The left joystick y value.
_xright The right joystick x value.
_yright The right joystick y value.
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_value 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:

_port The sensor port. See [NBC Input port constants](#).

_value 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:

_port The sensor port. See [NBC Input port constants](#).

_sec The seconds.

_min The minutes.

_hrs The hours.

_dow The day of week number.

_date The day.

_month The month.

_year The year.

_result 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:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.

- _x* The output x-axis tilt.
- _y* The output y-axis tilt.
- _z* The output z-axis tilt.
- _result* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _value* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _macro* The address of the macro to execute.
- _result* 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_level The sensitivity level. See [MindSensors ACCL-Nx sensitivity level constants](#).
_result 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_value The new Kd factor.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.

_value The new Kd value.

_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.

_value The new Ki factor.

_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.

_value The new Ki value.

_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _value* The new Kp factor.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _value* The new Kp value.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _value* The new setpoint value (10..80).
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _i2caddr* The sensor I2C address. See sensor documentation for this value.
- _servo* The servo number. See [MindSensors NXTServo servo numbers](#) group.
- _pos* The servo position. See [MindSensors NXTServo position constants](#) group.
- _result* A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_servo The servo number. See [MindSensors NXTServo servo numbers](#) group.
_qpos The servo quick position. See [MindSensors NXTServo quick position constants](#) group.
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The sensor port. See [NBC Input port constants](#).
_i2caddr The sensor I2C address. See sensor documentation for this value.
_servo The servo number. See [MindSensors NXTServo servo numbers](#) group.
_speed The servo speed. (0..255)
_result A status code indicating whether the operation completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The port to configure. See [NBC Input port constants](#).

**5.23.2.203 #define SetSensorMSDRODIInactive(_port) __-
SetSensorMSDRODIInactive(_port)**

Configure a mindsensors DROD inactive sensor. Configure the specified port for an inactive mindsensors DROD sensor.

Parameters:

port The port to configure. See [NBC Input port constants](#).

**5.23.2.204 #define SetSensorMSPressure(_port) __SetSensorMSPressure(_-
port)**

Configure a mindsensors pressure sensor. Configure the specified port for a mindsensors pressure sensor.

Parameters:

port The port to configure. See [NBC Input port constants](#).

**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:

port The port to configure. See [NBC Input port constants](#).

**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:

port The port to configure. See [NBC Input port constants](#).

**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:

_port The port to configure. See [NBC Input port constants](#).

**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:

_port The sensor port. See [NBC Input port constants](#).

_i2caddr The sensor I2C address. See sensor documentation for this value.

_bytes A byte array containing the data to write.

_result 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.

Defines

- #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 Define Documentation

5.24.2.1 #define **RFIDInit**(*_port*, *_result*) __RFIDInit(*_port*, *_result*)

RFIDInit function. Initialize the Codatex RFID sensor.

Parameters:

port The port to which the Codatex RFID sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.

result 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:

- _port* The port to which the Codatex RFID sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _mode* The RFID sensor mode. See the [Codatex RFID sensor modes](#) group.
- _result* 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:

- _port* The port to which the Codatex RFID sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _output* The five bytes of RFID data.
- _result* 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:

- _port* The port to which the Codatex RFID sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _output* The five bytes of RFID data.
- _result* 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:

- _port* The port to which the Codatex RFID sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _output* The five bytes of RFID data.
- _result* 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:

- _port* The port to which the Codatex RFID sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _result* The RFID sensor status.

5.24.2.7 #define RFIDStop(_port, _result) __RFIDStop(_port, _result)

RFIDStop function. Stop the Codatex RFID sensor.

Parameters:

- _port* The port to which the Codatex RFID sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _result* 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.

Defines

- `#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 Define 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:

- _port* The port to which the Dexter Industries IMU Accl sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _vector* A variable of type TVector which will contain the scaled X, Y, anx Z values.
- _result* 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:

- _port* The port to which the Dexter Industries IMU Accl sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _vector* A variable of type TVector which will contain the scaled X, Y, anx Z values.
- _result* 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:

- _port* The port to which the Dexter Industries IMU Accl sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _vector* A variable of type TVector which will contain the raw X, Y, anx Z values.
- _result* 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:

- _port* The port to which the Dexter Industries IMU Accl sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _x* The X axis 10-bit drift value.

- _y* The Y axis 10-bit drift value.
- _z* The Z axis 10-bit drift value.
- _result* 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:

- _port* The port to which the Dexter Industries IMU Accl sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _vector* A variable of type TVector which will contain the raw X, Y, anx Z values.
- _result* 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:

- _port* The port to which the Dexter Industries IMU Accl sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _out* The output status value.
- _result* 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:

_port The port to which the Dexter Industries GPS sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
_result 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:

_port The port to which the Dexter Industries GPS sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
_result 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:

_port The port to which the Dexter Industries GPS sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
_result 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 (dddddd; Positive = North; Negative = South).

Parameters:

_port The port to which the Dexter Industries GPS sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
_result 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:

- _port* The port to which the Dexter Industries GPS sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
_result 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:

- _port* The port to which the Dexter Industries GPS sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
_result The relative heading in degrees

**5.25.2.13 #define ReadSensorDIGPSSStatus(_port,
_status) __ReadSensorDIGPSSStatus(_port, _status)**

ReadSensorDIGPSSStatus function. Read the status of the GPS satellite link.

Parameters:

- _port* The port to which the Dexter Industries GPS sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
_status 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:

- _port* The port to which the Dexter Industries GPS sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
_result 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:

- _port* The port to which the Dexter Industries GPS sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
_result 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:

- _port* The port to which the Dexter Industries IMU Gyro sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
_vector A variable of type TVector which will contain the scaled X, Y, and Z values.
_result 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:

- _port* The port to which the Dexter Industries IMU Gyro sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _vector* A variable of type TVector which will contain the raw X, Y, anx Z values.
- _result* 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:

- _port* The port to which the Dexter Industries IMU Gyro sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _out* The output status value.
- _result* 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:

- _port* The port to which the Dexter Industries IMU Gyro sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _out* The output temperature value.
- _result* 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:

- _port* The port to which the Dexter Industries IMU Accl sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _result* 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:

- _port* The port to which the Dexter Industries IMU Accl sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _x* The X axis 10-bit drift value.
- _y* The Y axis 10-bit drift value.
- _z* The Z axis 10-bit drift value.
- _result* 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:

- _port* The port to which the Dexter Industries IMU Accl sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.

_mode The mode of the device (2G, 4G, or 8G). See the [Dexter Industries IMU Accelerometer mode control register constants](#) group. You may use a constant or a variable.

_result 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:

_port The port to which the Dexter Industries GPS sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.

_lat The latitude of the waypoint.

_long The longitude of the waypoint.

_result 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:

_port The port to which the Dexter Industries IMU Gyro sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.

_result 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:

- _port* The port to which the Dexter Industries IMU Gyro sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _scale* The full scale of the device (250dps, 500dps, or 2000dps). See the [Dexter Industries IMU Gyro control register 4 constants](#) group. You may use a constant or a variable.
- _odr* The output data rate of the device (100hz, 200hz, 400hz, or 800hz). See the [Dexter Industries IMU Gyro control register 1 constants](#) group. You may use a constant or a variable.
- _bw* The bandwidth of the device. See the [Dexter Industries IMU Gyro control register 1 constants](#) group. You may use a constant or a variable.
- _result* 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.

Defines

- #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(_port, _packet, _result)
ReadSensorMIXG1300L function.

5.26.1 Detailed Description

Functions for accessing and modifying Microinfinity devices.

5.26.2 Define 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _packet* The output XK1300L data structure. See [TXGPacket](#).
- _result* 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:

- _port* The sensor port. See [NBC Input port constants](#).
- _result* 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:

_port The sensor port. See [NBC Input port constants](#).

_result 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:

_port The sensor port. See [NBC Input port constants](#).

_scale This value must be a constant. See [Microinfinity CruizCore XG1300L](#).

_result The function call result.

5.27 RIC Macro Wrappers

Macro wrappers for use in defining RIC byte arrays.

Defines

- #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 Define Documentation

5.27.2.1 #define RICArg(_arg) ((_arg)|0x1000)

Output an RIC parameterized argument.

Parameters:

_arg 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:

_X The X coordinate.

_Y 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:

_Pt An ImgPoint. See [RICImgPoint](#).

_W The rectangle width.

_H The rectangle height.

5.27.2.4 #define RICMapArg(_mapidx, _arg) ((_arg)|0x1000|(((~-mapidx)&0xF)<<8))

Output an RIC parameterized and mapped argument.

Parameters:

_mapidx The varmap data address.

_arg The parameterized argument you want to pass through a varmap.

5.27.2.5 #define RICMapElement(_Domain, _Range) (_Domain)&0xFF, (_Domain)>>8, (_Range)&0xFF, (_Range)>>8

Output an RIC map element.

Parameters:

_Domain The map element domain.

_Range The map element range.

5.27.2.6 #define RICMapFunction(_MapElement, ...) _MapElement, __VA_ARGS__

Output an RIC VarMap function.

Parameters:

_MapElement An entry in the varmap function. At least 2 elements are required.
See [RICMapElement](#).

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:

_CopyOptions Circle copy options. See [Drawing option constants](#).

_Point The circle's center point. See [RICImgPoint](#).

_Radius 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:

_CopyOptions CopyBits copy options. See [Drawing option constants](#).

_DataAddr The address of the sprite from which to copy data.

_SrcRect The rectangular portion of the sprite to copy. See [RICImgRect](#).

_DstPoint The LCD coordinate to which to copy the data. See [RICImgPoint](#).

```
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:

_Options RIC options.

_Width The total RIC width.

_Height 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:

_CopyOptions Ellipse copy options. See [Drawing option constants](#).

_Point The center of the ellipse. See [RICImgPoint](#).

_RadiusX The x-axis radius of the ellipse.

_RadiusY 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:

_CopyOptions Line copy options. See [Drawing option constants](#).

_Point1 The starting point of the line. See [RICImgPoint](#).

_Point2 The ending point of the line. See [RICImgPoint](#).

```
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:

_CopyOptions NumBox copy options. See [Drawing option constants](#).

_Point The numbox bottom left corner. See [RICImgPoint](#).

_Value 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:

_CopyOptions Pixel copy options. See [Drawing option constants](#).

_Point The pixel coordinate. See [RICImgPoint](#).

_Value 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:

_CopyOptions Polygon copy options. See [Drawing option constants](#).
_Count The number of points in the polygon.
_ThePoints The list of polygon points. See [RICPolygonPoints](#).

```
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:

_CopyOptions Rect copy options. See [Drawing option constants](#).
_Point The rectangle's top left corner. See [RICImgPoint](#).
_Width The rectangle's width.
_Height The rectangle's height.

```
5.27.2.16 #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:

_DataAddr The address of the sprite.

_Rows The number of rows of data.

_BytesPerRow The number of bytes per row.

_SpriteData The actual sprite data. See [RICSpriteData](#).

**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:

_DataAddr The address of the varmap.

_MapCount The number of points in the function.

_MapFunction The definition of the varmap function. See [RICMapFunction](#).

**5.27.2.18 #define RICPolygonPoints(_pPoint1, _pPoint2, ...) _pPoint1,
_pPoint2, __VA_ARGS__**

Output RIC polygon points.

Parameters:

_pPoint1 The first polygon point. See [RICImgPoint](#).

_pPoint2 The second polygon point (at least 3 points are required). See [RICImgPoint](#).

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.

Defines

- #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 Define 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.

Defines

- #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

5.29.1 Detailed Description

Constant numeric IDs for all the NXT firmware modules.

5.29.2 Define 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 **#define InputModuleID 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

- [Property constants](#)

Use these constants for specifying the property for the GetProperty and SetProperty direct commands.

- [Data type limits](#)

Constants that define various data type limits.

Defines

- #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 Define 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 RADIAN_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 CodaTex.

Modules

- **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.

- **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.

- [HiTechnic API Functions](#)

Functions for accessing and modifying HiTechnic devices.

- [MindSensors API Functions](#)

Functions for accessing and modifying MindSensors devices.

- [Codatex API Functions](#)

Functions for accessing and modifying Codatex devices.

- [Dexter Industries API Functions](#)

Functions for accessing and modifying Dexter Industries devices.

- [Microinfinity API Functions](#)

Functions for accessing and modifying Microinfinity devices.

5.31.1 Detailed Description

Documentation for NXT devices made by companies other than LEGO such as HiTechnic, mindsensors.com, and CodaTex.

5.32 Standard-C API functions

Documentation for various Standard-C library routines.

Modules

- [cstdlib API](#)

Standard C cstdlib API functions.

- [cmath API](#)

Standard C cmath 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 begin modes](#)

Constants that are used to specify the polygon surface begin mode.

- [Graphics library actions](#)

Constants that are used to specify a graphics library action.

- [Graphics library settings](#)

Constants that are used to configure the graphics library settings.

- [Graphics library cull mode](#)

Constants to use when setting the graphics library cull mode.

Defines

- `#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(_glObjectId, _glAction, _glValue) __glObjectAction(_-glObjectId, _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(_glObjectId)` __`glCallObject(_glObjectId)`
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 Define 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:

_glValue 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:

_glValue 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:

_glValue The value to add to the Z axis angle.

5.33.2.4 #define glAddVertex(_glX, _glY, _glZ) __glAddVertex(_glX, _glY, _glZ)

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:

`_glX` The X axis coordinate.

`_glY` The Y axis coordinate.

`_glZ` 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:

`_glBeginMode` The desired mode. See [Graphics library begin modes](#).

5.33.2.6 #define glBeginObject(_glObjId) __glBeginObject(_glObjId)

Begin defining an object. Start the process of defining a graphics library object using low level functions such as `glBegin`, `glAddVertex`, and `glEnd`.

Parameters:

`_glObjId` 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,
_glObjId) __glBox(_glMode, _glSizeX, _glSizeY, _glSizeZ, _glObjId)**

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:

- `_glMode` The begin mode for each surface. See [Graphics library begin modes](#).
- `_glSizeX` The X axis size (width).
- `_glSizeY` The Y axis size (height).
- `_glSizeZ` The Z axis size (depth).
- `_glObjId` The object ID of the new object.

5.33.2.9 #define glCallObject(_glObjectId) __glCallObject(_glObjectId)

Call a graphic object. Tell the graphics library that you want it to include the specified object in the render.

Parameters:

- `_glObjectId` 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:

- `_glAngle` The angle in degrees.
- `_glResult` The cosine value scaled by 32768.

5.33.2.11 #define glCube(_glMode, _glSize, _glObjId) __glBox(_glMode, _glSize, _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:

_glMode The begin mode for each surface. See [Graphics library begin modes](#).

_glSize The cube's width, height, and depth.

_glObjId 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:

_glObjectId The object id.

_glAction The action to perform on the object. See [Graphics library actions](#).

_glValue 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:

_glMode The begin mode for each surface. See [Graphics library begin modes](#).

_glSizeX The X axis size (width).

_glSizeY The Y axis size (height).

_glSizeZ The Z axis size (depth).

_glObjId 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:

_glType The setting type. See [Graphics library settings](#).

_glValue The setting value. For culling modes see [Graphics library cull mode](#).

5.33.2.19 #define glSetAngleX(_glValue) __glSetAngleX(_glValue)

Set the X axis angle. Set the X axis angle to the specified value.

Parameters:

_glValue 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:

_glValue 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:

_glValue 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:

_glAngle The angle in degrees.

_glResult The sine value scaled by 32768.

5.34 Output module functions

Functions for accessing and modifying output module features.

Defines

- #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)
Run motors forward regulated.
- #define **OnRevReg**(_ports, _pwr, _regmode) OnRevRegEx(_ports, _pwr, _-
regmode, RESET_BLOCKANDTACHO)
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 with PID factors.
- #define **OnRevRegPID**(_ports, _pwr, _regmode, _p, _i, _d) OnRevRegExPID(_-
ports, _pwr, _regmode, RESET_BLOCKANDTACHO, _p, _i, _d)

Run motors reverse 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_BLOCKANDTACHO)

Run motors forward synchronised.

- #define `OnRevSync(_ports, _pwr, _turnpct)` OnRevSyncEx(_ports, _pwr, _-
turnpct, RESET_BLOCKANDTACHO)

Run motors backward synchronised.

- #define `OnFwdSyncPID(_ports, _pwr, _turnpct, _p, _i, _d)` __-
OnFwdSyncExPID(_ports, _pwr, _turnpct, RESET_BLOCKANDTACHO, _p, _i, _d)

Run motors forward synchronised with PID factors.

- #define `OnRevSyncPID(_ports, _pwr, _turnpct, _p, _i, _d)` OnRevSyncExPID(_-
ports, _pwr, _turnpct, RESET_BLOCKANDTACHO, _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_1,`
 `PID_0, PID_3)`
 Rotate motor.
- #define `RotateMotor(_ports, _pwr, _angle)` __`RotateMotorExPID(_ports, _pwr,`
 `_angle, 0, FALSE, TRUE, PID_1, PID_0, PID_3)`
 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)`
 Set motor regulation time.
- #define `GetOutRegulationTime(_n)` __`GetOutRegulationTime(_n)`
 Get motor regulation time.
- #define `SetOutRegulationOptions(_n)` __`setOutRegulationOptions(_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 Define 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:

ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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:

ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

reset Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).

5.34.2.3 #define Float(_ports) Coast(_ports)

Float motors. Make outputs float. Float is an alias for Coast.

Parameters:

ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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:

n 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:

_n The motor regulation options

5.34.2.6 #define GetOutRegulationTime(_n) __GetOutRegulationTime(_n)

Get motor regulation time. Get the current motor regulation time.

Parameters:

_n 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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_reset Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).

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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr 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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr Output power, 0 to 100. Can be negative to reverse direction.

_reset Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).

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:

- _ports* Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.
- _pwr* Output power, 0 to 100. Can be negative to reverse direction.
- _reset* Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).
- _p* Proportional factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _i* Integral factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _d* Derivative factor used by the firmware's PID motor control algorithm. See [PID constants](#).

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:

- _ports* Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.
- _pwr* Output power, 0 to 100. Can be negative to reverse direction.
- _regmode* Regulation mode, see [Output port regulation mode constants](#).

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:

- _ports* Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr Output power, 0 to 100. Can be negative to reverse direction.

_regmode Regulation mode, see [Output port regulation mode constants](#).

_reset Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).

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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr Output power, 0 to 100. Can be negative to reverse direction.

_regmode Regulation mode, see [Output port regulation mode constants](#).

_reset Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).

_p Proportional factor used by the firmware's PID motor control algorithm. See [PID constants](#).

_i Integral factor used by the firmware's PID motor control algorithm. See [PID constants](#).

_d Derivative factor used by the firmware's PID motor control algorithm. See [PID constants](#).

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:

- _ports* Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.
- _pwr* Output power, 0 to 100. Can be negative to reverse direction.
- _regmode* Regulation mode, see [Output port regulation mode constants](#).
- _p* Proportional factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _i* Integral factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _d* Derivative factor used by the firmware's PID motor control algorithm. See [PID constants](#).

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:

- _ports* Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.
- _pwr* Output power, 0 to 100. Can be negative to reverse direction.
- _turnpct* 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:

- _ports* Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr Output power, 0 to 100. Can be negative to reverse direction.

_turnpct Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.

_reset Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).

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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr Output power, 0 to 100. Can be negative to reverse direction.

_turnpct Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.

_reset Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).

_p Proportional factor used by the firmware's PID motor control algorithm. See [PID constants](#).

_i Integral factor used by the firmware's PID motor control algorithm. See [PID constants](#).

_d Derivative factor used by the firmware's PID motor control algorithm. See [PID constants](#).

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:

- _ports* Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.
- _pwr* Output power, 0 to 100. Can be negative to reverse direction.
- _turnpct* Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.
- _p* Proportional factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _i* Integral factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _d* Derivative factor used by the firmware's PID motor control algorithm. See [PID constants](#).

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:

- _ports* Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.
- _pwr* 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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr Output power, 0 to 100. Can be negative to reverse direction.

_reset Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).

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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr Output power, 0 to 100. Can be negative to reverse direction.

_reset Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).

_p Proportional factor used by the firmware's PID motor control algorithm. See [PID constants](#).

_i Integral factor used by the firmware's PID motor control algorithm. See [PID constants](#).

_d Derivative factor used by the firmware's PID motor control algorithm. See [PID constants](#).

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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr Output power, 0 to 100. Can be negative to reverse direction.

_regmode Regulation mode, see [Output port regulation mode constants](#).

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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr Output power, 0 to 100. Can be negative to reverse direction.

_regmode Regulation mode, see [Output port regulation mode constants](#).

_reset Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).

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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr Output power, 0 to 100. Can be negative to reverse direction.

- _regmode* Regulation mode, see [Output port regulation mode constants](#).
- _reset* Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).
- _p* Proportional factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _i* Integral factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _d* Derivative factor used by the firmware's PID motor control algorithm. See [PID constants](#).

**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:

- _ports* Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.
- _pwr* Output power, 0 to 100. Can be negative to reverse direction.
- _regmode* Regulation mode, see [Output port regulation mode constants](#).
- _p* Proportional factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _i* Integral factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _d* Derivative factor used by the firmware's PID motor control algorithm. See [PID constants](#).

**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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr Output power, 0 to 100. Can be negative to reverse direction.

_turnpct 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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr Output power, 0 to 100. Can be negative to reverse direction.

_turnpct Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.

_reset Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).

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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

- _pwr* Output power, 0 to 100. Can be negative to reverse direction.
- _turnpct* Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.
- _reset* Position counters reset control. It must be a constant, see [Tachometer counter reset flags](#).
- _p* Proportional factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _i* Integral factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _d* Derivative factor used by the firmware's PID motor control algorithm. See [PID constants](#).

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:

- _ports* Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.
- _pwr* Output power, 0 to 100. Can be negative to reverse direction.
- _turnpct* Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.
- _p* Proportional factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _i* Integral factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _d* Derivative factor used by the firmware's PID motor control algorithm. See [PID constants](#).

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:

- _p* Desired output ports. Can be a constant or a variable, see [Output port constants](#).
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:

- _p* Desired output ports. Can be a constant or a variable, see [Output port constants](#).
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:

- _p* Desired output ports. Can be a constant or a variable, see [Output port constants](#).
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:

- _p* Desired output ports. Can be a constant or a variable, see [Output port constants](#).
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_1, PID_0, PID_3)

Rotate motor. Run the specified outputs forward for the specified number of degrees.

Parameters:

- _ports* Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.
_pwr Output power, 0 to 100. Can be negative to reverse direction.
_angle 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_1, PID_0, PID_3)

Rotate motor. Run the specified outputs forward for the specified number of degrees.

Parameters:

- _ports* Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.
_pwr Output power, 0 to 100. Can be negative to reverse direction.
_angle Angle limit, in degree. Can be negative to reverse direction.
_turnpct Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.
_bSync Synchronise two motors. Should be set to true if a non-zero turn percent is specified or no turning will occur.
_bStop 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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

_pwr Output power, 0 to 100. Can be negative to reverse direction.

_angle Angle limit, in degree. Can be negative to reverse direction.

_turnpct Turn ratio, -100 to 100. The direction of your vehicle will depend on its construction.

_bSync Synchronise two motors. Should be set to true if a non-zero turn percent is specified or no turning will occur.

_bStop Specify whether the motor(s) should brake at the end of the rotation.

_p Proportional factor used by the firmware's PID motor control algorithm. See [PID constants](#).

_i Integral factor used by the firmware's PID motor control algorithm. See [PID constants](#).

_d Derivative factor used by the firmware's PID motor control algorithm. See [PID constants](#).

```
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:

_ports Desired output ports. Can be a constant or a variable, see [Output port constants](#). 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.

- _pwr* Output power, 0 to 100. Can be negative to reverse direction.
- _angle* Angle limit, in degree. Can be negative to reverse direction.
- _p* Proportional factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _i* Integral factor used by the firmware's PID motor control algorithm. See [PID constants](#).
- _d* Derivative factor used by the firmware's PID motor control algorithm. See [PID constants](#).

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:

- _n* 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:

- _n* 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:

- _n* The motor regulation time in milliseconds

5.35 Input module functions

Functions for accessing and modifying input module features.

Defines

- #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 Define 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:

_port The port to clear. See [NBC Input port constants](#).

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:

_p The sensor port. See [NBC Input port constants](#). Must be a constant.

_nc The color index. See [Color sensor array indices](#). Must be a constant.

_n 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _nc* The color index. See [Color sensor array indices](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _np* The calibration point. See [Color calibration constants](#). Must be a constant.
- _nc* The color index. See [Color sensor array indices](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _np* The calibration point. See [Color calibration constants](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _nc* The color index. See [Color sensor array indices](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _nc* The color index. See [Color sensor array indices](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#).
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#).
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#).
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _n* 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:

- _port* The sensor port. See [NBC Input port constants](#). A variable whose value is the desired sensor port may also be used.
- _value* 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:

- _port* The sensor port. See [NBC Input port constants](#).

_colorval The color value. See [Color values](#).

_rawVals An array containing four raw color values. See [Color sensor array indices](#).

_normVals An array containing four normalized color values. See [Color sensor array indices](#).

_scaledVals An array containing four scaled color values. See [Color sensor array indices](#).

_result 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:

_port The sensor port. See [NBC Input port constants](#).

_rawVals An array containing four raw color values. See [Color sensor array indices](#).

_result 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:

_port The port to reset. See [NBC Input port constants](#).

**5.35.2.20 #define SetInCustomActiveStatus(_p, _n) __-
setInCustomActiveStatus(_p, _n)**

Set active status. Sets the active status value of a custom sensor.

Parameters:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _n* 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:

- _p* The sensor port. See [NBC Input port constants](#). Must be a constant.
- _n* 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:

port The port to configure. See [NBC Input port constants](#).

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:

port The port to configure. See [NBC Input port constants](#).

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:

port The port to configure. See [NBC Input port constants](#).

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:

port The port to configure. See [NBC Input port constants](#).

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:

port The port to configure. See [NBC Input port constants](#).

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:

port The port to configure. See [NBC Input port constants](#).

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:

port The port to configure. See [NBC Input port constants](#).

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:

port The port to configure. See [NBC Input port constants](#).

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:

port The port to configure. See [NBC Input port constants](#).

m The desired sensor mode. See [NBC sensor mode constants](#).

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:

port The port to configure. See [NBC Input port constants](#).

5.35.2.37 #define SetSensorTemperature(_port)**Value:**

```
__SetSensorLowspeed(_port) \
__MSWriteToRegister(_port, LEGO_ADDR_TEMP, TEMP_REG_CONFIG, TEMP_RES_12BIT, __W
DSC_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:

_port The port to configure. See [NBC Input port constants](#).

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:

_port The port to configure. See [NBC Input port constants](#).

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 [Reset-Sensor](#) to give the firmware time to reconfigure the sensor port.

See also:

[SetSensorMode\(\)](#)

Parameters:

_port The port to configure. See [NBC Input port constants](#).

_t The desired sensor type. See [NBC sensor type constants](#).

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:

_port The port to configure. See [NBC Input port constants](#).

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.

Defines

- #define [ReadSensorUS\(_port, _value\)](#) __ReadSensorUS(_port, _value)
Read ultrasonic sensor value.
- #define [ReadSensorUSEx\(_port, _values, _result\)](#) __ReadSensorUSEx(_port, _values, _result)
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 [WriteI2CRegister](#)(_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 [SetI2COPTIONS](#)(_port, _options) __setI2COPTIONS(_port, _options)
Set I2C options.

5.36.1 Detailed Description

Functions for accessing and modifying low speed module features.

5.36.2 Define 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:

- _port* The port to which the LEGO EMeter sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _config* The temperature sensor configuration settings. See [LEGO temperature sensor constants](#) for configuration constants that can be ORed or added together.
- _result* A status code indicating whether the read completed successfully or not. See [TCommLSRead](#) 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:

_port The port to which the I2C device is attached. See the [NBC Input port constants](#) 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.

_i2caddr The I2C device address.

_cmd The command to send to the I2C device.

_result A status code indicating whether the write completed successfully or not. See [TCommLSCheckStatus](#) 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:

_port The port to which the I2C device is attached. See the [NBC Input port constants](#) 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.

_bready 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:

_port The port to which the I2C device is attached. See the [NBC Input port constants](#) 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.

_result A status code indicating whether the write completed successfully or not.

See [TCommLSCheckStatus](#) for possible Result values. If the return value is [NO_ERR](#) then the last operation did not cause any errors. Avoid calls to [LowspeedRead](#) or [LowspeedWrite](#) while [LowspeedCheckStatus](#) returns [STAT_COMM_PENDING](#).

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:

_port The port to which the I2C device is attached. See the [NBC Input port constants](#) 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.

_buflen The initial size of the output buffer.

_buffer 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.

_result A status code indicating whether the write completed successfully or not. See [TCommLSRead](#) for possible Result values. If the return value is [NO_ERR](#) 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:

_port The port to which the I2C device is attached. See the [NBC Input port constants](#) 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.

_bready The number of bytes available to be read from the internal I2C buffer. The maximum number of bytes that can be read is 16.

_result A status code indicating whether the write completed successfully or not. See [TCommLSCheckStatus](#) for possible Result values. If the return value is [NO_ERR](#) then the last operation did not cause any errors. Avoid calls to [LowspeedRead](#) or [LowspeedWrite](#) while LowspeedStatus returns [STAT_COMM_PENDING](#).

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:

_port The port to which the I2C device is attached. See the [NBC Input port constants](#) 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.

_retlen The number of bytes that should be returned by the I2C device.

_buffer 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.

_result A status code indicating whether the write completed successfully or not. See [TCommLSWrite](#) for possible Result values. If the return value is [NO_ERR](#) 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:

_port The port to which the I2C device is attached. See the [NBC Input port constants](#) 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.

_inbuf 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.

_count The number of bytes that should be returned by the I2C device. On output count is set to the number of bytes in outbuf.

_outbuf A byte array that contains the data read from the internal I2C buffer.

_result 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:

- *port* The port to which the I2C device is attached. See the [NBC Input port constants](#) 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.
- *i2caddr* The I2C device address.
- *strVal* 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:

- *port* The port to which the I2C device is attached. See the [NBC Input port constants](#) 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.
- *i2caddr* The I2C device address.
- *info* A value indicating the type of device information you are requesting. See [Standard I2C constants](#).
- *strVal* 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:

- *port* The port to which the I2C device is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- *i2caddr* The I2C device address.

_reg The I2C device register from which to read a single byte.

_out The single byte read from the I2C device.

_result A status code indicating whether the read completed successfully or not.

See [TCommLSRead](#) 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:

_port The port to which the I2C device is attached. See the [NBC Input port constants](#) 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.

_i2caddr The I2C device address.

_strVal 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:

_port The port to which the I2C device is attached. See the [NBC Input port constants](#) 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.

_i2caddr The I2C device address.

_strVal 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:

_port The port to which the LEGO EMeter sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
_vIn Input voltage
_aIn Input current
_vOut Output voltage
_aOut Output current
_joules The number of joules stored in E-Meter
_wIn The number of watts generated
_wOut The number of watts consumed
_result A status code indicating whether the read completed successfully or not.
See [TCommLSRead](#) 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:

_port The port to which the temperature sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
_temp The temperature sensor value in degrees celcius.

```
5.36.2.16 #define ReadSensorUS(_port, _value) __ReadSensorUS(_port,
    _value)
```

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:

_port The port to which the ultrasonic sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.

_value The ultrasonic sensor distance value (0..255)

5.36.2.17 #define ReadSensorUSEx(_port, _values, _result) __ReadSensorUSEx(_port, _values, _result)

Read multiple ultrasonic sensor values. Return eight ultrasonic sensor distance values.

Parameters:

_port The port to which the ultrasonic sensor is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.

_values An array of bytes that will contain the 8 distance values read from the ultrasonic sensor.

_result A status code indicating whether the read completed successfully or not. See [TCommLSRead](#) for possible Result values.

5.36.2.18 #define SetI2COOptions(_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:

_port The port whose I2C options you wish to change. See the [NBC Input port constants](#) group. You may use a constant or a variable.

_options The new option value. See [I2C option constants](#).

5.36.2.19 #define WriteI2CRegister(_port, _i2caddr, _reg, _val, _result) __MSWriteToRegister(_port, _i2caddr, _reg, _val, _result)

Write I2C register. Write a single byte to an I2C device register.

Parameters:

- _port* The port to which the I2C device is attached. See the [NBC Input port constants](#) group. You may use a constant or a variable.
- _i2caddr* The I2C device address.
- _reg* The I2C device register to which to write a single byte.
- _val* The byte to write to the I2C device.
- _result* A status code indicating whether the write completed successfully or not.
See [TCommLSCheckStatus](#) for possible Result values.

5.37 Low level LowSpeed module functions

Low level functions for accessing low speed module features.

Defines

- #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, _-
GetLSInputBufferBytesToRx(_p, _n)
Get I2C input buffer bytes to rx.
- #define [GetLSOutputBuffer](#)(_p, _offset, _cnt, _data) __getLSOutputBuffer(_p, _-
_offset, _cnt, _data)
Get I2C output buffer data.
- #define [GetLSOutputBufferInPtr](#)(_p, _n) __GetLSOutputBufferInPtr(_p, _n)
Get I2C output buffer in-pointer.

- #define `GetLSOutputBufferOutPtr(_p, _n)` __`GetLSOutputBufferOutPtr(_p, _n)`
Get I2C output buffer out-pointer.
- #define `GetLSOutputBufferBytesToRx(_p, _n)` __`GetLSOutputBufferBytesToRx(_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 Define 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:

p A constant port number (S1..S4). See [NBC Input port constants](#).

_n The I2C port channel state. See [LSChannelState constants](#).

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:

_p A constant port number (S1..S4). See [NBC Input port constants](#).

_n The I2C port error type. See [LSErrorType constants](#).

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:

_p A constant port number (S1..S4). See [NBC Input port constants](#).

_offset A constant offset into the I2C input buffer.

_cnt The number of bytes to read.

_data 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:

_p A constant port number (S1..S4). See [NBC Input port constants](#).

_n 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:

- _p* A constant port number (S1..S4). See [NBC Input port constants](#).
- _n* 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:

- _p* A constant port number (S1..S4). See [NBC Input port constants](#).
- _n* 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:

- _p* A constant port number (S1..S4). See [NBC Input port constants](#).
- _n* The I2C port mode. See [LSMode constants](#).

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:

- _n* The I2C no restart on read field. See [LSNoRestartOnRead constants](#).

**5.37.2.9 #define GetLSOutputBuffer(_p, _offset, _cnt,
_data) __getLSOutputBuffer(_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:

- _p* A constant port number (S1..S4). See [NBC Input port constants](#).
- _offset* A constant offset into the I2C output buffer.
- _cnt* The number of bytes to read.
- _data* The byte array reference which will contain the data read from the I2C output buffer.

**5.37.2.10 #define GetLSOutputBufferBytesToRx(_p,
_n) __GetLSOutputBufferBytesToRx(_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:

- _p* A constant port number (S1..S4). See [NBC Input port constants](#).
- _n* The I2C output buffer's bytes to rx value.

**5.37.2.11 #define GetLSOutputBufferInPtr(_p, _n) __-
GetLSOutputBufferInPtr(_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:

- _p* A constant port number (S1..S4). See [NBC Input port constants](#).
- _n* 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:

- _p* A constant port number (S1..S4). See [NBC Input port constants](#).
- _n* 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:

- _n* 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:

- _n* The I2C state. See [LSSState constants](#).

5.38 Display module functions

Functions for accessing and modifying display module features.

Defines

- #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)
Draw a polygon with drawing 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 Define 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:

- _x** The x value for the center of the circle.
- _y** The y value for the center of the circle.
- _r** 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:

- _x* The x value for the center of the circle.
- _y* The y value for the center of the circle.
- _r* The radius of the circle.
- _options* 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:

- _line* The line you want to clear. See [Line number constants](#).

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:

- _x* The x value for the center of the ellipse.
- _y* The y value for the center of the ellipse.
- _rX* The x axis radius.
- _rY* 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:

- _x* The x value for the center of the ellipse.
- _y* The y value for the center of the ellipse.
- _rX* The x axis radius.
- _rY* The y axis radius.
- _options* 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:

- _x* The x value for the start of the number output.
- _y* The y value for the start of the number output.
- _fnt* The filename of the RIC font.
- _num* 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:

- _x* The x value for the start of the number output.
- _y* The y value for the start of the number output.
- _fnt* The filename of the RIC font.
- _num* The value to output to the LCD screen. Any numeric type is supported.
- _options* 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:

- _x* The x value for the start of the text output.
- _y* The y value for the start of the text output.
- _fnt* The filename of the RIC font.
- _txt* 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:

- _x* The x value for the start of the text output.
- _y* The y value for the start of the text output.
- _fnt* The filename of the RIC font.
- _txt* The text to output to the LCD screen.
- _options* 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:

_n 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:

_n 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:

_n 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:

_n 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:

_n 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:

_x The desired x position from which to read pixel data.
_line The desired line from which to read pixel data.
_cnt The number of bytes of pixel data to read.
_data 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:

_x The desired x position from which to read pixel data.
_line The desired line from which to read pixel data.
_cnt The number of bytes of pixel data to read.
_data 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:

_n 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:

_n 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:

_x The x value for the position of the graphic image.

_y The y value for the position of the graphic image.

_data 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:

- _x* The x value for the position of the graphic image.
- _y* The y value for the position of the graphic image.
- _data* The byte array of the RIC graphic image.
- _vars* The byte array of parameters.
- _options* 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:

- _x* The x value for the position of the graphic image.
- _y* The y value for the position of the graphic image.
- _file* 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:

- _x* The x value for the position of the graphic image.
- _y* The y value for the position of the graphic image.
- _file* The filename of the RIC graphic image.
- _vars* The byte array of parameters.
- _options* 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:

- _x1* The x value for the start of the line.
- _y1* The y value for the start of the line.
- _x2* The x value for the end of the line.
- _y2* 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:

- _x1* The x value for the start of the line.

_y1 The y value for the start of the line.
_x2 The x value for the end of the line.
_y2 The y value for the end of the line.
_options 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:

_x The x value for the start of the number output.
_y The text line number for the number output.
_num 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:

_x The x value for the start of the number output.
_y The text line number for the number output.
_num The value to output to the LCD screen. Any numeric type is supported.
_options 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:

- _x* The x value for the point.
- _y* 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:

- _x* The x value for the point.
- _y* The y value for the point.
- _options* 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:

_points 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:

_points An array of [TLocation](#) points that define the polygon.

_options 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:

_x The x value for the top left corner of the rectangle.

_y The y value for the top left corner of the rectangle.

_w The width of the rectangle.

_h 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:

- _x* The x value for the top left corner of the rectangle.
- _y* The y value for the top left corner of the rectangle.
- _w* The width of the rectangle.
- _h* The height of the rectangle.
- _options* 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:

- _n* 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:

- _n* 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:

_n 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:

_n The new display flags. See [Display flags](#).

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:

_n 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:

_x The desired x position where you wish to write pixel data.

_line The desired line where you wish to write pixel data.

_cnt The number of bytes of pixel data to write.

_data 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:

_x The desired x position where you wish to write pixel data.

_line The desired line where you wish to write pixel data.

_cnt The number of bytes of pixel data to write.

_data 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:

_n 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:

_n 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:

- _x* The x value for the start of the text output.
- _y* The text line number for the text output.
- _txt* 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:

- _x* The x value for the start of the text output.
- _y* The text line number for the text output.
- _txt* The text to output to the LCD screen.
- _options* The optional drawing options.

5.39 Sound module functions

Functions for accessing and modifying sound module features.

Defines

- #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)`
Play a file.
- #define `PlayFileEx(_file, _vol, _loop)` __`PlayFileEx(_file,_vol,_loop)`
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 Define 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:

_n 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:

_state The current sound module state.

_flags 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:

_n The current sound volume.

5.39.2.7 #define PlayFile(_file) __PlayFileEx(_file,4,0)

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:

_file The name of the sound or melody file to play.

5.39.2.8 #define PlayFileEx(_file, _vol, _loop) __PlayFileEx(_file,_vol,_loop)

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:

_file The name of the sound or melody file to play.

_vol The desired tone volume.

_loop A boolean flag indicating whether to play the file repeatedly.

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:

- _freq* The desired tone frequency, in Hz.
- _dur* 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:

- _freq* The desired tone frequency, in Hz.
- _dur* The desired tone duration, in ms.
- _vol* The desired tone volume.
- _loop* 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:

- _n* 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:

_n 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:

_n 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:

_n 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:

_n 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:

_n 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 [Sound-State constants](#) group.

See also:

[GetSoundState](#)

Parameters:

_state The sound module state.
_flags The sound module flags.
_result The function call result.

5.39.2.18 #define SetSoundVolume(_n) __setSoundVolume(_n)

Set sound volume. Set the sound volume.

See also:[GetSoundVolume](#)**Parameters:**

_n The new volume

5.40 Command module functions

Functions for accessing and modifying Command module features.

Defines

- #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) __SetIOMapValue-ByID(CommandModuleID, _offset, _n)
Set Command module IOMap value.
- #define [SetIOCtrlModuleValue](#)(_offset, _n) __SetIOMapValue-ByID(IOCtrlModuleID, _offset, _n)
Set IOCtrl module IOMap value.
- #define [SetLoaderModuleValue](#)(_offset, _n) __SetIOMapValue-ByID(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) SetIOMapValue-
ByID(SoundModuleID, _offset, _n)
Set Sound module IOMap value.
- #define **SetButtonModuleValue**(_offset, _n) SetIOMapValue-
ByID(ButtonModuleID, _offset, _n)
Set Button module IOMap value.
- #define **SetInputModuleValue**(_offset, _n) SetIOMapValue-
ByID(InputModuleID, _offset, _n)
Set Input module IOMap value.
- #define **SetOutputModuleValue**(_offset, _n) SetIOMapValue-
ByID(OutputModuleID, _offset, _n)
Set Output module IOMap value.
- #define **SetLowSpeedModuleValue**(_offset, _n) SetIOMapValue-
ByID(LowSpeedModuleID, _offset, _n)
Set Lowspeed module IOMap value.
- #define **SetDisplayModuleValue**(_offset, _n) SetIOMapValue-
ByID(DisplayModuleID, _offset, _n)
Set Display module IOMap value.
- #define **SetCommModuleValue**(_offset, _n) SetIOMapValue-
ByID(CommModuleID, _offset, _n)
Set Comm module IOMap value.
- #define **SetCommandModuleBytes**(_offset, _cnt, _arrIn) SetIOMapBytes-
ByID(CommandModuleID, _offset, _cnt, _arrIn)
Set Command module IOMap bytes.
- #define **SetLowSpeedModuleBytes**(_offset, _cnt, _arrIn) SetIOMapBytes-
ByID(LowSpeedModuleID, _offset, _cnt, _arrIn)
Set Lowspeed module IOMap bytes.
- #define **SetDisplayModuleBytes**(_offset, _cnt, _arrIn) SetIOMapBytes-
ByID(DisplayModuleID, _offset, _cnt, _arrIn)
Set Display module IOMap bytes.
- #define **SetCommModuleBytes**(_offset, _cnt, _arrIn) SetIOMapBytes-
ByID(CommModuleID, _offset, _cnt, _arrIn)

Set Comm module IOMap bytes.

- #define **GetIOMapBytes**(_modName, _offset, _cnt, _arrOut) __-
getIOMapBytes(_modName, _offset, _cnt, _arrOut)
Get IOMap bytes by name.
- #define **GetIOMapValue**(_modName, _offset, _n) __getIOMapValue(_-
modName, _offset, _n)
Get IOMap value 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) GetIOMapValue-
ByID(CommandModuleID, _offset, _n)
Get Command module IOMap value.
- #define **GetLoaderModuleValue**(_offset, _n) GetIOMapValue-
ByID(LoaderModuleID, _offset, _n)
Get Loader module IOMap value.
- #define **GetSoundModuleValue**(_offset, _n) GetIOMapValue-
ByID(SoundModuleID, _offset, _n)
Get Sound module IOMap value.
- #define **GetButtonModuleValue**(_offset, _n) GetIOMapValue-
ByID(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) GetIOMapValue-
ByID(InputModuleID, _offset, _n)
Get Input module IOMap value.
- #define **GetOutputModuleValue**(_offset, _n) GetIOMapValue-
ByID(OutputModuleID, _offset, _n)
Get Output module IOMap value.

Get Output module IOMap value.

- #define **GetLowSpeedModuleValue**(_offset, _n) GetIOMapValue-
ByID(LowSpeedModuleID, _offset, _n)
Get LowSpeed module IOMap value.
- #define **GetDisplayModuleValue**(_offset, _n) GetIOMapValue-
ByID(DisplayModuleID, _offset, _n)
Get Display module IOMap value.
- #define **GetCommModuleValue**(_offset, _n) GetIOMapValue-
ByID(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 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 Define 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:

_offset The number of bytes offset from the start of the IOMap structure where the value should be read. See [Button module IOMAP offsets](#).

_n 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:

_offset The number of bytes offset from the start of the Command module IOMap structure where the data should be read. See [Command module IOMAP offsets](#).

_cnt The number of bytes to read from the specified Command module IOMap offset.

_arrOut 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:

_offset The number of bytes offset from the start of the IOMap structure where the value should be read. See [Command module IOMAP offsets](#).

_n 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:

_offset The number of bytes offset from the start of the Comm module IOMap structure where the data should be read. See [Comm module IOMAP offsets](#).

_cnt The number of bytes to read from the specified Comm module IOMap offset.

_arrOut 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:

_offset The number of bytes offset from the start of the IOMap structure where the value should be read. See [Comm module IOMAP offsets](#).

_n 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:

_offset The number of bytes offset from the start of the Display module IOMap structure where the data should be read. See [Display module IOMAP offsets](#).

_cnt The number of bytes to read from the specified Display module IOMap offset.

_arrOut 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:

_offset The number of bytes offset from the start of the IOMap structure where the value should be read. See [Display module IOMAP offsets](#).

_n 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:

_value 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:

_offset The number of bytes offset from the start of the IOMap structure where the value should be read. See [Input module IOMAP offsets](#).

_n 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:

_modName The module name of the IOMap. See [NXT firmware module names](#).

_offset The number of bytes offset from the start of the IOMap structure where the data should be read

_cnt The number of bytes to read from the specified IOMap offset.

_arrOut 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:

- _modID* The module ID of the IOMap. See [NXT firmware module IDs](#).
- _offset* The number of bytes offset from the start of the IOMap structure where the data should be read.
- _cnt* The number of bytes to read from the specified IOMap offset.
- _arrOut* 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:

- _modName* The module name of the IOMap. See [NXT firmware module names](#).
- _offset* The number of bytes offset from the start of the IOMap structure where the value should be read
- _n* 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:

- _modID* The module ID of the IOMap. See [NXT firmware module IDs](#).
- _offset* The number of bytes offset from the start of the IOMap structure where the value should be read.
- _n* 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:

- _Clear* A boolean value indicating whether to clear the response or not.
- _Length* The response packet length.
- _Command* The original command byte.
- _Buffer* The response packet buffer.
- _Result* 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:

_offset The number of bytes offset from the start of the IOMap structure where the value should be read. See [Loader module IOMAP offsets](#).

_n 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:

_offset The number of bytes offset from the start of the Lowspeed module IOMap structure where the data should be read. See [Low speed module IOMAP offsets](#).

_cnt The number of bytes to read from the specified Lowspeed module IOMap offset.

_arrOut 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:

_offset The number of bytes offset from the start of the IOMap structure where the value should be read. See [Low speed module IOMAP offsets](#).

_n 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:

- _Compact* A boolean value indicating whether to compact the dataspace or not.
- _PoolSize* The current pool size.
- _DataspaceSize* The current dataspace size.
- _Result* The function call result. It will be [NO_ERR](#) 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:

- _offset* The number of bytes offset from the start of the IOMap structure where the value should be read. See [Output module IOMAP offsets](#).
- _n* 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:

_offset The number of bytes offset from the start of the IOMap structure where the value should be read. See [Sound module IOMAP offsets](#).

_n 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:

_offset The number of bytes offset from the start of the IOMap structure where the value should be read. See [Ui module IOMAP offsets](#).

_n 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:

_offset The number of bytes offset from the start of the Button module IOMap structure where the new value should be written. See [Button module IOMAP offsets](#).

_n 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:

_offset The number of bytes offset from the start of the Command module IOMap structure where the data should be written. See [Command module IOMAP offsets](#).

_cnt The number of bytes to write at the specified Command module IOMap offset.

_arrIn 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:

_offset The number of bytes offset from the start of the Command module IOMap structure where the new value should be written. See [Command module IOMAP offsets](#).

_n 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:

_offset The number of bytes offset from the start of the Comm module IOMap structure where the data should be written. See [Comm module IOMAP offsets](#).

_cnt The number of bytes to write at the specified Comm module IOMap offset.

_arrIn 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:

_offset The number of bytes offset from the start of the Comm module IOMap structure where the new value should be written. See [Comm module IOMAP offsets](#).

_n 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:

_offset The number of bytes offset from the start of the Display module IOMap structure where the data should be written. See [Display module IOMAP offsets](#).

_cnt The number of bytes to write at the specified Display module IOMap offset.

_arrIn 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:

_offset The number of bytes offset from the start of the Display module IOMap structure where the new value should be written. See [Display module IOMAP offsets](#).

_n 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:

_offset The number of bytes offset from the start of the Input module IOMap structure where the new value should be written. See [Input module IOMAP offsets](#).

_n A variable containing the new value to write to the Input module IOMap.

5.40.2.31 #define SetIOCctrlModuleValue(_offset, _n)
SetIOMapValueByID(IOCctrlModuleID, _offset, _n)

Set IOCctrl module IOMap value. Set one of the fields of the IOCctrl module IOMap structure to a new value. You provide the offset into the IOCctrl module IOMap structure where you want to write the value along with a variable containing the new value.

Parameters:

_offset The number of bytes offset from the start of the IOCctrl module IOMap structure where the new value should be written. See [IOCctrl module IOMAP offsets](#).

_n A variable containing the new value to write to the IOCctrl 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:

_modName The module name of the IOMap to modify. See [NXT firmware module names](#).

_offset The number of bytes offset from the start of the IOMap structure where the data should be written

_cnt The number of bytes to write at the specified IOMap offset.

_arrIn 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:

- _modID* The module ID of the IOMap to modify. See [NXT firmware module IDs](#).
- _offset* The number of bytes offset from the start of the IOMap structure where the data should be written.
- _cnt* The number of bytes to write at the specified IOMap offset.
- _arrIn* 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:

- _modName* The module name of the IOMap to modify. See [NXT firmware module names](#).
- _offset* The number of bytes offset from the start of the IOMap structure where the new value should be written
- _n* 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:

- _modID* The module ID of the IOMap to modify. See [NXT firmware module IDs](#).
- _offset* The number of bytes offset from the start of the IOMap structure where the new value should be written.

_n 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)
Set IOMapValueByID(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:

_offset The number of bytes offset from the start of the Loader module IOMap structure where the new value should be written. See [Loader module IOMAP offsets](#).

_n 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:

_offset The number of bytes offset from the start of the Lowspeed module IOMap structure where the data should be written. See [Low speed module IOMAP offsets](#).

_cnt The number of bytes to write at the specified Lowspeed module IOMap offset.

_arrIn 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:

_offset The number of bytes offset from the start of the Lowspeed module IOMap structure where the new value should be written. See [Low speed module IOMAP offsets](#).

_n 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:

_offset The number of bytes offset from the start of the Output module IOMap structure where the new value should be written. See [Output module IOMAP offsets](#).

_n A variable containing the new value to write to the Output module IOMap.

**5.40.2.40 #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:

_offset The number of bytes offset from the start of the Sound module IOMap structure where the new value should be written. See [Sound module IOMAP offsets](#).

_n A variable containing the new value to write to the Sound module IOMap.

**5.40.2.41 #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:

_offset The number of bytes offset from the start of the Ui module IOMap structure where the new value should be written. See [Ui module IOMAP offsets](#).

_n A variable containing the new value to write to the Ui module IOMap.

5.40.2.42 #define Wait(_n) waitv _n

Wait some milliseconds. Make a task sleep for specified amount of time (in 1000ths of a second).

Parameters:

_n The number of milliseconds to sleep.

5.41 Button module functions

Functions for accessing and modifying Button module features.

Defines

- #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 Define 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:

- _b* The button to check. See [Button name constants](#).
- _n* 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:

- _b* The button to check. See [Button name constants](#).
- _n* 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:

- _b* The button to check. See [Button name constants](#).
- _n* 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:

- _b* The button to check. See [Button name constants](#).
- _n* 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:

- _b* The button to check. See [Button name constants](#).
- _n* 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:

- _b* The button to check. See [Button name constants](#).
- _n* 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:

- _idx* The button to check. See [Button name constants](#).
- _reset* Whether or not to reset the press counter.
- _pressed* The button pressed state.
- _count* The button press count.
- _result* 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:

- _b* The button number. See [Button name constants](#).
- _n* 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:

- _b* The button number. See [Button name constants](#).
- _n* 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:

- _b* The button number. See [Button name constants](#).
- _n* 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:

- _b* The button number. See [Button name constants](#).
- _n* 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:

- _b* The button number. See [Button name constants](#).
- _n* 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:

- _b* The button to check. See [Button name constants](#).
- _n* The new button state. See [ButtonState constants](#).

5.42 Ui module functions

Functions for accessing and modifying Ui module features.

Defines

- #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)
Get UI module USB state.
- #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 Define 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:

_n 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:

_n The current abort flag value. See [ButtonState constants](#).

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 [BluetoothState constants](#).

5.42.2.6 #define GetCommandFlags(_n) __GetCommandFlags(_n)

Get command flags. Return the command flags.

Parameters:

_n Command flags. See [CommandFlags constants](#)

**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 [UIButton constants](#).

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 [UIState constants](#).

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 [VM run state constants](#).

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 [ButtonState constants](#)

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 [BluetoothState constants](#).

5.42.2.19 #define SetCommandFlags(_n) __setCommandFlags(_n)

Set command flags. Set the command flags.

Parameters:

_n The new command flags. See [CommandFlags constants](#).

**5.42.2.20 #define SetOnBrickProgramPointer(_n) __-
setOnBrickProgramPointer(_n)**

Set on-brick program pointer. Set the current OBP (on-brick program) step.

Parameters:

_n 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:

_n 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:

_n The minutes left on the timer.

5.42.2.23 #define SetUIButton(_n) __setUIButton(_n)

Set UI button. Set user interface button information.

Parameters:

_n A user interface button value. See [UIButton constants](#).

5.42.2.24 #define SetUIState(_n) __setUIState(_n)

Set UI state. Set the user interface state.

Parameters:

_n A user interface state value. See [UIState constants](#).

5.42.2.25 #define SetUsbState(_n) __setUsbState(_n)

Set Usb state. This method sets the value of the Usb state.

Parameters:

_n The Usb state.

5.42.2.26 #define SetVMRunState(_n) __setVMRunState(_n)

Set VM run state. Set VM run state information.

Parameters:

_n The desired VM run state. See [VM run state constants](#).

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:

_n 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.

Defines

- #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`)

Send a string value to a remote mailbox.
- #define `UseRS485`() `__UseRS485()`

Use the RS485 port.
- #define `RS485Status`(`_sendingData`, `_dataAvail`) `__RS485Status`(`_sendingData`, `_dataAvail`)

Check RS485 status.
- #define `RS485Write`(`_buffer`, `_status`) `__RS485Write`(`_buffer`, `_status`)

Write RS485 data.
- #define `RS485Read`(`_buffer`, `_status`) `__RS485Read`(`_buffer`, `_status`)

Read RS485 data.
- #define `RS485ReadEx`(`_buffer`, `_buflen`, `_status`) `__RS485ReadEx`(`_buffer`, `_buflen`, `_status`)

Read limited RS485 data.
- #define `RS485Control`(`_cmd`, `_baud`, `_mode`, `_result`) `__RS485Control`(`_cmd`, `_baud`, `_mode`, `_result`)

Control the RS485 port.
- #define `RS485Uart`(`_baud`, `_mode`, `_result`) `__RS485Control(HS_CTRL_UART, _baud, _mode, _result)`

Configure RS485 UART.
- #define `RS485Initialize`(`_result`) `__RS485Control(HS_CTRL_UART, HS_BAUD_DEFAULT, HS_MODE_DEFAULT, _result)`

Initialize RS485 port.

- #define **RS485Enable**(_result) __RS485Control(HS_CTRL_INIT, HS_BAUD_DEFAULT, HS_MODE_DEFAULT, _result)

Enable RS485.

- #define **RS485Disable**(_result) __RS485Control(HS_CTRL_EXIT, HS_BAUD_DEFAULT, HS_MODE_DEFAULT, _result)

Disable RS485.

- #define **SendRS485Bool**(_bval, _status) __sendRS485Bool(_bval, _status)

Write RS485 boolean.

- #define **SendRS485Number**(_val, _status) __sendRS485Number(_val, _status)

Write RS485 numeric.

- #define **SendRS485String**(_str, _status) __sendRS485String(_str, _status)

Write RS485 string.

- #define **GetBTDeviceName**(_p, _str) __GetBTDeviceName(_p, _str)

Get bluetooth device name.

- #define **GetBTDeviceClass**(_p, _n) __GetBTDeviceClass(_p, _n)

Get bluetooth device class.

- #define **GetBTDeviceAddress**(_p, _btaddr) __getBTDeviceAddress(_p, _btaddr)

Get bluetooth device address.

- #define **GetBTDeviceStatus**(_p, _n) __GetBTDeviceStatus(_p, _n)

Get bluetooth device status.

- #define **GetBTConnectionName**(_p, _str) __GetBTConnectionName(_p, _str)

Get bluetooth device name.

- #define **GetBTConnectionClass**(_p, _n) __GetBTConnectionClass(_p, _n)

Get bluetooth device class.

- #define **GetBTConnectionPinCode**(_p, _code) __GetBTConnectionPinCode(_p, _code)

Get bluetooth device pin code.

- #define `GetBTConnectionAddress(_p, _btaddr)` `__getBTConnectionAddress(_p, _btaddr)`
Get bluetooth device address.
- #define `GetBTConnectionHandleNum(_p, _n)` `GetBTConnectionHandleNum(_p, _n)`
Get bluetooth device handle number.
- #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)**
Set hi-speed port input buffer in-pointer.
- #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 Define 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:

_conn The connection slot (0..3). Connections 0 through 3 are for bluetooth connections. See [Remote connection constants](#).

_result 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:

- _conn* The connection slot (0..3). Connections 0 through 3 are for bluetooth connections. See [Remote connection constants](#).
- _buffer* The data to be written (up to 128 bytes)
- _result* 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:

- _btaddr* 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:

- _n* 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) GetComm-
ModuleBytes(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:

_p The connection slot (0..3).

_btaddr 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:

_p The connection slot (0..3).

_n 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:

_p The connection slot (0..3).

_n 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:

_p The connection slot (0..3).

_n 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:

- _p* The connection slot (0..3).
- _str* 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:

- _p* The connection slot (0..3).
- _code* 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:

- _p* The connection slot (0..3).
- _n* 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:

_n The Bluetooth data mode. See [Data mode constants](#).

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:

_p The device table index.

_btaddr 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:

_p The device table index.

_n 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:

- _p* The device table index.
- _str* 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:

- _n* 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:

- _p* The device table index.
- _n* 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:

- _offset* A constant offset into the bluetooth input buffer.
- _cnt* The number of bytes to read.
- _data* 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:

- _n* 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:

- _n* 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:

- _offset* A constant offset into the bluetooth output buffer.

_cnt The number of bytes to read.

_data 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:

_n 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:

_n 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:

_n The hi-speed port address. See [Hi-speed port address constants](#).

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:

_n The hi-speed port data mode. See [Data mode constants](#).

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:

_n The hi-speed port flags. See [Hi-speed port flags constants](#).

**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:

_offset A constant offset into the hi-speed port input buffer.

_cnt The number of bytes to read.

_data 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 [Hi-speed port data bits constants](#), [Hi-speed port stop bits constants](#), [Hi-speed port parity constants](#), and [Hi-speed port combined UART constants](#).

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:

- _offset* A constant offset into the hi-speed port output buffer.
- _cnt* The number of bytes to read.
- _data* 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:

- _n* 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:

- _n* 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:

_n The hi-speed port speed (baud rate). See [Hi-speed port baud rate constants](#).

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:

_n The hi-speed port state. See [Hi-speed port state constants](#).

**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:

_offset A constant offset into the usb input buffer.

_cnt The number of bytes to read.

_data 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:

_n 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:

_n The USB port input buffer's out-pointer value.

**5.43.2.44 #define GetUSBOutputBuffer(_offset, _cnt,
_data) __getUSBOutputBuffer(_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:

_offset A constant offset into the usb output buffer.

_cnt The number of bytes to read.

_data The byte array reference which will contain the data read from the usb output buffer.

5.43.2.45 #define GetUSBOutputBufferInPtr(_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:

_n The USB port output buffer's in-pointer value.

5.43.2.46 #define GetUSBOutputBufferOutPtr(_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:

_n 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:

_offset A constant offset into the usb poll buffer.

_cnt The number of bytes to read.

_data 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:

- _queue* The mailbox number. See [Mailbox constants](#).
- _clear* A flag indicating whether to remove the message from the mailbox after it has been read.
- _msg* The message that is read from the mailbox.
- _result* 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:

- _queue* The mailbox number. See [Mailbox constants](#).
- _clear* A flag indicating whether to remove the message from the mailbox after it has been read.
- _bval* The boolean value that is read from the mailbox.
- _result* 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:

- _queue* The mailbox number. See [Mailbox constants](#).
- _clear* A flag indicating whether to remove the message from the mailbox after it has been read.
- _str* The string value that is read from the mailbox.
- _val* The numeric value that is read from the mailbox.
- _bval* The boolean value that is read from the mailbox.
- _result* 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:

- _queue* The mailbox number. See [Mailbox constants](#).
- _clear* A flag indicating whether to remove the message from the mailbox after it has been read.
- _val* The numeric value that is read from the mailbox.
- _result* 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:

- _queue* The mailbox number. See [Mailbox constants](#).
- _clear* A flag indicating whether to remove the message from the mailbox after it has been read.

_str The string value that is read from the mailbox.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_buffer The data to be written (up to 128 bytes)

_result 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:

- _cmd* The control command to send to the port. See [Hi-speed port SysCommHSControl constants](#).
- _baud* The baud rate for the RS485 port. See [Hi-speed port baud rate constants](#).
- _mode* The RS485 port mode (data bits, stop bits, parity). See [Hi-speed port data bits constants](#), [Hi-speed port stop bits constants](#), [Hi-speed port parity constants](#), and [Hi-speed port combined UART constants](#).
- _result* 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:

- _result* 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:

_result 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:

_result 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:

_buffer A byte array that will contain the data read from the RS485 port.

_status 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:

- _buffer* A byte array that will contain the data read from the RS485 port.
- _buflen* The number of bytes you want to read.
- _status* 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:

- _sendingData* A boolean value set to true on output if data is being sent.
- _dataAvail* 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:

- _baud* The baud rate for the RS485 port. See [Hi-speed port baud rate constants](#).
- _mode* The RS485 port mode (data bits, stop bits, parity). See [Hi-speed port data bits constants](#), [Hi-speed port stop bits constants](#), [Hi-speed port parity constants](#), and [Hi-speed port combined UART constants](#).

_result 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:

_buffer A byte array containing the data to write to the RS485 port.

_status 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:

_queue The mailbox number. See [Mailbox constants](#).

_msg The message to write to the mailbox.

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_queue The mailbox number. See [Mailbox constants](#).

_bval The boolean value to send.

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_queue The mailbox number. See [Mailbox constants](#).

_val The numeric value to send.

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_queue The mailbox number. See [Mailbox constants](#).

_str The string value to send.

_result 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:

- _queue* The mailbox number. See [Mailbox constants](#). This function shifts the specified value into the range of response mailbox numbers by adding 10.
- _bval* The boolean value to write.
- _result* 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:

- _queue* The mailbox number. See [Mailbox constants](#). This function shifts the specified value into the range of response mailbox numbers by adding 10.
- _val* The numeric value to write.
- _result* 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:

- _queue* The mailbox number. See [Mailbox constants](#). This function shifts the specified value into the range of response mailbox numbers by adding 10.
- _msg* The string value to write.
- _result* 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:

_bval A boolean value to write over the RS485 port.

_status 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:

_val A numeric value to write over the RS485 port.

_status 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:

_str A string value to write over the RS485 port.

_status 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:

_n The Bluetooth data mode. See [Data mode constants](#).

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:

_offset A constant offset into the input buffer

_cnt The number of bytes to write

_data 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:

_n 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:

_n 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:

- _offset* A constant offset into the output buffer
- _cnt* The number of bytes to write
- _data* 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:

- _n* 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:

- _n* 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:

- _n* The hi-speed port address. See [Hi-speed port address constants](#).

5.43.2.85 #define SetHSDataMode(_n) __setHSDataMode(_n)

Set hi-speed port data mode. This method sets the value of the hi-speed port data mode.

Parameters:

_n The hi-speed port data mode. See [Data mode constants](#).

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:

_n The hi-speed port flags. See [Hi-speed port flags constants](#).

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:

_offset A constant offset into the input buffer

_cnt The number of bytes to write

_data 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:

_n 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:

n 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:

n The hi-speed port mode (data bits, stop bits, parity). See [Hi-speed port data bits constants](#), [Hi-speed port stop bits constants](#), [Hi-speed port parity constants](#), and [Hi-speed port combined UART constants](#).

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:

offset A constant offset into the output buffer
cnt The number of bytes to write
data 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:

_n 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:

_n 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:

_n The hi-speed port speed (baud rate). See [Hi-speed port baud rate constants](#).

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:

_n The hi-speed port state. See [Hi-speed port state constants](#).

**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:

_offset A constant offset into the input buffer

_cnt The number of bytes to write
_data 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:

_n 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:

_n 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:

_offset A constant offset into the output buffer
_cnt The number of bytes to write
_data 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:

_n The new in-pointer value (0..63).

**5.43.2.101 #define SetUSBOutputBufferOutPtr(_n) __-
setUSBOutputBufferOutPtr(_n)**

Set USB output buffer out-pointer. Set the value of the output buffer out-pointer.

Parameters:

_n 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:

_offset A constant offset into the poll buffer

_cnt The number of bytes to write

_data 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:

_n 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:

_n 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:

_n 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.

Defines

- #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 SetInputModule 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)

Send a LowspeedRead message.

- #define **RemoteGetCurrentProgramName**(*_conn*, *_name*, *_result*) __-
remoteGetCurrentProgramName(*_conn*, *_name*, *_result*)

Send a GetCurrentProgramName message.

- #define **RemoteDatalogRead**(*_conn*, *_remove*, *_cnt*, *_log*, *_result*) __-
remoteDatalogRead(*_conn*, *_remove*, *_cnt*, *_log*, *_result*)

Send a DatalogRead 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 **Remote SetProperty**(*_conn*, *_prop*, *_value*, *_result*) __-
remote SetProperty(*_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 Define 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_remove Remove the datalog message from the queue after reading it (true or false).

_cnt The number of bytes read from the datalog.

_log A byte array containing the datalog contents.

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_synctime The datalog sync time.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_value The battery level value.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_cnt The number of connections.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_idx The index of the connection.

_name The name of the specified connection.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_cnt The number of contacts.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_idx The index of the contact.

_name The name of the specified contact.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_name The current program name.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_params The input and output parameters for the function call.

_result A char value indicating whether the function call succeeded or not.

5.44.2.10 #define RemoteGetOutputState(_conn, _params, _result)**Value:**

```
compchktype _params, TOoutputState \
__remoteGetOutputState(_conn, _params, _result)
```

Send a GetOutputState message. Send the GetOutputState direct command on the specified connection slot.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_params The input and output parameters for the function call.

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_property The property to read. See [Property constants](#).

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_value The count of available bytes to read.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_port The input port from which to read I2C data. See [NBC Input port constants](#).

_bread The number of bytes read.

_data A byte array containing the data read from the I2C device.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_port The I2C port. See [NBC Input port constants](#).

_txlen The number of bytes you are writing to the I2C device.

_rxlen The number of bytes want to read from the I2C device.

_data A byte array containing the data you are writing to the device.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_queue The mailbox to read. See [Mailbox constants](#).

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_queue The mailbox to write. See [Mailbox constants](#).

_msg The message to write to the mailbox.

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_filename The name of the sound file to play.

_bloop A boolean value indicating whether to loop the sound file or not.

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_frequency The frequency of the tone.

_duration The duration of the tone.

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_port The output port to reset.

_brelative A flag indicating whether the counter to reset is relative.

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_port The input port to reset.

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_port The output port to reset the tachometer count on. See [Output port constants](#).

_result A char value indicating whether the function call succeeded or not.

5.44.2.23 #define RemoteSetInputModule(_conn, _port, _type, _mode, _result) __remoteSetInputModule(_conn, _port, _type, _mode, _result)

Send a SetInputModule message. Send the SetInputModule direct command on the specified connection slot. Use [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_port The input port to configure. See [NBC Input port constants](#).

_type The sensor type. See [NBC sensor type constants](#).

_mode The sensor mode. See [NBC sensor mode constants](#).

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_port The output port to configure. See [Output port constants](#).

_speed The motor speed. (-100..100)

_mode The motor mode. See [Output port mode constants](#).

_regmode The motor regulation mode. See [Output port regulation mode constants](#).

_turnpct The motor synchronized turn percentage. (-100..100)

_runstate The motor run state. See [Output port run state constants](#).

_tacholimit The motor tachometer limit.

_result A char value indicating whether the function call succeeded or not.

5.44.2.25 #define RemoteSetProperty(_conn, _prop, _value, _result) __remoteSetProperty(_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:

- _conn* The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).
- _prop* The property to set. See [Property constants](#).
- _value* The new property value.
- _result* A char value indicating whether the function call succeeded or not.

5.44.2.26 #define RemoteStartProgram(_conn, _filename, _result) __remoteStartProgram(_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:

- _conn* The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).
- _filename* The name of the program to start running.
- _result* 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:

- _conn* The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_result A char value indicating whether the function call succeeded or not.

5.45 System Command functions

Functions for sending system commands to another NXT.

Defines

- #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 Define 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 [RemoteConnection-Idle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_handle The handle of the file to close.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_filename The name of the program to delete.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_result 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:

- _conn* The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).
- _mask* The filename mask for the files you want to find.
- _handle* The handle of the found file.
- _name* The name of the found file.
- _size* The size of the found file.
- _result* 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:

- _conn* The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).
- _handle* The handle returned by the last [FindFirstFile](#) or FindNextFile call.
- _result* 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_btaddr The bluetooth address of the remote device.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_name The name of the remote device.

_btaddr The bluetooth address of the remote device.

_btsignal The signal strength of each connection on the remote device.

_freemem The number of bytes of free flash memory on the remote device.

_result 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:

- _conn* The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).
- _pmin* The protocol minor version byte.
- _pmaj* The protocol major version byte.
- _fmin* The firmware minor version byte.
- _fmaj* The firmware major version byte.
- _result* 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:

- _conn* The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).
- _id* The ID of the module from which to read data.
- _offset* The offset into the IOMap structure from which to read.
- _numbytes* The number of bytes of data to read.
- _result* 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_id The ID of the module to which to write data.

_offset The offset into the IOMap structure to which to write.

_data A byte array containing the data you are writing to the IOMap structure.

_result 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 [RemoteConnectionIdle](#) to determine when this write request is completed.

Parameters:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_id The ID of the module to which to write data.

_offset The offset into the IOMap structure to which to write.

_value A scalar variable containing the value you are writing to the IOMap structure.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_filename The name of the file to open for appending.

_handle The handle of the file.

_size The size of the file.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_filename The name of the file to open for reading.

_handle The handle of the file.

_size The size of the file.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_filename The name of the program to open for writing (i.e., create the file).

_size The size for the new file.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_filename The name of the program to open for writing (i.e., create the file).

_size The size for the new file.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_filename The name of the program to open for writing (i.e., create the file).

_size The size for the new file.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_bufnum The buffer from which to read data (0=USBPoll, 1=HiSpeed).

_len The number of bytes to read.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_bufnum The poll buffer you want to query (0=USBPoll, 1=HiSpeed).

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_handle The handle of the file you are reading from.

_numbytes The number of bytes you want to read.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_oldname The old filename.

_newname The new filename.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_name The new brick name.

_result 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:

_conn The connection slot (0..4). Connections 0 through 3 are for bluetooth connections. Connection 4 refers to the RS485 hi-speed port. See [Remote connection constants](#).

_handle The handle of the file you are writing to.

_data A byte array containing the data you are writing.

_result A char value indicating whether the function call succeeded or not.

5.46 IOCtrl module functions

Functions for accessing and modifying IOCtrl module features.

Defines

- `#define PowerDown SetIOCtrlModuleValue(IOCtlOffsetPowerOn, IOCTRL_POWERDOWN)`
Power down the NXT.
- `#define RebootInFirmwareMode SetIOCtrlModuleValue(IOCtlOffsetPowerOn, IOCTRL_BOOT)`
Reboot the NXT in firmware download mode.

5.46.1 Detailed Description

Functions for accessing and modifying IOCtrl module features.

5.46.2 Define Documentation

5.46.2.1 `#define PowerDown SetIOCtrlModuleValue(IOCtlOffsetPowerOn, IOCTRL_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(IOCtlOffsetPowerOn, IOCTRL_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.

Defines

- #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, _writeable, _result)` `__resolveHandle(_fname, _handle, _writeable, _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, _newszie, _result)` `__fileResize(_fname, _newszie, _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 Define 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:

`_handle` The file handle.
`_result` The function call result. See [Loader module error codes](#).

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:

`_fname` The name of the file to create.
`_fsize` The size of the file.

_handle The file handle output from the function call.

_result The function call result. See [Loader module error codes](#).

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:

_fname The name of the file to create.

_fsize The size of the file.

_handle The file handle output from the function call.

_result The function call result. See [Loader module error codes](#).

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:

_fname The name of the file to create.

_fsize The size of the file.

_handle The file handle output from the function call.

_result The function call result. See [Loader module error codes](#).

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:

_fname The name of the file to delete.

_result The function call result. See [Loader module error codes](#).

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:

_fname 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.

_handle The search handle input to and output from the function call.

_result The function call result. See [Loader module error codes](#).

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:

_fname On output this contains the name of the next file found that matches the pattern used when the search began by calling [FindFirstFile](#).

_handle The search handle input to and output from the function call.

_result The function call result. See [Loader module error codes](#).

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:

_value 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:

_fname The name of the file to open.

_fsize The size of the file returned by the function.

_handle The file handle output from the function call.

_result The function call result. See [Loader module error codes](#).

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:

- _fname* The name of the file to open.
- _fsize* The size of the file returned by the function.
- _handle* The file handle output from the function call.
- _result* The function call result. See [Loader module error codes](#).

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:

- _fname* The name of the file to open.
- _fsize* The size of the file returned by the function.
- _handle* The file handle output from the function call.
- _result* The function call result. See [Loader module error codes](#).

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:

- _handle* The file handle.
- _n* The variable to store the data read from the file.
- _result* The function call result. See [Loader module error codes](#).

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:

- _handle* The file handle.
- _len* The number of bytes to read. Returns the number of bytes actually read.
- _buf* The byte array where the data is stored on output.
- _result* The function call result. See [Loader module error codes](#).

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:

- _handle* The file handle.
- _n* The variable to store the data read from the file.
- _result* The function call result. See [Loader module error codes](#).

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:

- _handle* The file handle.
- _output* The variable to store the string read from the file.
- _result* The function call result. See [Loader module error codes](#).

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:

- _oldname* The old filename.
- _newname* The new filename.
- _result* The function call result. See [Loader module error codes](#).

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:

- _fname* The name of the file to resize.
- _newszie* The new size for the file.
- _result* The function call result. See [Loader module error codes](#).

5.47.2.18 #define ResolveHandle(_fname, _handle, _writeable, _result) __resolveHandle(_fname, _handle, _writeable, _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:

- _fname* The name of the file for which to resolve a handle.
- _handle* The file handle output from the function call.
- _writeable* A boolean flag indicating whether the handle is to a file open for writing (true) or reading (false).
- _result* The function call result. See [Loader module error codes](#).

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:

- _n* The variable.
- _result* 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:

- _handle* The file handle.
- _n* The value to write to the file.
- _result* The function call result. See [Loader module error codes](#).

**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:

- _handle* The file handle.
- _buf* The byte array or string containing the data to write.
- _cnt* The number of bytes actually written to the file.
- _result* The function call result. See [Loader module error codes](#).

**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:

- _handle* The file handle.
- _len* The maximum number of bytes to write on input. Returns the actual number of bytes written.
- _buf* The byte array or string containing the data to write.
- _result* The function call result. See [Loader module error codes](#).

**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:

_handle The file handle.
_n The value to write to the file.
_result The function call result. See [Loader module error codes](#).

**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:

_handle The file handle.
_str The string to write to the file.
_cnt The number of bytes actually written to the file.
_result The function call result. See [Loader module error codes](#).

**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:

_handle The file handle.
_str The string to write to the file.
_cnt The number of bytes actually written to the file.
_result The function call result. See [Loader module error codes](#).

5.48 cstdlib API

Standard C cstdlib API functions.

Defines

- #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 Define 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:

- `_arg` An unsigned random number.
`_max` 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:

- `_arg` A signed random number

5.49 cmath API

Standard C cmath API functions.

Defines

- #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 Define 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:

_bcd The value you want to convert from bcd to decimal.

_result 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.

Defines

- #define `RC_PROP_BTONOFF` 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 Define Documentation

5.50.2.1 #define RC_PROP_BTTONOFF 0x0

Set/get whether bluetooth is on or off

5.50.2.2 #define RC_PROP_DEBUGGING 0xF

Set/get enhanced firmware debugging information

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 Array operation constants

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

Defines

- #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

5.51.1 Detailed Description

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

5.51.2 Define Documentation

5.51.2.1 #define OPARR_MAX 0x05

Calculate the maximum value of the elements in the numeric input array

5.51.2.2 #define OPARR_MEAN 0x01

Calculate the mean value for the elements in the numeric input array

5.51.2.3 #define OPARR_MIN 0x04

Calculate the minimum value of the elements in the numeric input array

5.51.2.4 #define OPARR_SORT 0x06

Sort the elements in the numeric input array

5.51.2.5 #define OPARR_STD 0x03

Calculate the standard deviation of the elements in the numeric input array

5.51.2.6 #define OPARR_SUM 0x00

Calculate the sum of the elements in the numeric input array

5.51.2.7 #define OPARR_SUMSQR 0x02

Calculate the sum of the squares of the elements in the numeric input array

5.52 System Call function constants

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

Defines

- #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 CommHSControl 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

5.52.1 Detailed Description

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

5.52.2 Define Documentation

5.52.2.1 #define ColorSensorRead 34

Read data from the NXT 2.0 color sensor

5.52.2.2 #define CommBTCheckStatus 28

Check the bluetooth status

5.52.2.3 #define CommBTConnection 36

Connect or disconnect to a known bluetooth device

5.52.2.4 #define CommBTOnOff 35

Turn the bluetooth radio on or off

5.52.2.5 #define CommBTRead 30

Read from a bluetooth connection

5.52.2.6 #define CommBTWrite 29

Write to a bluetooth connections

5.52.2.7 #define CommExecuteFunction 81

Execute one of the Comm module's internal functions

5.52.2.8 #define CommHSCheckStatus 39

Check the status of the hi-speed port

5.52.2.9 #define CommHSControl 88

Control the hi-speed port

5.52.2.10 #define CommHSRead 38

Read data from the hi-speed port

5.52.2.11 #define CommHSWrite 37

Write data to the hi-speed port

5.52.2.12 #define CommLSCheckStatus 23

Check the status of a lowspeed (aka I2C) device

5.52.2.13 #define CommLSRead 22

Read from a lowspeed (aka I2C) device

5.52.2.14 #define CommLSWrite 21

Write to a lowspeed (aka I2C) device

5.52.2.15 #define CommLSWriteEx 89

Write to a lowspeed (aka I2C) device with optional restart on read

5.52.2.16 #define ComputeCalibValue 42

Compute a calibration value

5.52.2.17 #define DatalogGetTimes 45

Get datalog timing information

5.52.2.18 #define DatalogWrite 44

Write to the datalog

5.52.2.19 #define DisplayExecuteFunction 80

Execute one of the Display module's internal functions

5.52.2.20 #define DrawCircle 16

Draw a circle on the LCD screen

5.52.2.21 #define DrawEllipse 94

Draw an ellipse on the LCD screen

5.52.2.22 #define DrawFont 95

Draw text using a custom RIC-based font to the LCD screen

5.52.2.23 #define DrawGraphic 18

Draw a graphic image on the LCD screen

5.52.2.24 #define DrawGraphicArray 92

Draw a graphic image from a byte array to the LCD screen

5.52.2.25 #define DrawLine 15

Draw a line on the LCD screen

5.52.2.26 #define DrawPoint 14

Draw a single pixel on the LCD screen

5.52.2.27 #define DrawPolygon 93

Draw a polygon on the LCD screen

5.52.2.28 #define DrawRect 17

Draw a rectangle on the LCD screen

5.52.2.29 #define DrawText 13

Draw text to one of 8 LCD lines

5.52.2.30 #define FileClose 5

Close the specified file

5.52.2.31 #define FileDelete 8

Delete a file

5.52.2.32 #define FileFindFirst 83

Start a search for a file using a filename pattern

5.52.2.33 #define FileFindNext 84

Continue searching for a file

5.52.2.34 #define FileOpenAppend 2

Open a file for appending to the end of the file

5.52.2.35 #define FileOpenRead 0

Open a file for reading

5.52.2.36 #define FileOpenReadLinear 87

Open a linear file for reading

5.52.2.37 #define FileOpenWrite 1

Open a file for writing (creates a new file)

5.52.2.38 #define FileOpenWriteLinear 85

Open a linear file for writing

5.52.2.39 #define FileOpenWriteNonLinear 86

Open a non-linear file for writing

5.52.2.40 #define FileRead 3

Read from the specified file

5.52.2.41 #define FileRename 7

Rename a file

5.52.2.42 #define FileResize 91

Resize a file (not yet implemented)

5.52.2.43 #define FileResolveHandle 6

Get a file handle for the specified filename if it is already open

5.52.2.44 #define FileSeek 90

Seek to a specific position in an open file

5.52.2.45 #define FileTell 98

Return the current file position in an open file

5.52.2.46 #define FileWrite 4

Write to the specified file

5.52.2.47 #define GetStartTick 25

Get the current system tick count

5.52.2.48 #define InputPinFunction 77

Execute the Input module's pin function

5.52.2.49 #define IOMapRead 32

Read data from one of the firmware module's IOMap structures using the module's name

5.52.2.50 #define IOMapReadByID 78

Read data from one of the firmware module's IOMap structures using the module's ID

5.52.2.51 #define IOMapWrite 33

Write data to one of the firmware module's IOMap structures using the module's name

5.52.2.52 #define IOMapWriteByID 79

Write data to one of the firmware module's IOMap structures using the module's ID

5.52.2.53 #define KeepAlive 31

Reset the NXT sleep timer

5.52.2.54 #define ListFiles 47

List files that match the specified filename pattern

5.52.2.55 #define LoaderExecuteFunction 82

Execute one of the Loader module's internal functions

5.52.2.56 #define MemoryManager 96

Read memory manager information, optionally compacting the dataspace first

5.52.2.57 #define MessageRead 27

Read a message from a mailbox

5.52.2.58 #define MessageWrite 26

Write a message to a mailbox

5.52.2.59 #define RandomEx 99

Generate a random number or seed the RNG.

5.52.2.60 #define RandomNumber 24

Generate a random number

5.52.2.61 #define ReadButton 20

Read the current button state

5.52.2.62 #define ReadLastResponse 97

Read the last response packet received by the NXT. Optionally clear the value after reading it.

5.52.2.63 #define ReadSemData 40

Read motor semaphore data

5.52.2.64 #define SetScreenMode 19

Set the screen mode

5.52.2.65 #define SetSleepTimeoutVal 46

Set the NXT sleep timeout value

5.52.2.66 #define SoundGetState 11

Get the current sound module state

5.52.2.67 #define SoundPlayFile 9

Play a sound or melody file

5.52.2.68 #define SoundPlayTone 10

Play a simple tone with the specified frequency and duration

5.52.2.69 #define SoundSetState 12

Set the sound module state

5.52.2.70 #define UpdateCalibCacheInfo 43

Update sensor calibration cache information

5.52.2.71 #define WriteSemData 41

Write motor semaphore data

5.53 Line number constants

Line numbers for use with DrawText system function.

Defines

- #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

5.53.1 Detailed Description

Line numbers for use with DrawText system function.

See also:

[SysDrawText\(\)](#), [TextOut\(\)](#), [NumOut\(\)](#)

5.53.2 Define Documentation**5.53.2.1 #define LCD_LINE1 56**

The 1st line of the LCD screen

5.53.2.2 #define LCD_LINE2 48

The 2nd line of the LCD screen

5.53.2.3 #define LCD_LINE3 40

The 3rd line of the LCD screen

5.53.2.4 #define LCD_LINE4 32

The 4th line of the LCD screen

5.53.2.5 #define LCD_LINE5 24

The 5th line of the LCD screen

5.53.2.6 #define LCD_LINE6 16

The 6th line of the LCD screen

5.53.2.7 #define LCD_LINE7 8

The 7th line of the LCD screen

5.53.2.8 #define LCD_LINE8 0

The 8th line of the LCD screen

5.54 Time constants

Constants for use with the [Wait\(\)](#) function.

Defines

- #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

5.54.1 Detailed Description

Constants for use with the [Wait\(\)](#) function.

See also:

[Wait\(\)](#)

5.54.2 Define Documentation

5.54.2.1 #define MIN_1 60000

1 minute

5.54.2.2 #define MS_1 1

1 millisecond

5.54.2.3 #define MS_10 10

10 milliseconds

5.54.2.4 #define MS_100 100

100 milliseconds

5.54.2.5 #define MS_150 150

150 milliseconds

5.54.2.6 #define MS_2 2

2 milliseconds

5.54.2.7 #define MS_20 20

20 milliseconds

5.54.2.8 #define MS_200 200

200 milliseconds

5.54.2.9 #define MS_250 250

250 milliseconds

5.54.2.10 #define MS_3 3

3 milliseconds

5.54.2.11 #define MS_30 30

30 milliseconds

5.54.2.12 #define MS_300 300

300 milliseconds

5.54.2.13 #define MS_350 350

350 milliseconds

5.54.2.14 #define MS_4 4

4 milliseconds

5.54.2.15 #define MS_40 40

40 milliseconds

5.54.2.16 #define MS_400 400

400 milliseconds

5.54.2.17 #define MS_450 450

450 milliseconds

5.54.2.18 #define MS_5 5

5 milliseconds

5.54.2.19 #define MS_50 50

50 milliseconds

5.54.2.20 #define MS_500 500

500 milliseconds

5.54.2.21 #define MS_6 6

6 milliseconds

5.54.2.22 #define MS_60 60

60 milliseconds

5.54.2.23 #define MS_600 600

600 milliseconds

5.54.2.24 #define MS_7 7

7 milliseconds

5.54.2.25 #define MS_70 70

70 milliseconds

5.54.2.26 #define MS_700 700

700 milliseconds

5.54.2.27 #define MS_8 8

8 milliseconds

5.54.2.28 #define MS_80 80

80 milliseconds

5.54.2.29 #define MS_800 800

800 milliseconds

5.54.2.30 #define MS_9 9

9 milliseconds

5.54.2.31 #define MS_90 90

90 milliseconds

5.54.2.32 #define MS_900 900

900 milliseconds

5.54.2.33 #define SEC_1 1000

1 second

5.54.2.34 #define SEC_10 10000

10 seconds

5.54.2.35 #define SEC_15 15000

15 seconds

5.54.2.36 #define SEC_2 2000

2 seconds

5.54.2.37 #define SEC_20 20000

20 seconds

5.54.2.38 #define SEC_3 3000

3 seconds

5.54.2.39 #define SEC_30 30000

30 seconds

5.54.2.40 #define SEC_4 4000

4 seconds

5.54.2.41 #define SEC_5 5000

5 seconds

5.54.2.42 #define SEC_6 6000

6 seconds

5.54.2.43 #define SEC_7 7000

7 seconds

5.54.2.44 #define SEC_8 8000

8 seconds

5.54.2.45 #define SEC_9 9000

9 seconds

5.55 Mailbox constants

Mailbox number constants should be used to avoid confusing NXT-G users.

Defines

- #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

5.55.1 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\(\)](#)

5.55.2 Define Documentation**5.55.2.1 #define MAILBOX1 0**

Mailbox number 1

5.55.2.2 #define MAILBOX10 9

Mailbox number 10

5.55.2.3 #define MAILBOX2 1

Mailbox number 2

5.55.2.4 #define MAILBOX3 2

Mailbox number 3

5.55.2.5 #define MAILBOX4 3

Mailbox number 4

5.55.2.6 #define MAILBOX5 4

Mailbox number 5

5.55.2.7 #define MAILBOX6 5

Mailbox number 6

5.55.2.8 #define MAILBOX7 6

Mailbox number 7

5.55.2.9 #define MAILBOX8 7

Mailbox number 8

5.55.2.10 #define MAILBOX9 8

Mailbox number 9

5.56 VM state constants

Constants defining possible VM states.

Defines

- #define **TIMES_UP** 6
- #define **ROTATE_QUEUE** 5
- #define **STOP_REQ** 4
- #define **BREAKOUT_REQ** 3
- #define **CLUMP_SUSPEND** 2
- #define **CLUMP_DONE** 1

5.56.1 Detailed Description

Constants defining possible VM states.

5.56.2 Define Documentation**5.56.2.1 #define BREAKOUT_REQ 3**

VM should break out of current thread

5.56.2.2 #define CLUMP_DONE 1

VM has finished executing thread

5.56.2.3 #define CLUMP_SUSPEND 2

VM should suspend thread

5.56.2.4 #define ROTATE_QUEUE 5

VM should rotate queue

5.56.2.5 #define STOP_REQ 4

VM should stop executing program

5.56.2.6 #define TIMES_UP 6

VM time is up

5.57 Fatal errors

Constants defining various fatal error conditions.

Defines

- #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

5.57.1 Detailed Description

Constants defining various fatal error conditions.

5.57.2 Define Documentation**5.57.2.1 #define ERR_ARG -1**

0xFF Bad arguments

5.57.2.2 #define ERR_BAD_POOL_SIZE -10

0xF6 VarsCmd.PoolSize > POOL_MAX_SIZE

5.57.2.3 #define ERR_BAD_PTR -6

0xFA Someone passed us a bad pointer!

5.57.2.4 #define ERR_CLUMP_COUNT -7

0xF9 (FileClumpCount == 0 || FileClumpCount >= NOT_A_CLUMP)

5.57.2.5 #define ERR_DEFAULT_OFFSETS -14

0xF2 (DefaultsOffset != FileOffsets.DynamicDefaults) || (DefaultsOffset + FileOffsets.DynamicDefaultsSize != FileOffsets.DSDefaultsSize)

5.57.2.6 #define ERR_FILE -3

0xFD Malformed file contents

5.57.2.7 #define ERR_INSANE_OFFSET -9

0xF7 CurrOffset != (DataSize - VarsCmd.CodespaceCount * 2)

5.57.2.8 #define ERR_INSTR -2

0xFE Illegal bytecode instruction

5.57.2.9 #define ERR_LOADER_ERR -11

0xF5 LOADER_ERR(LStatus) != SUCCESS || pData == NULL || DataSize == 0

5.57.2.10 #define ERR_MEM -5

0xFB Insufficient memory available

5.57.2.11 #define ERR_MEMMGR_FAIL -15

0xF1 (UBYTE *)VarsCmd.MemMgr.pDopeVectorArray != VarsCmd.pDataspace + DV_ARRAY[0].Offset

5.57.2.12 #define ERR_NO_ACTIVE_CLUMP -13

0xF3 VarsCmd.RunQ.Head == NOT_A_CLUMP

5.57.2.13 #define ERR_NO_CODE -8

0xF8 VarsCmd.CodespaceCount == 0

5.57.2.14 #define ERR_NON_FATAL -16

Fatal errors are greater than this value

5.57.2.15 #define ERR_SPOTCHECK_FAIL -12

0xF4 ((UBYTE*)(VarsCmd.pCodespace) < pData) (c_cmd.c 1893)

5.57.2.16 #define ERR_VER -4

0xFC Version mismatch between firmware and compiler

5.58 General errors

Constants defining general error conditions.

Defines

- #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

5.58.1 Detailed Description

Constants defining general error conditions.

5.58.2 Define Documentation**5.58.2.1 #define ERR_INVALID_FIELD -17**

0xEF Attempted to access invalid field of a structure

5.58.2.2 #define ERR_INVALID_PORT -16

0xF0 Bad input or output port specified

5.58.2.3 #define ERR_INVALID_QUEUE -18

0xEE Illegal queue ID specified

5.58.2.4 #define ERR_INVALID_SIZE -19

0xED Illegal size specified

5.58.2.5 #define ERR_NO_PROG -20

0xEC No active program

5.59 Communications specific errors

Constants defining communication error conditions.

Defines

- #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

5.59.1 Detailed Description

Constants defining communication error conditions.

5.59.2 Define Documentation**5.59.2.1 #define ERR_COMM_BUFFER_FULL -34**

0xDE No room in comm buffer

5.59.2.2 #define ERR_COMM_BUS_ERR -35

0xDD Something went wrong on the communications bus

5.59.2.3 #define ERR_COMM_CHAN_INVALID -33

0xDF Specified channel/connection is not valid

5.59.2.4 #define ERR_COMM_CHAN_NOT_READY -32

0xE0 Specified channel/connection not configured or busy

5.60 Remote control (direct commands) errors

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

Defines

- #define [ERR_RC_ILLEGAL_VAL](#) -64
- #define [ERR_RC_BAD_PACKET](#) -65
- #define [ERR_RC_UNKNOWN_CMD](#) -66
- #define [ERR_RC_FAILED](#) -67

5.60.1 Detailed Description

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

5.60.2 Define Documentation**5.60.2.1 #define ERR_RC_BAD_PACKET -65**

0xBF Clearly insane packet

5.60.2.2 #define ERR_RC_FAILED -67

0xBD Request failed (i.e. specified file not found)

5.60.2.3 #define ERR_RC_ILLEGAL_VAL -64

0xC0 Data contains out-of-range values

5.60.2.4 #define ERR_RC_UNKNOWN_CMD -66

0xBE Unknown command opcode

5.61 Program status constants

Constants defining various states of the command module virtual machine.

Defines

- #define PROG_IDLE 0
- #define PROG_OK 1
- #define PROG_RUNNING 2
- #define PROG_ERROR 3
- #define PROG_ABORT 4
- #define PROG_RESET 5

5.61.1 Detailed Description

Constants defining various states of the command module virtual machine.

5.61.2 Define Documentation**5.61.2.1 #define PROG_ABORT 4**

Program has been aborted

5.61.2.2 #define PROG_ERROR 3

A program error has occurred

5.61.2.3 #define PROG_IDLE 0

Program state is idle

5.61.2.4 #define PROG_OK 1

Program state is okay

5.61.2.5 #define PROG_RESET 5

Program has been reset

5.61.2.6 #define PROG_RUNNING 2

Program is running

5.62 Command module IOMAP offsets

Constant offsets into the Command module IOMAP structure.

Defines

- #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

5.62.1 Detailed Description

Constant offsets into the Command module IOMAP structure.

5.62.2 Define Documentation**5.62.2.1 #define CommandOffsetActivateFlag 30**

Offset to the activate flag

5.62.2.2 #define CommandOffsetAwake 29

Offset to the VM's awake state

5.62.2.3 #define CommandOffsetDeactivateFlag 31

Offset to the deactivate flag

5.62.2.4 #define CommandOffsetFileName 32

Offset to the running program's filename

5.62.2.5 #define CommandOffsetFormatString 0

Offset to the format string

5.62.2.6 #define CommandOffsetMemoryPool 52

Offset to the VM's memory pool

5.62.2.7 #define CommandOffsetOffsetDS 24

Offset to the running program's data space (DS)

5.62.2.8 #define CommandOffsetOffsetDVA 26

Offset to the running program's DOPE vector address (DVA)

5.62.2.9 #define CommandOffsetPRCHandler 16

Offset to the RC Handler function pointer

5.62.2.10 #define CommandOffsetProgStatus 28

Offset to the running program's status

5.62.2.11 #define CommandOffsetSyncTick 32824

Offset to the VM sync tick

5.62.2.12 #define CommandOffsetSyncTime 32820

Offset to the VM sync time

5.62.2.13 #define CommandOffsetTick 20

Offset to the VM's current tick

5.63 IOCtrl module constants

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

Modules

- [PowerOn constants](#)

Use these constants to power down the NXT or boot it into SAMBA (aka firmware download) mode.

- [IOCtrl module IOMAP offsets](#)

Constant offsets into the IOCtrl module IOMAP structure.

5.63.1 Detailed Description

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

5.64 PowerOn constants

Use these constants to power down the NXT or boot it into SAMBA (aka firmware download) mode.

Defines

- #define **IOCTRL_POWERDOWN** 0x5A00
- #define **IOCTRL_BOOT** 0xA55A

5.64.1 Detailed Description

Use these constants to power down the NXT or boot it into SAMBA (aka firmware download) mode.

5.64.2 Define Documentation**5.64.2.1 #define IOCTRL_BOOT 0xA55A**

Reboot the NXT into SAMBA mode

5.64.2.2 #define IOCTRL_POWERDOWN 0x5A00

Power down the NXT

5.65 IOCtrl module IOMAP offsets

Constant offsets into the IOCtrl module IOMAP structure.

Defines

- #define **IOCctrlOffsetPowerOn** 0

5.65.1 Detailed Description

Constant offsets into the IOCtrl module IOMAP structure.

5.65.2 Define Documentation**5.65.2.1 #define IOCctrlOffsetPowerOn 0**

Offset to power on field

5.66 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.

Defines

- `#define EOF -1`
- `#define NULL 0`

5.66.1 Detailed Description

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

5.66.2 Define Documentation

5.66.2.1 #define EOF -1

A constant representing end of file

5.66.2.2 #define NULL 0

A constant representing NULL

5.67 Loader module IOMAP offsets

Constant offsets into the Loader module IOMAP structure.

Defines

- #define `LoaderOffsetPFunc` 0
- #define `LoaderOffsetFreeUserFlash` 4

5.67.1 Detailed Description

Constant offsets into the Loader module IOMAP structure.

5.67.2 Define Documentation**5.67.2.1 #define LoaderOffsetFreeUserFlash 4**

Offset to the amount of free user flash

5.67.2.2 #define LoaderOffsetPFunc 0

Offset to the Loader module function pointer

5.68 Loader module error codes

Error codes returned by functions in the Loader module (file access).

Defines

- #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_OUTOFCBOUNDARY** 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

5.68.1 Detailed Description

Error codes returned by functions in the Loader module (file access).

5.68.2 Define Documentation

5.68.2.1 #define **LDR_APPENDNOTPOSSIBLE** 0x8D00

Only datafiles can be appended to.

5.68.2.2 #define **LDR_BTBUSY** 0x9400

The bluetooth system is busy.

5.68.2.3 #define **LDR_BTCONNECTFAIL** 0x9500

Bluetooth connection attempt failed.

5.68.2.4 #define **LDR_BTTIMEOUT** 0x9600

A timeout in the bluetooth system has occurred.

5.68.2.5 #define **LDR_ENDOFFILE** 0x8500

The end of the file has been reached.

5.68.2.6 #define LDR_EOFEXPECTED 0x8400

EOF expected.

5.68.2.7 #define LDR_FILEEXISTS 0x8F00

A file with the same name already exists.

5.68.2.8 #define LDR_FILEISBUSY 0x8B00

The file is already being used.

5.68.2.9 #define LDR_FILEISFULL 0x8E00

The allocated file size has been filled.

5.68.2.10 #define LDR_FILENOFOUND 0x8700

No files matched the search criteria.

5.68.2.11 #define LDR_FILETX_CLOSEERROR 0x9B00

Error transmitting file: attempt to close file failed.

5.68.2.12 #define LDR_FILETX_DSEXISTS 0x9800

Error transmitting file: destination file exists.

5.68.2.13 #define LDR_FILETX_SRCMISSING 0x9900

Error transmitting file: source file is missing.

5.68.2.14 #define LDR_FILETX_STREAMERROR 0x9A00

Error transmitting file: a stream error occurred.

5.68.2.15 #define LDR_FILETX_TIMEOUT 0x9700

Error transmitting file: a timeout occurred.

5.68.2.16 #define LDR_HANDLEALREADYCLOSED 0x8800

The file handle has already been closed.

5.68.2.17 #define LDR_ILLEGALFILENAME 0x9200

Filename length to long or attempted open a system file (*.rxe, *.rtm, or *.sys) for writing as a datafile.

5.68.2.18 #define LDR_ILLEGALHANDLE 0x9300

Invalid file handle.

5.68.2.19 #define LDR_INPROGRESS 0x0001

The function is executing but has not yet completed.

5.68.2.20 #define LDR_INVALIDSEEK 0x9C00

Invalid file seek operation.

5.68.2.21 #define LDR_MODULENOTFOUND 0x9000

No modules matched the specified search criteria.

5.68.2.22 #define LDR_NOLINEARSPACE 0x8900

Not enough linear flash memory is available.

5.68.2.23 #define LDR_NOMOREFILES 0x8300

The maximum number of files has been reached.

5.68.2.24 #define LDR_NOMOREHANDLES 0x8100

All available file handles are in use.

5.68.2.25 #define LDR_NOSPACE 0x8200

Not enough free flash memory for the specified file size.

5.68.2.26 #define LDR_NOTLINEARFILE 0x8600

The specified file is not linear.

5.68.2.27 #define LDR_NOWRITEBUFFERS 0x8C00

No more write buffers are available.

5.68.2.28 #define LDR_OUTOFCBOUNDARY 0x9100

Specified IOMap offset is outside the bounds of the IOMap.

5.68.2.29 #define LDR_REQPIN 0x0002

A PIN exchange request is in progress.

5.68.2.30 #define LDR_SUCCESS 0x0000

The function completed successfully.

5.68.2.31 #define LDR_UNDEFINEDERROR 0x8A00

An undefined error has occurred.

5.69 Loader module function constants

Constants defining the functions provided by the Loader module.

Defines

- #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_OPENWRITELINEAR 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

5.69.1 Detailed Description

Constants defining the functions provided by the Loader module.

5.69.2 Define Documentation

5.69.2.1 #define LDR_CMD_BOOTCMD 0x97

Reboot the NXT into SAMBA mode

5.69.2.2 #define LDR_CMD_BTFACTORYRESET 0xA4

Reset bluetooth configuration to factory defaults

5.69.2.3 #define LDR_CMD_BTGETADR 0x9A

Get the NXT's bluetooth brick address

5.69.2.4 #define LDR_CMD_CLOSE 0x84

Close a file handle

5.69.2.5 #define LDR_CMD_CLOSEMODHANDLE 0x92

Close a module handle

5.69.2.6 #define LDR_CMD_CROPDATAFILE 0x8D

Crop a data file to its used space

5.69.2.7 #define LDR_CMD_DELETE 0x85

Delete a file

5.69.2.8 #define LDR_CMD_DELETEUSERFLASH 0xA0

Delete all files from user flash memory

5.69.2.9 #define LDR_CMD_DEVICEINFO 0x9B

Read device information

5.69.2.10 #define LDR_CMD_FINDFIRST 0x86

Find the first file matching the specified pattern

5.69.2.11 #define LDR_CMD_FINDFIRSTMODULE 0x90

Find the first module matching the specified pattern

5.69.2.12 #define LDR_CMD_FINDNEXT 0x87

Find the next file matching the specified pattern

5.69.2.13 #define LDR_CMD_FINDNEXTMODULE 0x91

Find the next module matching the specified pattern

5.69.2.14 #define LDR_CMD_IOMAPREAD 0x94

Read data from a module IOMAP

5.69.2.15 #define LDR_CMD_IOMAPWRITE 0x95

Write data to a module IOMAP

5.69.2.16 #define LDR_CMD_OPENAPPENDDATA 0x8C

Open a data file for appending

5.69.2.17 #define LDR_CMD_OPENREAD 0x80

Open a file for reading

5.69.2.18 #define LDR_CMD_OPENREADLINEAR 0x8A

Open a linear file for reading

5.69.2.19 #define LDR_CMD_OPENWRITE 0x81

Open a file for writing

5.69.2.20 #define LDR_CMD_OPENWRITEDATA 0x8B

Open a data file for writing

5.69.2.21 #define LDR_CMD_OPENWITELINEAR 0x89

Open a linear file for writing

5.69.2.22 #define LDR_CMD_POLLCMD 0xA2

Poll command

5.69.2.23 #define LDR_CMD_POLLCMDLEN 0xA1

Read poll command length

5.69.2.24 #define LDR_CMD_READ 0x82

Read from a file

5.69.2.25 #define LDR_CMD_RENAMEFILE 0xA3

Rename a file

5.69.2.26 #define LDR_CMD_RESIZEDATAFILE 0xD0

Resize a data file

5.69.2.27 #define LDR_CMD_SEEKFROMCURRENT 0xD2

Seek from the current position

5.69.2.28 #define LDR_CMD_SEEKFROMEND 0xD3

Seek from the end of the file

5.69.2.29 #define LDR_CMD_SEEKFROMSTART 0xD1

Seek from the start of the file

5.69.2.30 #define LDR_CMD_SETBRICKNAME 0x98

Set the NXT's brick name

5.69.2.31 #define LDR_CMD_VERSIONS 0x88

Read firmware version information

5.69.2.32 #define LDR_CMD_WRITE 0x83

Write to a file

5.70 Sound module constants

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

Modules

- [SoundFlags constants](#)

Constants for use with the SoundFlags() function.

- [SoundState constants](#)

Constants for use with the SoundState() function.

- [SoundMode constants](#)

Constants for use with the SoundMode() function.

- [Sound module IOMAP offsets](#)

Constant offsets into the Sound module IOMAP structure.

- [Sound module miscellaneous constants](#)

Constants defining miscellaneous sound module aspects.

- [Tone constants](#)

Constants for use in the SoundPlayTone() API function.

5.70.1 Detailed Description

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

5.71 SoundFlags constants

Constants for use with the SoundFlags() function.

Defines

- #define [SOUND_FLAGS_IDLE](#) 0x00
- #define [SOUND_FLAGS_UPDATE](#) 0x01
- #define [SOUND_FLAGS_RUNNING](#) 0x02

5.71.1 Detailed Description

Constants for use with the SoundFlags() function.

See also:

[SoundFlags\(\)](#)

5.71.2 Define Documentation**5.71.2.1 #define SOUND_FLAGS_IDLE 0x00**

R - Sound is idle

5.71.2.2 #define SOUND_FLAGS_RUNNING 0x02

R - Currently processing a tone or file

5.71.2.3 #define SOUND_FLAGS_UPDATE 0x01

W - Make changes take effect

5.72 SoundState constants

Constants for use with the SoundState() function.

Defines

- #define **SOUND_STATE_IDLE** 0x00
- #define **SOUND_STATE_FILE** 0x02
- #define **SOUND_STATE_TONE** 0x03
- #define **SOUND_STATE_STOP** 0x04

5.72.1 Detailed Description

Constants for use with the SoundState() function.

See also:

SoundState()

5.72.2 Define Documentation**5.72.2.1 #define SOUND_STATE_FILE 0x02**

R - Processing a file of sound/melody data

5.72.2.2 #define SOUND_STATE_IDLE 0x00

R - Idle, ready for start sound (SOUND_UPDATE)

5.72.2.3 #define SOUND_STATE_STOP 0x04

W - Stop sound immediately and close hardware

5.72.2.4 #define SOUND_STATE_TONE 0x03

R - Processing a play tone request

5.73 SoundMode constants

Constants for use with the SoundMode() function.

Defines

- #define **SOUND_MODE_ONCE** 0x00
- #define **SOUND_MODE_LOOP** 0x01
- #define **SOUND_MODE_TONE** 0x02

5.73.1 Detailed Description

Constants for use with the SoundMode() function.

See also:

SoundMode()

5.73.2 Define Documentation**5.73.2.1 #define SOUND_MODE_LOOP 0x01**

W - Play file until writing SOUND_STATE_STOP into SoundState

5.73.2.2 #define SOUND_MODE_ONCE 0x00

W - Only play file once

5.73.2.3 #define SOUND_MODE_TONE 0x02

W - Play tone specified in Frequency for Duration ms

5.74 Sound module IOMAP offsets

Constant offsets into the Sound module IOMAP structure.

Defines

- #define [SoundOffsetFreq](#) 0
- #define [SoundOffsetDuration](#) 2
- #define [SoundOffsetSampleRate](#) 4
- #define [SoundOffsetSoundFilename](#) 6
- #define [SoundOffsetFlags](#) 26
- #define [SoundOffsetState](#) 27
- #define [SoundOffsetMode](#) 28
- #define [SoundOffsetVolume](#) 29

5.74.1 Detailed Description

Constant offsets into the Sound module IOMAP structure.

5.74.2 Define Documentation

5.74.2.1 #define SoundOffsetDuration 2

RW - Tone duration [mS] (2 bytes)

5.74.2.2 #define SoundOffsetFlags 26

RW - Play flag - described above (1 byte) [SoundFlags constants](#)

5.74.2.3 #define SoundOffsetFreq 0

RW - Tone frequency [Hz] (2 bytes)

5.74.2.4 #define SoundOffsetMode 28

RW - Play mode - described above (1 byte) [SoundMode constants](#)

5.74.2.5 #define SoundOffsetSampleRate 4

RW - Sound file sample rate [2000..16000] (2 bytes)

5.74.2.6 #define SoundOffsetSoundFilename 6

RW - Sound/melody filename (20 bytes)

5.74.2.7 #define SoundOffsetState 27

RW - Play state - described above (1 byte) [SoundState constants](#)

5.74.2.8 #define SoundOffsetVolume 29

RW - Sound/melody volume [0..4] 0 = off (1 byte)

5.75 Sound module miscellaneous constants

Constants defining miscellaneous sound module aspects.

Defines

- #define [FREQUENCY_MIN](#) 220
- #define [FREQUENCY_MAX](#) 14080
- #define [SAMPLERATE_MIN](#) 2000
- #define [SAMPLERATE_DEFAULT](#) 8000
- #define [SAMPLERATE_MAX](#) 16000

5.75.1 Detailed Description

Constants defining miscellaneous sound module aspects.

5.75.2 Define Documentation**5.75.2.1 #define FREQUENCY_MAX 14080**

Maximum frequency [Hz]

5.75.2.2 #define FREQUENCY_MIN 220

Minimum frequency [Hz]

5.75.2.3 #define SAMPLERATE_DEFAULT 8000

Default sample rate [sps]

5.75.2.4 #define SAMPLERATE_MAX 16000

Max sample rate [sps]

5.75.2.5 #define SAMPLERATE_MIN 2000

Min sample rate [sps]

5.76 Tone constants

Constants for use in the [SoundPlayTone\(\)](#) API function.

Defines

- #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

5.76.1 Detailed Description

Constants for use in the [SoundPlayTone\(\)](#) API function.

See also:

[SoundPlayTone\(\)](#)

5.76.2 Define Documentation

5.76.2.1 #define TONE_A3 220

Third octave A

5.76.2.2 #define TONE_A4 440

Fourth octave A

5.76.2.3 #define TONE_A5 880

Fifth octave A

5.76.2.4 #define TONE_A6 1760

Sixth octave A

5.76.2.5 #define TONE_A7 3520

Seventh octave A

5.76.2.6 #define TONE_AS3 233

Third octave A sharp

5.76.2.7 #define TONE_AS4 466

Fourth octave A sharp

5.76.2.8 #define TONE_AS5 932

Fifth octave A sharp

5.76.2.9 #define TONE_AS6 1865

Sixth octave A sharp

5.76.2.10 #define TONE_AS7 3729

Seventh octave A sharp

5.76.2.11 #define TONE_B3 247

Third octave B

5.76.2.12 #define TONE_B4 494

Fourth octave B

5.76.2.13 #define TONE_B5 988

Fifth octave B

5.76.2.14 #define TONE_B6 1976

Sixth octave B

5.76.2.15 #define TONE_B7 3951

Seventh octave B

5.76.2.16 #define TONE_C4 262

Fourth octave C

5.76.2.17 #define TONE_C5 523

Fifth octave C

5.76.2.18 #define TONE_C6 1047

Sixth octave C

5.76.2.19 #define TONE_C7 2093

Seventh octave C

5.76.2.20 #define TONE_CS4 277

Fourth octave C sharp

5.76.2.21 #define TONE_CS5 554

Fifth octave C sharp

5.76.2.22 #define TONE_CS6 1109

Sixth octave C sharp

5.76.2.23 #define TONE_CS7 2217

Seventh octave C sharp

5.76.2.24 #define TONE_D4 294

Fourth octave D

5.76.2.25 #define TONE_D5 587

Fifth octave D

5.76.2.26 #define TONE_D6 1175

Sixth octave D

5.76.2.27 #define TONE_D7 2349

Seventh octave D

5.76.2.28 #define TONE_DS4 311

Fourth octave D sharp

5.76.2.29 #define TONE_DS5 622

Fifth octave D sharp

5.76.2.30 #define TONE_DS6 1245

Sixth octave D sharp

5.76.2.31 #define TONE_DS7 2489

Seventh octave D sharp

5.76.2.32 #define TONE_E4 330

Fourth octave E

5.76.2.33 #define TONE_E5 659

Fifth octave E

5.76.2.34 #define TONE_E6 1319

Sixth octave E

5.76.2.35 #define TONE_E7 2637

Seventh octave E

5.76.2.36 #define TONE_F4 349

Fourth octave F

5.76.2.37 #define TONE_F5 698

Fifth octave F

5.76.2.38 #define TONE_F6 1397

Sixth octave F

5.76.2.39 #define TONE_F7 2794

Seventh octave F

5.76.2.40 #define TONE_FS4 370

Fourth octave F sharp

5.76.2.41 #define TONE_FS5 740

Fifth octave F sharp

5.76.2.42 #define TONE_FS6 1480

Sixth octave F sharp

5.76.2.43 #define TONE_FS7 2960

Seventh octave F sharp

5.76.2.44 #define TONE_G4 392

Fourth octave G

5.76.2.45 #define TONE_G5 784

Fifth octave G

5.76.2.46 #define TONE_G6 1568

Sixth octave G

5.76.2.47 #define TONE_G7 3136

Seventh octave G

5.76.2.48 #define TONE_GS4 415

Fourth octave G sharp

5.76.2.49 #define TONE_GS5 831

Fifth octave G sharp

5.76.2.50 #define TONE_GS6 1661

Sixth octave G sharp

5.76.2.51 #define TONE_GS7 3322

Seventh octave G sharp

5.77 Button module constants

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

Modules

- [Button name constants](#)

Constants to specify which button to use with button module functions.

- [ButtonState constants](#)

Constants for use with the ButtonState() function.

- [Button module IOMAP offsets](#)

Constant offsets into the Button module IOMAP structure.

5.77.1 Detailed Description

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

5.78 Button name constants

Constants to specify which button to use with button module functions.

Defines

- #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

5.78.1 Detailed Description

Constants to specify which button to use with button module functions.

See also:

ButtonPressed(), ButtonState(), ButtonCount(), [ReadButtonEx\(\)](#), SysReadButton(), ReadButtonType

5.78.2 Define Documentation**5.78.2.1 #define BTN1 0**

The exit button.

5.78.2.2 #define BTN2 1

The right button.

5.78.2.3 #define BTN3 2

The left button.

5.78.2.4 #define BTN4 3

The enter button.

5.78.2.5 #define BTNCENTER BTN4

The enter button.

5.78.2.6 #define BTNEXIT BTN1

The exit button.

5.78.2.7 #define BTNLEFT BTN3

The left button.

5.78.2.8 #define BTNRIGHT BTN2

The right button.

5.78.2.9 #define NO_OF_BTNS 4

The number of NXT buttons.

5.79 ButtonState constants

Constants for use with the ButtonState() function.

Defines

- #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

5.79.1 Detailed Description

Constants for use with the ButtonState() function. The _EV values can be combined together using a bitwise OR operation.

See also:

ButtonState()

5.79.2 Define Documentation**5.79.2.1 #define BTNSTATE_LONG_PRESSED_EV 0x04**

Button is in the long pressed state.

5.79.2.2 #define BTNSTATE_LONG_RELEASED_EV 0x08

Button is in the long released state.

5.79.2.3 #define BTNSTATE_NONE 0x10

The default button state.

5.79.2.4 #define BTNSTATE_PRESSED_EV 0x01

Button is in the pressed state.

5.79.2.5 #define BTNSTATE_PRESSED_STATE 0x80

A bitmask for the button pressed state

5.79.2.6 #define BTNSTATE_SHORT_RELEASED_EV 0x02

Button is in the short released state.

5.80 Button module IOMAP offsets

Constant offsets into the Button module IOMAP structure.

Defines

- #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)

5.80.1 Detailed Description

Constant offsets into the Button module IOMAP structure.

5.80.2 Define Documentation

5.80.2.1 #define ButtonOffsetLongPressCnt(b) (((b)*8)+1)

Offset to the LongPressCnt field. This field stores the long press count.

5.80.2.2 #define ButtonOffsetLongRelCnt(b) (((b)*8)+3)

Offset to the LongRelCnt field. This field stores the long release count.

5.80.2.3 #define ButtonOffsetPressedCnt(b) (((b)*8)+0)

Offset to the PressedCnt field. This field stores the press count.

5.80.2.4 #define ButtonOffsetRelCnt(b) (((b)*8)+4)

Offset to the RelCnt field. This field stores the release count.

5.80.2.5 #define ButtonOffsetShortRelCnt(b) (((b)*8)+2)

Offset to the ShortRelCnt field. This field stores the short release count.

5.80.2.6 #define ButtonOffsetState(b) ((b)+32)

Offset to the State field. This field stores the current button state.

5.81 Ui module constants

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

Modules

- [CommandFlags constants](#)

Constants for use with the CommandFlags() function.

- [UIState constants](#)

Constants for use with the UIState() function.

- [UIButton constants](#)

Constants for use with the UIButton() function.

- [BluetoothState constants](#)

Constants for use with the BluetoothState() function.

- [VM run state constants](#)

Constants for use with the VMRunState() function.

- [Ui module IOMAP offsets](#)

Constant offsets into the Ui module IOMAP structure.

5.81.1 Detailed Description

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

5.82 CommandFlags constants

Constants for use with the CommandFlags() function.

Defines

- #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

5.82.1 Detailed Description

Constants for use with the CommandFlags() function.

See also:

[CommandFlags\(\)](#)

5.82.2 Define Documentation

5.82.2.1 #define `UI_FLAGS_BUSY` 0x40

R - UI busy running or datalogging (popup disabled)

5.82.2.2 #define `UI_FLAGS_DISABLE_EXIT` 0x04

RW - Disable exit button

5.82.2.3 #define `UI_FLAGS_DISABLE_LEFT_RIGHT_ENTER` 0x02

RW - Disable left, right and enter button

5.82.2.4 #define UI_FLAGS_ENABLE_STATUS_UPDATE 0x80

W - Enable status line to be updated

5.82.2.5 #define UI_FLAGS_EXECUTE_LMS_FILE 0x20

W - Execute LMS file in "LMSfilename" (Try It)

5.82.2.6 #define UI_FLAGS_REDRAW_STATUS 0x08

W - Redraw entire status line

5.82.2.7 #define UI_FLAGS_RESET_SLEEP_TIMER 0x10

W - Reset sleep timeout timer

5.82.2.8 #define UI_FLAGS_UPDATE 0x01

W - Make changes take effect

5.83 UIState constants

Constants for use with the UIState() function.

Defines

- #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

5.83.1 Detailed Description

Constants for use with the `UIState()` function.

See also:

`UIState()`

5.83.2 Define Documentation

5.83.2.1 #define `UI_STATE_BT_ERROR` 16

R - BT error

5.83.2.2 #define `UI_STATE_CONNECT_REQUEST` 12

RW - Request for connection accept

5.83.2.3 #define `UI_STATE_DRAW_MENU` 6

RW - Execute function and draw menu icons

5.83.2.4 #define `UI_STATE_ENTER_PRESSED` 10

RW - Load selected function and next menu id

5.83.2.5 #define `UI_STATE_EXECUTE_FILE` 13

RW - Execute file in "LMSfilename"

5.83.2.6 #define `UI_STATE_EXECUTING_FILE` 14

R - Executing file in "LMSfilename"

5.83.2.7 #define `UI_STATE_EXIT_PRESSED` 11

RW - Load selected function and next menu id

5.83.2.8 #define UI_STATE_INIT_DISPLAY 0

RW - Init display and load font, menu etc.

5.83.2.9 #define UI_STATE_INIT_INTRO 2

R - Display intro

5.83.2.10 #define UI_STATE_INIT_LOW_BATTERY 1

R - Low battery voltage at power on

5.83.2.11 #define UI_STATE_INIT_MENU 4

RW - Init menu system

5.83.2.12 #define UI_STATE_INIT_WAIT 3

RW - Wait for initialization end

5.83.2.13 #define UI_STATE_LEFT_PRESSED 8

RW - Load selected function and next menu id

5.83.2.14 #define UI_STATE_LOW_BATTERY 15

R - Low battery at runtime

5.83.2.15 #define UI_STATE_NEXT_MENU 5

RW - Next menu icons ready for drawing

5.83.2.16 #define UI_STATE_RIGHT_PRESSED 9

RW - Load selected function and next menu id

5.83.2.17 #define UI_STATE_TEST_BUTTONS 7

RW - Wait for buttons to be pressed

5.84 UIButton constants

Constants for use with the UIButton() function.

Defines

- #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

5.84.1 Detailed Description

Constants for use with the UIButton() function.

See also:

[UIButton\(\)](#)

5.84.2 Define Documentation

5.84.2.1 #define `UI_BUTTON_ENTER` 2

W - Insert enter button

5.84.2.2 #define `UI_BUTTON_EXIT` 4

W - Insert exit button

5.84.2.3 #define `UI_BUTTON_LEFT` 1

W - Insert left arrow button

5.84.2.4 #define `UI_BUTTON_NONE` 0

R - Button inserted are executed

5.84.2.5 #define `UI_BUTTON_RIGHT` 3

W - Insert right arrow button

5.85 BluetoothState constants

Constants for use with the BluetoothState() function.

Defines

- #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

5.85.1 Detailed Description

Constants for use with the BluetoothState() function.

See also:

`BluetoothState()`

5.85.2 Define Documentation

5.85.2.1 #define `UI_BT_CONNECT_REQUEST` 0x40

RW - BT get connect accept in progress

5.85.2.2 #define `UI_BT_ERROR_ATTENTION` 0x08

W - BT error attention

5.85.2.3 #define `UI_BT_PIN_REQUEST` 0x80

RW - BT get pin code

5.85.2.4 #define `UI_BT_STATE_CONNECTED` 0x02

RW - BT connected to something

5.85.2.5 #define `UI_BT_STATE_OFF` 0x04

RW - BT power off

5.85.2.6 #define UI_BT_STATE_VISIBLE 0x01

RW - BT visible

5.86 VM run state constants

Constants for use with the VMRunState() function.

Defines

- #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

5.86.1 Detailed Description

Constants for use with the VMRunState() function.

See also:[VMRunState\(\)](#)**5.86.2 Define Documentation****5.86.2.1 #define UI_VM_IDLE 0**

VM_IDLE: Just sitting around. Request to run program will lead to ONE of the VM_RUN* states.

5.86.2.2 #define UI_VM_RESET1 4

VM_RESET1: Initialize state variables and some I/O devices -- executed when programs end

5.86.2.3 #define UI_VM_RESET2 5

VM_RESET2: Final clean up and return to IDLE

5.86.2.4 #define UI_VM_RUN_FREE 1

VM_RUN_FREE: Attempt to run as many instructions as possible within our timeslice

5.86.2.5 #define UI_VM_RUN_PAUSE 3

VM_RUN_PAUSE: Program still "active", but someone has asked us to pause

5.86.2.6 #define UI_VM_RUN_SINGLE 2

VM_RUN_SINGLE: Run exactly one instruction per timeslice

5.87 Ui module IOMAP offsets

Constant offsets into the Ui module IOMAP structure.

Defines

- #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

5.87.1 Detailed Description

Constant offsets into the Ui module IOMAP structure.

5.87.2 Define Documentation**5.87.2.1 #define UIOffsetAbortFlag 40**

RW - Long Abort (true == use long press to abort) (1 byte)

5.87.2.2 #define UIOffsetBatteryState 30

W - Battery state (0..4 capacity) (1 byte)

5.87.2.3 #define UIOffsetBatteryVoltage 4

R - Battery voltage in millivolts (2 bytes)

5.87.2.4 #define UIOffsetBluetoothState 31

W - Bluetooth state (0=on, 1=visible, 2=conn, 3=conn.visible, 4=off, 5=dfu) (1 byte)

5.87.2.5 #define UIOffsetButton 28

RW - Insert button (buttons enumerated above) (1 byte)

5.87.2.6 #define UIOffsetError 37

W - Error code (1 byte)

5.87.2.7 #define UIOffsetFlags 26

RW - Update command flags (flags enumerated above) (1 byte)

5.87.2.8 #define UIOffsetForceOff 39

W - Force off (> 0 = off) (1 byte)

5.87.2.9 #define UIOffsetLMSfilename 6

W - LMS filename to execute (Try It) (20 bytes)

5.87.2.10 #define UIOffsetOBPPointer 38

W - Actual OBP step (0 - 4) (1 byte)

5.87.2.11 #define UIOffsetPMenu 0

W - Pointer to menu file (4 bytes)

5.87.2.12 #define UIOffsetRechargeable 35

R - Rechargeable battery (0 = no, 1 = yes) (1 byte)

5.87.2.13 #define UIOffsetRunState 29

W - VM Run state (0 = stopped, 1 = running) (1 byte)

5.87.2.14 #define UIOffsetSleepTimeout 33

RW - Sleep timeout time (min) (1 byte)

5.87.2.15 #define UIOffsetSleepTimer 34

RW - Sleep timer (min) (1 byte)

5.87.2.16 #define UIOffsetState 27

RW - UI state (states enumerated above) (1 byte)

5.87.2.17 #define UIOffsetUsbState 32

W - Usb state (0=disconnected, 1=connected, 2=working) (1 byte)

5.87.2.18 #define UIOffsetVolume 36

RW - Volume used in UI (0 - 4) (1 byte)

5.88 NBC Input port constants

Input port constants are used when calling sensor control API functions.

Defines

- #define IN_1 0x00
- #define IN_2 0x01
- #define IN_3 0x02
- #define IN_4 0x03

5.88.1 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

5.88.2 Define Documentation**5.88.2.1 #define IN_1 0x00**

Input port 1

5.88.2.2 #define IN_2 0x01

Input port 2

5.88.2.3 #define IN_3 0x02

Input port 3

5.88.2.4 #define IN_4 0x03

Input port 4

5.89 NBC sensor type constants

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

Defines

- #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

5.89.1 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\(\)](#)

5.89.2 Define Documentation**5.89.2.1 #define IN_TYPE_ANGLE 0x04**

RCX rotation sensor

5.89.2.2 #define IN_TYPE_COLORBLUE 0x10

NXT 2.0 color sensor with blue light

5.89.2.3 #define IN_TYPE_COLOREXIT 0x12

NXT 2.0 color sensor internal state

5.89.2.4 #define IN_TYPE_COLORFULL 0x0D

NXT 2.0 color sensor in full color mode

5.89.2.5 #define IN_TYPE_COLORGREEN 0x0F

NXT 2.0 color sensor with green light

5.89.2.6 #define IN_TYPE_COLORNONE 0x11

NXT 2.0 color sensor with no light

5.89.2.7 #define IN_TYPE_COLORRED 0x0E

NXT 2.0 color sensor with red light

5.89.2.8 #define IN_TYPE_CUSTOM 0x09

NXT custom sensor

5.89.2.9 #define IN_TYPE_HISPEED 0x0C

NXT Hi-speed port (only S4)

5.89.2.10 #define IN_TYPE_LIGHT_ACTIVE 0x05

NXT light sensor with light

5.89.2.11 #define IN_TYPE_LIGHT_INACTIVE 0x06

NXT light sensor without light

5.89.2.12 #define IN_TYPE_LOWSPEED 0x0A

NXT I2C digital sensor

5.89.2.13 #define IN_TYPE_LOWSPEED_9V 0x0B

NXT I2C digital sensor with 9V power

5.89.2.14 #define IN_TYPE_NO_SENSOR 0x00

No sensor configured

5.89.2.15 #define IN_TYPE_REFLECTION 0x03

RCX light sensor

5.89.2.16 #define IN_TYPE_SOUND_DB 0x07

NXT sound sensor with dB scaling

5.89.2.17 #define IN_TYPE_SOUND_DB_A 0x08

NXT sound sensor with dBA scaling

5.89.2.18 #define IN_TYPE_SWITCH 0x01

NXT or RCX touch sensor

5.89.2.19 #define IN_TYPE_TEMPERATURE 0x02

RCX temperature sensor

5.90 NBC sensor mode constants

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

Defines

- #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

5.90.1 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\(\)](#)

5.90.2 Define Documentation

5.90.2.1 #define IN_MODE_ANGLESTEP 0xE0

RCX rotation sensor (16 ticks per revolution)

5.90.2.2 #define IN_MODE_BOOLEAN 0x20

Boolean value (0 or 1)

5.90.2.3 #define IN_MODE_CELSIUS 0xA0

RCX temperature sensor value in degrees celcius

5.90.2.4 #define IN_MODE_FAHRENHEIT 0xC0

RCX temperature sensor value in degrees fahrenheit

5.90.2.5 #define IN_MODE_MODEMASK 0xE0

Mask for the mode without any slope value

5.90.2.6 #define IN_MODE_PCTFULLSCALE 0x80

Scaled value from 0 to 100

5.90.2.7 #define IN_MODE_PERIODCOUNTER 0x60

Counts the number of boolean periods

5.90.2.8 #define IN_MODE_RAW 0x00

Raw value from 0 to 1023

5.90.2.9 #define IN_MODE_SLOPEMASK 0x1F

Mask for slope parameter added to mode

5.90.2.10 #define IN_MODE_TRANSITIONCNT 0x40

Counts the number of boolean transitions

5.91 Input field constants

Constants for use with SetInput() and GetInput().

Defines

- #define TypeField 0
- #define InputModeField 1
- #define RawValueField 2
- #define NormalizedValueField 3
- #define ScaledValueField 4
- #define InvalidDataField 5

5.91.1 Detailed Description

Constants for use with SetInput() and GetInput(). Each sensor has six fields that are used to define its state.

5.91.2 Define Documentation**5.91.2.1 #define InputModeField 1**

Input mode field. Contains one of the sensor mode constants. Read/write.

5.91.2.2 #define InvalidDataField 5

Invalid data field. Contains a boolean value indicating whether the sensor data is valid or not. Read/write.

5.91.2.3 #define NormalizedValueField 3

Normalized value field. Contains the current normalized analog sensor value. Read only.

5.91.2.4 #define RawValueField 2

Raw value field. Contains the current raw analog sensor value. Read only.

5.91.2.5 #define ScaledValueField 4

Scaled value field. Contains the current scaled analog sensor value. Read/write.

5.91.2.6 #define TypeField 0

Type field. Contains one of the sensor type constants. Read/write.

5.92 Input port digital pin constants

Constants for use when directly controlling or reading a port's digital pin state.

Defines

- #define INPUT_DIGI0 0x01
- #define INPUT_DIGI1 0x02

5.92.1 Detailed Description

Constants for use when directly controlling or reading a port's digital pin state.

5.92.2 Define Documentation**5.92.2.1 #define INPUT_DIGI0 0x01**

Digital pin 0

5.92.2.2 #define INPUT_DIGI1 0x02

Digital pin 1

5.93 Color sensor array indices

Constants for use with color sensor value arrays to index RGB and blank return values.

Defines

- #define INPUT_RED 0
- #define INPUT_GREEN 1
- #define INPUT_BLUE 2
- #define INPUT_BLANK 3
- #define INPUT_NO_OF_COLORS 4

5.93.1 Detailed Description

Constants for use with color sensor value arrays to index RGB and blank return values.

See also:

[ReadSensorColorEx\(\)](#), [ReadSensorColorRaw\(\)](#), [SysColorSensorRead\(\)](#), [ColorSensorReadType](#)

5.93.2 Define Documentation

5.93.2.1 #define INPUT_BLANK 3

Access the blank value from color sensor value arrays

5.93.2.2 #define INPUT_BLUE 2

Access the blue value from color sensor value arrays

5.93.2.3 #define INPUT_GREEN 1

Access the green value from color sensor value arrays

5.93.2.4 #define INPUT_NO_OF_COLORS 4

The number of entries in the color sensor value arrays

5.93.2.5 #define INPUT_RED 0

Access the red value from color sensor value arrays

5.94 Color values

Constants for use with the ColorValue returned by the color sensor in full color mode.

Defines

- #define INPUT_BLACKCOLOR 1
- #define INPUT_BLUECOLOR 2
- #define INPUT_GREENCOLOR 3
- #define INPUT_YELLOWCOLOR 4
- #define INPUT_REDCOLOR 5
- #define INPUT_WHITECOLOR 6

5.94.1 Detailed Description

Constants for use with the ColorValue returned by the color sensor in full color mode.

See also:

SensorValue(), SysColorSensorRead(), ColorSensorReadType

5.94.2 Define Documentation

5.94.2.1 #define INPUT_BLACKCOLOR 1

The color value is black

5.94.2.2 #define INPUT_BLUECOLOR 2

The color value is blue

5.94.2.3 #define INPUT_GREENCOLOR 3

The color value is green

5.94.2.4 #define INPUT_REDCOLOR 5

The color value is red

5.94.2.5 #define INPUT_WHITECOLOR 6

The color value is white

5.94.2.6 #define INPUT_YELLOWCOLOR 4

The color value is yellow

5.95 Color calibration state constants

Constants for use with the color calibration state function.

Defines

- #define INPUT_SENSORCAL 0x01
- #define INPUT_SENSOROFF 0x02
- #define INPUT_RUNNINGCAL 0x20
- #define INPUT_STARTCAL 0x40
- #define INPUT_RESETCAL 0x80

5.95.1 Detailed Description

Constants for use with the color calibration state function.

See also:

ColorCalibrationState()

5.95.2 Define Documentation**5.95.2.1 #define INPUT_RESETCAL 0x80**

Unused calibration state constant

5.95.2.2 #define INPUT_RUNNINGCAL 0x20

Unused calibration state constant

5.95.2.3 #define INPUT_SENSORCAL 0x01

The state returned while the color sensor is calibrating

5.95.2.4 #define INPUT_SENSOROFF 0x02

The state returned once calibration has completed

5.95.2.5 #define INPUT_STARTCAL 0x40

Unused calibration state constant

5.96 Color calibration constants

Constants for use with the color calibration functions.

Defines

- #define INPUT_CAL_POINT_0 0
- #define INPUT_CAL_POINT_1 1
- #define INPUT_CAL_POINT_2 2
- #define INPUT_NO_OF_POINTS 3

5.96.1 Detailed Description

Constants for use with the color calibration functions.

See also:

ColorCalibration(), ColorCalLimits()

5.96.2 Define Documentation**5.96.2.1 #define INPUT_CAL_POINT_0 0**

Calibration point 0

5.96.2.2 #define INPUT_CAL_POINT_1 1

Calibration point 1

5.96.2.3 #define INPUT_CAL_POINT_2 2

Calibration point 2

5.96.2.4 #define INPUT_NO_OF_POINTS 3

The number of calibration points

5.97 Input module IOMAP offsets

Constant offsets into the Input module IOMAP structure.

Defines

- #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)`

5.97.1 Detailed Description

Constant offsets into the Input module IOMAP structure.

5.97.2 Define Documentation

5.97.2.1 #define InputOffsetADRaw(p) (((p)*20)+2)

Read the AD raw sensor value (2 bytes) uword

5.97.2.2 #define InputOffsetColorADRaw(p, nc) (80+((p)*84)+52+((nc)*2))

Read AD raw color sensor values

5.97.2.3 #define InputOffsetColorBoolean(p, nc) (80+((p)*84)+76+((nc)*2))

Read color sensor boolean values

5.97.2.4 #define InputOffsetColorCalibration(p, np, nc) (80+((p)*84)+0+((np)*16)+((nc)*4))

Read/write color calibration point values

5.97.2.5 #define InputOffsetColorCalibrationState(p) (80+((p)*84)+80)

Read color sensor calibration state

5.97.2.6 #define InputOffsetColorCalLimits(p, np) (80+((p)*84)+48+((np)*2))

Read/write color calibration limits

5.97.2.7 #define InputOffsetColorSensorRaw(p, nc) (80+((p)*84)+60+((nc)*2))

Read raw color sensor values

5.97.2.8 #define InputOffsetColorSensorValue(p, nc) (80+((p)*84)+68+((nc)*2))

Read scaled color sensor values

5.97.2.9 #define InputOffsetCustomActiveStatus(p) (((p)*20)+15)

Read/write the active or inactive state of the custom sensor

5.97.2.10 #define InputOffsetCustomPctFullScale(p) (((p)*20)+14)

Read/write the Pct full scale of the custom sensor

5.97.2.11 #define InputOffsetCustomZeroOffset(p) (((p)*20)+0)

Read/write the zero offset of a custom sensor (2 bytes) uword

5.97.2.12 #define InputOffsetDigiPinsDir(p) (((p)*20)+11)

Read/write the direction of the Digital pins (1 is output, 0 is input)

5.97.2.13 #define InputOffsetDigiPinsIn(p) (((p)*20)+12)

Read/write the status of the digital pins

5.97.2.14 #define InputOffsetDigiPinsOut(p) (((p)*20)+13)

Read/write the output level of the digital pins

5.97.2.15 #define InputOffsetInvalidData(p) (((p)*20)+16)

Indicates whether data is invalid (1) or valid (0)

5.97.2.16 #define InputOffsetSensorBoolean(p) (((p)*20)+10)

Read the sensor boolean value

5.97.2.17 #define InputOffsetSensorMode(p) (((p)*20)+9)

Read/write the sensor mode

5.97.2.18 #define InputOffsetSensorRaw(p) (((p)*20)+4)

Read the raw sensor value (2 bytes) uword

5.97.2.19 #define InputOffsetSensorType(p) (((p)*20)+8)

Read/write the sensor type

5.97.2.20 #define InputOffsetSensorValue(p) (((p)*20)+6)

Read/write the scaled sensor value (2 bytes) sword

5.98 Constants to use with the Input module's Pin function

Constants for use with the Input module's Pin function.

Defines

- #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

5.98.1 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.

5.98.2 Define Documentation**5.98.2.1 #define INPUT_PINCMD_CLEAR 0x02**

Clear digital pin(s)

5.98.2.2 #define INPUT_PINCMD_DIR 0x00

Set digital pin(s) direction

5.98.2.3 #define INPUT_PINCMD_MASK 0x03

Mask for the two bits used by pin function commands

5.98.2.4 #define INPUT_PINCMD_READ 0x03

Read digital pin(s)

5.98.2.5 #define INPUT_PINCMD_SET 0x01

Set digital pin(s)

5.98.2.6 #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

5.98.2.7 #define INPUT_PINDIR_INPUT 0x04

Use with the direction command to set direction to output. OR this with the pin value.

5.98.2.8 #define INPUT_PINDIR_OUTPUT 0x00

Use with the direction command to set direction to input. OR this with the pin value.

5.99 Output port constants

Output port constants are used when calling motor control API functions.

Defines

- #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

5.99.1 Detailed Description

Output port constants are used when calling motor control API functions.

5.99.2 Define Documentation

5.99.2.1 #define OUT_A 0x00

Output port A

5.99.2.2 #define OUT_AB 0x03

Output ports A and B

5.99.2.3 #define OUT_ABC 0x06

Output ports A, B, and C

5.99.2.4 #define OUT_AC 0x04

Output ports A and C

5.99.2.5 #define OUT_B 0x01

Output port B

5.99.2.6 #define OUT_BC 0x05

Output ports B and C

5.99.2.7 #define OUT_C 0x02

Output port C

5.100 PID constants

PID constants are for adjusting the Proportional, Integral, and Derivative motor controller parameters.

Defines

- #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

5.100.1 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\(\)](#)

5.100.2 Define Documentation**5.100.2.1 #define PID_0 0**

PID zero

5.100.2.2 #define PID_1 32

PID one

5.100.2.3 #define PID_2 64

PID two

5.100.2.4 #define PID_3 96

PID three

5.100.2.5 #define PID_4 128

PID four

5.100.2.6 #define PID_5 160

PID five

5.100.2.7 #define PID_6 192

PID six

5.100.2.8 #define PID_7 224

PID seven

5.101 Output port update flag constants

Use these constants to specify which motor values need to be updated.

Defines

- #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

5.101.1 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\(\)](#)

5.101.2 Define Documentation

5.101.2.1 #define **UF_PENDING_UPDATES** 0x80

Are there any pending motor updates?

5.101.2.2 #define **UF_UPDATE_MODE** 0x01

Commits changes to the [OutputModeField](#) output property

5.101.2.3 #define **UF_UPDATE_PID_VALUES** 0x10

Commits changes to the PID motor regulation properties

5.101.2.4 #define **UF_UPDATE_RESET_BLOCK_COUNT** 0x20

Resets the NXT-G block-relative rotation counter

5.101.2.5 #define UF_UPDATE_RESET_COUNT 0x08

Resets all rotation counters, cancels the current goal, and resets the rotation error-correction system

5.101.2.6 #define UF_UPDATE_RESET_ROTATION_COUNT 0x40

Resets the program-relative (user) rotation counter

5.101.2.7 #define UF_UPDATE_SPEED 0x02

Commits changes to the [PowerField](#) output property

5.101.2.8 #define UF_UPDATE_TACHO_LIMIT 0x04

Commits changes to the [TachoLimitField](#) output property

5.102 Tachometer counter reset flags

Use these constants to specify which of the three tachometer counters should be reset.

Defines

- #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

5.102.1 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...

5.102.2 Define Documentation**5.102.2.1 #define RESET_ALL 0x68**

Reset all three tachometer counters

5.102.2.2 #define RESET_BLOCK_COUNT 0x20

Reset the NXT-G block tachometer counter

5.102.2.3 #define RESET_BLOCKANDTACHO 0x28

Reset both the internal counter and the NXT-G block counter

5.102.2.4 #define RESET_COUNT 0x08

Reset the internal tachometer counter

5.102.2.5 #define RESET_NONE 0x00

No counters will be reset

5.102.2.6 #define RESET_ROTATION_COUNT 0x40

Reset the rotation counter

5.103 Output port mode constants

Use these constants to configure the desired mode for the specified motor(s): coast, motoron, brake, or regulated.

Defines

- #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

5.103.1 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()`

5.103.2 Define Documentation

5.103.2.1 `#define OUT_MODE_BRAKE 0x02`

Uses electronic braking to outputs

5.103.2.2 `#define OUT_MODE_COAST 0x00`

No power and no braking so motors rotate freely.

5.103.2.3 `#define OUT_MODE_MOTORON 0x01`

Enables PWM power to the outputs given the power setting

5.103.2.4 `#define OUT_MODE_REGMETHOD 0xF0`

Mask for unimplemented regulation mode

5.103.2.5 `#define OUT_MODE_REGULATED 0x04`

Enables active power regulation using the regulation mode value

5.104 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.

Defines

- `#define OUT_OPTION_HOLDATLIMIT 0x10`
- `#define OUT_OPTION_RAMPDOWNTOLIMIT 0x20`

5.104.1 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.

See also:

[SetOutput\(\)](#)

5.104.2 Define Documentation

5.104.2.1 **#define OUT_OPTION_HOLDATLIMIT 0x10**

Option to have the firmware hold the motor when it reaches the tachometer limit

5.104.2.2 **#define OUT_OPTION_RAMPDOWNTOLIMIT 0x20**

Option to have the firmware rampdown the motor power as it approaches the tachometer limit

5.105 Output regulation option constants

Use these constants to configure the desired options for position regulation.

Defines

- [#define OUT_REGOPTION_NO_SATURATION 0x01](#)

5.105.1 Detailed Description

Use these constants to configure the desired options for position regulation.

5.105.2 Define Documentation

5.105.2.1 **#define OUT_REGOPTION_NO_SATURATION 0x01**

Do not limit intermediary regulation results

5.106 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.

Defines

- #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

5.106.1 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\(\)](#)

5.106.2 Define Documentation**5.106.2.1 #define OUT_RUNSTATE_HOLD 0x60**

Set motor run state to hold at the current position.

5.106.2.2 #define OUT_RUNSTATE_IDLE 0x00

Disable all power to motors.

5.106.2.3 #define OUT_RUNSTATE_RAMPDOWN 0x40

Enable ramping down from a current power to a new (lower) power over a specified [TachoLimitField](#) goal.

5.106.2.4 #define OUT_RUNSTATE_RAMPUP 0x10

Enable ramping up from a current power to a new (higher) power over a specified [TachoLimitField](#) goal.

5.106.2.5 #define OUT_RUNSTATE_RUNNING 0x20

Enable power to motors at the specified power level.

5.107 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+).

Defines

- #define OUT_REGMODE_IDLE 0
- #define OUT_REGMODE_SPEED 1
- #define OUT_REGMODE_SYNC 2
- #define OUT_REGMODE_POS 4

5.107.1 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\(\)](#)

5.107.2 Define Documentation

5.107.2.1 #define OUT_REGMODE_IDLE 0

No motor regulation.

5.107.2.2 #define OUT_REGMODE_POS 4

Regulate a motor's position.

5.107.2.3 #define OUT_REGMODE_SPEED 1

Regulate a motor's speed (aka power).

5.107.2.4 #define OUT_REGMODE_SYNC 2

Synchronize the rotation of two motors.

5.108 Output field constants

Constants for use with SetOutput() and GetOutput().

Defines

- #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.

5.108.1 Detailed Description

Constants for use with SetOutput() and GetOutput().

See also:

[SetOutput\(\)](#), [GetOutput\(\)](#)

5.108.2 Define Documentation

5.108.2.1 #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.

5.108.2.2 #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.

5.108.2.3 #define MaxAccelerationField 17

MaxAcceleration field. Contains the current max acceleration value. Read/write. Set the maximum acceleration to be used during position regulation.

5.108.2.4 #define MaxSpeedField 16

MaxSpeed field. Contains the current max speed value. Read/write. Set the maximum speed to be used during position regulation.

5.108.2.5 #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.

5.108.2.6 #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.

5.108.2.7 #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](#).

5.108.2.8 #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.

5.108.2.9 #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.

5.108.2.10 #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.

5.108.2.11 #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.

5.108.2.12 #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.

5.108.2.13 #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.

5.108.2.14 #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 en-

able 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](#).

5.108.2.15 #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.

5.108.2.16 #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.

5.108.2.17 #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.

5.108.2.18 #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.

5.109 Output module IOMAP offsets

Constant offsets into the Output module IOMAP structure.

Defines

- #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](#)

5.109.1 Detailed Description

Constant offsets into the Output module IOMAP structure.

5.109.2 Define Documentation

5.109.2.1 #define OutputOffsetActualSpeed(p) (((p)*32)+21)

R - Holds the current motor speed (1 byte) sbyte

5.109.2.2 #define OutputOffsetBlockTachoCount(p) (((p)*32)+4)

R - Holds current number of counts for the current output block (4 bytes) slong

5.109.2.3 #define OutputOffsetFlags(p) (((p)*32)+18)

RW - Holds flags for which data should be updated (1 byte) ubyte

5.109.2.4 #define OutputOffsetMaxAccel(p) (((p)*32)+31)

RW - holds the maximum acceleration for position regulation (1 byte) sbyte (NBC/NXC)

5.109.2.5 #define OutputOffsetMaxSpeed(p) (((p)*32)+30)

RW - holds the maximum speed for position regulation (1 byte) sbyte (NBC/NXC)

5.109.2.6 #define OutputOffsetMode(p) (((p)*32)+19)

RW - Holds motor mode: Run, Break, regulated, ... (1 byte) ubyte

5.109.2.7 #define OutputOffsetMotorRPM(p) (((p)*32)+16)

Not updated, will be removed later !! (2 bytes) sword

5.109.2.8 #define OutputOffsetOptions(p) (((p)*32)+29)

RW - holds extra motor options related to the tachometer limit (1 byte) ubyte (NBC/NXC)

5.109.2.9 #define OutputOffsetOverloaded(p) (((p)*32)+27)

R - True if the motor has been overloaded within speed control regulation (1 byte) ubyte

5.109.2.10 #define OutputOffsetRegDParameter(p) (((p)*32)+24)

RW - Holds the D-constant used in the regulation (1 byte) ubyte

5.109.2.11 #define OutputOffsetRegIParameter(p) (((p)*32)+23)

RW - Holds the I-constant used in the regulation (1 byte) ubyte

5.109.2.12 #define OutputOffsetRegMode(p) (((p)*32)+26)

RW - Tells which regulation mode should be used (1 byte) ubyte

5.109.2.13 #define OutputOffsetRegPParameter(p) (((p)*32)+22)

RW - Holds the P-constant used in the regulation (1 byte) ubyte

5.109.2.14 #define OutputOffsetRegulationOptions 97

use for position regulation options (1 byte) ubyte (NBC/NXC)

5.109.2.15 #define OutputOffsetRegulationTime 96

use for frequency of checking regulation mode (1 byte) ubyte (NBC/NXC)

5.109.2.16 #define OutputOffsetRotationCount(p) (((p)*32)+8)

R - Holds current number of counts for the rotation counter to the output (4 bytes)
slong

5.109.2.17 #define OutputOffsetRunState(p) (((p)*32)+25)

RW - Holds the current motor run state in the output module (1 byte) ubyte

5.109.2.18 #define OutputOffsetSpeed(p) (((p)*32)+20)

RW - Holds the wanted speed (1 byte) sbyte

5.109.2.19 #define OutputOffsetSyncTurnParameter(p) (((p)*32)+28)

RW - Holds the turning parameter need within MoveBlock (1 byte) sbyte

5.109.2.20 #define OutputOffsetTachoCount(p) (((p)*32)+0)

R - Holds current number of counts, since last reset, updated every 1 mS (4 bytes)
slong

5.109.2.21 #define OutputOffsetTachoLimit(p) (((p)*32)+12)

RW - Holds number of counts to travel, 0 => Run forever (4 bytes) ulong

5.110 LowSpeed module constants

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

Modules

- [LSState constants](#)

Constants for the low speed module LSState function.

- [LSCurrentState constants](#)

Constants for the low speed module LSCurrentState function.

- [LSMode constants](#)

Constants for the low speed module LSMode function.

- [LSErrorType constants](#)

Constants for the low speed module LSErrorType function.

- [Low speed module IOMAP offsets](#)

Constant offsets into the low speed module IOMAP structure.

- [LSNoRestartOnRead constants](#)

Constants for the low speed module LSNoRestartOnRead and SetLSNoRestartOnRead functions.

- Standard I2C constants

Constants for use with standard I2C devices.

- LEGO I2C address constants

Constants for LEGO I2C device addresses.

- Ultrasonic sensor constants

Constants for use with the ultrasonic sensor.

- LEGO temperature sensor constants

Constants for use with the LEGO temperature sensor.

- E-Meter sensor constants

Constants for use with the e-meter sensor.

- I2C option constants

Constants for the SetI2COPTIONS function.

5.110.1 Detailed Description

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

5.111 LSState constants

Constants for the low speed module LSState function.

Defines

- #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

5.111.1 Detailed Description

Constants for the low speed module LSState function. These values are combined together using a bitwise OR operation.

See also:

LSState()

5.111.2 Define Documentation**5.111.2.1 #define COM_CHANNEL_FOUR_ACTIVE 0x08**

Low speed channel 4 is active

5.111.2.2 #define COM_CHANNEL_NONE_ACTIVE 0x00

None of the low speed channels are active

5.111.2.3 #define COM_CHANNEL_ONE_ACTIVE 0x01

Low speed channel 1 is active

5.111.2.4 #define COM_CHANNEL_THREE_ACTIVE 0x04

Low speed channel 3 is active

5.111.2.5 #define COM_CHANNEL_TWO_ACTIVE 0x02

Low speed channel 2 is active

5.112 LSChannelState constants

Constants for the low speed module LSChannelState function.

Defines

- #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

5.112.1 Detailed Description

Constants for the low speed module LSChannelState function.

See also:

LSChannelState()

5.112.2 Define Documentation**5.112.2.1 #define LOWSPEED_COMMUNICATING 3**

Channel is actively communicating

5.112.2.2 #define LOWSPEED_DONE 5

Channel is done communicating

5.112.2.3 #define LOWSPEED_ERROR 4

Channel is in an error state

5.112.2.4 #define LOWSPEED_IDLE 0

Channel is idle

5.112.2.5 #define LOWSPEED_INIT 1

Channel is being initialized

5.112.2.6 #define LOWSPEED_LOAD_BUFFER 2

Channel buffer is loading

5.113 LSMode constants

Constants for the low speed module LSMode function.

Defines

- #define **LOWSPEED_TRANSMITTING** 1
- #define **LOWSPEED RECEIVING** 2
- #define **LOWSPEED_DATA_RECEIVED** 3

5.113.1 Detailed Description

Constants for the low speed module LSMode function.

See also:

LSMode()

5.113.2 Define Documentation**5.113.2.1 #define LOWSPEED_DATA_RECEIVED 3**

Lowspeed port is in data received mode

5.113.2.2 #define LOWSPEED RECEIVING 2

Lowspeed port is in receiving mode

5.113.2.3 #define LOWSPEED_TRANSMITTING 1

Lowspeed port is in transmitting mode

5.114 LSErrorType constants

Constants for the low speed module LSErrorType function.

Defines

- #define **LOWSPEED_NO_ERROR** 0
- #define **LOWSPEED_CH_NOT_READY** 1
- #define **LOWSPEED_TX_ERROR** 2
- #define **LOWSPEED_RX_ERROR** 3

5.114.1 Detailed Description

Constants for the low speed module LSErrorType function.

See also:

LSErrorType()

5.114.2 Define Documentation

5.114.2.1 #define LOWSPEED_CH_NOT_READY 1

Lowspeed port is not ready

5.114.2.2 #define LOWSPEED_NO_ERROR 0

Lowspeed port has no error

5.114.2.3 #define LOWSPEED_RX_ERROR 3

Lowspeed port encountered an error while receiving data

5.114.2.4 #define LOWSPEED_TX_ERROR 2

Lowspeed port encountered an error while transmitting data

5.115 Low speed module IOMAP offsets

Constant offsets into the low speed module IOMAP structure.

Defines

- #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`

5.115.1 Detailed Description

Constant offsets into the low speed module IOMAP structure.

5.115.2 Define Documentation

5.115.2.1 #define `LowSpeedOffsetChannelState(p) ((p)+156)`

R - Lowspeed channel state (1 byte)

5.115.2.2 #define `LowSpeedOffsetErrorType(p) ((p)+160)`

R - Lowspeed port error type (1 byte)

5.115.2.3 #define `LowSpeedOffsetInBufBuf(p) (((p)*19)+0)`

RW - Input buffer data buffer field offset (16 bytes)

5.115.2.4 #define `LowSpeedOffsetInBufBytesToRx(p) (((p)*19)+18)`

RW - Input buffer bytes to receive field offset (1 byte)

5.115.2.5 #define `LowSpeedOffsetInBufInPtr(p) (((p)*19)+16)`

RW - Input buffer in pointer field offset (1 byte)

5.115.2.6 #define `LowSpeedOffsetInBufOutPtr(p) (((p)*19)+17)`

RW - Input buffer out pointer field offset (1 byte)

5.115.2.7 #define `LowSpeedOffsetMode(p) ((p)+152)`

R - Lowspeed port mode (1 byte)

5.115.2.8 #define LowSpeedOffsetNoRestartOnRead 166

RW - Lowspeed option for no restart on read (all channels) (NBC/NXC)

5.115.2.9 #define LowSpeedOffsetOutBufBuf(p) (((p)*19)+76)

RW - Output buffer data buffer field offset (16 bytes)

5.115.2.10 #define LowSpeedOffsetOutBufBytesToRx(p) (((p)*19)+94)

RW - Output buffer bytes to receive field offset (1 byte)

5.115.2.11 #define LowSpeedOffsetOutBufInPtr(p) (((p)*19)+92)

RW - Output buffer in pointer field offset (1 byte)

5.115.2.12 #define LowSpeedOffsetOutBufOutPtr(p) (((p)*19)+93)

RW - Output buffer out pointer field offset (1 byte)

5.115.2.13 #define LowSpeedOffsetSpeed 165

R - Lowspeed speed (unused)

5.115.2.14 #define LowSpeedOffsetState 164

R - Lowspeed state (all channels)

5.116 LSNoRestartOnRead constants

Constants for the low speed module LSNoRestartOnRead and SetLSNoRestartOnRead functions.

Defines

- #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

5.116.1 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()

5.116.2 Define Documentation

5.116.2.1 #define LSREAD_NO_RESTART_1 0x01

No restart on read for channel 1

5.116.2.2 #define LSREAD_NO_RESTART_2 0x02

No restart on read for channel 2

5.116.2.3 #define LSREAD_NO_RESTART_3 0x04

No restart on read for channel 3

5.116.2.4 #define LSREAD_NO_RESTART_4 0x08

No restart on read for channel 4

5.116.2.5 #define LSREAD_NO_RESTART_MASK 0x10

No restart mask

5.116.2.6 #define LSREAD_RESTART_ALL 0x00

Restart on read for all channels (default)

5.116.2.7 #define LSREAD_RESTART_NONE 0x0F

No restart on read for all channels

5.117 Standard I2C constants

Constants for use with standard I2C devices.

Defines

- #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

5.117.1 Detailed Description

Constants for use with standard I2C devices.

5.117.2 Define Documentation

5.117.2.1 #define I2C_ADDR_DEFAULT 0x02

Standard NXT I2C device address

5.117.2.2 #define I2C_REG_CMD 0x41

Standard NXT I2C device command register

5.117.2.3 #define I2C_REG_DEVICE_ID 0x10

Standard NXT I2C device ID register

5.117.2.4 #define I2C_REG_VENDOR_ID 0x08

Standard NXT I2C vendor ID register

5.117.2.5 #define I2C_REG_VERSION 0x00

Standard NXT I2C version register

5.118 LEGO I2C address constants

Constants for LEGO I2C device addresses.

Defines

- #define **LEGO_ADDR_US** 0x02
- #define **LEGO_ADDR_TEMP** 0x98
- #define **LEGO_ADDR_EMETER** 0x04

5.118.1 Detailed Description

Constants for LEGO I2C device addresses.

5.118.2 Define Documentation

5.118.2.1 #define LEGO_ADDR_EMETER 0x04

The LEGO e-meter sensor's I2C address

5.118.2.2 #define LEGO_ADDR_TEMP 0x98

The LEGO temperature sensor's I2C address

5.118.2.3 #define LEGO_ADDR_US 0x02

The LEGO ultrasonic sensor's I2C address

5.119 Ultrasonic sensor constants

Constants for use with the ultrasonic sensor.

Defines

- #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

5.119.1 Detailed Description

Constants for use with the ultrasonic sensor.

5.119.2 Define Documentation

5.119.2.1 #define **US_CMD_CONTINUOUS** 0x02

Command to put the ultrasonic sensor into continuous polling mode (default)

5.119.2.2 #define **US_CMD_EVENTCAPTURE** 0x03

Command to put the ultrasonic sensor into event capture mode

5.119.2.3 #define **US_CMD_OFF** 0x00

Command to turn off the ultrasonic sensor

5.119.2.4 #define **US_CMD_SINGLESHOT** 0x01

Command to put the ultrasonic sensor into single shot mode

5.119.2.5 #define **US_CMD_WARMRESET** 0x04

Command to warm reset the ultrasonic sensor

5.119.2.6 #define **US_REG_ACTUAL_ZERO** 0x50

The register address used to store the actual zero value

5.119.2.7 #define US_REG_CM_INTERVAL 0x40

The register address used to store the CM interval

5.119.2.8 #define US_REG_FACTORY_ACTUAL_ZERO 0x11

The register address containing the factory setting for the actual zero value

5.119.2.9 #define US_REG_FACTORY_SCALE_DIVISOR 0x13

The register address containing the factory setting for the scale divisor value

5.119.2.10 #define US_REG_FACTORY_SCALE_FACTOR 0x12

The register address containing the factory setting for the scale factor value

5.119.2.11 #define US_REG_MEASUREMENT_UNITS 0x14

The register address containing the measurement units (degrees C or F)

5.119.2.12 #define US_REG_SCALE_DIVISOR 0x52

The register address used to store the scale divisor value

5.119.2.13 #define US_REG_SCALE_FACTOR 0x51

The register address used to store the scale factor value

5.120 LEGO temperature sensor constants

Constants for use with the LEGO temperature sensor.

Defines

- #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

5.120.1 Detailed Description

Constants for use with the LEGO temperature sensor.

5.120.2 Define Documentation

5.120.2.1 #define TEMP_FQ_1 0x00

Set fault queue to 1 fault before alert

5.120.2.2 #define TEMP_FQ_2 0x08

Set fault queue to 2 faults before alert

5.120.2.3 #define TEMP_FQ_4 0x10

Set fault queue to 4 faults before alert

5.120.2.4 #define TEMP_FQ_6 0x18

Set fault queue to 6 faults before alert

5.120.2.5 #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.

5.120.2.6 #define TEMP_POL_HIGH 0x04

Set polarity of ALERT pin to be active HIGH

5.120.2.7 #define TEMP_POL_LOW 0x00

Set polarity of ALERT pin to be active LOW

5.120.2.8 #define TEMP_REG_CONFIG 0x01

The register for reading/writing sensor configuration values

5.120.2.9 #define TEMP_REG_TEMP 0x00

The register where temperature values can be read

5.120.2.10 #define TEMP_REG_THIGH 0x03

The register for reading/writing a user-defined high temperature limit

5.120.2.11 #define TEMP_REG_TLOW 0x02

The register for reading/writing a user-defined low temperature limit

5.120.2.12 #define TEMP_RES_10BIT 0x20

Set the temperature conversion resolution to 10 bit

5.120.2.13 #define TEMP_RES_11BIT 0x40

Set the temperature conversion resolution to 11 bit

5.120.2.14 #define TEMP_RES_12BIT 0x60

Set the temperature conversion resolution to 12 bit

5.120.2.15 #define TEMP_RES_9BIT 0x00

Set the temperature conversion resolution to 9 bit

5.120.2.16 #define TEMP_SD_CONTINUOUS 0x00

Set the sensor mode to continuous

5.120.2.17 #define TEMP_SD_SHUTDOWN 0x01

Set the sensor mode to shutdown. The device will shut down after the current conversion is completed.

5.120.2.18 #define TEMP_TM_COMPARATOR 0x00

Set the thermostat mode to comparator

5.120.2.19 #define TEMP_TM_INTERRUPT 0x02

Set the thermostat mode to interrupt

5.121 E-Meter sensor constants

Constants for use with the e-meter sensor.

Defines

- #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

5.121.1 Detailed Description

Constants for use with the e-meter sensor.

5.121.2 Define Documentation**5.121.2.1 #define EMETER_REG_AIN 0x0c**

The register address for amps in

5.121.2.2 #define EMETER_REG_AOUT 0x10

The register address for amps out

5.121.2.3 #define EMETER_REG_JOULES 0x12

The register address for joules

5.121.2.4 #define EMETER_REG_VIN 0x0a

The register address for voltage in

5.121.2.5 #define EMETER_REG_VOUT 0x0e

The register address for voltage out

5.121.2.6 #define EMETER_REG_WIN 0x14

The register address for watts in

5.121.2.7 #define EMETER_REG_WOUT 0x16

The register address for watts out

5.122 I2C option constants

Constants for the SetI2COptions function.

Defines

- #define I2C_OPTION_STANDARD 0x00
- #define I2C_OPTION_NORESTART 0x04
- #define I2C_OPTION_FAST 0x08

5.122.1 Detailed Description

Constants for the SetI2COptions function. These values are combined with a bitwise OR operation.

See also:

[SetI2COptions\(\)](#)

5.122.2 Define Documentation**5.122.2.1 #define I2C_OPTION_FAST 0x08**

Fast I2C speed

5.122.2.2 #define I2C_OPTION_NORESTART 0x04

Use no restart on I2C read

5.122.2.3 #define I2C_OPTION_STANDARD 0x00

Standard I2C speed

5.123 Display module constants

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

Modules

- [Line number constants](#)

Line numbers for use with the DrawText system function.

- [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.

- [Display flags](#)

Constants that are for use with the display flags functions.

- [Display contrast constants](#)

Constants that are for use with the display contrast API functions.

- [Text line constants](#)

Constants that are for use with getting/setting display data.

- **Display module IOMAP offsets**

Constant offsets into the display module IOMAP structure.

Defines

- #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

5.123.1 Detailed Description

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

5.123.2 Define Documentation

5.123.2.1 `#define BITMAP_1 0`

Bitmap 1

5.123.2.2 `#define BITMAP_2 1`

Bitmap 2

5.123.2.3 `#define BITMAP_3 2`

Bitmap 3

5.123.2.4 `#define BITMAP_4 3`

Bitmap 4

5.123.2.5 `#define BITMAPS 4`

The number of bitmap bits

5.123.2.6 `#define DISPLAY_HEIGHT 64`

The height of the LCD screen in pixels

5.123.2.7 `#define DISPLAY_MENUICONS_X_DIFF 31`

5.123.2.8 `#define DISPLAY_MENUICONS_X_OFFSET 7`

5.123.2.9 #define DISPLAY_MENUICONS_Y 40

The width of the LCD screen in pixels

5.123.2.10 #define DISPLAY_WIDTH 100

Center icon select frame

5.123.2.11 #define FRAME_SELECT 0

Center icon

5.123.2.12 #define MENUICON_CENTER 1

Center icon

5.123.2.13 #define MENUICON_LEFT 0

Left icon

Right icon

5.123.2.15 #define MENUICONS 3

The number of menu icons

5.123.2.19 #define SCREEN_MODE_CLEAR 0x01

Clear the screen

See also:

[SetScreenMode\(\)](#)

5.123.2.20 #define SCREEN_MODE_RESTORE 0x00

Restore the screen

See also:

[SetScreenMode\(\)](#)

5.123.2.21 #define SCREEN_SMALL 2

Screen between menu icons and status line

5.123.2.22 #define SCREENS 3

The number of screen bits

5.123.2.23 #define SPECIALS 5

The number of special bit values

5.123.2.24 #define STATUSICON_BATTERY 3

Battery status icon collection

5.123.2.25 #define STATUSICON_BLUETOOTH 0

BlueTooth status icon collection

5.123.2.26 #define STATUSICON_USB 1

USB status icon collection

5.123.2.27 #define STATUSICON_VM 2

VM status icon collection

5.123.2.28 #define STATUSICONS 4

The number of status icons

5.123.2.29 #define STATUSTEXT 1

Status text (BT name)

5.123.2.30 #define STEPICON_1 0

Left most step icon

5.123.2.31 #define STEPICON_2 1

5.123.2.32 #define STEPICON_3 2

5.123.2.33 #define STEPICON_4 3

5.123.2.34 #define STEPICON_5 4

Right most step icon

5.123.2.35 #define STEPICONS 5

5.123.2.36 #define STEPLINE 3

Step collection lines

5.123.2.37 #define TOPLINE 4

Top status underline

5.124 DisplayExecuteFunction constants

Constants that are for use with the DisplayExecuteFunction system call.

Defines

- #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

5.124.1 Detailed Description

Constants that are for use with the DisplayExecuteFunction system call.

5.124.2 Define Documentation**5.124.2.1 #define DISPLAY_CHAR 0x04**

W - draw char (actual font) (CMD,TRUE,X1,Y1,Char,x)

5.124.2.2 #define DISPLAY_ERASE_ALL 0x00

W - erase entire screen (CMD,x,x,x,x,x)

5.124.2.3 #define DISPLAY_ERASE_LINE 0x05

W - erase a single line (CMD,x,LINE,x,x,x)

5.124.2.4 #define DISPLAY_FILL_REGION 0x06

W - fill screen region (CMD,TRUE/FALSE,X1,Y1,X2,Y2)

5.124.2.5 #define DISPLAY_FRAME 0x07

W - draw a frame (on/off) (CMD,TRUE/FALSE,X1,Y1,X2,Y2)

5.124.2.6 #define DISPLAY_HORIZONTAL_LINE 0x02

W - draw horizontal line (CMD,TRUE/FALSE,X1,Y1,X2,x)

5.124.2.7 #define DISPLAY_PIXEL 0x01

W - set pixel (on/off) (CMD,TRUE/FALSE,X,Y,x,x)

5.124.2.8 #define DISPLAY_VERTICAL_LINE 0x03

W - draw vertical line (CMD,TRUE/FALSE,X1,Y1,x,Y2)

5.125 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.

Defines

- #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)

5.125.1 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\(\)](#)

5.125.2 Define Documentation

5.125.2.1 #define DRAW_OPT_CLEAR (0x0004)

Clear pixels while drawing (aka draw in white)

5.125.2.2 #define DRAW_OPT_CLEAR_EXCEPT_STATUS_-SCREEN (0x0002)

Clear the screen except for the status line before drawing

5.125.2.3 #define DRAW_OPT_CLEAR_PIXELS (0x0004)

Clear pixels while drawing (aka draw in white)

5.125.2.4 #define DRAW_OPT_CLEAR_SCREEN_MODES (0x0003)

Bit mask for the clear screen modes

5.125.2.5 #define DRAW_OPT_CLEAR_WHOLE_SCREEN (0x0001)

Clear the entire screen before drawing

5.125.2.6 #define DRAW_OPT_FILL_SHAPE (0x0020)

Fill the shape while drawing (rectangle, circle, ellipses, and polygon)

5.125.2.7 #define DRAW_OPT_INVERT (0x0004)

Invert text or graphics

5.125.2.8 #define DRAW_OPT_LOGICAL_AND (0x0008)

Draw pixels using a logical AND operation

5.125.2.9 #define DRAW_OPT_LOGICAL_COPY (0x0000)

Draw pixels using a logical copy operation

5.125.2.10 #define DRAW_OPT_LOGICAL_OPERATIONS (0x0018)

Bit mask for the logical drawing operations

5.125.2.11 #define DRAW_OPT_LOGICAL_OR (0x0010)

Draw pixels using a logical OR operation

5.125.2.12 #define DRAW_OPT_LOGICAL_XOR (0x0018)

Draw pixels using a logical XOR operation

5.125.2.13 #define DRAW_OPT_NORMAL (0x0000)

Normal drawing

5.125.2.14 #define DRAW_OPT_POLYGON_POLYLINE (0x0400)

When drawing polygons, do not close (i.e., draw a polyline instead)

5.126 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.

Defines

- #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)

5.126.1 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\(\)](#)

5.126.2 Define Documentation

5.126.2.1 #define DRAW_OPT_FONT_DIR_B2TL (0x0100)

Font bottom to top left align

5.126.2.2 #define DRAW_OPT_FONT_DIR_B2TR (0x0140)

Font bottom to top right align

5.126.2.3 #define DRAW_OPT_FONT_DIR_L2RB (0x0000)

Font left to right bottom align

5.126.2.4 #define DRAW_OPT_FONT_DIR_L2RT (0x0040)

Font left to right top align

5.126.2.5 #define DRAW_OPT_FONT_DIR_R2LB (0x0080)

Font right to left bottom align

5.126.2.6 #define DRAW_OPT_FONT_DIR_R2LT (0x00C0)

Font right to left top align

5.126.2.7 #define DRAW_OPT_FONT_DIR_T2BL (0x0180)

Font top to bottom left align

5.126.2.8 #define DRAW_OPT_FONT_DIR_T2BR (0x01C0)

Font top to bottom right align

5.126.2.9 #define DRAW_OPT_FONT_DIRECTIONS (0x01C0)

Bit mask for the font direction bits

5.126.2.10 #define DRAW_OPT_FONT_WRAP (0x0200)

Option to have text wrap in [FontNumOut](#) and [FontTextOut](#) calls

5.127 Display flags

Constants that are for use with the display flags functions.

Defines

- #define [DISPLAY_ON](#) 0x01
- #define [DISPLAY_REFRESH](#) 0x02
- #define [DISPLAY_POPUP](#) 0x08
- #define [DISPLAY_REFRESH_DISABLED](#) 0x40
- #define [DISPLAY_BUSY](#) 0x80

5.127.1 Detailed Description

Constants that are for use with the display flags functions.

See also:

[SetDisplayFlags\(\)](#), [DisplayFlags\(\)](#)

5.127.2 Define Documentation

5.127.2.1 #define DISPLAY_BUSY 0x80

R - Refresh in progress

5.127.2.2 #define DISPLAY_ON 0x01

W - Display on

5.127.2.3 #define DISPLAY_POPUP 0x08

W - Use popup display memory

5.127.2.4 #define DISPLAY_REFRESH 0x02

W - Enable refresh

5.127.2.5 #define DISPLAY_REFRESH_DISABLED 0x40

R - Refresh disabled

5.128 Display contrast constants

Constants that are for use with the display contrast API functions.

Defines

- #define [DISPLAY_CONTRAST_DEFAULT](#) 0x5A
- #define [DISPLAY_CONTRAST_MAX](#) 0x7F

5.128.1 Detailed Description

Constants that are for use with the display contrast API functions.

See also:

[SetDisplayContrast\(\)](#), [DisplayContrast\(\)](#)

5.128.2 Define Documentation

5.128.2.1 #define DISPLAY_CONTRAST_DEFAULT 0x5A

Default display contrast value

5.128.2.2 #define DISPLAY_CONTRAST_MAX 0x7F

Maximum display contrast value

5.129 Text line constants

Constants that are for use with getting/setting display data.

Defines

- #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

5.129.1 Detailed Description

Constants that are for use with getting/setting display data.

See also:

[SetDisplayNormal\(\)](#), [GetDisplayNormal\(\)](#), [SetDisplayPopup\(\)](#), [GetDisplayPopup\(\)](#)

5.129.2 Define Documentation**5.129.2.1 #define TEXTLINE_1 0**

Text line 1

5.129.2.2 #define TEXTLINE_2 1

Text line 2

5.129.2.3 #define TEXTLINE_3 2

Text line 3

5.129.2.4 #define TEXTLINE_4 3

Text line 4

5.129.2.5 #define TEXTLINE_5 4

Text line 5

5.129.2.6 #define TEXTLINE_6 5

Text line 6

5.129.2.7 #define TEXTLINE_7 6

Text line 7

5.129.2.8 #define TEXTLINE_8 7

Text line 8

5.129.2.9 #define TEXTLINES 8

The number of text lines on the LCD

5.130 Display module IOMAP offsets

Constant offsets into the display module IOMAP structure.

Defines

- #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

5.130.1 Detailed Description

Constant offsets into the display module IOMAP structure.

5.130.2 Define Documentation

5.130.2.1 #define `DisplayOffsetContrast` 1719

Adjust the display contrast with this field

5.130.2.2 #define `DisplayOffsetDisplay` 104

Display content copied to physical display every 17 mS

5.130.2.3 #define DisplayOffsetEraseMask 4

Section erase mask (executed first)

5.130.2.4 #define DisplayOffsetFlags 117

Update flags enumerated above

5.130.2.5 #define DisplayOffsetNormal(l, w) (((l)*100)+(w)+119)

Raw display memory for normal screen

5.130.2.6 #define DisplayOffsetPBitmaps(p) (((p)*4)+68)

Pointer to free bitmap files

5.130.2.7 #define DisplayOffsetPFont 12

Pointer to font file

5.130.2.8 #define DisplayOffsetPFunc 0

Simple draw entry

5.130.2.9 #define DisplayOffsetPMenuIcons(p) (((p)*4)+88)

Pointer to menu icon images (NULL == none)

5.130.2.10 #define DisplayOffsetPMenuText 84

Pointer to menu icon text (NULL == none)

5.130.2.11 #define DisplayOffsetPopup(l, w) (((l)*100)+(w)+919)

Raw display memory for popup screen

5.130.2.12 #define DisplayOffsetPScreens(p) (((p)*4)+56)

Pointer to screen bitmap file

5.130.2.13 #define DisplayOffsetPStatusIcons 52

Pointer to status icon collection file

5.130.2.14 #define DisplayOffsetPStatusText 48

Pointer to status text string

5.130.2.15 #define DisplayOffsetPStepIcons 100

Pointer to step icon collection file

5.130.2.16 #define DisplayOffsetPTextLines(p) (((p)*4)+16)

Pointer to text strings

5.130.2.17 #define DisplayOffsetStatusIcons(p) ((p)+108)

Index in status icon collection file (index = 0 -> none)

5.130.2.18 #define DisplayOffsetStepIcons(p) ((p)+112)

Index in step icon collection file (index = 0 -> none)

5.130.2.19 #define DisplayOffsetTextLinesCenterFlags 118

Mask to center TextLines

5.130.2.20 #define DisplayOffsetUpdateMask 8

Section update mask (executed next)

5.131 Comm module constants

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

Modules

- [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.

- [Bluetooth State constants](#)

Constants related to the bluetooth state.

- [Data mode constants](#)

Constants related to the bluetooth and hi-speed data modes.

- [Bluetooth state status constants](#)

Constants related to the bluetooth state status.

- [Remote connection constants](#)

Constants for specifying remote connection slots.

- [Bluetooth hardware status constants](#)

Constants related to the bluetooth hardware status.

- [Hi-speed port constants](#)

Constants related to the hi-speed port.

- [Device status constants](#)

Constants referring to DeviceStatus within DeviceTable.

- [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.

- [Comm module IOMAP offsets](#)

Constant offsets into the Comm module IOMAP structure.

5.131.1 Detailed Description

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

5.132 Miscellaneous Comm module constants

Miscellaneous constants related to the Comm module.

Defines

- #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

5.132.1 Detailed Description

Miscellaneous constants related to the Comm module.

5.132.2 Define Documentation

5.132.2.1 #define BT_CMD_BYTE 1

Size of Bluetooth command

5.132.2.2 #define BT_DEFAULT_INQUIRY_MAX 0

Bluetooth default inquiry Max (0 == unlimited)

5.132.2.3 #define BT_DEFAULT_INQUIRY_TIMEOUT_LO 15

Bluetooth inquiry timeout (15*1.28 sec = 19.2 sec)

5.132.2.4 #define MAX_BT_MSG_SIZE 60000

Max Bluetooth Message Size

5.132.2.5 #define SIZE_OF_BDADDR 7

Size of Bluetooth Address

5.132.2.6 #define SIZE_OF_BRICK_NAME 8

Size of NXT Brick name

5.132.2.7 #define SIZE_OF_BT_CONNECT_TABLE 4

Size of Bluetooth connection table -- Index 0 is always incoming connection

5.132.2.8 #define SIZE_OF_BT_DEVICE_TABLE 30

Size of Bluetooth device table

5.132.2.9 #define SIZE_OF_BT_NAME 16

Size of Bluetooth name

5.132.2.10 #define SIZE_OF_BT_PINCODE 16

Size of Bluetooth PIN

5.132.2.11 #define SIZE_OF_BTBUF 128

Size of Bluetooth buffer

5.132.2.12 #define SIZE_OF_CLASS_OF_DEVICE 4

Size of class of device

5.132.2.13 #define SIZE_OF_HSBUF 128

Size of High Speed Port 4 buffer

5.132.2.14 #define SIZE_OF_USBBUF 64

Size of USB Buffer in bytes

5.132.2.15 #define SIZE_OF_USBDATA 62

Size of USB Buffer available for data

5.132.2.16 #define USB_PROTOCOL_OVERHEAD 2

Size of USB Overhead in bytes -- Command type byte + Command

5.133 Bluetooth State constants

Constants related to the bluetooth state.

Defines

- #define BT_ARM_OFF 0
- #define BT_ARM_CMD_MODE 1
- #define BT_ARM_DATA_MODE 2

5.133.1 Detailed Description

Constants related to the bluetooth state.

5.133.2 Define Documentation**5.133.2.1 #define BT_ARM_CMD_MODE 1**

BtState constant bluetooth command mode

5.133.2.2 #define BT_ARM_DATA_MODE 2

BtState constant bluetooth data mode

5.133.2.3 #define BT_ARM_OFF 0

BtState constant bluetooth off

5.134 Data mode constants

Constants related to the bluetooth and hi-speed data modes.

Defines

- #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

5.134.1 Detailed Description

Constants related to the bluetooth and hi-speed data modes.

5.134.2 Define Documentation

5.134.2.1 #define DATA_MODE_GPS 0x01

Use GPS data mode

5.134.2.2 #define DATA_MODE_MASK 0x07

A mask for the data mode bits.

5.134.2.3 #define DATA_MODE_NXT 0x00

Use NXT data mode

5.134.2.4 #define DATA_MODE_RAW 0x02

Use RAW data mode

5.134.2.5 #define DATA_MODE_UPDATE 0x08

Indicates that the data mode has been changed.

5.135 Bluetooth state status constants

Constants related to the bluetooth state status.

Defines

- #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

5.135.1 Detailed Description

Constants related to the bluetooth state status.

5.135.2 Define Documentation

5.135.2.1 #define BT_BRICK_PORT_OPEN 0x02

BtStateStatus port open bit

5.135.2.2 #define BT_BRICK_VISIBILITY 0x01

BtStateStatus brick visibility bit

5.135.2.3 #define BT_CONNECTION_0_ENABLE 0x10

BtStateStatus connection 0 enable/disable bit

5.135.2.4 #define BT_CONNECTION_1_ENABLE 0x20

BtStateStatus connection 1 enable/disable bit

5.135.2.5 #define BT_CONNECTION_2_ENABLE 0x40

BtStateStatus connection 2 enable/disable bit

5.135.2.6 #define BT_CONNECTION_3_ENABLE 0x80

BtStateStatus connection 3 enable/disable bit

5.136 Remote connection constants

Constants for specifying remote connection slots.

Defines

- #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

5.136.1 Detailed Description

Constants for specifying remote connection slots.

5.136.2 Define Documentation

5.136.2.1 #define CONN_BT0 0x0

Bluetooth connection 0

5.136.2.2 #define CONN_BT1 0x1

Bluetooth connection 1

5.136.2.3 #define CONN_BT2 0x2

Bluetooth connection 2

5.136.2.4 #define CONN_BT3 0x3

Bluetooth connection 3

5.136.2.5 #define CONN_HS4 0x4

RS485 (hi-speed) connection (port 4, all devices)

5.136.2.6 #define CONN_HS_1 0x5

RS485 (hi-speed) connection (port 4, device address 1)

5.136.2.7 #define CONN_HS_2 0x6

RS485 (hi-speed) connection (port 4, device address 2)

5.136.2.8 #define CONN_HS_3 0x7

RS485 (hi-speed) connection (port 4, device address 3)

5.136.2.9 #define CONN_HS_4 0x8

RS485 (hi-speed) connection (port 4, device address 4)

5.136.2.10 #define CONN_HS_5 0x9

RS485 (hi-speed) connection (port 4, device address 5)

5.136.2.11 #define CONN_HS_6 0xa

RS485 (hi-speed) connection (port 4, device address 6)

5.136.2.12 #define CONN_HS_7 0xb

RS485 (hi-speed) connection (port 4, device address 7)

5.136.2.13 #define CONN_HS_8 0xc

RS485 (hi-speed) connection (port 4, device address 8)

5.136.2.14 #define CONN_HS_ALL 0x4

RS485 (hi-speed) connection (port 4, all devices)

5.137 Bluetooth hardware status constants

Constants related to the bluetooth hardware status.

Defines

- #define BT_ENABLE 0x00
- #define BT_DISABLE 0x01

5.137.1 Detailed Description

Constants related to the bluetooth hardware status.

5.137.2 Define Documentation**5.137.2.1 #define BT_DISABLE 0x01**

BtHwStatus bluetooth disable

5.137.2.2 #define BT_ENABLE 0x00

BtHwStatus bluetooth enable

5.138 Hi-speed port constants

Constants related to the hi-speed port.

Modules

- [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.

- [Hi-speed port SysCommHSControl constants](#)

Constants for use with the SysCommHSControl API function.

- [Hi-speed port baud rate constants](#)

Constants for configuring the hi-speed port baud rate (HsSpeed).

- [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.

5.138.1 Detailed Description

Constants related to the hi-speed port.

5.139 Hi-speed port flags constants

Constants related to the hi-speed port flags.

Defines

- #define HS_UPDATE 1

5.139.1 Detailed Description

Constants related to the hi-speed port flags.

5.139.2 Define Documentation

5.139.2.1 #define HS_UPDATE 1

HsFlags high speed update required

5.140 Hi-speed port state constants

Constants related to the hi-speed port state.

Defines

- #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

5.140.1 Detailed Description

Constants related to the hi-speed port state.

5.140.2 Define Documentation

5.140.2.1 #define HS_BYTES_REMAINING 16

HsState bytes remaining to be sent

5.140.2.2 #define HS_DEFAULT 6

HsState default

5.140.2.3 #define HS_DISABLE 4

HsState disable

5.140.2.4 #define HS_ENABLE 5

HsState enable

5.140.2.5 #define HS_INIT_RECEIVER 2

HsState initialize receiver

5.140.2.6 #define HS_INITIALISE 1

HsState initialize

5.140.2.7 #define HS_SEND_DATA 3

HsState send data

5.141 Hi-speed port SysCommHSControl constants

Constants for use with the SysCommHSControl API function.

Defines

- #define HS_CTRL_INIT 0
- #define HS_CTRL_UART 1
- #define HS_CTRL_EXIT 2

5.141.1 Detailed Description

Constants for use with the SysCommHSControl API function.

See also:

SysCommHSControl()

5.141.2 Define Documentation**5.141.2.1 #define HS_CTRL_EXIT 2**

Ddisable the high speed port

5.141.2.2 #define HS_CTRL_INIT 0

Enable the high speed port

5.141.2.3 #define HS_CTRL_UART 1

Setup the high speed port UART configuration

5.142 Hi-speed port baud rate constants

Constants for configuring the hi-speed port baud rate (HsSpeed).

Defines

- #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

5.142.1 Detailed Description

Constants for configuring the hi-speed port baud rate (HsSpeed).

5.142.2 Define Documentation

5.142.2.1 #define HS_BAUD_115200 12

HsSpeed 115200 Baud

5.142.2.2 #define HS_BAUD_1200 0

HsSpeed 1200 Baud

5.142.2.3 #define HS_BAUD_14400 6

HsSpeed 14400 Baud

5.142.2.4 #define HS_BAUD_19200 7

HsSpeed 19200 Baud

5.142.2.5 #define HS_BAUD_230400 13

HsSpeed 230400 Baud

5.142.2.6 #define HS_BAUD_2400 1

HsSpeed 2400 Baud

5.142.2.7 #define HS_BAUD_28800 8

HsSpeed 28800 Baud

5.142.2.8 #define HS_BAUD_3600 2

HsSpeed 3600 Baud

5.142.2.9 #define HS_BAUD_38400 9

HsSpeed 38400 Baud

5.142.2.10 #define HS_BAUD_460800 14

HsSpeed 460800 Baud

5.142.2.11 #define HS_BAUD_4800 3

HsSpeed 4800 Baud

5.142.2.12 #define HS_BAUD_57600 10

HsSpeed 57600 Baud

5.142.2.13 #define HS_BAUD_7200 4

HsSpeed 7200 Baud

5.142.2.14 #define HS_BAUD_76800 11

HsSpeed 76800 Baud

5.142.2.15 #define HS_BAUD_921600 15

HsSpeed 921600 Baud

5.142.2.16 #define HS_BAUD_9600 5

HsSpeed 9600 Baud

5.142.2.17 #define HS_BAUD_DEFAULT 15

HsSpeed default Baud (921600)

5.143 Hi-speed port UART mode constants

Constants referring to HsMode UART configuration settings.

Modules

- [Hi-speed port data bits constants](#)

Constants referring to HsMode (number of data bits).

- [Hi-speed port stop bits constants](#)

Constants referring to HsMode (number of stop bits).

- [Hi-speed port parity constants](#)

Constants referring to HsMode (parity).

- [Hi-speed port combined UART constants](#)

Constants that combine data bits, parity, and stop bits into a single value.

Defines

- [#define HS_MODE_UART_RS485 0x0](#)
- [#define HS_MODE_UART_RS232 0x1](#)
- [#define HS_MODE_MASK 0xFFFF](#)

- #define HS_UART_MASK 0x000F
- #define HS_MODE_DEFAULT HS_MODE_8N1

5.143.1 Detailed Description

Constants referring to HsMode UART configuration settings.

5.143.2 Define Documentation

5.143.2.1 #define HS_MODE_DEFAULT HS_MODE_8N1

HsMode default mode (8 data bits, no parity, 1 stop bit)

5.143.2.2 #define HS_MODE_MASK 0xFFFF

HsMode mode mask

5.143.2.3 #define HS_MODE_UART_RS232 0x1

HsMode UART in normal or RS232 mode

5.143.2.4 #define HS_MODE_UART_RS485 0x0

HsMode UART in default or RS485 mode

5.143.2.5 #define HS_UART_MASK 0x000F

HsMode UART mask

5.144 Hi-speed port data bits constants

Constants referring to HsMode (number of data bits).

Defines

- #define HS_MODE_5_DATA 0x0000
- #define HS_MODE_6_DATA 0x0040
- #define HS_MODE_7_DATA 0x0080
- #define HS_MODE_8_DATA 0x00C0

5.144.1 Detailed Description

Constants referring to HsMode (number of data bits).

5.144.2 Define Documentation

5.144.2.1 `#define HS_MODE_5_DATA 0x0000`

HsMode 5 data bits

5.144.2.2 `#define HS_MODE_6_DATA 0x0040`

HsMode 6 data bits

5.144.2.3 `#define HS_MODE_7_DATA 0x0080`

HsMode 7 data bits

5.144.2.4 `#define HS_MODE_8_DATA 0x00C0`

HsMode 8 data bits

5.145 Hi-speed port stop bits constants

Constants referring to HsMode (number of stop bits).

Defines

- `#define HS_MODE_10_STOP 0x0000`
- `#define HS_MODE_15_STOP 0x1000`
- `#define HS_MODE_20_STOP 0x2000`

5.145.1 Detailed Description

Constants referring to HsMode (number of stop bits).

5.145.2 Define Documentation**5.145.2.1 #define HS_MODE_10_STOP 0x0000**

HsMode 1 stop bit

5.145.2.2 #define HS_MODE_15_STOP 0x1000

HsMode 1.5 stop bits

5.145.2.3 #define HS_MODE_20_STOP 0x2000

HsMode 2 stop bits

5.146 Hi-speed port parity constants

Constants referring to HsMode (parity).

Defines

- #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

5.146.1 Detailed Description

Constants referring to HsMode (parity).

5.146.2 Define Documentation**5.146.2.1 #define HS_MODE_E_PARITY 0x0000**

HsMode Even parity

5.146.2.2 #define HS_MODE_M_PARITY 0x0600

HsMode Mark parity

5.146.2.3 #define HS_MODE_N_PARITY 0x0800

HsMode No parity

5.146.2.4 #define HS_MODE_O_PARITY 0x0200

HsMode Odd parity

5.146.2.5 #define HS_MODE_S_PARITY 0x0400

HsMode Space parity

5.147 Hi-speed port combined UART constants

Constants that combine data bits, parity, and stop bits into a single value.

Defines

- #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)

5.147.1 Detailed Description

Constants that combine data bits, parity, and stop bits into a single value.

5.147.2 Define Documentation**5.147.2.1 #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

5.147.2.2 #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

5.148 Hi-speed port address constants

Constants that are used to specify the Hi-speed (RS-485) port device address.

Defines

- #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

5.148.1 Detailed Description

Constants that are used to specify the Hi-speed (RS-485) port device address.

5.148.2 Define Documentation

5.148.2.1 #define HS_ADDRESS_1 1

HsAddress device address 1

5.148.2.2 #define HS_ADDRESS_2 2

HsAddress device address 2

5.148.2.3 #define HS_ADDRESS_3 3

HsAddress device address 3

5.148.2.4 #define HS_ADDRESS_4 4

HsAddress device address 4

5.148.2.5 #define HS_ADDRESS_5 5

HsAddress device address 5

5.148.2.6 #define HS_ADDRESS_6 6

HsAddress device address 6

5.148.2.7 #define HS_ADDRESS_7 7

HsAddress device address 7

5.148.2.8 #define HS_ADDRESS_8 8

HsAddress device address 8

5.148.2.9 #define HS_ADDRESS_ALL 0

HsAddress all devices

5.149 Device status constants

Constants referring to DeviceStatus within DeviceTable.

Defines

- #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

5.149.1 Detailed Description

Constants referring to DeviceStatus within DeviceTable.

5.149.2 Define Documentation**5.149.2.1 #define BT_DEVICE_AWAY 0x80**

Bluetooth device away

5.149.2.2 #define BT_DEVICE_EMPTY 0x00

Bluetooth device table empty

5.149.2.3 #define BT_DEVICE_KNOWN 0x02

Bluetooth device known

5.149.2.4 #define BT_DEVICE_NAME 0x40

Bluetooth device name

5.149.2.5 #define BT_DEVICE_UNKNOWN 0x01

Bluetooth device unknown

5.150 Comm module interface function constants

Constants for all the Comm module interface functions executable via SysCommExecuteFunction.

Defines

- #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

5.150.1 Detailed Description

Constants for all the Comm module interface functions executable via SysCommExecuteFunction.

See also:

SysCommExecuteFunction()

5.150.2 Define Documentation

5.150.2.1 #define INTF_BTOFF 13

Turn off the bluetooth radio

5.150.2.2 #define INTF_BTTON 12

Turn on the bluetooth radio

5.150.2.3 #define INTF_CONNECT 3

Connect to one of the known devices

5.150.2.4 #define INTF_CONNECTBYNAME 18

Connect to a bluetooth device by name

5.150.2.5 #define INTF_CONNECTREQ 17

Connection request from another device

5.150.2.6 #define INTF_DISCONNECT 4

Disconnect from one of the connected devices

5.150.2.7 #define INTF_DISCONNECTALL 5

Disconnect all devices

5.150.2.8 #define INTF_EXTREAD 15

External read request

5.150.2.9 #define INTF_FACTORYRESET 11

Reset bluetooth settings to factory values

5.150.2.10 #define INTF_OPENSTREAM 9

Open a bluetooth stream

5.150.2.11 #define INTF_PINREQ 16

Bluetooth PIN request

5.150.2.12 #define INTF_REMOVEDevice 6

Remove a device from the known devices table

5.150.2.13 #define INTF_SEARCH 1

Search for bluetooth devices

5.150.2.14 #define INTF_SENDDATA 10

Send data over a bluetooth connection

5.150.2.15 #define INTF_SENDFILE 0

Send a file via bluetooth to another device

5.150.2.16 #define INTF_SETBTNAME 14

Set the bluetooth name

5.150.2.17 #define INTF_SETCMDMODE 8

Set bluetooth into command mode

5.150.2.18 #define INTF_STOPSEARCH 2

Stop searching for bluetooth devices

5.150.2.19 #define INTF_VISIBILITY 7

Set the bluetooth visibility on or off

5.151 Comm module status code constants

Constants for Comm module status codes.

Defines

- #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

5.151.1 Detailed Description

Constants for Comm module status codes.

5.151.2 Define Documentation**5.151.2.1 #define BT_CMD_READY 0x02**

A constant representing bluetooth direct command

5.151.2.2 #define HS_CMD_READY 0x04

A constant representing high speed direct command

5.151.2.3 #define LR_COULD_NOT_SAVE 0x51

Bluetooth list result could not save

5.151.2.4 #define LR_ENTRY_REMOVED 0x53

Bluetooth list result entry removed

5.151.2.5 #define LR_STORE_IS_FULL 0x52

Bluetooth list result store is full

5.151.2.6 #define LR_SUCCESS 0x50

Bluetooth list result success

5.151.2.7 #define LR_UNKNOWN_ADDR 0x54

Bluetooth list result unknown address

5.151.2.8 #define USB_CMD_READY 0x01

A constant representing usb direct command

5.152 Comm module IOMAP offsets

Constant offsets into the Comm module IOMAP structure.

Defines

- #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

5.152.1 Detailed Description

Constant offsets into the Comm module IOMAP structure.

5.152.2 Define Documentation**5.152.2.1 #define CommOffsetBrickDataBdAddr 1144**

Offset to Bluetooth address (7 bytes)

5.152.2.2 #define CommOffsetBrickDataBluecoreVersion 1142

Offset to Bluecore version (2 bytes)

5.152.2.3 #define CommOffsetBrickDataBtHwStatus 1152

Offset to BtHwStatus (1 byte)

5.152.2.4 #define CommOffsetBrickDataBtStateStatus 1151

Offset to BtStateStatus (1 byte)

5.152.2.5 #define CommOffsetBrickDataName 1126

Offset to brick name (16 bytes)

5.152.2.6 #define CommOffsetBrickDataTableBdAddr(p) (((p)*47)+974)

Offset to data timeout value (1 byte)

5.152.2.7 #define CommOffsetBtConnectTableBdAddr(p) (((p)*47)+974)

Offset to Bluetooth connect table address (7 bytes)

5.152.2.8 #define CommOffsetBtConnectTableClassOfDevice(p) (((p)*47)+954)

Offset to Bluetooth connect table device class (4 bytes)

5.152.2.9 #define CommOffsetBtConnectTableHandleNr(p) (((p)*47)+981)

Offset to Bluetooth connect table handle (1 byte)

5.152.2.10 #define CommOffsetBtConnectTableLinkQuality(p) (((p)*47)+983)

Offset to Bluetooth connect table link quality (1 byte)

5.152.2.11 #define CommOffsetBtConnectTableName(p) (((p)*47)+938)

Offset to Bluetooth connect table name (16 bytes)

5.152.2.12 #define CommOffsetBtConnectTablePinCode(p) (((p)*47)+958)

Offset to Bluetooth connect table pin code (16 bytes)

5.152.2.13 #define CommOffsetBtConnectTableStreamStatus(p) (((p)*47)+982)

Offset to Bluetooth connect table stream status (1 byte)

5.152.2.14 #define CommOffsetBtDataMode 1898

Offset to Bluetooth data mode (1 byte)

5.152.2.15 #define CommOffsetBtDeviceCnt 1889

Offset to Bluetooth device count (1 byte)

5.152.2.16 #define CommOffsetBtDeviceNameCnt 1890

Offset to Bluetooth device name count (1 byte)

5.152.2.17 #define CommOffsetBtDeviceTableBdAddr(p) (((p)*31)+28)

Offset to Bluetooth device table address (7 bytes)

5.152.2.18 #define CommOffsetBtDeviceTableClassOfDevice(p) (((p)*31)+24)

Offset to Bluetooth device table device class (4 bytes)

5.152.2.19 #define CommOffsetBtDeviceTableDeviceStatus(p) (((p)*31)+35)

Offset to Bluetooth device table status (1 byte)

5.152.2.20 #define CommOffsetBtDeviceTableName(p) (((p)*31)+8)

Offset to BT device table name (16 bytes)

5.152.2.21 #define CommOffsetBtInBufBuf 1157

Offset to Bluetooth input buffer data (128 bytes)

5.152.2.22 #define CommOffsetBtInBufInPtr 1285

Offset to Bluetooth input buffer front pointer (1 byte)

5.152.2.23 #define CommOffsetBtInBufOutPtr 1286

Offset to Bluetooth output buffer back pointer (1 byte)

5.152.2.24 #define CommOffsetBtOutBufBuf 1289

Offset to Bluetooth output buffer offset data (128 bytes)

5.152.2.25 #define CommOffsetBtOutBufInPtr 1417

Offset to Bluetooth output buffer front pointer (1 byte)

5.152.2.26 #define CommOffsetBtOutBufOutPtr 1418

Offset to Bluetooth output buffer back pointer (1 byte)

5.152.2.27 #define CommOffsetHsAddress 1895

Offset to High Speed address (1 byte)

5.152.2.28 #define CommOffsetHsDataMode 1899

Offset to High Speed data mode (1 byte)

5.152.2.29 #define CommOffsetHsFlags 1891

Offset to High Speed flags (1 byte)

5.152.2.30 #define CommOffsetHsInBufBuf 1421

Offset to High Speed input buffer data (128 bytes)

5.152.2.31 #define CommOffsetHsInBufInPtr 1549

Offset to High Speed input buffer front pointer (1 byte)

5.152.2.32 #define CommOffsetHsInBufOutPtr 1550

Offset to High Speed input buffer back pointer (1 byte)

5.152.2.33 #define CommOffsetHsMode 1896

Offset to High Speed mode (2 bytes)

5.152.2.34 #define CommOffsetHsOutBufBuf 1553

Offset to High Speed output buffer data (128 bytes)

5.152.2.35 #define CommOffsetHsOutBufInPtr 1681

Offset to High Speed output buffer front pointer (1 byte)

5.152.2.36 #define CommOffsetHsOutBufOutPtr 1682

Offset to High Speed output buffer back pointer (1 byte)

5.152.2.37 #define CommOffsetHsSpeed 1892

Offset to High Speed speed (1 byte)

5.152.2.38 #define CommOffsetHsState 1893

Offset to High Speed state (1 byte)

5.152.2.39 #define CommOffsetPFunc 0

Offset to the Comm module first function pointer (4 bytes)

5.152.2.40 #define CommOffsetPFuncTwo 4

Offset to the Comm module second function pointer (4 bytes)

5.152.2.41 #define CommOffsetUsbInBufBuf 1685

Offset to Usb input buffer data (64 bytes)

5.152.2.42 #define CommOffsetUsbInBufInPtr 1749

Offset to Usb input buffer front pointer (1 byte)

5.152.2.43 #define CommOffsetUsbInBufOutPtr 1750

Offset to Usb input buffer back pointer (1 byte)

5.152.2.44 #define CommOffsetUsbOutBufBuf 1753

Offset to Usb output buffer data (64 bytes)

5.152.2.45 #define CommOffsetUsbOutBufInPtr 1817

Offset to Usb output buffer front pointer (1 byte)

5.152.2.46 #define CommOffsetUsbOutBufOutPtr 1818

Offset to Usb output buffer back pointer (1 byte)

5.152.2.47 #define CommOffsetUsbPollBufBuf 1821

Offset to Usb Poll buffer data (64 bytes)

5.152.2.48 #define CommOffsetUsbPollBufInPtr 1885

Offset to Usb Poll buffer front pointer (1 byte)

5.152.2.49 #define CommOffsetUsbPollBufOutPtr 1886

Offset to Usb Poll buffer back pointer (1 byte)

5.152.2.50 #define CommOffsetUsbState 1894

Offset to Usb State (1 byte)

5.153 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 output constants](#)

Constants for use when choosing RCX outputs.

- [RCX output mode constants](#)

Constants for use when configuring RCX output mode.

- [RCX output direction constants](#)

Constants for use when configuring RCX output direction.

- [RCX output power constants](#)

Constants for use when configuring RCX output power.

- [RCX IR remote constants](#)

Constants for use when simulating RCX IR remote messages.

- [RCX and Scout sound constants](#)

Constants for use when playing standard RCX and Scout sounds.

- [Scout constants](#)

Constants for use when controlling the Scout brick.

- [RCX and Scout source constants](#)

Constants for use when specifying RCX and Scout sources.

- [RCX and Scout opcode constants](#)

Constants for use when specifying RCX and Scout opcodes.

5.153.1 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.

5.154 RCX output constants

Constants for use when choosing RCX outputs.

Defines

- #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

5.154.1 Detailed Description

Constants for use when choosing RCX outputs.

5.154.2 Define Documentation

5.154.2.1 #define `RCX_OUT_A` 0x01

RCX Output A

5.154.2.2 #define `RCX_OUT_AB` 0x03

RCX Outputs A and B

5.154.2.3 #define `RCX_OUT_ABC` 0x07

RCX Outputs A, B, and C

5.154.2.4 #define `RCX_OUT_AC` 0x05

RCX Outputs A and C

5.154.2.5 #define RCX_OUT_B 0x02

RCX Output B

5.154.2.6 #define RCX_OUT_BC 0x06

RCX Outputs B and C

5.154.2.7 #define RCX_OUT_C 0x04

RCX Output C

5.155 RCX output mode constants

Constants for use when configuring RCX output mode.

Defines

- #define [RCX_OUT_FLOAT](#) 0
- #define [RCX_OUT_OFF](#) 0x40
- #define [RCX_OUT_ON](#) 0x80

5.155.1 Detailed Description

Constants for use when configuring RCX output mode.

5.155.2 Define Documentation**5.155.2.1 #define RCX_OUT_FLOAT 0**

Set RCX output to float

5.155.2.2 #define RCX_OUT_OFF 0x40

Set RCX output to off

5.155.2.3 #define RCX_OUT_ON 0x80

Set RCX output to on

5.156 RCX output direction constants

Constants for use when configuring RCX output direction.

Defines

- #define `RCX_OUT_REV` 0
- #define `RCX_OUT_TOGGLE` 0x40
- #define `RCX_OUT_FWD` 0x80

5.156.1 Detailed Description

Constants for use when configuring RCX output direction.

5.156.2 Define Documentation

5.156.2.1 #define `RCX_OUT_FWD` 0x80

Set RCX output direction to forward

5.156.2.2 #define `RCX_OUT_REV` 0

Set RCX output direction to reverse

5.156.2.3 #define `RCX_OUT_TOGGLE` 0x40

Set RCX output direction to toggle

5.157 RCX output power constants

Constants for use when configuring RCX output power.

Defines

- #define `RCX_OUT_LOW` 0
- #define `RCX_OUT_HALF` 3
- #define `RCX_OUT_FULL` 7

5.157.1 Detailed Description

Constants for use when configuring RCX output power.

5.157.2 Define Documentation

5.157.2.1 #define RCX_OUT_FULL 7

Set RCX output power level to full

5.157.2.2 #define RCX_OUT_HALF 3

Set RCX output power level to half

5.157.2.3 #define RCX_OUT_LOW 0

Set RCX output power level to low

5.158 RCX IR remote constants

Constants for use when simulating RCX IR remote messages.

Defines

- #define `RCX_RemoteKeysReleased` 0x0000
- #define `RCX_RemotePBMessage1` 0x0100
- #define `RCX_RemotePBMessage2` 0x0200
- #define `RCX_RemotePBMessage3` 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

5.158.1 Detailed Description

Constants for use when simulating RCX IR remote messages.

5.158.2 Define Documentation

5.158.2.1 **#define RCX_RemoteKeysReleased 0x0000**

All remote keys have been released

5.158.2.2 **#define RCX_RemoteOutABackward 0x4000**

Set output A backward

5.158.2.3 **#define RCX_RemoteOutAForward 0x0800**

Set output A forward

5.158.2.4 **#define RCX_RemoteOutBBackward 0x8000**

Set output B backward

5.158.2.5 **#define RCX_RemoteOutBForward 0x1000**

Set output B forward

5.158.2.6 **#define RCX_RemoteOutCBackward 0x0001**

Set output C backward

5.158.2.7 **#define RCX_RemoteOutCForward 0x2000**

Set output C forward

5.158.2.8 **#define RCX_RemotePBMessage1 0x0100**

Send PB message 1

5.158.2.9 #define RCX_RemotePBMessage2 0x0200

Send PB message 2

5.158.2.10 #define RCX_RemotePBMessage3 0x0400

Send PB message 3

5.158.2.11 #define RCX_RemotePlayASound 0x0080

Play a sound

5.158.2.12 #define RCX_RemoteSelProgram1 0x0002

Select program 1

5.158.2.13 #define RCX_RemoteSelProgram2 0x0004

Select program 2

5.158.2.14 #define RCX_RemoteSelProgram3 0x0008

Select program 3

5.158.2.15 #define RCX_RemoteSelProgram4 0x0010

Select program 4

5.158.2.16 #define RCX_RemoteSelProgram5 0x0020

Select program 5

5.158.2.17 #define RCX_RemoteStopOutOff 0x0040

Stop and turn off outputs

5.159 RCX and Scout sound constants

Constants for use when playing standard RCX and Scout sounds.

Defines

- #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

5.159.1 Detailed Description

Constants for use when playing standard RCX and Scout sounds.

5.159.2 Define Documentation**5.159.2.1 #define SOUND_CLICK 0**

Play the standard key click sound

5.159.2.2 #define SOUND_DOUBLE_BEEP 1

Play the standard double beep sound

5.159.2.3 #define SOUND_DOWN 2

Play the standard sweep down sound

5.159.2.4 #define SOUND_FAST_UP 5

Play the standard fast up sound

5.159.2.5 #define SOUND_LOW_BEEP 4

Play the standard low beep sound

5.159.2.6 #define SOUND_UP 3

Play the standard sweep up sound

5.160 Scout constants

Constants for use when controlling the Scout brick.

Modules

- [Scout light constants](#)

Constants for use when controlling the Scout light settings.

- [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 mode constants](#)

Constants for use when setting the scout mode.

- [Scout motion rule constants](#)

Constants for use when setting the scout motion rule.

- [Scout touch rule constants](#)

Constants for use when setting the scout touch rule.

- [Scout light rule constants](#)

Constants for use when setting the scout light rule.

- [Scout transmit rule constants](#)

Constants for use when setting the scout transmit rule.

- [Scout special effect constants](#)

Constants for use when setting the scout special effect.

5.160.1 Detailed Description

Constants for use when controlling the Scout brick.

5.161 Scout light constants

Constants for use when controlling the Scout light settings.

Defines

- #define SCOUT_LIGHT_ON 0x80
- #define SCOUT_LIGHT_OFF 0

5.161.1 Detailed Description

Constants for use when controlling the Scout light settings.

5.161.2 Define Documentation**5.161.2.1 #define SCOUT_LIGHT_OFF 0**

Turn off the scout light

5.161.2.2 #define SCOUT_LIGHT_ON 0x80

Turn on the scout light

5.162 Scout sound constants

Constants for use when playing standard Scout sounds.

Defines

- #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

5.162.1 Detailed Description

Constants for use when playing standard Scout sounds.

5.162.2 Define Documentation

5.162.2.1 #define SCOUT_SOUND_1_BLINK 17

Play the Scout 1 blink sound

5.162.2.2 #define SCOUT_SOUND_2_BLINK 18

Play the Scout 2 blink sound

5.162.2.3 #define SCOUT_SOUND_COUNTER1 19

Play the Scout counter 1 sound

5.162.2.4 #define SCOUT_SOUND_COUNTER2 20

Play the Scout counter 2 sound

5.162.2.5 #define SCOUT_SOUND_ENTER_BRIGHT 14

Play the Scout enter bright sound

5.162.2.6 #define SCOUT_SOUND_ENTER_DARK 16

Play the Scout enter dark sound

5.162.2.7 #define SCOUT_SOUND_ENTER_NORMAL 15

Play the Scout enter normal sound

5.162.2.8 #define SCOUT_SOUND_ENTERSA 7

Play the Scout enter standalone sound

5.162.2.9 #define SCOUT_SOUND_KEYERROR 8

Play the Scout key error sound

5.162.2.10 #define SCOUT_SOUND_MAIL_RECEIVED 24

Play the Scout mail received sound

5.162.2.11 #define SCOUT_SOUND_NONE 9

Play the Scout none sound

5.162.2.12 #define SCOUT_SOUND_REMOTE 6

Play the Scout remote sound

5.162.2.13 #define SCOUT_SOUND_SPECIAL1 25

Play the Scout special 1 sound

5.162.2.14 #define SCOUT_SOUND_SPECIAL2 26

Play the Scout special 2 sound

5.162.2.15 #define SCOUT_SOUND_SPECIAL3 27

Play the Scout special 3 sound

5.162.2.16 #define SCOUT_SOUND_TIMER1 21

Play the Scout timer 1 sound

5.162.2.17 #define SCOUT_SOUND_TIMER2 22

Play the Scout timer 2 sound

5.162.2.18 #define SCOUT_SOUND_TIMER3 23

Play the Scout timer 3 sound

5.162.2.19 #define SCOUT_SOUND_TOUCH1_PRES 10

Play the Scout touch 1 pressed sound

5.162.2.20 #define SCOUT_SOUND_TOUCH1_REL 11

Play the Scout touch 1 released sound

5.162.2.21 #define SCOUT_SOUND_TOUCH2_PRES 12

Play the Scout touch 2 pressed sound

5.162.2.22 #define SCOUT_SOUND_TOUCH2_REL 13

Play the Scout touch 2 released sound

5.163 Scout sound set constants

Constants for use when choosing standard Scout sound sets.

Defines

- #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

5.163.1 Detailed Description

Constants for use when choosing standard Scout sound sets.

5.163.2 Define Documentation**5.163.2.1 #define SCOUT_SNDSET_ALARM 3**

Set sound set to alarm

5.163.2.2 #define SCOUT_SNDSET_BASIC 1

Set sound set to basic

5.163.2.3 #define SCOUT_SNDSET_BUG 2

Set sound set to bug

5.163.2.4 #define SCOUT_SNDSET_NONE 0

Set sound set to none

5.163.2.5 #define SCOUT_SNDSET_RANDOM 4

Set sound set to random

5.163.2.6 #define SCOUT_SNDSET_SCIENCE 5

Set sound set to science

5.164 Scout mode constants

Constants for use when setting the scout mode.

Defines

- #define SCOUT_MODE_STANDALONE 0
- #define SCOUT_MODE_POWER 1

5.164.1 Detailed Description

Constants for use when setting the scout mode.

5.164.2 Define Documentation**5.164.2.1 #define SCOUT_MODE_POWER 1**

Enter power mode

5.164.2.2 #define SCOUT_MODE_STANDALONE 0

Enter stand alone mode

5.165 Scout motion rule constants

Constants for use when setting the scout motion rule.

Defines

- #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

5.165.1 Detailed Description

Constants for use when setting the scout motion rule.

5.165.2 Define Documentation**5.165.2.1 #define SCOUT_MR_CIRCLE_LEFT 4**

Motion rule circle left

5.165.2.2 #define SCOUT_MR_CIRCLE_RIGHT 3

Motion rule circle right

5.165.2.3 #define SCOUT_MR_FORWARD 1

Motion rule forward

5.165.2.4 #define SCOUT_MR_LOOP_A 5

Motion rule loop A

5.165.2.5 #define SCOUT_MR_LOOP_AB 7

Motion rule loop A then B

5.165.2.6 #define SCOUT_MR_LOOP_B 6

Motion rule loop B

5.165.2.7 #define SCOUT_MR_NO_MOTION 0

Motion rule none

5.165.2.8 #define SCOUT_MR_ZIGZAG 2

Motion rule zigzag

5.166 Scout touch rule constants

Constants for use when setting the scout touch rule.

Defines

- #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

5.166.1 Detailed Description

Constants for use when setting the scout touch rule.

5.166.2 Define Documentation**5.166.2.1 #define SCOUT_TR_AVOID 2**

Touch rule avoid

5.166.2.2 #define SCOUT_TR_IGNORE 0

Touch rule ignore

5.166.2.3 #define SCOUT_TR_OFF_WHEN 4

Touch rule off when

5.166.2.4 #define SCOUT_TR_REVERSE 1

Touch rule reverse

5.166.2.5 #define SCOUT_TR_WAIT_FOR 3

Touch rule wait for

5.167 Scout light rule constants

Constants for use when setting the scout light rule.

Defines

- #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

5.167.1 Detailed Description

Constants for use when setting the scout light rule.

5.167.2 Define Documentation**5.167.2.1 #define SCOUT_LR_AVOID 3**

Light rule avoid

5.167.2.2 #define SCOUT_LR_IGNORE 0

Light rule ignore

5.167.2.3 #define SCOUT_LR_OFF_WHEN 5

Light rule off when

5.167.2.4 #define SCOUT_LR_SEEK_DARK 2

Light rule seek dark

5.167.2.5 #define SCOUT_LR_SEEK_LIGHT 1

Light rule seek light

5.167.2.6 #define SCOUT_LR_WAIT_FOR 4

Light rule wait for

5.168 Scout transmit rule constants

Constants for use when setting the scout transmit rule.

Defines

- #define SCOUT_TGS_SHORT 0
- #define SCOUT_TGS_MEDIUM 1
- #define SCOUT_TGS_LONG 2

5.168.1 Detailed Description

Constants for use when setting the scout transmit rule.

5.168.2 Define Documentation**5.168.2.1 #define SCOUT_TGS_LONG 2**

Transmit level long

5.168.2.2 #define SCOUT_TGS_MEDIUM 1

Transmit level medium

5.168.2.3 #define SCOUT_TGS_SHORT 0

Transmit level short

5.169 Scout special effect constants

Constants for use when setting the scout special effect.

Defines

- #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

5.169.1 Detailed Description

Constants for use when setting the scout special effect.

5.169.2 Define Documentation**5.169.2.1 #define SCOUT_FXR_ALARM 2**

Alarm special effects

5.169.2.2 #define SCOUT_FXR_BUG 1

Bug special effects

5.169.2.3 #define SCOUT_FXR_NONE 0

No special effects

5.169.2.4 #define SCOUT_FXR_RANDOM 3

Random special effects

5.169.2.5 #define SCOUT_FXR_SCIENCE 4

Science special effects

5.170 RCX and Scout source constants

Constants for use when specifying RCX and Scout sources.

Defines

- #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

5.170.1 Detailed Description

Constants for use when specifying RCX and Scout sources.

5.170.2 Define Documentation

5.170.2.1 #define `RCX_BatteryLevelSrc` 34

The RCX battery level source

5.170.2.2 #define `RCX_ClickCounterSrc` 27

The RCX event click counter source

5.170.2.3 #define `RCX_ConstantSrc` 2

The RCX constant value source

5.170.2.4 #define `RCX_CounterSrc` 21

The RCX counter source

5.170.2.5 #define `RCX_DatalogRawDirectSrc` 42

The RCX direct datalog raw source

5.170.2.6 #define RCX_DatalogRawIndirectSrc 41

The RCX indirect datalog raw source

5.170.2.7 #define RCX_DatalogSrcDirectSrc 38

The RCX direct datalog source source

5.170.2.8 #define RCX_DatalogSrcIndirectSrc 37

The RCX indirect datalog source source

5.170.2.9 #define RCX_DatalogValueDirectSrc 40

The RCX direct datalog value source

5.170.2.10 #define RCX_DatalogValueIndirectSrc 39

The RCX indirect datalog value source

5.170.2.11 #define RCX_DurationSrc 31

The RCX event duration source

5.170.2.12 #define RCX_EventStateSrc 25

The RCX event static source

5.170.2.13 #define RCX_FirmwareVersionSrc 35

The RCX firmware version source

5.170.2.14 #define RCX_GlobalMotorStatusSrc 17

The RCX global motor status source

5.170.2.15 #define RCX_HysteresisSrc 30

The RCX event hysteresis source

5.170.2.16 #define RCX_IndirectVarSrc 36

The RCX indirect variable source

5.170.2.17 #define RCX_InputBooleanSrc 13

The RCX input boolean source

5.170.2.18 #define RCX_InputModeSrc 11

The RCX input mode source

5.170.2.19 #define RCX_InputRawSrc 12

The RCX input raw source

5.170.2.20 #define RCX_InputTypeSrc 10

The RCX input type source

5.170.2.21 #define RCX_InputValueSrc 9

The RCX input value source

5.170.2.22 #define RCX_LowerThresholdSrc 29

The RCX event lower threshold source

5.170.2.23 #define RCX_MessageSrc 15

The RCX message source

5.170.2.24 #define RCX_OutputStatusSrc 3

The RCX output status source

5.170.2.25 #define RCX_ProgramSlotSrc 8

The RCX program slot source

5.170.2.26 #define RCX_RandomSrc 4

The RCX random number source

5.170.2.27 #define RCX_ScoutCounterLimitSrc 22

The Scout counter limit source

5.170.2.28 #define RCX_ScoutEventFBSrc 24

The Scout event feedback source

5.170.2.29 #define RCX_ScoutLightParamsSrc 19

The Scout light parameters source

5.170.2.30 #define RCX_ScoutRulesSrc 18

The Scout rules source

5.170.2.31 #define RCX_ScoutTimerLimitSrc 20

The Scout timer limit source

5.170.2.32 #define RCX_TaskEventsSrc 23

The RCX task events source

5.170.2.33 #define RCX_TenMSTimerSrc 26

The RCX 10ms timer source

5.170.2.34 #define RCX_TimerSrc 1

The RCX timer source

5.170.2.35 #define RCX_UARTSetupSrc 33

The RCX UART setup source

5.170.2.36 #define RCX_UpperThresholdSrc 28

The RCX event upper threshold source

5.170.2.37 #define RCX_VariableSrc 0

The RCX variable source

5.170.2.38 #define RCX_WatchSrc 14

The RCX watch source

5.171 RCX and Scout opcode constants

Constants for use when specifying RCX and Scout opcodes.

Defines

- #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

5.171.1 Detailed Description

Constants for use when specifying RCX and Scout opcodes.

5.171.2 Define Documentation

5.171.2.1 #define **RCX_AbsVarOp** 0x74

Absolute value function

5.171.2.2 #define **RCX_AndVarOp** 0x84

AND function

5.171.2.3 #define **RCX_AutoOffOp** 0xb1

Set auto off timer

5.171.2.4 #define **RCX_BatteryLevelOp** 0x30

Read the battery level

5.171.2.5 #define **RCX_BootModeOp** 0x65

Set into book mode

5.171.2.6 #define **RCX_CalibrateEventOp** 0x04

Calibrate event

5.171.2.7 #define RCX_ClearAllEventsOp 0x06

Clear all events

5.171.2.8 #define RCX_ClearCounterOp 0xb7

Clear a counter

5.171.2.9 #define RCX_ClearMsgOp 0x90

Clear message

5.171.2.10 #define RCX_ClearSensorOp 0xd1

Clear a sensor

5.171.2.11 #define RCX_ClearSoundOp 0x80

Clear sound

5.171.2.12 #define RCX_ClearTimerOp 0xa1

Clear a timer

5.171.2.13 #define RCX_DatalogOp 0x62

Datalog the specified source/value

5.171.2.14 #define RCX_DecCounterOp 0xa7

Decrement a counter

5.171.2.15 #define RCX_DeleteSubOp 0xc1

Delete a subroutine

5.171.2.16 #define RCX_DeleteSubsOp 0x70

Delete subroutines

5.171.2.17 #define RCX_DeleteTaskOp 0x61

Delete a task

5.171.2.18 #define RCX_DeleteTasksOp 0x40

Delete tasks

5.171.2.19 #define RCX_DirectEventOp 0x03

Fire an event

5.171.2.20 #define RCX_DisplayOp 0x33

Set LCD display value

5.171.2.21 #define RCX_DivVarOp 0x44

Divide function

5.171.2.22 #define RCX_GOutputDirOp 0x77

Set global motor direction

5.171.2.23 #define RCX_GOutputModeOp 0x67

Set global motor mode

5.171.2.24 #define RCX_GOutputPowerOp 0xa3

Set global motor power levels

5.171.2.25 #define RCX_IncCounterOp 0x97

Increment a counter

5.171.2.26 #define RCX_InputModeOp 0x42

Set the input mode

5.171.2.27 #define RCX_InputTypeOp 0x32

Set the input type

5.171.2.28 #define RCX_IRModeOp 0x31

Set the IR transmit mode

5.171.2.29 #define RCX_LightOp 0x87

Light opcode

5.171.2.30 #define RCX_LSBlinkTimeOp 0xe3

Set the light sensor blink time

5.171.2.31 #define RCX_LSCalibrateOp 0xc0

Calibrate the light sensor

5.171.2.32 #define RCX_LSHysteresisOp 0xd3

Set the light sensor hysteresis

5.171.2.33 #define RCX_LSLowerThreshOp 0xc3

Set the light sensor lower threshold

5.171.2.34 #define RCX_LSSupperThreshOp 0xb3

Set the light sensor upper threshold

5.171.2.35 #define RCX_MessageOp 0xf7

Set message

5.171.2.36 #define RCX_MulVarOp 0x54

Multiply function

5.171.2.37 #define RCX_MuteSoundOp 0xd0

Mute sound

5.171.2.38 #define RCX_OnOffFloatOp 0x21

Control motor state - on, off, float

5.171.2.39 #define RCX_OrVarOp 0x94

OR function

5.171.2.40 #define RCX_OutputDirOp 0xe1

Set the motor direction

5.171.2.41 #define RCX_OutputPowerOp 0x13

Set the motor power level

5.171.2.42 #define RCX_PBTurnOffOp 0x60

Turn off the brick

5.171.2.43 #define RCX_PingOp 0x10

Ping the brick

5.171.2.44 #define RCX_PlaySoundOp 0x51

Play a sound

5.171.2.45 #define RCX_PlayToneOp 0x23

Play a tone

5.171.2.46 #define RCX_PlayToneVarOp 0x02

Play a tone using a variable

5.171.2.47 #define RCX_PollMemoryOp 0x63

Poll a memory location

5.171.2.48 #define RCX_PollOp 0x12

Poll a source/value combination

5.171.2.49 #define RCX_RemoteOp 0xd2

Execute simulated remote control buttons

5.171.2.50 #define RCX_ScoutOp 0x47

Scout opcode

5.171.2.51 #define RCX_ScoutRulesOp 0xd5

Set Scout rules

5.171.2.52 #define RCX_SelectProgramOp 0x91

Select a program slot

5.171.2.53 #define RCX_SendUARTDataOp 0xc2

Send data via IR using UART settings

5.171.2.54 #define RCX_SetCounterOp 0xd4

Set counter value

5.171.2.55 #define RCX_SetDatalogOp 0x52

Set the datalog size

5.171.2.56 #define RCX_SetEventOp 0x93

Set an event

5.171.2.57 #define RCX_SetFeedbackOp 0x83

Set Scout feedback

5.171.2.58 #define RCX_SetPriorityOp 0xd7

Set task priority

5.171.2.59 #define RCX_SetSourceValueOp 0x05

Set a source/value

5.171.2.60 #define RCX_SetTimerLimitOp 0xc4

Set timer limit

5.171.2.61 #define RCX_SetVarOp 0x14

Set function

5.171.2.62 #define RCX_SetWatchOp 0x22

Set the watch source/value

5.171.2.63 #define RCX_SgnVarOp 0x64

Sign function

5.171.2.64 #define RCX_SoundOp 0x57

Sound opcode

5.171.2.65 #define RCX_StartTaskOp 0x71

Start a task

5.171.2.66 #define RCX_StopAllTasksOp 0x50

Stop all tasks

5.171.2.67 #define RCX_StopTaskOp 0x81

Stop a task

5.171.2.68 #define RCX_SubVarOp 0x34

Subtract function

5.171.2.69 #define RCX_SumVarOp 0x24

Sum function

5.171.2.70 #define RCX_UnlockFirmOp 0xa5

Unlock the firmware

5.171.2.71 #define RCX_UnlockOp 0x15

Unlock the brick

5.171.2.72 #define RCX_UnmuteSoundOp 0xe0

Unmute sound

5.171.2.73 #define RCX_UploadDatalogOp 0xa4

Upload datalog contents

5.171.2.74 #define RCX_ViewSourceValOp 0xe5

View a source/value

5.171.2.75 #define RCX_VLLOp 0xe2

Send visual light link (VLL) data

5.172 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

- [Power Function command constants](#)

Constants that are for sending Power Function commands.

- [Power Function channel constants](#)

Constants that are for specifying Power Function channels.

- [Power Function mode constants](#)

Constants that are for choosing Power Function modes.

- [PF/IR Train function constants](#)

Constants that are for sending PF/IR Train functions.

- [IR Train channel constants](#)

Constants that are for specifying IR Train channels.

- [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.

- [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.

5.172.1 Detailed Description

Constants that are for use with the HiTechnic IRLink or mindsensors nRLink in Power Function or IR Train mode.

5.173 Power Function command constants

Constants that are for sending Power Function commands.

Defines

- #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

5.173.1 Detailed Description

Constants that are for sending Power Function commands.

5.173.2 Define Documentation

5.173.2.1 #define PF_CMD_BRAKE 3

Power function command brake

5.173.2.2 #define PF_CMD_FLOAT 0

Power function command float (same as stop)

5.173.2.3 #define PF_CMD_FWD 1

Power function command forward

5.173.2.4 #define PF_CMD_REV 2

Power function command reverse

5.173.2.5 #define PF_CMD_STOP 0

Power function command stop

5.174 Power Function channel constants

Constants that are for specifying Power Function channels.

Defines

- #define PF_CHANNEL_1 0
- #define PF_CHANNEL_2 1
- #define PF_CHANNEL_3 2
- #define PF_CHANNEL_4 3

5.174.1 Detailed Description

Constants that are for specifying Power Function channels.

5.174.2 Define Documentation

5.174.2.1 #define PF_CHANNEL_1 0

Power function channel 1

5.174.2.2 #define PF_CHANNEL_2 1

Power function channel 2

5.174.2.3 #define PF_CHANNEL_3 2

Power function channel 3

5.174.2.4 #define PF_CHANNEL_4 3

Power function channel 4

5.175 Power Function mode constants

Constants that are for choosing Power Function modes.

Defines

- #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

5.175.1 Detailed Description

Constants that are for choosing Power Function modes.

5.175.2 Define Documentation**5.175.2.1 #define PF_MODE_COMBO_DIRECT 1**

Power function mode combo direct

5.175.2.2 #define PF_MODE_COMBO_PWM 4

Power function mode combo pulse width modulation (PWM)

5.175.2.3 #define PF_MODE_SINGLE_OUTPUT_CST 6

Power function mode single output clear, set, toggle (CST)

5.175.2.4 #define PF_MODE_SINGLE_OUTPUT_PWM 4

Power function mode single output pulse width modulation (PWM)

5.175.2.5 #define PF_MODE_SINGLE_PIN_CONT 2

Power function mode single pin continuous

5.175.2.6 #define PF_MODE_SINGLE_PIN_TIME 3

Power function mode single pin timed

5.175.2.7 #define PF_MODE_TRAIN 0

Power function mode IR Train

5.176 PF/IR Train function constants

Constants that are for sending PF/IR Train functions.

Defines

- #define TRAIN_FUNC_STOP 0
- #define TRAIN_FUNC_INCR_SPEED 1
- #define TRAIN_FUNC_DECR_SPEED 2
- #define TRAIN_FUNC_TOGGLE_LIGHT 4

5.176.1 Detailed Description

Constants that are for sending PF/IR Train functions.

5.176.2 Define Documentation**5.176.2.1 #define TRAIN_FUNC_DECR_SPEED 2**

PF/IR Train function decrement speed

5.176.2.2 #define TRAIN_FUNC_INCR_SPEED 1

PF/IR Train function increment speed

5.176.2.3 #define TRAIN_FUNC_STOP 0

PF/IR Train function stop

5.176.2.4 #define TRAIN_FUNC_TOGGLE_LIGHT 4

PF/IR Train function toggle light

5.177 IR Train channel constants

Constants that are for specifying IR Train channels.

Defines

- #define TRAIN_CHANNEL_1 0
- #define TRAIN_CHANNEL_2 1
- #define TRAIN_CHANNEL_3 2
- #define TRAIN_CHANNEL_ALL 3

5.177.1 Detailed Description

Constants that are for specifying IR Train channels.

5.177.2 Define Documentation**5.177.2.1 #define TRAIN_CHANNEL_1 0**

IR Train channel 1

5.177.2.2 #define TRAIN_CHANNEL_2 1

IR Train channel 2

5.177.2.3 #define TRAIN_CHANNEL_3 2

IR Train channel 3

5.177.2.4 #define TRAIN_CHANNEL_ALL 3

IR Train channel all

5.178 Power Function output constants

Constants that are for choosing a Power Function output.

Defines

- #define PF_OUT_A 0
- #define PF_OUT_B 1

5.178.1 Detailed Description

Constants that are for choosing a Power Function output.

5.178.2 Define Documentation**5.178.2.1 #define PF_OUT_A 0**

Power function output A

5.178.2.2 #define PF_OUT_B 1

Power function output B

5.179 Power Function pin constants

Constants that are for choosing a Power Function pin.

Defines

- #define PF_PIN_C1 0
- #define PF_PIN_C2 1

5.179.1 Detailed Description

Constants that are for choosing a Power Function pin.

5.179.2 Define Documentation**5.179.2.1 #define PF_PIN_C1 0**

Power function pin C1

5.179.2.2 #define PF_PIN_C2 1

Power function pin C2

5.180 Power Function single pin function constants

Constants that are for sending Power Function single pin functions.

Defines

- #define PF_FUNC_NOCHANGE 0
- #define PF_FUNC_CLEAR 1
- #define PF_FUNC_SET 2
- #define PF_FUNC_TOGGLE 3

5.180.1 Detailed Description

Constants that are for sending Power Function single pin functions.

5.180.2 Define Documentation**5.180.2.1 #define PF_FUNC_CLEAR 1**

Power function single pin - clear

5.180.2.2 #define PF_FUNC_NOCHANGE 0

Power function single pin - no change

5.180.2.3 #define PF_FUNC_SET 2

Power function single pin - set

5.180.2.4 #define PF_FUNC_TOGGLE 3

Power function single pin - toggle

5.181 Power Function CST options constants

Constants that are for specifying Power Function CST options.

Defines

- #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

5.181.1 Detailed Description

Constants that are for specifying Power Function CST options.

5.181.2 Define Documentation

5.181.2.1 #define PF_CST_CLEAR1_CLEAR2 0

Power function CST clear 1 and clear 2

5.181.2.2 #define PF_CST_CLEAR1_SET2 2

Power function CST clear 1 and set 2

5.181.2.3 #define PF_CST_DECREMENT_PWM 5

Power function CST decrement PWM

5.181.2.4 #define PF_CST_FULL_FWD 6

Power function CST full forward

5.181.2.5 #define PF_CST_FULL_REV 7

Power function CST full reverse

5.181.2.6 #define PF_CST_INCREMENT_PWM 4

Power function CST increment PWM

5.181.2.7 #define PF_CST_SET1_CLEAR2 1

Power function CST set 1 and clear 2

5.181.2.8 #define PF_CST_SET1_SET2 3

Power function CST set 1 and set 2

5.181.2.9 #define PF_CST_TOGGLE_DIR 8

Power function CST toggle direction

5.182 Power Function PWM option constants

Constants that are for specifying Power Function PWM options.

Defines

- #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

5.182.1 Detailed Description

Constants that are for specifying Power Function PWM options.

5.182.2 Define Documentation**5.182.2.1 #define PF_PWM_BRAKE 8**

Power function PWM brake

5.182.2.2 #define PF_PWM_FLOAT 0

Power function PWM float

5.182.2.3 #define PF_PWM_FWD1 1

Power function PWM foward level 1

5.182.2.4 #define PF_PWM_FWD2 2

Power function PWM foward level 2

5.182.2.5 #define PF_PWM_FWD3 3

Power function PWM foward level 3

5.182.2.6 #define PF_PWM_FWD4 4

Power function PWM foward level 4

5.182.2.7 #define PF_PWM_FWD5 5

Power function PWM foward level 5

5.182.2.8 #define PF_PWM_FWD6 6

Power function PWM foward level 6

5.182.2.9 #define PF_PWM_FWD7 7

Power function PWM foward level 7

5.182.2.10 #define PF_PWM_REV1 15

Power function PWM reverse level 1

5.182.2.11 #define PF_PWM_REV2 14

Power function PWM reverse level 2

5.182.2.12 #define PF_PWM_REV3 13

Power function PWM reverse level 3

5.182.2.13 #define PF_PWM_REV4 12

Power function PWM reverse level 4

5.182.2.14 #define PF_PWM_REV5 11

Power function PWM reverse level 5

5.182.2.15 #define PF_PWM_REV6 10

Power function PWM reverse level 6

5.182.2.16 #define PF_PWM_REV7 9

Power function PWM reverse level 7

5.183 HiTechnic device constants

Constants that are for use with HiTechnic devices.

Modules

- [HiTechnic IRSeeker2 constants](#)

Constants that are for use with the HiTechnic IRSeeker2 device.

- [HiTechnic IRReceiver constants](#)

Constants that are for use with the HiTechnic IRReceiver device.

- [HiTechnic Color2 constants](#)

Constants that are for use with the HiTechnic Color2 device.

- [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 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.

Defines

- #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

5.183.1 Detailed Description

Constants that are for use with HiTechnic devices.

5.183.2 Define Documentation

5.183.2.1 #define HT_ADDR_ACCEL 0x02

HiTechnic Accel I2C address

5.183.2.2 #define HT_ADDR_ANGLE 0x02

HiTechnic Angle I2C address

5.183.2.3 #define HT_ADDR_BAROMETRIC 0x02

HiTechnic Barometric I2C address

5.183.2.4 #define HT_ADDR_COLOR 0x02

HiTechnic Color I2C address

5.183.2.5 #define HT_ADDR_COLOR2 0x02

HiTechnic Color2 I2C address

5.183.2.6 #define HT_ADDR_COMPASS 0x02

HiTechnic Compass I2C address

5.183.2.7 #define HT_ADDR_IRLINK 0x02

HiTechnic IRLink I2C address

5.183.2.8 #define HT_ADDR_IRRECEIVER 0x02

HiTechnic IRReceiver I2C address

5.183.2.9 #define HT_ADDR_IRSEEKER 0x02

HiTechnic IRSeeker I2C address

5.183.2.10 #define HT_ADDR_IRSEEKER2 0x10

HiTechnic IRSeeker2 I2C address

5.183.2.11 #define HT_ADDR_PROTOBOARD 0x02

HiTechnic Prototype board I2C address

5.183.2.12 #define HT_ADDR_SUPERPRO 0x10

HiTechnic SuperPro board I2C address

5.184 HiTechnic IRSeeker2 constants

Constants that are for use with the HiTechnic IRSeeker2 device.

Defines

- #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

5.184.1 Detailed Description

Constants that are for use with the HiTechnic IRSeeker2 device.

5.184.2 Define Documentation**5.184.2.1 #define HTIR2_MODE_1200 0**

Set IRSeeker2 to 1200 mode

5.184.2.2 #define HTIR2_MODE_600 1

Set IRSeeker2 to 600 mode

5.184.2.3 #define HTIR2_REG_AC01 0x4A

IRSeeker2 AC 01 register

5.184.2.4 #define HTIR2_REG_AC02 0x4B

IRSeeker2 AC 02 register

5.184.2.5 #define HTIR2_REG_AC03 0x4C

IRSeeker2 AC 03 register

5.184.2.6 #define HTIR2_REG_AC04 0x4D

IRSeeker2 AC 04 register

5.184.2.7 #define HTIR2_REG_AC05 0x4E

IRSeeker2 AC 05 register

5.184.2.8 #define HTIR2_REG_ACDIR 0x49

IRSeeker2 AC direction register

5.184.2.9 #define HTIR2_REG_DC01 0x43

IRSeeker2 DC 01 register

5.184.2.10 #define HTIR2_REG_DC02 0x44

IRSeeker2 DC 02 register

5.184.2.11 #define HTIR2_REG_DC03 0x45

IRSeeker2 DC 03 register

5.184.2.12 #define HTIR2_REG_DC04 0x46

IRSeeker2 DC 04 register

5.184.2.13 #define HTIR2_REG_DC05 0x47

IRSeeker2 DC 05 register

5.184.2.14 #define HTIR2_REG_DCAVG 0x48

IRSeeker2 DC average register

5.184.2.15 #define HTIR2_REG_DCDIR 0x42

IRSeeker2 DC direction register

5.184.2.16 #define HTIR2_REG_MODE 0x41

IRSeeker2 mode register

5.185 HiTechnic IRReceiver constants

Constants that are for use with the HiTechnic IRReceiver device.

Defines

- #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

5.185.1 Detailed Description

Constants that are for use with the HiTechnic IRReceiver device.

5.185.2 Define Documentation**5.185.2.1 #define HT_CH1_A 0**

Use IRReceiver channel 1 output A

5.185.2.2 #define HT_CH1_B 1

Use IRReceiver channel 1 output B

5.185.2.3 #define HT_CH2_A 2

Use IRReceiver channel 2 output A

5.185.2.4 #define HT_CH2_B 3

Use IRReceiver channel 2 output B

5.185.2.5 #define HT_CH3_A 4

Use IRReceiver channel 3 output A

5.185.2.6 #define HT_CH3_B 5

Use IRReceiver channel 3 output B

5.185.2.7 #define HT_CH4_A 6

Use IRReceiver channel 4 output A

5.185.2.8 #define HT_CH4_B 7

Use IRReceiver channel 4 output B

5.186 HiTechnic Color2 constants

Constants that are for use with the HiTechnic Color2 device.

Defines

- #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

5.186.1 Detailed Description

Constants that are for use with the HiTechnic Color2 device.

5.186.2 Define Documentation**5.186.2.1 #define HT_CMD_COLOR2_50HZ 0x35**

Set the Color2 sensor to 50Hz mode

5.186.2.2 #define HT_CMD_COLOR2_60HZ 0x36

Set the Color2 sensor to 60Hz mode

5.186.2.3 #define HT_CMD_COLOR2_ACTIVE 0x00

Set the Color2 sensor to active mode

5.186.2.4 #define HT_CMD_COLOR2_BLCAL 0x42

Set the Color2 sensor to black level calibration mode

5.186.2.5 #define HT_CMD_COLOR2_FAR 0x46

Set the Color2 sensor to far mode

5.186.2.6 #define HT_CMD_COLOR2_LED_HI 0x48

Set the Color2 sensor to LED high mode

5.186.2.7 #define HT_CMD_COLOR2_LED_LOW 0x4C

Set the Color2 sensor to LED low mode

5.186.2.8 #define HT_CMD_COLOR2_NEAR 0x4E

Set the Color2 sensor to near mode

5.186.2.9 #define HT_CMD_COLOR2_PASSIVE 0x01

Set the Color2 sensor to passive mode

5.186.2.10 #define HT_CMD_COLOR2_RAW 0x03

Set the Color2 sensor to raw mode

5.186.2.11 #define HT_CMD_COLOR2_WBCAL 0x43

Set the Color2 sensor to white level calibration mode

5.187 HiTechnic Angle sensor constants

Constants that are for use with the HiTechnic Angle sensor device.

Defines

- #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

5.187.1 Detailed Description

Constants that are for use with the HiTechnic Angle sensor device.

5.187.2 Define Documentation**5.187.2.1 #define HTANGLE_MODE_CALIBRATE 0x43**

Resets 0 degree position to current shaft angle

5.187.2.2 #define HTANGLE_MODE_NORMAL 0x00

Normal angle measurement mode

5.187.2.3 #define HTANGLE_MODE_RESET 0x52

Resets the accumulated angle

5.187.2.4 #define HTANGLE_REG_ACDIR 0x49

Angle 16 bit revolutions per minute, low byte register

5.187.2.5 #define HTANGLE_REG_DC01 0x43

Angle current angle (1 degree adder) register

5.187.2.6 #define HTANGLE_REG_DC02 0x44

Angle 32 bit accumulated angle, high byte register

5.187.2.7 #define HTANGLE_REG_DC03 0x45

Angle 32 bit accumulated angle, mid byte register

5.187.2.8 #define HTANGLE_REG_DC04 0x46

Angle 32 bit accumulated angle, mid byte register

5.187.2.9 #define HTANGLE_REG_DC05 0x47

Angle 32 bit accumulated angle, low byte register

5.187.2.10 #define HTANGLE_REG_DCAVG 0x48

Angle 16 bit revolutions per minute, high byte register

5.187.2.11 #define HTANGLE_REG_DCDIR 0x42

Angle current angle (2 degree increments) register

5.187.2.12 #define HTANGLE_REG_MODE 0x41

Angle mode register

5.188 HiTechnic Barometric sensor constants

Constants that are for use with the HiTechnic Barometric sensor device.

Defines

- #define HTBAR_REG_COMMAND 0x40
- #define HTBAR_REG_TEMPERATURE 0x42
- #define HTBAR_REG_PRESSURE 0x44
- #define HTBAR_REG_CALIBRATION 0x46

5.188.1 Detailed Description

Constants that are for use with the HiTechnic Barometric sensor device.

5.188.2 Define Documentation

5.188.2.1 #define HTBAR_REG_CALIBRATION 0x46

Barometric sensor calibration register (2 bytes msb/lsb)

5.188.2.2 #define HTBAR_REG_COMMAND 0x40

Barometric sensor command register

5.188.2.3 #define HTBAR_REG_PRESSURE 0x44

Barometric sensor pressure register (2 bytes msb/lsb)

5.188.2.4 #define HTBAR_REG_TEMPERATURE 0x42

Barometric sensor temperature register (2 bytes msb/lsb)

5.189 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.

Defines

- #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

5.189.1 Detailed Description

Constants that are for use with the HiTechnic Prototype board.

5.189.2 Define Documentation

5.189.2.1 #define HTProto_REG_A0 0x42

Prototype board analog 0 register (2 bytes msb/lsb)

5.189.2.2 #define HTProto_REG_A1 0x44

Prototype board analog 1 register (2 bytes msb/lsb)

5.189.2.3 #define HTProto_REG_A2 0x46

Prototype board analog 2 register (2 bytes msb/lsb)

5.189.2.4 #define HTProto_REG_A3 0x48

Prototype board analog 3 register (2 bytes msb/lsb)

5.189.2.5 #define HTPROTO_REG_A4 0x4A

Prototype board analog 4 register (2 bytes msb/lsb)

5.189.2.6 #define HTPROTO_REG_DCTRL 0x4E

Prototype board digital pin control register (6 bits)

5.189.2.7 #define HTPROTO_REG_DIN 0x4C

Prototype board digital pin input register (6 bits)

5.189.2.8 #define HTPROTO_REG_DOUT 0x4D

Prototype board digital pin output register (6 bits)

5.189.2.9 #define HTPROTO_REG_SRATE 0x4F

Prototype board sample rate register

5.190 HiTechnic Prototype board analog input constants

Constants that are for use with reading the HiTechnic Prototype board analog input values.

Defines

- #define HTPROTO_A0 0x42
- #define HTPROTO_A1 0x44
- #define HTPROTO_A2 0x46
- #define HTPROTO_A3 0x48
- #define HTPROTO_A4 0x4A

5.190.1 Detailed Description

Constants that are for use with reading the HiTechnic Prototype board analog input values.

5.190.2 Define Documentation**5.190.2.1 #define HTPROTO_A0 0x42**

Read Prototype board analog input 0

5.190.2.2 #define HTPROTO_A1 0x44

Read Prototype board analog input 1

5.190.2.3 #define HTPROTO_A2 0x46

Read Prototype board analog input 2

5.190.2.4 #define HTPROTO_A3 0x48

Read Prototype board analog input 3

5.190.2.5 #define HTPROTO_A4 0x4A

Read Prototype board analog input 4

5.191 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 analog output mode constants](#)

Constants for controlling the 2 analog output modes.

- SuperPro digital pin constants

Constants for controlling the 8 digital pins.

- SuperPro Strobe control constants

Constants for manipulating the six digital strobe outputs.

Defines

- #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

5.191.1 Detailed Description

Constants that are for use with the HiTechnic SuperPro board.

5.191.2 Define Documentation

5.191.2.1 #define HTSPRO_REG_A0 0x42

SuperPro analog 0 register (10 bits)

5.191.2.2 #define HTSPRO_REG_A1 0x44

SuperPro analog 1 register (10 bits)

5.191.2.3 #define HTSPRO_REG_A2 0x46

SuperPro analog 2 register (10 bits)

5.191.2.4 #define HTSPRO_REG_A3 0x48

SuperPro analog 3 register (10 bits)

5.191.2.5 #define HTSPRO_REG_CTRL 0x40

SuperPro program control register

5.191.2.6 #define HTSPRO_REG_DAC0_FREQ 0x53

SuperPro analog output 0 frequency register (2 bytes msb/lsb)

5.191.2.7 #define HTSPRO_REG_DAC0_MODE 0x52

SuperPro analog output 0 mode register

5.191.2.8 #define HTSPRO_REG_DAC0_VOLTAGE 0x55

SuperPro analog output 0 voltage register (10 bits)

5.191.2.9 #define HTSPRO_REG_DAC1_FREQ 0x58

SuperPro analog output 1 frequency register (2 bytes msb/lsb)

5.191.2.10 #define HTSPRO_REG_DAC1_MODE 0x57

SuperPro analog output 1 mode register

5.191.2.11 #define HTSPRO_REG_DAC1_VOLTAGE 0x5A

SuperPro analog output 1 voltage register (10 bits)

5.191.2.12 #define HTSPRO_REG_DCTRL 0x4E

SuperPro digital pin control register (8 bits)

5.191.2.13 #define HTSPRO_REG_DIN 0x4C

SuperPro digital pin input register (8 bits)

5.191.2.14 #define HTSPRO_REG_DLADDRESS 0x60

SuperPro download address register (2 bytes msb/lsb)

5.191.2.15 #define HTSPRO_REG_DLCKSUM 0x6A

SuperPro download checksum register

5.191.2.16 #define HTSPRO_REG_DLCONTROL 0x6B

SuperPro download control register

5.191.2.17 #define HTSPRO_REG_DLDATA 0x62

SuperPro download data register (8 bytes)

5.191.2.18 #define HTSPRO_REG_DOUT 0x4D

SuperPro digital pin output register (8 bits)

5.191.2.19 #define HTSPRO_REG_LED 0x51

SuperPro LED control register

5.191.2.20 #define HTSPRO_REG_MEMORY_20 0x80

SuperPro memory address 0x20 register (4 bytes msb/lsb)

5.191.2.21 #define HTSPRO_REG_MEMORY_21 0x84

SuperPro memory address 0x21 register (4 bytes msb/lsb)

5.191.2.22 #define HTSPRO_REG_MEMORY_22 0x88

SuperPro memory address 0x22 register (4 bytes msb/lsb)

5.191.2.23 #define HTSPRO_REG_MEMORY_23 0x8C

SuperPro memory address 0x23 register (4 bytes msb/lsb)

5.191.2.24 #define HTSPRO_REG_MEMORY_24 0x90

SuperPro memory address 0x24 register (4 bytes msb/lsb)

5.191.2.25 #define HTSPRO_REG_MEMORY_25 0x94

SuperPro memory address 0x25 register (4 bytes msb/lsb)

5.191.2.26 #define HTSPRO_REG_MEMORY_26 0x98

SuperPro memory address 0x26 register (4 bytes msb/lsb)

5.191.2.27 #define HTSPRO_REG_MEMORY_27 0x9C

SuperPro memory address 0x27 register (4 bytes msb/lsb)

5.191.2.28 #define HTSPRO_REG_MEMORY_28 0xA0

SuperPro memory address 0x28 register (4 bytes msb/lsb)

5.191.2.29 #define HTSPRO_REG_MEMORY_29 0xA4

SuperPro memory address 0x29 register (4 bytes msb/lsb)

5.191.2.30 #define HTSPRO_REG_MEMORY_2A 0xA8

SuperPro memory address 0x2A register (4 bytes msb/lsb)

5.191.2.31 #define HTSPRO_REG_MEMORY_2B 0xAC

SuperPro memory address 0x2B register (4 bytes msb/lsb)

5.191.2.32 #define HTSPRO_REG_MEMORY_2C 0xB0

SuperPro memory address 0x2C register (4 bytes msb/lsb)

5.191.2.33 #define HTSPRO_REG_MEMORY_2D 0xB4

SuperPro memory address 0x2D register (4 bytes msb/lsb)

5.191.2.34 #define HTSPRO_REG_MEMORY_2E 0xB8

SuperPro memory address 0x2E register (4 bytes msb/lsb)

5.191.2.35 #define HTSPRO_REG_MEMORY_2F 0xBC

SuperPro memory address 0x2F register (4 bytes msb/lsb)

5.191.2.36 #define HTSPRO_REG_MEMORY_30 0xC0

SuperPro memory address 0x30 register (4 bytes msb/lsb)

5.191.2.37 #define HTSPRO_REG_MEMORY_31 0xC4

SuperPro memory address 0x31 register (4 bytes msb/lsb)

5.191.2.38 #define HTSPRO_REG_MEMORY_32 0xC8

SuperPro memory address 0x32 register (4 bytes msb/lsb)

5.191.2.39 #define HTSPRO_REG_MEMORY_33 0xCC

SuperPro memory address 0x33 register (4 bytes msb/lsb)

5.191.2.40 #define HTSPRO_REG_MEMORY_34 0xD0

SuperPro memory address 0x34 register (4 bytes msb/lsb)

5.191.2.41 #define HTSPRO_REG_MEMORY_35 0xD4

SuperPro memory address 0x35 register (4 bytes msb/lsb)

5.191.2.42 #define HTSPRO_REG_MEMORY_36 0xD8

SuperPro memory address 0x36 register (4 bytes msb/lsb)

5.191.2.43 #define HTSPRO_REG_MEMORY_37 0xDC

SuperPro memory address 0x37 register (4 bytes msb/lsb)

5.191.2.44 #define HTSPRO_REG_MEMORY_38 0xE0

SuperPro memory address 0x38 register (4 bytes msb/lsb)

5.191.2.45 #define HTSPRO_REG_MEMORY_39 0xE4

SuperPro memory address 0x39 register (4 bytes msb/lsb)

5.191.2.46 #define HTSPRO_REG_MEMORY_3A 0xE8

SuperPro memory address 0x3A register (4 bytes msb/lsb)

5.191.2.47 #define HTSPRO_REG_MEMORY_3B 0xEC

SuperPro memory address 0x3B register (4 bytes msb/lsb)

5.191.2.48 #define HTSPRO_REG_MEMORY_3C 0xF0

SuperPro memory address 0x3C register (4 bytes msb/lsb)

5.191.2.49 #define HTSPRO_REG_MEMORY_3D 0xF4

SuperPro memory address 0x3D register (4 bytes msb/lsb)

5.191.2.50 #define HTSPRO_REG_MEMORY_3E 0xF8

SuperPro memory address 0x3E register (4 bytes msb/lsb)

5.191.2.51 #define HTSPRO_REG_MEMORY_3F 0xFC

SuperPro memory address 0x3F register (4 bytes msb/lsb)

5.191.2.52 #define HTSPRO_REG_STROBE 0x50

SuperPro strobe control register

5.192 HiTechnic SuperPro analog input index constants

Constants that are for use with reading the HiTechnic SuperPro analog input values.

Defines

- #define HTSPRO_A0 0x42
- #define HTSPRO_A1 0x44
- #define HTSPRO_A2 0x46
- #define HTSPRO_A3 0x48

5.192.1 Detailed Description

Constants that are for use with reading the HiTechnic SuperPro analog input values.

5.192.2 Define Documentation**5.192.2.1 #define HTSPRO_A0 0x42**

Read SuperPro analog input 0

5.192.2.2 #define HTSPRO_A1 0x44

Read SuperPro analog input 1

5.192.2.3 #define HTSPRO_A2 0x46

Read SuperPro analog input 2

5.192.2.4 #define HTSPRO_A3 0x48

Read SuperPro analog input 3

5.193 HiTechnic SuperPro analog output index constants

Constants that are for use with configuraing the HiTechnic SuperPro analog outputs.

Defines

- #define HTSPRO_DAC0 0x52
- #define HTSPRO_DAC1 0x57

5.193.1 Detailed Description

Constants that are for use with configuraing the HiTechnic SuperPro analog outputs.

5.193.2 Define Documentation

5.193.2.1 `#define HTSPRO_DAC0 0x52`

Set SuperPro analog output 0 configuration

5.193.2.2 `#define HTSPRO_DAC1 0x57`

Set SuperPro analog output 1 configuration

5.194 MindSensors device constants

Constants that are for use with MindSensors devices.

Modules

- [MindSensors DIST-Nx constants](#)

Constants that are for use with the MindSensors DIST-Nx 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.

- [MindSensors ACCL-Nx constants](#)

Constants that are for use with the MindSensors ACCL-Nx device.

- [MindSensors PFmate constants](#)

Constants that are for use with the MindSensors PFmate device.

- [MindSensors NXTservo constants](#)

Constants that are for use with the MindSensors NXTservo device.

- [MindSensors NXTHID constants](#)

Constants that are for use with the MindSensors NXTHID device.

- [MindSensors NXTPowerMeter constants](#)

Constants that are for use with the MindSensors NXTPowerMeter device.

- [MindSensors NXTSumoEyes constants](#)

Constants that are for use with the MindSensors NXTSumoEyes device.

- [MindSensors NXTLineLeader constants](#)

Constants that are for use with the MindSensors NXTLineLeader device.

Defines

- #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_LINEADR](#) 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

5.194.1 Detailed Description

Constants that are for use with MindSensors devices.

5.194.2 Define Documentation

5.194.2.1 #define [MS_ADDR_ACCLNX](#) 0x02

MindSensors ACCL-Nx I2C address

5.194.2.2 #define MS_ADDR_CMPSNX 0x02

MindSensors CMPS-Nx I2C address

5.194.2.3 #define MS_ADDR_DISTNX 0x02

MindSensors DIST-Nx I2C address

5.194.2.4 #define MS_ADDR_IVSENS 0x12

MindSensors IVSens (NXTPowerMeter) I2C address

5.194.2.5 #define MS_ADDR_LINELDR 0x02

MindSensors LineLdr I2C address

5.194.2.6 #define MS_ADDR_MTRMUX 0xB4

MindSensors MTRMux I2C address

5.194.2.7 #define MS_ADDR_NRLINK 0x02

MindSensors NRLink I2C address

5.194.2.8 #define MS_ADDR_NXTCAM 0x02

MindSensors NXTCam I2C address

5.194.2.9 #define MS_ADDR_NXTHID 0x04

MindSensors NXTHID I2C address

5.194.2.10 #define MS_ADDR_NXTMMX 0x06

MindSensors NXTMMX I2C address

5.194.2.11 #define MS_ADDR_NXTSERVO 0xB0

MindSensors NXTServo I2C address

5.194.2.12 #define MS_ADDR_NXTSERVO_EM 0x40

MindSensors NXTServo in edit macro mode I2C address

5.194.2.13 #define MS_ADDR_PFMATE 0x48

MindSensors PFMate I2C address

5.194.2.14 #define MS_ADDR_PSPNX 0x02

MindSensors PSP-Nx I2C address

5.194.2.15 #define MS_ADDR_RTCLOCK 0xD0

MindSensors RTClock I2C address

5.194.2.16 #define MS_ADDR_RXMUX 0x7E

MindSensors RXMux I2C address

5.194.2.17 #define MS_CMD_ADPA_OFF 0x4F

Turn MindSensors ADPA mode off

5.194.2.18 #define MS_CMD_ADPA_ON 0x4E

Turn MindSensors ADPA mode on

5.194.2.19 #define MS_CMD_DEENERGIZED 0x44

De-energize the MindSensors device

5.194.2.20 #define MS_CMD_ENERGIZED 0x45

Energize the MindSensors device

5.195 MindSensors DIST-Nx constants

Constants that are for use with the MindSensors DIST-Nx device.

Defines

- #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

5.195.1 Detailed Description

Constants that are for use with the MindSensors DIST-Nx device.

5.195.2 Define Documentation**5.195.2.1 #define DIST_CMD_CUSTOM 0x35**

Set the DIST-Nx to a custom mode

5.195.2.2 #define DIST_CMD_GP2D12 0x31

Set the DIST-Nx to GP2D12 mode

5.195.2.3 #define DIST_CMD_GP2D120 0x32

Set the DIST-Nx to GP2D120 mode

5.195.2.4 #define DIST_CMD_GP2YA02 0x34

Set the DIST-Nx to GP2YA02 mode

5.195.2.5 #define DIST_CMD_GP2YA21 0x33

Set the DIST-Nx to GP2YA21 mode

5.195.2.6 #define DIST_REG_DIST 0x42

The DIST-Nx distance register

5.195.2.7 #define DIST_REG_DIST1 0x58

The DIST-Nx distance 1 register

5.195.2.8 #define DIST_REG_DIST_MAX 0x54

The DIST-Nx maximum distance register

5.195.2.9 #define DIST_REG_DIST_MIN 0x52

The DIST-Nx minimum distance register

5.195.2.10 #define DIST_REG_MODULE_TYPE 0x50

The DIST-Nx module type register

5.195.2.11 #define DIST_REG_NUM_POINTS 0x51

The DIST-Nx number of data points in Custom curve register

5.195.2.12 #define DIST_REG_VOLT 0x44

The DIST-Nx voltage register

5.195.2.13 #define DIST_REG_VOLT1 0x56

The DIST-Nx voltage 1 register

5.196 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.

Defines

- `#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`

5.196.1 Detailed Description

Constants that are for use with the MindSensors PSP-Nx device.

5.196.2 Define Documentation

5.196.2.1 `#define PSP_CMD_ANALOG 0x73`

Set the PSP-Nx to analog mode

5.196.2.2 `#define PSP_CMD_DIGITAL 0x41`

Set the PSP-Nx to digital mode

5.196.2.3 `#define PSP_REG_BTNSET1 0x42`

The PSP-Nx button set 1 register

5.196.2.4 `#define PSP_REG_BTNSET2 0x43`

The PSP-Nx button set 2 register

5.196.2.5 #define PSP_REG_XLEFT 0x44

The PSP-Nx X left register

5.196.2.6 #define PSP_REG_XRIGHT 0x46

The PSP-Nx X right register

5.196.2.7 #define PSP_REG_YLEFT 0x45

The PSP-Nx Y left register

5.196.2.8 #define PSP_REG_YRIGHT 0x47

The PSP-Nx Y right register

5.197 MindSensors PSP-Nx button set 1 constants

Constants that are for interpreting MindSensors PSP-Nx button set 1 values.

Defines

- #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

5.197.1 Detailed Description

Constants that are for interpreting MindSensors PSP-Nx button set 1 values.

5.197.2 Define Documentation**5.197.2.1 #define PSP_BTNSET1_DOWN 0x40**

The PSP-Nx button set 1 down arrow

5.197.2.2 #define PSP_BTNSET1_L3 0x02

The PSP-Nx button set 1 L3

5.197.2.3 #define PSP_BTNSET1_LEFT 0x80

The PSP-Nx button set 1 left arrow

5.197.2.4 #define PSP_BTNSET1_R3 0x04

The PSP-Nx button set 1 R3

5.197.2.5 #define PSP_BTNSET1_RIGHT 0x20

The PSP-Nx button set 1 right arrow

5.197.2.6 #define PSP_BTNSET1_SELECT 0x01

The PSP-Nx button set 1 select

5.197.2.7 #define PSP_BTNSET1_START 0x08

The PSP-Nx button set 1 start

5.197.2.8 #define PSP_BTNSET1_UP 0x10

The PSP-Nx button set 1 up arrow

5.198 MindSensors PSP-Nx button set 2 constants

Constants that are for interpreting MindSensors PSP-Nx button set 2 values.

Defines

- #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

5.198.1 Detailed Description

Constants that are for interpreting MindSensors PSP-Nx button set 2 values.

5.198.2 Define Documentation

5.198.2.1 #define PSP_BTNSET2_CIRCLE 0x20

The PSP-Nx button set 2 circle

5.198.2.2 #define PSP_BTNSET2_CROSS 0x40

The PSP-Nx button set 2 cross

5.198.2.3 #define PSP_BTNSET2_L1 0x04

The PSP-Nx button set 2 L1

5.198.2.4 #define PSP_BTNSET2_L2 0x01

The PSP-Nx button set 2 L2

5.198.2.5 #define PSP_BTNSET2_R1 0x08

The PSP-Nx button set 2 R1

5.198.2.6 #define PSP_BTNSET2_R2 0x02

The PSP-Nx button set 2 R2

5.198.2.7 #define PSP_BTNSET2_SQUARE 0x80

The PSP-Nx button set 2 square

5.198.2.8 #define PSP_BTNSET2_TRIANGLE 0x10

The PSP-Nx button set 2 triangle

5.199 MindSensors nRLink constants

Constants that are for use with the MindSensors nRLink device.

Defines

- #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

5.199.1 Detailed Description

Constants that are for use with the MindSensors nRLink device.

5.199.2 Define Documentation**5.199.2.1 #define NRLINK_CMD_2400 0x44**

Set NRLink to 2400 baud

5.199.2.2 #define NRLINK_CMD_4800 0x48

Set NRLink to 4800 baud

5.199.2.3 #define NRLINK_CMD_FLUSH 0x46

Flush the NRLink

5.199.2.4 #define NRLINK_CMD_IR_LONG 0x4C

Set the NRLink to long range IR

5.199.2.5 #define NRLINK_CMD_IR_SHORT 0x53

Set the NRLink to short range IR

5.199.2.6 #define NRLINK_CMD_RUN_MACRO 0x52

Run an NRLink macro

5.199.2.7 #define NRLINK_CMD_SET_PF 0x50

Set the NRLink to Power Function mode

5.199.2.8 #define NRLINK_CMD_SET_RCX 0x58

Set the NRLink to RCX mode

5.199.2.9 #define NRLINK_CMD_SET_TRAIN 0x54

Set the NRLink to IR Train mode

5.199.2.10 #define NRLINK_CMD_TX_RAW 0x55

Set the NRLink to transmit raw bytes

5.199.2.11 #define NRLINK_REG_BYTES 0x40

The NRLink bytes register

5.199.2.12 #define NRLINK_REG_DATA 0x42

The NRLink data register

5.199.2.13 #define NRLINK_REG_EEPROM 0x50

The NRLink eeprom register

5.200 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.

Defines

- #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

5.200.1 Detailed Description

Constants that are for use with the MindSensors ACCL-Nx device.

5.200.2 Define Documentation

5.200.2.1 #define ACCL_CMD_RESET_CAL 0x52

Reset to factory calibration

5.200.2.2 #define ACCL_CMD_X_CAL 0x58

Acquire X-axis calibration point

5.200.2.3 #define ACCL_CMD_X_CAL_END 0x78

Acquire X-axis calibration point and end calibration

5.200.2.4 #define ACCL_CMD_Y_CAL 0x59

Acquire Y-axis calibration point

5.200.2.5 #define ACCL_CMD_Y_CAL_END 0x79

Acquire Y-axis calibration point and end calibration

5.200.2.6 #define ACCL_CMD_Z_CAL 0x5a

Acquire Z-axis calibration point

5.200.2.7 #define ACCL_CMD_Z_CAL_END 0x7a

Acquire Z-axis calibration point and end calibration

5.200.2.8 #define ACCL_REG_SENS_LVL 0x19

The current sensitivity

5.200.2.9 #define ACCL_REG_X_ACCEL 0x45

The X-axis acceleration data

5.200.2.10 #define ACCL_REG_X_OFFSET 0x4b

The X-axis offset

5.200.2.11 #define ACCL_REG_X_RANGE 0x4d

The X-axis range

5.200.2.12 #define ACCL_REG_X_TILT 0x42

The X-axis tilt data

5.200.2.13 #define ACCL_REG_Y_ACCEL 0x47

The Y-axis acceleration data

5.200.2.14 #define ACCL_REG_Y_OFFSET 0x4f

The Y-axis offset

5.200.2.15 #define ACCL_REG_Y_RANGE 0x51

The Y-axis range

5.200.2.16 #define ACCL_REG_Y_TILT 0x43

The Y-axis tilt data

5.200.2.17 #define ACCL_REG_Z_ACCEL 0x49

The Z-axis acceleration data

5.200.2.18 #define ACCL_REG_Z_OFFSET 0x53

The Z-axis offset

5.200.2.19 #define ACCL_REG_Z_RANGE 0x55

The Z-axis range

5.200.2.20 #define ACCL_REG_Z_TILT 0x44

The Z-axis tilt data

5.201 MindSensors ACCL-Nx sensitivity level constants

Constants that are for setting the MindSensors ACCL-Nx sensitivity level.

Defines

- #define ACCL_SENSITIVITY_LEVEL_1 0x31
- #define ACCL_SENSITIVITY_LEVEL_2 0x32
- #define ACCL_SENSITIVITY_LEVEL_3 0x33
- #define ACCL_SENSITIVITY_LEVEL_4 0x34

5.201.1 Detailed Description

Constants that are for setting the MindSensors ACCL-Nx sensitivity level.

5.201.2 Define Documentation**5.201.2.1 #define ACCL_SENSITIVITY_LEVEL_1 0x31**

The ACCL-Nx sensitivity level 1

5.201.2.2 #define ACCL_SENSITIVITY_LEVEL_2 0x32

The ACCL-Nx sensitivity level 2

5.201.2.3 #define ACCL_SENSITIVITY_LEVEL_3 0x33

The ACCL-Nx sensitivity level 3

5.201.2.4 #define ACCL_SENSITIVITY_LEVEL_4 0x34

The ACCL-Nx sensitivity level 4

5.202 MindSensors PFMate constants

Constants that are for use with the MindSensors PFMate device.

Modules

- [PFMate motor constants](#)

Constants that are for specifying PFMate motors.

- [PFMate channel constants](#)

Constants that are for specifying PFMate channels.

Defines

- #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

5.202.1 Detailed Description

Constants that are for use with the MindSensors PFmate device.

5.202.2 Define Documentation**5.202.2.1 #define PFMATE_CMD_GO 0x47**

Send IR signal to IR receiver

5.202.2.2 #define PFMATE_CMD_RAW 0x52

Send raw IR signal to IR receiver

5.202.2.3 #define PFMATE_REG_A_CMD 0x44

PF command for motor A? (PF_CMD_FLOAT, PF_CMD_FWD, PF_CMD_REV,
PF_CMD_BRAKE)

5.202.2.4 #define PFMATE_REG_A_SPEED 0x45

PF speed for motor A? (0-7)

5.202.2.5 #define PFMATE_REG_B_CMD 0x46

PF command for motor B? (PF_CMD_FLOAT, PF_CMD_FWD, PF_CMD_REV,
PF_CMD_BRAKE)

5.202.2.6 #define PFMATE_REG_B_SPEED 0x47

PF speed for motor B? (0-7)

5.202.2.7 #define PFMATE_REG_CHANNEL 0x42

PF channel? 1, 2, 3, or 4

5.202.2.8 #define PFMATE_REG_CMD 0x41

PFMate command

5.202.2.9 #define PFMATE_REG_MOTORS 0x43

PF motors? (0 = both, 1 = A, 2 = B)

5.203 PFMate motor constants

Constants that are for specifying PFMate motors.

Defines

- #define PFMATE_MOTORS_BOTH 0x00
- #define PFMATE_MOTORS_A 0x01
- #define PFMATE_MOTORS_B 0x02

5.203.1 Detailed Description

Constants that are for specifying PFMate motors.

5.203.2 Define Documentation**5.203.2.1 #define PFMATE_MOTORS_A 0x01**

Control only motor A

5.203.2.2 #define PFMATE_MOTORS_B 0x02

Control only motor B

5.203.2.3 #define PFMATE_MOTORS_BOTH 0x00

Control both motors

5.204 PFMate channel constants

Constants that are for specifying PFMate channels.

Defines

- #define PFMATE_CHANNEL_1 1
- #define PFMATE_CHANNEL_2 2
- #define PFMATE_CHANNEL_3 3
- #define PFMATE_CHANNEL_4 4

5.204.1 Detailed Description

Constants that are for specifying PFMate channels.

5.204.2 Define Documentation**5.204.2.1 #define PFMATE_CHANNEL_1 1**

Power function channel 1

5.204.2.2 #define PFMATE_CHANNEL_2 2

Power function channel 2

5.204.2.3 #define PFMATE_CHANNEL_3 3

Power function channel 3

5.204.2.4 #define PFMATE_CHANNEL_4 4

Power function channel 4

5.205 MindSensors NXTServo constants

Constants that are for use with the MindSensors NXTServo device.

Modules

- [MindSensors NXTServo registers](#)
NXTServo device register constants.
- [MindSensors NXTServo position constants](#)
NXTServo device position constants.
- [MindSensors NXTServo quick position constants](#)
NXTServo device quick position constants.
- [MindSensors NXTServo servo numbers](#)
NXTServo device servo number constants.
- [MindSensors NXTServo commands](#)
NXTServo device command constants.

5.205.1 Detailed Description

Constants that are for use with the MindSensors NXTServo device.

5.206 MindSensors NXTServo registers

NXTServo device register constants.

Defines

- #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

5.206.1 Detailed Description

NXTServo device register constants.

5.206.2 Define Documentation

5.206.2.1 #define NXTSERVO_EM_REG_CMD 0x00

NXTServo in macro edit mode command register.

5.206.2.2 #define NXTSERVO_EM_REG_EEPROM_END 0xFF

NXTServo in macro edit mode EEPROM end register.

5.206.2.3 #define NXTSERVO_EM_REG_EEPROM_START 0x21

NXTServo in macro edit mode EEPROM start register.

5.206.2.4 #define NXTSERVO_REG_CMD 0x41

NXTServo command register. See [MindSensors NXTServo commands](#) group. (write only)

5.206.2.5 #define NXTSERVO_REG_S1_POS 0x42

NXTServo servo 1 position register.

5.206.2.6 #define NXTSERVO_REG_S1_QPOS 0x5A

NXTServo servo 1 quick position register. (write only)

5.206.2.7 #define NXTSERVO_REG_S1_SPEED 0x52

NXTServo servo 1 speed register.

5.206.2.8 #define NXTSERVO_REG_S2_POS 0x44

NXTServo servo 2 position register.

5.206.2.9 #define NXTSERVO_REG_S2_QPOS 0x5B

NXTServo servo 2 quick position register. (write only)

5.206.2.10 #define NXTSERVO_REG_S2_SPEED 0x53

NXTServo servo 2 speed register.

5.206.2.11 #define NXTSERVO_REG_S3_POS 0x46

NXTServo servo 3 position register.

5.206.2.12 #define NXTSERVO_REG_S3_QPOS 0x5C

NXTServo servo 3 quick position register. (write only)

5.206.2.13 #define NXTSERVO_REG_S3_SPEED 0x54

NXTServo servo 3 speed register.

5.206.2.14 #define NXTSERVO_REG_S4_POS 0x48

NXTServo servo 4 position register.

5.206.2.15 #define NXTSERVO_REG_S4_QPOS 0x5D

NXTServo servo 4 quick position register. (write only)

5.206.2.16 #define NXTSERVO_REG_S4_SPEED 0x55

NXTServo servo 4 speed register.

5.206.2.17 #define NXTSERVO_REG_S5_POS 0x4A

NXTServo servo 5 position register.

5.206.2.18 #define NXTSERVO_REG_S5_QPOS 0x5E

NXTServo servo 5 quick position register. (write only)

5.206.2.19 #define NXTSERVO_REG_S5_SPEED 0x56

NXTServo servo 5 speed register.

5.206.2.20 #define NXTSERVO_REG_S6_POS 0x4C

NXTServo servo 6 position register.

5.206.2.21 #define NXTSERVO_REG_S6_QPOS 0x5F

NXTServo servo 6 quick position register. (write only)

5.206.2.22 #define NXTSERVO_REG_S6_SPEED 0x57

NXTServo servo 6 speed register.

5.206.2.23 #define NXTSERVO_REG_S7_POS 0x4E

NXTServo servo 7 position register.

5.206.2.24 #define NXTSERVO_REG_S7_QPOS 0x60

NXTServo servo 7 quick position register. (write only)

5.206.2.25 #define NXTSERVO_REG_S7_SPEED 0x58

NXTServo servo 7 speed register.

5.206.2.26 #define NXTSERVO_REG_S8_POS 0x50

NXTServo servo 8 position register.

5.206.2.27 #define NXTSERVO_REG_S8_QPOS 0x61

NXTServo servo 8 quick position register. (write only)

5.206.2.28 #define NXTSERVO_REG_S8_SPEED 0x59

NXTServo servo 8 speed register.

5.206.2.29 #define NXTSERVO_REG_VOLTAGE 0x41

Battery voltage register. (read only)

5.207 MindSensors NXTServo position constants

NXTServo device position constants.

Defines

- #define [NXTSERVO_POS_CENTER](#) 1500
- #define [NXTSERVO_POS_MIN](#) 500
- #define [NXTSERVO_POS_MAX](#) 2500

5.207.1 Detailed Description

NXTServo device position constants.

5.207.2 Define Documentation**5.207.2.1 #define NXTSERVO_POS_CENTER 1500**

Center position for 1500us servos.

5.207.2.2 #define NXTSERVO_POS_MAX 2500

Maximum position for 1500us servos.

5.207.2.3 #define NXTSERVO_POS_MIN 500

Minimum position for 1500us servos.

5.208 MindSensors NXTServo quick position constants

NXTServo device quick position constants.

Defines

- #define [NXTSERVO_QPOS_CENTER](#) 150
- #define [NXTSERVO_QPOS_MIN](#) 50
- #define [NXTSERVO_QPOS_MAX](#) 250

5.208.1 Detailed Description

NXTServo device quick position constants.

5.208.2 Define Documentation**5.208.2.1 #define NXTSERVO_QPOS_CENTER 150**

Center quick position for 1500us servos.

5.208.2.2 #define NXTSERVO_QPOS_MAX 250

Maximum quick position for 1500us servos.

5.208.2.3 #define NXTSERVO_QPOS_MIN 50

Minimum quick position for 1500us servos.

5.209 MindSensors NXTServo servo numbers

NXTServo device servo number constants.

Defines

- #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

5.209.1 Detailed Description

NXTServo device servo number constants.

5.209.2 Define Documentation**5.209.2.1 #define NXTSERVO_SERVO_1 0**

NXTServo server number 1.

5.209.2.2 #define NXTSERVO_SERVO_2 1

NXTServo server number 2.

5.209.2.3 #define NXTSERVO_SERVO_3 2

NXTServo server number 3.

5.209.2.4 #define NXTSERVO_SERVO_4 3

NXTServo server number 4.

5.209.2.5 #define NXTSERVO_SERVO_5 4

NXTServo server number 5.

5.209.2.6 #define NXTSERVO_SERVO_6 5

NXTServo server number 6.

5.209.2.7 #define NXTSERVO_SERVO_7 6

NXTServo server number 7.

5.209.2.8 #define NXTSERVO_SERVO_8 7

NXTServo server number 8.

5.210 MindSensors NXTServo commands

NXTServo device command constants.

Defines

- #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

5.210.1 Detailed Description

NXTServo device command constants. These are written to the command register to control the device.

5.210.2 Define Documentation**5.210.2.1 #define NXTSERVO_CMD_EDIT1 0x45**

Edit Macro (part 1 of 2 character command sequence)

5.210.2.2 #define NXTSERVO_CMD_EDIT2 0x4D

Edit Macro (part 2 of 2 character command sequence)

5.210.2.3 #define NXTSERVO_CMD_GOTO 0x47

Goto EEPROM position x. This command re-initializes the macro environment.

5.210.2.4 #define NXTSERVO_CMD_HALT 0x48

Halt Macro. This command re-initializes the macro environment.

5.210.2.5 #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.

5.210.2.6 #define NXTSERVO_CMD_PAUSE 0x50

Pause Macro. This command will pause the macro, and save the environment for subsequent resumption.

5.210.2.7 #define NXTSERVO_CMD_RESET 0x53

Reset servo properties to factory default. Initial Position of servos to 1500, and speed to 0.

5.210.2.8 #define NXTSERVO_CMD_RESUME 0x52

Resume macro Execution. This command resumes macro where it was paused last, using the same environment.

5.210.2.9 #define NXTSERVO_EM_CMD_QUIT 0x51

Exit edit macro mode

5.211 MindSensors NXTHID constants

Constants that are for use with the MindSensors NXTHID device.

Modules

- [MindSensors NXTHID registers](#)

NXTHID device register constants.

- [MindSensors NXTHID modifier keys](#)
NXTHID device modifier key constants.
- [MindSensors NXTHID commands](#)
NXTHID device command constants.

5.211.1 Detailed Description

Constants that are for use with the MindSensors NXTHID device.

5.212 MindSensors NXTHID registers

NXTHID device register constants.

Defines

- #define [NXTHID_REG_CMD](#) 0x41
- #define [NXTHID_REG_MODIFIER](#) 0x42
- #define [NXTHID_REG_DATA](#) 0x43

5.212.1 Detailed Description

NXTHID device register constants.

5.212.2 Define Documentation

5.212.2.1 #define NXTHID_REG_CMD 0x41

NXTHID command register. See [MindSensors NXTHID commands](#) group.

5.212.2.2 #define NXTHID_REG_DATA 0x43

NXTHID data register.

5.212.2.3 #define NXTHID_REG_MODIFIER 0x42

NXTHID modifier register. See [MindSensors NXTHID modifier keys](#) group.

5.213 MindSensors NXTHID modifier keys

NXTHID device modifier key constants.

Defines

- #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

5.213.1 Detailed Description

NXTHID device modifier key constants.

5.213.2 Define Documentation

5.213.2.1 #define NXTHID_MOD_LEFT_ALT 0x04

NXTHID left alt modifier.

5.213.2.2 #define NXTHID_MOD_LEFT_CTRL 0x01

NXTHID left control modifier.

5.213.2.3 #define NXTHID_MOD_LEFT_GUI 0x08

NXTHID left gui modifier.

5.213.2.4 #define NXTHID_MOD_LEFT_SHIFT 0x02

NXTHID left shift modifier.

5.213.2.5 #define NXTHID_MOD_NONE 0x00

NXTHID no modifier.

5.213.2.6 #define NXTHID_MOD_RIGHT_ALT 0x40

NXTHID right alt modifier.

5.213.2.7 #define NXTHID_MOD_RIGHT_CTRL 0x10

NXTHID right control modifier.

5.213.2.8 #define NXTHID_MOD_RIGHT_GUI 0x80

NXTHID right gui modifier.

5.213.2.9 #define NXTHID_MOD_RIGHT_SHIFT 0x20

NXTHID right shift modifier.

5.214 MindSensors NXTHID commands

NXTHID device command constants.

Defines

- #define [NXTHID_CMD_ASCII](#) 0x41
- #define [NXTHID_CMD_DIRECT](#) 0x44
- #define [NXTHID_CMD_TRANSMIT](#) 0x54

5.214.1 Detailed Description

NXTHID device command constants. These are written to the command register to control the device.

5.214.2 Define Documentation**5.214.2.1 #define NXTHID_CMD_ASCII 0x41**

Use ASCII data mode. In ASCII mode no non-printable characters can be sent.

5.214.2.2 #define NXTHID_CMD_DIRECT 0x44

Use direct data mode In direct mode any character can be sent.

5.214.2.3 #define NXTHID_CMD_TRANSMIT 0x54

Transmit data to the host computer.

5.215 MindSensors NXTPowerMeter constants

Constants that are for use with the MindSensors NXTPowerMeter device.

Modules

- [MindSensors NXTPowerMeter registers](#)
NXTPowerMeter device register constants.
- [MindSensors NXTPowerMeter commands](#)
NXTPowerMeter device command constants.

5.215.1 Detailed Description

Constants that are for use with the MindSensors NXTPowerMeter device.

5.216 MindSensors NXTPowerMeter registers

NXTPowerMeter device register constants.

Defines

- #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

5.216.1 Detailed Description

NXTPowerMeter device register constants.

5.216.2 Define Documentation

5.216.2.1 `#define NXTPM_REG_CAPACITY 0x46`

NXTPowerMeter capacity used since last reset register. (2 bytes)

5.216.2.2 `#define NXTPM_REG_CMD 0x41`

NXTPowerMeter command register. See the [MindSensors NXTPowerMeter commands](#) group.

5.216.2.3 `#define NXTPM_REG_CURRENT 0x42`

NXTPowerMeter present current in mA register. (2 bytes)

5.216.2.4 `#define NXTPM_REG_ERRORCOUNT 0x5F`

NXTPowerMeter error count register. (2 bytes)

5.216.2.5 `#define NXTPM_REG_GAIN 0x5E`

NXTPowerMeter gain register. (1 byte)

5.216.2.6 `#define NXTPM_REG_MAXCURRENT 0x4E`

NXTPowerMeter max current register. (2 bytes)

5.216.2.7 `#define NXTPM_REG_MAXVOLTAGE 0x52`

NXTPowerMeter max voltage register. (2 bytes)

5.216.2.8 `#define NXTPM_REG_MINCURRENT 0x50`

NXTPowerMeter min current register. (2 bytes)

5.216.2.9 #define NXTPM_REG_MINVOLTAGE 0x54

NXTPowerMeter min voltage register. (2 bytes)

5.216.2.10 #define NXTPM_REG_POWER 0x48

NXTPowerMeter present power register. (2 bytes)

5.216.2.11 #define NXTPM_REG_TIME 0x56

NXTPowerMeter time register. (4 bytes)

5.216.2.12 #define NXTPM_REG_TOTALPOWER 0x4A

NXTPowerMeter total power consumed since last reset register. (4 bytes)

5.216.2.13 #define NXTPM_REG_USERGAIN 0x5A

NXTPowerMeter user gain register. Not yet implemented. (4 bytes)

5.216.2.14 #define NXTPM_REG_VOLTAGE 0x44

NXTPowerMeter present voltage in mV register. (2 bytes)

5.217 MindSensors NXTPowerMeter commands

NXTPowerMeter device command constants.

Defines

- #define [NXTPM_CMD_RESET](#) 0x52

5.217.1 Detailed Description

NXTPowerMeter device command constants. These are written to the command register to control the device.

5.217.2 Define Documentation**5.217.2.1 #define NXTPM_CMD_RESET 0x52**

Reset counters.

5.218 MindSensors NXTSumoEyes constants

Constants that are for use with the MindSensors NXTSumoEyes device.

Defines

- #define NXTSE_ZONE_NONE 0
- #define NXTSE_ZONE_FRONT 1
- #define NXTSE_ZONE_LEFT 2
- #define NXTSE_ZONE_RIGHT 3

5.218.1 Detailed Description

Constants that are for use with the MindSensors NXTSumoEyes device.

5.218.2 Define Documentation**5.218.2.1 #define NXTSE_ZONE_FRONT 1**

Obstacle zone front.

5.218.2.2 #define NXTSE_ZONE_LEFT 2

Obstacle zone left.

5.218.2.3 #define NXTSE_ZONE_NONE 0

Obstacle zone none.

5.218.2.4 #define NXTSE_ZONE_RIGHT 3

Obstacle zone right.

5.219 MindSensors NXTLineLeader constants

Constants that are for use with the MindSensors NXTLineLeader device.

Modules

- [MindSensors NXTLineLeader registers](#)
NXTLineLeader device register constants.
- [MindSensors NXTLineLeader commands](#)
NXTLineLeader device command constants.

5.219.1 Detailed Description

Constants that are for use with the MindSensors NXTLineLeader device.

5.220 MindSensors NXTLineLeader registers

NXTLineLeader device register constants.

Defines

- #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

5.220.1 Detailed Description

NXTLineLeader device register constants.

5.220.2 Define Documentation

5.220.2.1 **#define NXTLL_REG_AVERAGE 0x43**

NXTLineLeader average result register.

5.220.2.2 **#define NXTLL_REG_BLACKDATA 0x6C**

NXTLineLeader black calibration data registers. 8 bytes.

5.220.2.3 **#define NXTLL_REG_BLACKLIMITS 0x59**

NXTLineLeader black limit registers. 8 bytes.

5.220.2.4 **#define NXTLL_REG_CALIBRATED 0x49**

NXTLineLeader calibrated sensor reading registers. 8 bytes.

5.220.2.5 **#define NXTLL_REG_CMD 0x41**

NXTLineLeader command register. See the [MindSensors NXTLineLeader commands](#) group.

5.220.2.6 **#define NXTLL_REG_KD_FACTOR 0x63**

NXTLineLeader Kd factor register. Default = 32.

5.220.2.7 **#define NXTLL_REG_KD_VALUE 0x48**

NXTLineLeader Kd value register. Default = 8.

5.220.2.8 **#define NXTLL_REG_KI_FACTOR 0x62**

NXTLineLeader Ki factor register. Default = 32.

5.220.2.9 #define NXTLL_REG_KI_VALUE 0x47

NXTLineLeader Ki value register. Default = 0.

5.220.2.10 #define NXTLL_REG_KP_FACTOR 0x61

NXTLineLeader Kp factor register. Default = 32.

5.220.2.11 #define NXTLL_REG_KP_VALUE 0x46

NXTLineLeader Kp value register. Default = 25.

5.220.2.12 #define NXTLL_REG_RAWVOLTAGE 0x74

NXTLineLeader uncalibrated sensor voltage registers. 16 bytes.

5.220.2.13 #define NXTLL_REG_RESULT 0x44

NXTLineLeader result register (sensor bit values).

5.220.2.14 #define NXTLL_REG_SETPOINT 0x45

NXTLineLeader user settable average (setpoint) register. Default = 45.

5.220.2.15 #define NXTLL_REG_STEERING 0x42

NXTLineLeader steering register.

5.220.2.16 #define NXTLL_REG_WHITEDATA 0x64

NXTLineLeader white calibration data registers. 8 bytes.

5.220.2.17 #define NXTLL_REG_WHITELIMITS 0x51

NXTLineLeader white limit registers. 8 bytes.

5.221 MindSensors NXTLineLeader commands

NXTLineLeader device command constants.

Defines

- #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

5.221.1 Detailed Description

NXTLineLeader device command constants. These are written to the command register to control the device.

5.221.2 Define Documentation**5.221.2.1 #define NXTLL_CMD_BLACK 0x42**

Black calibration.

5.221.2.2 #define NXTLL_CMD_EUROPEAN 0x45

European power frequency. (50hz)

5.221.2.3 #define NXTLL_CMD_INVERT 0x49

Invert color.

5.221.2.4 #define NXTLL_CMD_POWERDOWN 0x44

Power down the device.

5.221.2.5 #define NXTLL_CMD_POWERUP 0x50

Power up the device.

5.221.2.6 #define NXTLL_CMD_RESET 0x52

Reset inversion.

5.221.2.7 #define NXTLL_CMD_SNAPSHOT 0x53

Setpoint based on snapshot (automatically sets invert if needed).

5.221.2.8 #define NXTLL_CMD_UNIVERSAL 0x55

Universal power frequency. The sensor auto adjusts for any frequency. This is the default mode.

5.221.2.9 #define NXTLL_CMD_USA 0x41

USA power frequency. (60hz)

5.221.2.10 #define NXTLL_CMD_WHITE 0x57

White balance calibration.

5.222 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.

5.222.1 Detailed Description

Constants that are for use with Codatex devices.

5.223 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.

Defines

- #define [CT_ADDR_RFID](#) 0x04
- #define [CT_REG_STATUS](#) 0x32
- #define [CT_REG_MODE](#) 0x41
- #define [CT_REG_DATA](#) 0x42

5.223.1 Detailed Description

Constants that are for use with the Codatex RFID sensor device.

5.223.2 Define Documentation

5.223.2.1 #define CT_ADDR_RFID 0x04

RFID I2C address

5.223.2.2 #define CT_REG_DATA 0x42

RFID data register

5.223.2.3 #define CT_REG_MODE 0x41

RFID mode register

5.223.2.4 #define CT_REG_STATUS 0x32

RFID status register

5.224 Codatex RFID sensor modes

Constants that are for configuring the Codatex RFID sensor mode.

Defines

- #define **RFID_MODE_STOP** 0
- #define **RFID_MODE_SINGLE** 1
- #define **RFID_MODE_CONTINUOUS** 2

5.224.1 Detailed Description

Constants that are for configuring the Codatex RFID sensor mode.

5.224.2 Define Documentation**5.224.2.1 #define RFID_MODE_CONTINUOUS 2**

Configure the RFID device for continuous reading

5.224.2.2 #define RFID_MODE_SINGLE 1

Configure the RFID device for a single reading

5.224.2.3 #define RFID_MODE_STOP 0

Stop the RFID device

5.225 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.

5.225.1 Detailed Description

Constants that are for use with Dexter Industries devices.

5.226 Dexter Industries GPS sensor constants

Constants that are for use with the Dexter Industries GPS sensor.

Defines

- #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

5.226.1 Detailed Description

Constants that are for use with the Dexter Industries GPS sensor.

5.226.2 Define Documentation

5.226.2.1 #define DGPS_REG_DISTANCE 0x08

Read distance to current waypoint in meters.

5.226.2.2 #define DGPS_REG_HEADING 0x07

Read heading in degrees.

5.226.2.3 #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.

5.226.2.4 #define DGPS_REG_LATITUDE 0x02

Read integer latitude.(dddddd; Positive = North; Negative = South).

5.226.2.5 #define DGPS_REG_LONGITUDE 0x04

Read integer longitude (ddddddddd; Positive = East; Negative = West).

5.226.2.6 #define DGPS_REG_SETLATITUDE 0x0B

Set waypoint latitude as a 4 byte integer.

5.226.2.7 #define DGPS_REG_SETLONGITUDE 0x0C

Set waypoint longitude as a 4 byte integer.

5.226.2.8 #define DGPS_REG_STATUS 0x01

Read status of the GPS (0 - invalid signal, 1 - valid signal).

5.226.2.9 #define DGPS_REG_TIME 0x00

Read time in UTC (hhmmss).

5.226.2.10 #define DGPS_REG_VELOCITY 0x06

Read velocity in cm/s.

5.226.2.11 #define DGPS_REG_WAYANGLE 0x09

Read angle to current waypoint in degrees.

5.226.2.12 #define DI_ADDR_DGPS 0x06

Dexter Industries DGPS I2C address

5.227 Dexter Industries IMU sensor constants

Constants that are for use with the Dexter Industries IMU sensor.

Modules

- [Dexter Industries IMU Gyro register constants](#)

Constants that define the Dexter Industries IMU Gyro registers.

- [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 FIFO control register onstants](#)

Constants that are for use with the Dexter Industries IMU Gyro sensor's FIFO control register.

- [Dexter Industries IMU Gyro status register constants](#)

Constants that are for use with the Dexter Industries IMU Gyro sensor's status 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 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 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 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.

Defines

- `#define DI_ADDR_GYRO 0xD2`
- `#define DI_ADDR_ACCL 0x3A`

5.227.1 Detailed Description

Constants that are for use with the Dexter Industries IMU sensor.

5.227.2 Define Documentation

5.227.2.1 `#define DI_ADDR_ACCL 0x3A`

Dexter Industries DIMU Accelerometer I2C address

5.227.2.2 `#define DI_ADDR_GYRO 0xD2`

Dexter Industries DIMU Gyro I2C address

5.228 Dexter Industries IMU Gyro register constants

Constants that define the Dexter Industries IMU Gyro registers.

Defines

- `#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

5.228.1 Detailed Description

Constants that define the Dexter Industries IMU Gyro registers.

5.228.2 Define Documentation

5.228.2.1 #define **DIGYRO_REG_CTRL1** 0x20

Gyro control register 1

5.228.2.2 #define **DIGYRO_REG_CTRL1AUTO** 0xA0

Gyro control register 1 - auto increment write

5.228.2.3 #define DIGYRO_REG_CTRL2 0x21

Gyro control register 2

5.228.2.4 #define DIGYRO_REG_CTRL3 0x22

Gyro control register 3

5.228.2.5 #define DIGYRO_REG_CTRL4 0x23

Gyro control register 4

5.228.2.6 #define DIGYRO_REG_CTRL5 0x24

Gyro control register 5

5.228.2.7 #define DIGYRO_REG_FIFOCTRL 0x2E

Gyro FIFO control register

5.228.2.8 #define DIGYRO_REG_FIFOSRC 0x2F

Gyro FIFO source register (read only)

5.228.2.9 #define DIGYRO_REG_INT1_CFG 0x30

Gyro interrupt 1 config register

5.228.2.10 #define DIGYRO_REG_INT1_DUR 0x38

Gyro interrupt 1 duration register

5.228.2.11 #define DIGYRO_REG_INT1_SRC 0x31

Gyro interrupt 1 source register

5.228.2.12 #define DIGYRO_REG_INT1_XHI 0x32

Gyro interrupt 1 x-axis high threshold register

5.228.2.13 #define DIGYRO_REG_INT1_XLO 0x33

Gyro interrupt 1 x-axis low threshold register

5.228.2.14 #define DIGYRO_REG_INT1_YHI 0x34

Gyro interrupt 1 y-axis high threshold register

5.228.2.15 #define DIGYRO_REG_INT1_YLO 0x35

Gyro interrupt 1 y-axis low threshold register

5.228.2.16 #define DIGYRO_REG_INT1_ZHI 0x36

Gyro interrupt 1 z-axis high threshold register

5.228.2.17 #define DIGYRO_REG_INT1_ZLO 0x37

Gyro interrupt 1 z-axis low threshold register

5.228.2.18 #define DIGYRO_REG_OUTTEMP 0x26

Gyro temperature register (read only) - stores temperature data

5.228.2.19 #define DIGYRO_REG_REFERENCE 0x25

Gyro reference register - stores the reference value used for interrupt generation

5.228.2.20 #define DIGYRO_REG_STATUS 0x27

Gyro status register (read only)

5.228.2.21 #define DIGYRO_REG_TEMPAUTO 0xA6

Gyro temperature register - read burst mode (read only)

5.228.2.22 #define DIGYRO_REG_WHOAMI 0x0F

Gyro device identification register (read only)

5.228.2.23 #define DIGYRO_REG_XHIGH 0x29

Gyro x-axis high byte register (read only)

5.228.2.24 #define DIGYRO_REG_XLOW 0x28

Gyro x-axis low byte register (read only)

5.228.2.25 #define DIGYRO_REG_XLOWBURST 0xA8

Gyro x-axis low byte register - read burst mode (read only)

5.228.2.26 #define DIGYRO_REG_YHIGH 0x2B

Gyro y-axis high byte register (read only)

5.228.2.27 #define DIGYRO_REG_YLOW 0x2A

Gyro y-axis low byte register (read only)

5.228.2.28 #define DIGYRO_REG_YLOWBURST 0xAA

Gyro y-axis low byte register - read burst mode (read only)

5.228.2.29 #define DIGYRO_REG_ZHIGH 0x2D

Gyro z-axis high byte register (read only)

5.228.2.30 #define DIGYRO_REG_ZLOW 0x2C

Gyro z-axis low byte register (read only)

5.228.2.31 #define DIGYRO_REG_ZLOWBURST 0xAC

Gyro z-axis low byte register - read burst mode (read only)

5.229 Dexter Industries IMU Gyro control register 1 constants

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 1.

Defines

- #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

5.229.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 1.

5.229.2 Define Documentation

5.229.2.1 #define **DIGYRO_CTRL1_BANDWIDTH_1** 0x00

Gyro LPF2 cut-off frequency bandwidth level 1 (12.5hz, 12.5hz, 20hz, 30hz)

5.229.2.2 #define **DIGYRO_CTRL1_BANDWIDTH_2** 0x10

Gyro LPF2 cut-off frequency bandwidth level 2 (12.5hz, 25hz, 50hz, 70hz)

5.229.2.3 #define **DIGYRO_CTRL1_BANDWIDTH_3** 0x20

Gyro LPF2 cut-off frequency bandwidth level 3 (20hz, 25hz, 50hz, 110hz)

5.229.2.4 #define DIGYRO_CTRL1_BANDWIDTH_4 0x30

Gyro LPF2 cut-off frequency bandwidth level 4 (30hz, 35hz, 50hz, 110hz)

5.229.2.5 #define DIGYRO_CTRL1_DATARATE_100 0x00

Gyro output data rate 100 hz

5.229.2.6 #define DIGYRO_CTRL1_DATARATE_200 0x40

Gyro output data rate 200 hz

5.229.2.7 #define DIGYRO_CTRL1_DATARATE_400 0x80

Gyro output data rate 400 hz

5.229.2.8 #define DIGYRO_CTRL1_DATARATE_800 0xC0

Gyro output data rate 800 hz

5.229.2.9 #define DIGYRO_CTRL1_NORMAL 0x08

Gyro disable power down mode

5.229.2.10 #define DIGYRO_CTRL1_POWERDOWN 0x00

Gyro enable power down mode

5.229.2.11 #define DIGYRO_CTRL1_XENABLE 0x01

Gyro enable X axis

5.229.2.12 #define DIGYRO_CTRL1_YENABLE 0x02

Gyro enable Y axis

5.229.2.13 #define DIGYRO_CTRL1_ZENABLE 0x04

Gyro enable Z axis

5.230 Dexter Industries IMU Gyro control register 2 constants

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 2.

Defines

- #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

5.230.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 2.

5.230.2 Define Documentation

5.230.2.1 #define `DIGYRO_CTRL2_CUTOFF_FREQ_001 0x09`

Gyro high pass filter cutoff frequency 0.01 hz

5.230.2.2 #define `DIGYRO_CTRL2_CUTOFF_FREQ_002 0x08`

Gyro high pass filter cutoff frequency 0.02 hz

5.230.2.3 #define `DIGYRO_CTRL2_CUTOFF_FREQ_005 0x07`

Gyro high pass filter cutoff frequency 0.05 hz

5.230.2.4 #define DIGYRO_CTRL2_CUTOFF_FREQ_01 0x06

Gyro high pass filter cutoff frequency 0.1 hz

5.230.2.5 #define DIGYRO_CTRL2_CUTOFF_FREQ_02 0x05

Gyro high pass filter cutoff frequency 0.2 hz

5.230.2.6 #define DIGYRO_CTRL2_CUTOFF_FREQ_05 0x04

Gyro high pass filter cutoff frequency 0.5 hz

5.230.2.7 #define DIGYRO_CTRL2_CUTOFF_FREQ_1 0x03

Gyro high pass filter cutoff frequency 1 hz

5.230.2.8 #define DIGYRO_CTRL2_CUTOFF_FREQ_2 0x02

Gyro high pass filter cutoff frequency 2 hz

5.230.2.9 #define DIGYRO_CTRL2_CUTOFF_FREQ_4 0x01

Gyro high pass filter cutoff frequency 4 hz

5.230.2.10 #define DIGYRO_CTRL2_CUTOFF_FREQ_8 0x00

Gyro high pass filter cutoff frequency 8 hz

5.230.2.11 #define DIGYRO_CTRL2_HPMODE_AUTOINT 0x30

Gyro high pass filter autoreset on interrupt event mode

5.230.2.12 #define DIGYRO_CTRL2_HPMODE_NORMAL 0x20

Gyro high pass filter normal mode

5.230.2.13 #define DIGYRO_CTRL2_HPMODE_REFSIG 0x10

Gyro high pass filter reference signal mode

5.230.2.14 #define DIGYRO_CTRL2_HPMODE_RESET 0x00

Gyro high pass filter reset mode

5.231 Dexter Industries IMU Gyro control register 3 constants

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 3.

Defines

- #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

5.231.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 3.

5.231.2 Define Documentation**5.231.2.1 #define DIGYRO_CTRL3_INT1_BOOT 0x40**

Gyro boot status available on INT1

5.231.2.2 #define DIGYRO_CTRL3_INT1_ENABLE 0x80

Gyro interrupt enable on INT1 pin

5.231.2.3 #define DIGYRO_CTRL3_INT1_LOWACTIVE 0x20

Gyro interrupt active low on INT1

5.231.2.4 #define DIGYRO_CTRL3_INT2_DATAREADY 0x08

Gyro data ready on DRDY/INT2

5.231.2.5 #define DIGYRO_CTRL3_INT2_EMPTY 0x01

Gyro FIFO empty interrupt on DRDY/INT2

5.231.2.6 #define DIGYRO_CTRL3_INT2_OVERRUN 0x02

Gyro FIFO overrun interrupt on DRDY/INT2

5.231.2.7 #define DIGYRO_CTRL3_INT2_WATERMARK 0x04

Gyro FIFO watermark interrupt on DRDY/INT2

5.231.2.8 #define DIGYRO_CTRL3_OPENDRAIN 0x10

Gyro use open drain rather than push-pull

5.232 Dexter Industries IMU Gyro control register 4 constants

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 4.

Defines

- #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

5.232.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 4.

5.232.2 Define Documentation

5.232.2.1 #define DIGYRO_CTRL4_BIGENDIAN 0x40

Gyro use big endian - MSB/LSB rather than LSB/MSB in output registers

5.232.2.2 #define DIGYRO_CTRL4_BLOCKDATA 0x80

Gyro block data update - output registers are not updated until MSB and LSB reading

5.232.2.3 #define DIGYRO_CTRL4_SCALE_2000 0x30

Gyro 2000 degrees per second scale

5.232.2.4 #define DIGYRO_CTRL4_SCALE_250 0x00

Gyro 250 degrees per second scale

5.232.2.5 #define DIGYRO_CTRL4_SCALE_500 0x10

Gyro 500 degrees per second scale

5.233 Dexter Industries IMU Gyro control register 5 constants

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 5.

Defines

- #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

5.233.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's control register 5.

5.233.2 Define Documentation

5.233.2.1 **#define DIGYRO_CTRL5_FIFOENABLE 0x40**

Gyro enable FIFO

5.233.2.2 **#define DIGYRO_CTRL5_HPENABLE 0x10**

Gyro enable high pass filter

5.233.2.3 **#define DIGYRO_CTRL5_INT1_SEL_1 0x00**

Gyro non-high-pass-filtered data are used for interrupt generation

5.233.2.4 **#define DIGYRO_CTRL5_INT1_SEL_2 0x04**

Gyro high-pass-filtered data are used for interrupt generation

5.233.2.5 **#define DIGYRO_CTRL5_INT1_SEL_3 0x08**

Gyro low-pass-filtered data are used for interrupt generation

5.233.2.6 **#define DIGYRO_CTRL5_OUT_SEL_1 0x00**

Gyro data in data registers and FIFO are not high-pass filtered

5.233.2.7 **#define DIGYRO_CTRL5_OUT_SEL_2 0x01**

Gyro data in data registers and FIFO are high-pass filtered

5.233.2.8 **#define DIGYRO_CTRL5_OUT_SEL_3 0x02**

Gyro data in data registers and FIFO are low-pass filtered by LPF2

5.233.2.9 #define DIGYRO_CTRL5_REBOOTMEM 0x80

Gyro reboot memory content

5.234 Dexter Industries IMU Gyro FIFO control register onstants

Constants that are for use with the Dexter Industries IMU Gyro sensor's FIFO control register.

Defines

- #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

5.234.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's FIFO control register.

5.234.2 Define Documentation**5.234.2.1 #define DIGYRO_FIFOCTRL_BYPASS 0x00**

Gyro FIFO bypass mode

5.234.2.2 #define DIGYRO_FIFOCTRL_BYPASS2STREAM 0x80

Gyro FIFO bypass-to-stream mode

5.234.2.3 #define DIGYRO_FIFOCTRL_FIFO 0x20

Gyro FIFO mode

5.234.2.4 #define DIGYRO_FIFOCTRL_STREAM 0x40

Gyro FIFO stream mode

5.234.2.5 #define DIGYRO_FIFOCTRL_STREAM2FIFO 0x60

Gyro FIFO stream-to-FIFO mode

5.234.2.6 #define DIGYRO_FIFOCTRL_WATERMARK_MASK 0x1F

Gyro FIFO threshold. Watermark level setting mask (values from 0x00 to 0x1F)

5.235 Dexter Industries IMU Gyro status register constants

Constants that are for use with the Dexter Industries IMU Gyro sensor's status register.

Defines

- #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

5.235.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Gyro sensor's status register.

5.235.2 Define Documentation**5.235.2.1 #define DIGYRO_STATUS_XDATA 0x01**

Gyro X-axis new data available

5.235.2.2 #define DIGYRO_STATUS_XOVER 0x10

Gyro X-axis data overrun - new data for the X-axis has overwritten the previous one

5.235.2.3 #define DIGYRO_STATUS_XYZDATA 0x08

Gyro X, Y, or Z-axis new data available - a new set of data is available

5.235.2.4 #define DIGYRO_STATUS_XYZOVER 0x80

Gyro X, Y, or Z-axis data overrun - new data has overwritten the previous one before it was read

5.235.2.5 #define DIGYRO_STATUS_YDATA 0x02

Gyro Y-axis new data available

5.235.2.6 #define DIGYRO_STATUS_YOVER 0x20

Gyro Y-axis data overrun - new data for the Y-axis has overwritten the previous one

5.235.2.7 #define DIGYRO_STATUS_ZDATA 0x04

Gyro Z-axis new data available

5.235.2.8 #define DIGYRO_STATUS_ZOVER 0x40

Gyro Z-axis data overrun - new data for the Z-axis has overwritten the previous one

5.236 Dexter Industries IMU Accelerometer register constants

Constants that define the Dexter Industries IMU Accelerometer registers.

Defines

- #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

5.236.1 Detailed Description

Constants that define the Dexter Industries IMU Accelerometer registers.

5.236.2 Define Documentation

5.236.2.1 #define DIACCL_REG_CTRL1 0x18

Accelerometer control register 1 (read/write)

5.236.2.2 #define DIACCL_REG_CTRL2 0x19

Accelerometer control register 1 (read/write)

5.236.2.3 #define DIACCL_REG_DETECTSRC 0x0A

Accelerometer detection source register (read only)

5.236.2.4 #define DIACCL_REG_I2CADDR 0x0D

Accelerometer I2C address register (read only)

5.236.2.5 #define DIACCL_REG_INTLATCH 0x17

Accelerometer interrupt latch reset register (read/write)

5.236.2.6 #define DIACCL_REG_LATENCYTM 0x1D

Accelerometer latency time value register (read/write)

5.236.2.7 #define DIACCL_REG_LVLDETTTHR 0x1A

Accelerometer level detection threshold limit value register (read/write)

5.236.2.8 #define DIACCL_REG_MODECTRL 0x16

Accelerometer mode control register (read/write)

5.236.2.9 #define DIACCL_REG_OUTTEMP 0x0B

Accelerometer temperature output register (read only)

5.236.2.10 #define DIACCL_REG_PLSDETTTHR 0x1B

Accelerometer pulse detection threshold limit value register (read/write)

5.236.2.11 #define DIACCL_REG_PLSDURVAL 0x1C

Accelerometer pulse duration value register (read/write)

5.236.2.12 #define DIACCL_REG_STATUS 0x09

Accelerometer status register (read only)

5.236.2.13 #define DIACCL_REG_TIMEWINDOW 0x1E

Accelerometer time window for 2nd pulse value register (read/write)

5.236.2.14 #define DIACCL_REG_USERINFO 0x0E

Accelerometer user information register (read only)

5.236.2.15 #define DIACCL_REG_WHOAMI 0x0F

Accelerometer device identification register (read only)

5.236.2.16 #define DIACCL_REG_X8 0x06

Accelerometer x-axis 8-bit register (read only)

5.236.2.17 #define DIACCL_REG_XHIGH 0x01

Accelerometer x-axis high byte register (read only)

5.236.2.18 #define DIACCL_REG_XHIGHDRIFT 0x11

Accelerometer x-axis offset drift high byte register (read/write)

5.236.2.19 #define DIACCL_REG_XLOW 0x00

Accelerometer x-axis low byte register (read only)

5.236.2.20 #define DIACCL_REG_XLOWDRIFT 0x10

Accelerometer x-axis offset drift low byte register (read/write)

5.236.2.21 #define DIACCL_REG_Y8 0x07

Accelerometer x-axis 8-bit register (read only)

5.236.2.22 #define DIACCL_REG_YHIGH 0x03

Accelerometer y-axis high byte register (read only)

5.236.2.23 #define DIACCL_REG_YHIGHDRIFT 0x13

Accelerometer y-axis offset drift high byte register (read/write)

5.236.2.24 #define DIACCL_REG_YLOW 0x02

Accelerometer y-axis low byte register (read only)

5.236.2.25 #define DIACCL_REG_YLOWDRIFT 0x12

Accelerometer y-axis offset drift low byte register (read/write)

5.236.2.26 #define DIACCL_REG_Z8 0x08

Accelerometer x-axis 8-bit register (read only)

5.236.2.27 #define DIACCL_REG_ZHIGH 0x05

Accelerometer z-axis high byte register (read only)

5.236.2.28 #define DIACCL_REG_ZHIGHDRIFT 0x15

Accelerometer z-axis offset drift high byte register (read/write)

5.236.2.29 #define DIACCL_REG_ZLOW 0x04

Accelerometer z-axis low byte register (read only)

5.236.2.30 #define DIACCL_REG_ZLOWDRIFT 0x14

Accelerometer z-axis offset drift low byte register (read/write)

5.237 Dexter Industries IMU Accelerometer status register constants

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's status register.

Defines

- #define **DIACCL_STATUS_DATAREADY** 0x01
- #define **DIACCL_STATUS_DATAOVER** 0x02
- #define **DIACCL_STATUS_PARITYERR** 0x04

5.237.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's status register.

5.238 Dexter Industries IMU Accelerometer mode control register constants

5.237.2 Define Documentation

5.237.2.1 #define DIACCL_STATUS_DATAOVER 0x02

Accelerometer data is overwritten

5.237.2.2 #define DIACCL_STATUS_DATAREADY 0x01

Accelerometer data is ready

5.237.2.3 #define DIACCL_STATUS_PARITYERR 0x04

Accelerometer parity error is detected in trim data

5.238 Dexter Industries IMU Accelerometer mode control register constants

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's mode control register.

Defines

- #define DIACCL_MODE_STANDBY 0x00
- #define DIACCL_MODE_MEASURE 0x01
- #define DIACCL_MODE_LVLDETECT 0x02
- #define DIACCL_MODE_PLSDETECT 0x03
- #define DIACCL_MODE_GLBL8 0x00
- #define DIACCL_MODE_GLBL2 0x04
- #define DIACCL_MODE_GLBL4 0x08

5.238.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's mode control register.

5.238.2 Define Documentation

5.238.2.1 #define DIACCL_MODE_GLBL2 0x04

Accelerometer 2G measurement range

5.238.2.2 #define DIACCL_MODE_GLVL4 0x08

Accelerometer 4G measurement range

5.238.2.3 #define DIACCL_MODE_GLVL8 0x00

Accelerometer 8G measurement range

5.238.2.4 #define DIACCL_MODE_LVLDETECT 0x02

Accelerometer level detect mode

5.238.2.5 #define DIACCL_MODE_MEASURE 0x01

Accelerometer measurement mode

5.238.2.6 #define DIACCL_MODE_PLSDETECT 0x03

Accelerometer pulse detect mode

5.238.2.7 #define DIACCL_MODE_STANDBY 0x00

Accelerometer standby mode

5.239 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.

Defines

- #define DIACCL_INTERRUPT_LATCH_CLEAR1 0x01
- #define DIACCL_INTERRUPT_LATCH_CLEAR2 0x02

5.239.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's interrupt latch reset register.

5.239.2 Define Documentation

5.239.2.1 #define DIACCL_INTERRUPT_LATCH_CLEAR1 0x01

Accelerometer clear interrupt 1

5.239.2.2 #define DIACCL_INTERRUPT_LATCH_CLEAR2 0x02

Accelerometer clear interrupt 2

5.240 Dexter Industries IMU Accelerometer control register 1 constants

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's control register 1.

Defines

- #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

5.240.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's control register 1.

5.240.2 Define Documentation

5.240.2.1 #define DIACCL_CTRL1_filt_BW125 0x80

Accelerometer digital filter band width is 125 Hz.

5.240.2.2 #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)

5.240.2.3 #define DIACCL_CTRL1_LEVELPULSE 0x00

Accelerometer INT1 register is detecting Level while INT2 is detecting pulse

5.240.2.4 #define DIACCL_CTRL1_NO_XDETECT 0x08

Accelerometer disable x-axis detection.

5.240.2.5 #define DIACCL_CTRL1_NO_YDETECT 0x10

Accelerometer disable y-axis detection.

5.240.2.6 #define DIACCL_CTRL1_NO_ZDETECT 0x20

Accelerometer disable z-axis detection.

5.240.2.7 #define DIACCL_CTRL1_PULSELEVEL 0x02

Accelerometer INT1 Register is detecting Pulse while INT2 is detecting Level

5.240.2.8 #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.

5.240.2.9 #define DIACCL_CTRL1_THRESH_INT 0x40

Accelerometer threshold value can be an integer.

5.241 Dexter Industries IMU Accelerometer control register 2 constants

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's control register 2.

Defines

- #define DIACCL_CTRL2_LVLPO_L_NEGAND 0x01
- #define DIACCL_CTRL2_DETPOL_NEGAND 0x02
- #define DIACCL_CTRL2_DRIVE_STRONG 0x04

5.241.1 Detailed Description

Constants that are for use with the Dexter Industries IMU Accelerometer sensor's control register 2.

5.241.2 Define Documentation**5.241.2.1 #define DIACCL_CTRL2_DETPOL_NEGAND 0x02**

Accelerometer pulse detection polarity is negative and detecting condition is AND all 3 axes

5.241.2.2 #define DIACCL_CTRL2_DRIVE_STRONG 0x04

Accelerometer strong drive strength on SDA/SDO pin

5.241.2.3 #define DIACCL_CTRL2_LVLPO_L_NEGAND 0x01

Accelerometer level detection polarity is negative and detecting condition is AND all 3 axes

5.242 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.

5.242.1 Detailed Description

Constants that are for use with Microinfinity devices.

5.243 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.

Defines

- #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

5.243.1 Detailed Description

Constants that are for use with the CruizCore XG1300L sensor.

5.243.2 Define Documentation

5.243.2.1 #define MI_ADDR_XG1300L 0x02

XG1300L I2C address

5.243.2.2 #define XG1300L_REG_2G 0x61

Select +/- 2G accelerometer range.

5.243.2.3 #define XG1300L_REG_4G 0x62

Select +/- 4G accelerometer range.

5.243.2.4 #define XG1300L_REG_8G 0x63

Select +/- 8G accelerometer range.

5.243.2.5 #define XG1300L_REG_ANGLE 0x42

Read accumulated angle (2 bytes little endian) in 1/100s of degrees.

5.243.2.6 #define XG1300L_REG_RESET 0x60

Reset the XG1300L device.

5.243.2.7 #define XG1300L_REG_TURNRATE 0x44

Read rate of turn (2 bytes little endian) in 1/100s of degrees/second.

5.243.2.8 #define XG1300L_REG_XAXIS 0x46

Read x-axis acceleration (2 bytes little endian) in m/s² scaled by 100/ACC_RANGE*2, where ACC_RANGE is 2, 4, or 8.

5.243.2.9 #define XG1300L_REG_YAXIS 0x48

Read y-axis acceleration (2 bytes little endian) in m/s² scaled by 100/ACC_RANGE*2, where ACC_RANGE is 2, 4, or 8.

5.243.2.10 #define XG1300L_REG_ZAXIS 0x4A

Read z-axis acceleration (2 bytes little endian) in m/s² scaled by 100/ACC_RANGE*2, where ACC_RANGE is 2, 4, or 8.

5.244 Microinfinity CruizCore XG1300L

sensor scale factor constants Constants for setting the scale factor of the CruizCore XG1300L sensor.

Defines

- #define [XG1300L_SCALE_2G](#) 0x01
- #define [XG1300L_SCALE_4G](#) 0x02

- #define XG1300L_SCALE_8G 0x04

5.244.1 Detailed Description

sensor scale factor constants Constants for setting the scale factor of the CruzCore XG1300L sensor.

5.244.2 Define Documentation

5.244.2.1 #define XG1300L_SCALE_2G 0x01

Select +/- 2G accelerometer range.

5.244.2.2 #define XG1300L_SCALE_4G 0x02

Select +/- 4G accelerometer range.

5.244.2.3 #define XG1300L_SCALE_8G 0x04

Select +/- 8G accelerometer range.

5.245 Data type limits

Constants that define various data type limits.

Defines

- #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

5.245.1 Detailed Description

Constants that define various data type limits.

5.245.2 Define Documentation

5.245.2.1 #define CHAR_BIT 8

The number of bits in the char type

5.245.2.2 #define CHAR_MAX 127

The maximum value of the char type

5.245.2.3 #define CHAR_MIN -128

The minimum value of the char type

5.245.2.4 #define INT_MAX 32767

The maximum value of the int type

5.245.2.5 #define INT_MIN -32768

The minimum value of the int type

5.245.2.6 #define LONG_MAX 2147483647

The maximum value of the long type

5.245.2.7 #define LONG_MIN -2147483648

The minimum value of the long type

5.245.2.8 #define RAND_MAX 2147483646

The maximum long random number returned by rand

5.245.2.9 #define SCHAR_MAX 127

The maximum value of the signed char type

5.245.2.10 #define SCHAR_MIN -128

The minimum value of the signed char type

5.245.2.11 #define SHRT_MAX 32767

The maximum value of the short type

5.245.2.12 #define SHRT_MIN -32768

The minimum value of the short type

5.245.2.13 #define UCHAR_MAX 255

The maximum value of the unsigned char type

5.245.2.14 #define UINT_MAX 65535

The maximum value of the unsigned int type

5.245.2.15 #define ULONG_MAX 4294967295

The maximum value of the unsigned long type

5.245.2.16 #define USHRT_MAX 65535

The maximum value of the unsigned short type

5.246 Graphics library begin modes

Constants that are used to specify the polygon surface begin mode.

Defines

- #define `GL_POLYGON` 1
- #define `GL_LINE` 2
- #define `GL_POINT` 3
- #define `GL_CIRCLE` 4

5.246.1 Detailed Description

Constants that are used to specify the polygon surface begin mode.

5.246.2 Define Documentation**5.246.2.1 #define GL_CIRCLE 4**

Use circle mode.

5.246.2.2 #define GL_LINE 2

Use line mode.

5.246.2.3 #define GL_POINT 3

Use point mode.

5.246.2.4 #define GL_POLYGON 1

Use polygon mode.

5.247 Graphics library actions

Constants that are used to specify a graphics library action.

Defines

- #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

5.247.1 Detailed Description

Constants that are used to specify a graphics library action.

5.247.2 Define Documentation

5.247.2.1 #define GL_ROTATE_X 4

Rotate around the X axis.

5.247.2.2 #define GL_ROTATE_Y 5

Rotate around the Y axis.

5.247.2.3 #define GL_ROTATE_Z 6

Rotate around the Z axis.

5.247.2.4 #define GL_SCALE_X 7

Scale along the X axis.

5.247.2.5 #define GL_SCALE_Y 8

Scale along the Y axis.

5.247.2.6 #define GL_SCALE_Z 9

Scale along the Z axis.

5.247.2.7 #define GL_TRANSLATE_X 1

Translate along the X axis.

5.247.2.8 #define GL_TRANSLATE_Y 2

Translate along the Y axis.

5.247.2.9 #define GL_TRANSLATE_Z 3

Translate along the Z axis.

5.248 Graphics library settings

Constants that are used to configure the graphics library settings.

Defines

- #define [GL_CIRCLE_SIZE](#) 1
- #define [GL_CULL_MODE](#) 2
- #define [GL_CAMERA_DEPTH](#) 3
- #define [GL_ZOOM_FACTOR](#) 4

5.248.1 Detailed Description

Constants that are used to configure the graphics library settings.

5.248.2 Define Documentation**5.248.2.1 #define GL_CAMERA_DEPTH 3**

Set the camera depth.

5.248.2.2 #define GL_CIRCLE_SIZE 1

Set the circle size.

5.248.2.3 #define GL_CULL_MODE 2

Set the cull mode.

5.248.2.4 #define GL_ZOOM_FACTOR 4

Set the zoom factor.

5.249 Graphics library cull mode

Constants to use when setting the graphics library cull mode.

Defines

- #define **GL_CULL_BACK** 2
- #define **GL_CULL_FRONT** 3
- #define **GL_CULL_NONE** 4

5.249.1 Detailed Description

Constants to use when setting the graphics library cull mode.

5.249.2 Define Documentation

5.249.2.1 #define GL_CULL_BACK 2

Cull lines in back.

5.249.2.2 #define GL_CULL_FRONT 3

Cull lines in front.

5.249.2.3 #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) 2009-2010 John Hansen. All Rights Reserved.

Author:

John Hansen (bricxcc_at_comcast.net)

Date:

2011-10-10

Version:

9

6.2 NBCCommon.h File Reference

Constants and macros common to both NBC and NXC.

Defines

- #define TRUE 1
- #define FALSE 0
- #define NA 0xFFFF
- #define RC_PROP_BTTONOFF 0x0
- #define RC_PROP_SOUND_LEVEL 0x1
- #define RC_PROP_SLEEP_TIMEOUT 0x2
- #define RC_PROP_DEBUGGING 0xF
- #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 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 [CommHSControl](#) 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 **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 `IOCTRL_POWERDOWN` 0x5A00
- #define `IOCTRL_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_NMOREFILES` 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_OPENWRITELINEAR 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_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_DIGI0 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 **OutputOffsetRegIPParameter**(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 `LowSpeedOffsetInBufBuff`(p) (((p)*19)+0)
- #define `LowSpeedOffsetInBufInPtr`(p) (((p)*19)+16)
- #define `LowSpeedOffsetInBufOutPtr`(p) (((p)*19)+17)
- #define `LowSpeedOffsetInBufBytesToRx`(p) (((p)*19)+18)
- #define `LowSpeedOffsetOutBufBuff`(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_SINGLESHT 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_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 BT_ARM_OFF 0
- #define BT_ARM_CMD_MODE 1
- #define BT_ARM_DATA_MODE 2
- #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
- #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 0xFFFF
- #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_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
- #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 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_ANGLE_MODE_NORMAL 0x00
- #define HT_ANGLE_MODE_CALIBRATE 0x43
- #define HT_ANGLE_MODE_RESET 0x52
- #define HT_ANGLE_REG_MODE 0x41
- #define HT_ANGLE_REG_DCDIR 0x42
- #define HT_ANGLE_REG_DC01 0x43
- #define HT_ANGLE_REG_DC02 0x44
- #define HT_ANGLE_REG_DC03 0x45
- #define HT_ANGLE_REG_DC04 0x46
- #define HT_ANGLE_REG_DC05 0x47
- #define HT_ANGLE_REG_DCAVG 0x48
- #define HT_ANGLE_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_SAWNFWAVE 4
- #define DAC_MODE_TRIANGLEWAVE 5
- #define DAC_MODE_PWMVOLTAGE 6
- #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_S0 0x01
- #define STROBE_S1 0x02
- #define STROBE_S2 0x04
- #define STROBE_S3 0x08
- #define STROBE_READ 0x10
- #define STROBE_WRITE 0x20

- #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 **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 **NXT HID MOD_RIGHT_ALT** 0x40
- #define **NXT HID MOD_RIGHT_GUI** 0x80
- #define **NXT HID CMD ASCII** 0x41
- #define **NXT HID CMD DIRECT** 0x44
- #define **NXT HID CMD TRANSMIT** 0x54
- #define **NXT PM REG_CMD** 0x41
- #define **NXT PM REG_CURRENT** 0x42
- #define **NXT PM REG_VOLTAGE** 0x44
- #define **NXT PM REG_CAPACITY** 0x46
- #define **NXT PM REG_POWER** 0x48
- #define **NXT PM REG_TOTALPOWER** 0x4A
- #define **NXT PM REG_MAXCURRENT** 0x4E
- #define **NXT PM REG_MINCURRENT** 0x50
- #define **NXT PM REG_MAXVOLTAGE** 0x52
- #define **NXT PM REG_MINVOLTAGE** 0x54
- #define **NXT PM REG_TIME** 0x56
- #define **NXT PM REG_USERGAIN** 0x5A
- #define **NXT PM REG_GAIN** 0x5E
- #define **NXT PM REG_ERRORCOUNT** 0x5F
- #define **NXT PM CMD_RESET** 0x52
- #define **NXT SE ZONE_NONE** 0
- #define **NXT SE ZONE_FRONT** 1
- #define **NXT SE ZONE_LEFT** 2
- #define **NXT SE ZONE_RIGHT** 3
- #define **NXT LL REG_CMD** 0x41
- #define **NXT LL REG_STEERING** 0x42
- #define **NXT LL REG_AVERAGE** 0x43
- #define **NXT LL REG_RESULT** 0x44
- #define **NXT LL REG_SETPOINT** 0x45
- #define **NXT LL REG_KP_VALUE** 0x46
- #define **NXT LL REG_KI_VALUE** 0x47
- #define **NXT LL REG_KD_VALUE** 0x48
- #define **NXT LL REG_CALIBRATED** 0x49
- #define **NXT LL REG_WHITELIMITS** 0x51
- #define **NXT LL REG_BLACKLIMITS** 0x59
- #define **NXT LL REG_KP_FACTOR** 0x61
- #define **NXT LL REG_KI_FACTOR** 0x62
- #define **NXT LL REG_KD_FACTOR** 0x63
- #define **NXT LL REG_WHITEDATA** 0x64
- #define **NXT LL REG_BLACKDATA** 0x6C
- #define **NXT LL REG_RAWVOLTAGE** 0x74
- #define **NXT LL CMD_USA** 0x41
- #define **NXT LL 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 **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 **RICOOpDescription**(_Options, _Width, _Height) 8, 0, 0, 0, 0, (_Options)&0xFF, (_Options)>>8, (_Width)&0xFF, (_Width)>>8, (_Height)&0xFF, (_Height)>>8

Output an RIC Description opcode.

- #define **RICOOpCopyBits**(_CopyOptions, _DataAddr, _SrcRect, _DstPoint) 18, 0, 3, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, (_DataAddr)&0xFF, (_DataAddr)>>8, _SrcRect, _DstPoint

Output an RIC CopyBits opcode.

- #define **RICOOpPixel**(_CopyOptions, _Point, _Value) 10, 0, 4, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point, (_Value)&0xFF, (_Value)>>8

Output an RIC Pixel opcode.

- #define **RICOOpLine**(_CopyOptions, _Point1, _Point2) 12, 0, 5, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point1, _Point2

Output an RIC Line opcode.

- #define **RICOOpRect**(_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 **RICOOpCircle**(_CopyOptions, _Point, _Radius) 10, 0, 7, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point, (_Radius)&0xFF, (_Radius)>>8

Output an RIC Circle opcode.

- #define **RICOOpNumBox**(_CopyOptions, _Point, _Value) 10, 0, 8, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point, (_Value)&0xFF, (_Value)>>8

Output an RIC NumBox opcode.

- #define **RICOOpSprite**(_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 **RICOOpVarMap**(_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 **RICOOpPolygon**(_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 **RICOOpEllipse**(_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) 2009-2011 John Hansen. All Rights Reserved.

Author:

John Hansen (bricxcc_at_comcast.net)

Date:

2011-10-10

Version:

69

6.2.2 Define 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 BTNEXIT 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 CommLSWrite 21

Write to a lowspeed (aka I2C) device

6.2.2.114 #define CommLSWriteEx 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_SAWNEGWAVE 4

Negative going sawtooth output.

6.2.2.189 #define DAC_MODE_SAWPOSWAVE 3

Positive going sawtooth output.

6.2.2.190 #define DAC_MODE_SINEWAVE 1

Sine wave output.

6.2.2.191 #define DAC_MODE_SQUAREWAVE 2

Square wave output.

6.2.2.192 #define DAC_MODE_TRIANGLEWAVE 5

Triangle wave output.

6.2.2.193 #define DATA_MODE_GPS 0x01

Use GPS data mode

6.2.2.194 #define DATA_MODE_MASK 0x07

A mask for the data mode bits.

6.2.2.195 #define DATA_MODE_NXT 0x00

Use NXT data mode

6.2.2.196 #define DATA_MODE_RAW 0x02

Use RAW data mode

6.2.2.197 #define DATA_MODE_UPDATE 0x08

Indicates that the data mode has been changed.

6.2.2.198 #define DatalogGetTimes 45

Get datalog timing information

6.2.2.199 #define DatalogWrite 44

Write to the datalog

6.2.2.200 #define DEGREES_PER_RADIAN 180/PI

Used for converting from radians to degrees

6.2.2.201 #define DGPS_REG_DISTANCE 0x08

Read distance to current waypoint in meters.

6.2.2.202 #define DGPS_REG_HEADING 0x07

Read heading in degrees.

6.2.2.203 #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.

6.2.2.204 #define DGPS_REG_LATITUDE 0x02

Read integer latitude.(dddddd; Positive = North; Negative = South).

6.2.2.205 #define DGPS_REG_LONGITUDE 0x04

Read integer longitude (dddddd; Positive = East; Negative = West).

6.2.2.206 #define DGPS_REG_SETLATITUDE 0x0B

Set waypoint latitude as a 4 byte integer.

6.2.2.207 #define DGPS_REG_SETLONGITUDE 0x0C

Set waypoint longitude as a 4 byte integer.

6.2.2.208 #define DGPS_REG_STATUS 0x01

Read status of the GPS (0 - invalid signal, 1 - valid signal).

6.2.2.209 #define DGPS_REG_TIME 0x00

Read time in UTC (hhmmss).

6.2.2.210 #define DGPS_REG_VELOCITY 0x06

Read velocity in cm/s.

6.2.2.211 #define DGPS_REG_WAYANGLE 0x09

Read angle to current waypoint in degrees.

6.2.2.212 #define DI_ADDR_ACCL 0x3A

Dexter Industries DIMU Accelerometer I2C address

6.2.2.213 #define DI_ADDR_DGPS 0x06

Dexter Industries DGPS I2C address

6.2.2.214 #define DI_ADDR_GYRO 0xD2

Dexter Industries DIMU Gyro I2C address

6.2.2.215 #define DIACCL_CTRL1_filt_BW125 0x80

Accelerometer digital filter band width is 125 Hz.

6.2.2.216 #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)

6.2.2.217 #define DIACCL_CTRL1_LEVELPULSE 0x00

Accelerometer INT1 register is detecting Level while INT2 is detecting pulse

6.2.2.218 #define DIACCL_CTRL1_NO_XDETECT 0x08

Accelerometer disable x-axis detection.

6.2.2.219 #define DIACCL_CTRL1_NO_YDETECT 0x10

Accelerometer disable y-axis detection.

6.2.2.220 #define DIACCL_CTRL1_NO_ZDETECT 0x20

Accelerometer disable z-axis detection.

6.2.2.221 #define DIACCL_CTRL1_PULSELEVEL 0x02

Accelerometer INT1 Register is detecting Pulse while INT2 is detecting Level

6.2.2.222 #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.

6.2.2.223 #define DIACCL_CTRL1_THRESH_INT 0x40

Accelerometer threshold value can be an integer.

6.2.2.224 #define DIACCL_CTRL2_DETPOL_NEGAND 0x02

Accelerometer pulse detection polarity is negative and detecting condition is AND all 3 axes

6.2.2.225 #define DIACCL_CTRL2_DRIVE_STRONG 0x04

Accelerometer strong drive strength on SDA/SDO pin

6.2.2.226 #define DIACCL_CTRL2_LVLPOLE_NEGAND 0x01

Accelerometer level detection polarity is negative and detecting condition is AND all 3 axes

6.2.2.227 #define DIACCL_INTERRUPT_LATCH_CLEAR1 0x01

Accelerometer clear interrupt 1

6.2.2.228 #define DIACCL_INTERRUPT_LATCH_CLEAR2 0x02

Accelerometer clear interrupt 2

6.2.2.229 #define DIACCL_MODE_GLVL2 0x04

Accelerometer 2G measurement range

6.2.2.230 #define DIACCL_MODE_GLVL4 0x08

Accelerometer 4G measurement range

6.2.2.231 #define DIACCL_MODE_GLVL8 0x00

Accelerometer 8G measurement range

6.2.2.232 #define DIACCL_MODE_LVLDETECT 0x02

Accelerometer level detect mode

6.2.2.233 #define DIACCL_MODE_MEASURE 0x01

Accelerometer measurement mode

6.2.2.234 #define DIACCL_MODE_PLSDETECT 0x03

Accelerometer pulse detect mode

6.2.2.235 #define DIACCL_MODE_STANDBY 0x00

Accelerometer standby mode

6.2.2.236 #define DIACCL_REG_CTRL1 0x18

Accelerometer control register 1 (read/write)

6.2.2.237 #define DIACCL_REG_CTRL2 0x19

Accelerometer control register 1 (read/write)

6.2.2.238 #define DIACCL_REG_DETECTSRC 0x0A

Accelerometer detection source register (read only)

6.2.2.239 #define DIACCL_REG_I2CADDR 0x0D

Accelerometer I2C address register (read only)

6.2.2.240 #define DIACCL_REG_INTLATCH 0x17

Accelerometer interrupt latch reset register (read/write)

6.2.2.241 #define DIACCL_REG_LATENCYTM 0x1D

Accelerometer latency time value register (read/write)

6.2.2.242 #define DIACCL_REG_LVLDETTTHR 0x1A

Accelerometer level detection threshold limit value register (read/write)

6.2.2.243 #define DIACCL_REG_MODECTRL 0x16

Accelerometer mode control register (read/write)

6.2.2.244 #define DIACCL_REG_OUTTEMP 0x0B

Accelerometer temperature output register (read only)

6.2.2.245 #define DIACCL_REG_PLSDETTTHR 0x1B

Accelerometer pulse detection threshold limit value register (read/write)

6.2.2.246 #define DIACCL_REG_PLSDURVAL 0x1C

Accelerometer pulse duration value register (read/write)

6.2.2.247 #define DIACCL_REG_STATUS 0x09

Accelerometer status register (read only)

6.2.2.248 #define DIACCL_REG_TIMEWINDOW 0x1E

Accelerometer time window for 2nd pulse value register (read/write)

6.2.2.249 #define DIACCL_REG_USERINFO 0x0E

Accelerometer user information register (read only)

6.2.2.250 #define DIACCL_REG_WHOAMI 0x0F

Accelerometer device identification register (read only)

6.2.2.251 #define DIACCL_REG_X8 0x06

Accelerometer x-axis 8-bit register (read only)

6.2.2.252 #define DIACCL_REG_XHIGH 0x01

Accelerometer x-axis high byte register (read only)

6.2.2.253 #define DIACCL_REG_XHIGHDRIFT 0x11

Accelerometer x-axis offset drift high byte register (read/write)

6.2.2.254 #define DIACCL_REG_XLOW 0x00

Accelerometer x-axis low byte register (read only)

6.2.2.255 #define DIACCL_REG_XLOWDRIFT 0x10

Accelerometer x-axis offset drift low byte register (read/write)

6.2.2.256 #define DIACCL_REG_Y8 0x07

Accelerometer x-axis 8-bit register (read only)

6.2.2.257 #define DIACCL_REG_YHIGH 0x03

Accelerometer y-axis high byte register (read only)

6.2.2.258 #define DIACCL_REG_YHIGHDRIFT 0x13

Accelerometer y-axis offset drift high byte register (read/write)

6.2.2.259 #define DIACCL_REG_YLOW 0x02

Accelerometer y-axis low byte register (read only)

6.2.2.260 #define DIACCL_REG_YLOWDRIFT 0x12

Accelerometer y-axis offset drift low byte register (read/write)

6.2.2.261 #define DIACCL_REG_Z8 0x08

Accelerometer x-axis 8-bit register (read only)

6.2.2.262 #define DIACCL_REG_ZHIGH 0x05

Accelerometer z-axis high byte register (read only)

6.2.2.263 #define DIACCL_REG_ZHIGHDRIFT 0x15

Accelerometer z-axis offset drift high byte register (read/write)

6.2.2.264 #define DIACCL_REG_ZLOW 0x04

Accelerometer z-axis low byte register (read only)

6.2.2.265 #define DIACCL_REG_ZLOWDRIFT 0x14

Accelerometer z-axis offset drift low byte register (read/write)

6.2.2.266 #define DIACCL_STATUS_DATAOVER 0x02

Accelerometer data is overwritten

6.2.2.267 #define DIACCL_STATUS_DATAREADY 0x01

Accelerometer data is ready

6.2.2.268 #define DIACCL_STATUS_PARITYERR 0x04

Accelerometer parity error is detected in trim data

6.2.2.269 #define DIGI_PIN0 0x01

Access digital pin 0 (B0)

6.2.2.270 #define DIGI_PIN1 0x02

Access digital pin 1 (B1)

6.2.2.271 #define DIGI_PIN2 0x04

Access digital pin 2 (B2)

6.2.2.272 #define DIGI_PIN3 0x08

Access digital pin 3 (B3)

6.2.2.273 #define DIGI_PIN4 0x10

Access digital pin 4 (B4)

6.2.2.274 #define DIGI_PIN5 0x20

Access digital pin 5 (B5)

6.2.2.275 #define DIGI_PIN6 0x40

Access digital pin 6 (B6)

6.2.2.276 #define DIGI_PIN7 0x80

Access digital pin 7 (B7)

6.2.2.277 #define DIGYRO_CTRL1_BANDWIDTH_1 0x00

Gyro LPF2 cut-off frequency bandwidth level 1 (12.5hz, 12.5hz, 20hz, 30hz)

6.2.2.278 #define DIGYRO_CTRL1_BANDWIDTH_2 0x10

Gyro LPF2 cut-off frequency bandwidth level 2 (12.5hz, 25hz, 50hz, 70hz)

6.2.2.279 #define DIGYRO_CTRL1_BANDWIDTH_3 0x20

Gyro LPF2 cut-off frequency bandwidth level 3 (20hz, 25hz, 50hz, 110hz)

6.2.2.280 #define DIGYRO_CTRL1_BANDWIDTH_4 0x30

Gyro LPF2 cut-off frequency bandwidth level 4 (30hz, 35hz, 50hz, 110hz)

6.2.2.281 #define DIGYRO_CTRL1_DATARATE_100 0x00

Gyro output data rate 100 hz

6.2.2.282 #define DIGYRO_CTRL1_DATARATE_200 0x40

Gyro output data rate 200 hz

6.2.2.283 #define DIGYRO_CTRL1_DATARATE_400 0x80

Gyro output data rate 400 hz

6.2.2.284 #define DIGYRO_CTRL1_DATARATE_800 0xC0

Gyro output data rate 800 hz

6.2.2.285 #define DIGYRO_CTRL1_NORMAL 0x08

Gyro disable power down mode

6.2.2.286 #define DIGYRO_CTRL1_POWERDOWN 0x00

Gyro enable power down mode

6.2.2.287 #define DIGYRO_CTRL1_XENABLE 0x01

Gyro enable X axis

6.2.2.288 #define DIGYRO_CTRL1_YENABLE 0x02

Gyro enable Y axis

6.2.2.289 #define DIGYRO_CTRL1_ZENABLE 0x04

Gyro enable Z axis

6.2.2.290 #define DIGYRO_CTRL2_CUTOFF_FREQ_001 0x09

Gyro high pass filter cutoff frequency 0.01 hz

6.2.2.291 #define DIGYRO_CTRL2_CUTOFF_FREQ_002 0x08

Gyro high pass filter cutoff frequency 0.02 hz

6.2.2.292 #define DIGYRO_CTRL2_CUTOFF_FREQ_005 0x07

Gyro high pass filter cutoff frequency 0.05 hz

6.2.2.293 #define DIGYRO_CTRL2_CUTOFF_FREQ_01 0x06

Gyro high pass filter cutoff frequency 0.1 hz

6.2.2.294 #define DIGYRO_CTRL2_CUTOFF_FREQ_02 0x05

Gyro high pass filter cutoff frequency 0.2 hz

6.2.2.295 #define DIGYRO_CTRL2_CUTOFF_FREQ_05 0x04

Gyro high pass filter cutoff frequency 0.5 hz

6.2.2.296 #define DIGYRO_CTRL2_CUTOFF_FREQ_1 0x03

Gyro high pass filter cutoff frequency 1 hz

6.2.2.297 #define DIGYRO_CTRL2_CUTOFF_FREQ_2 0x02

Gyro high pass filter cutoff frequency 2 hz

6.2.2.298 #define DIGYRO_CTRL2_CUTOFF_FREQ_4 0x01

Gyro high pass filter cutoff frequency 4 hz

6.2.2.299 #define DIGYRO_CTRL2_CUTOFF_FREQ_8 0x00

Gyro high pass filter cutoff frequency 8 hz

6.2.2.300 #define DIGYRO_CTRL2_HPMODE_AUTOINT 0x30

Gyro high pass filter autoreset on interrupt event mode

6.2.2.301 #define DIGYRO_CTRL2_HPMODE_NORMAL 0x20

Gyro high pass filter normal mode

6.2.2.302 #define DIGYRO_CTRL2_HPMODE_REFSIG 0x10

Gyro high pass filter reference signal mode

6.2.2.303 #define DIGYRO_CTRL2_HPMODE_RESET 0x00

Gyro high pass filter reset mode

6.2.2.304 #define DIGYRO_CTRL3_INT1_BOOT 0x40

Gyro boot status available on INT1

6.2.2.305 #define DIGYRO_CTRL3_INT1_ENABLE 0x80

Gyro interrupt enable on INT1 pin

6.2.2.306 #define DIGYRO_CTRL3_INT1_LOWACTIVE 0x20

Gyro interrupt active low on INT1

6.2.2.307 #define DIGYRO_CTRL3_INT2_DATAREADY 0x08

Gyro data ready on DRDY/INT2

6.2.2.308 #define DIGYRO_CTRL3_INT2_EMPTY 0x01

Gyro FIFO empty interrupt on DRDY/INT2

6.2.2.309 #define DIGYRO_CTRL3_INT2_OVERRUN 0x02

Gyro FIFO overrun interrupt on DRDY/INT2

6.2.2.310 #define DIGYRO_CTRL3_INT2_WATERMARK 0x04

Gyro FIFO watermark interrupt on DRDY/INT2

6.2.2.311 #define DIGYRO_CTRL3_OPENDRAIN 0x10

Gyro use open drain rather than push-pull

6.2.2.312 #define DIGYRO_CTRL4_BIGENDIAN 0x40

Gyro use big endian - MSB/LSB rather than LSB/MSB in output registers

6.2.2.313 #define DIGYRO_CTRL4_BLOCKDATA 0x80

Gyro block data update - output registers are not updated until MSB and LSB reading

6.2.2.314 #define DIGYRO_CTRL4_SCALE_2000 0x30

Gyro 2000 degrees per second scale

6.2.2.315 #define DIGYRO_CTRL4_SCALE_250 0x00

Gyro 250 degrees per second scale

6.2.2.316 #define DIGYRO_CTRL4_SCALE_500 0x10

Gyro 500 degrees per second scale

6.2.2.317 #define DIGYRO_CTRL5_FIFOENABLE 0x40

Gyro enable FIFO

6.2.2.318 #define DIGYRO_CTRL5_HPENABLE 0x10

Gyro enable high pass filter

6.2.2.319 #define DIGYRO_CTRL5_INT1_SEL_1 0x00

Gyro non-high-pass-filtered data are used for interrupt generation

6.2.2.320 #define DIGYRO_CTRL5_INT1_SEL_2 0x04

Gyro high-pass-filtered data are used for interrupt generation

6.2.2.321 #define DIGYRO_CTRL5_INT1_SEL_3 0x08

Gyro low-pass-filtered data are used for interrupt generation

6.2.2.322 #define DIGYRO_CTRL5_OUT_SEL_1 0x00

Gyro data in data registers and FIFO are not high-pass filtered

6.2.2.323 #define DIGYRO_CTRL5_OUT_SEL_2 0x01

Gyro data in data registers and FIFO are high-pass filtered

6.2.2.324 #define DIGYRO_CTRL5_OUT_SEL_3 0x02

Gyro data in data registers and FIFO are low-pass filtered by LPF2

6.2.2.325 #define DIGYRO_CTRL5_REBOOTMEM 0x80

Gyro reboot memory content

6.2.2.326 #define DIGYRO_FIFOCTRL_BYPASS 0x00

Gyro FIFO bypass mode

6.2.2.327 #define DIGYRO_FIFOCTRL_BYPASS2STREAM 0x80

Gyro FIFO bypass-to-stream mode

6.2.2.328 #define DIGYRO_FIFOCTRL_FIFO 0x20

Gyro FIFO mode

6.2.2.329 #define DIGYRO_FIFOCTRL_STREAM 0x40

Gyro FIFO stream mode

6.2.2.330 #define DIGYRO_FIFOCTRL_STREAM2FIFO 0x60

Gyro FIFO stream-to-FIFO mode

6.2.2.331 #define DIGYRO_FIFOCTRL_WATERMARK_MASK 0x1F

Gyro FIFO threshold. Watermark level setting mask (values from 0x00 to 0x1F)

6.2.2.332 #define DIGYRO_REG_CTRL1 0x20

Gyro control register 1

6.2.2.333 #define DIGYRO_REG_CTRL1AUTO 0xA0

Gyro control register 1 - auto increment write

6.2.2.334 #define DIGYRO_REG_CTRL2 0x21

Gyro control register 2

6.2.2.335 #define DIGYRO_REG_CTRL3 0x22

Gyro control register 3

6.2.2.336 #define DIGYRO_REG_CTRL4 0x23

Gyro control register 4

6.2.2.337 #define DIGYRO_REG_CTRL5 0x24

Gyro control register 5

6.2.2.338 #define DIGYRO_REG_FIFOCTRL 0x2E

Gyro FIFO control register

6.2.2.339 #define DIGYRO_REG_FIFOSRC 0x2F

Gyro FIFO source register (read only)

6.2.2.340 #define DIGYRO_REG_INT1_CFG 0x30

Gyro interrupt 1 config register

6.2.2.341 #define DIGYRO_REG_INT1_DUR 0x38

Gyro interrupt 1 duration register

6.2.2.342 #define DIGYRO_REG_INT1_SRC 0x31

Gyro interrupt 1 source register

6.2.2.343 #define DIGYRO_REG_INT1_XHI 0x32

Gyro interrupt 1 x-axis high threshold register

6.2.2.344 #define DIGYRO_REG_INT1_XLO 0x33

Gyro interrupt 1 x-axis low threshold register

6.2.2.345 #define DIGYRO_REG_INT1_YHI 0x34

Gyro interrupt 1 y-axis high threshold register

6.2.2.346 #define DIGYRO_REG_INT1_YLO 0x35

Gyro interrupt 1 y-axis low threshold register

6.2.2.347 #define DIGYRO_REG_INT1_ZHI 0x36

Gyro interrupt 1 z-axis high threshold register

6.2.2.348 #define DIGYRO_REG_INT1_ZLO 0x37

Gyro interrupt 1 z-axis low threshold register

6.2.2.349 #define DIGYRO_REG_OUTTEMP 0x26

Gyro temperature register (read only) - stores temperature data

6.2.2.350 #define DIGYRO_REG_REFERENCE 0x25

Gyro reference register - stores the reference value used for interrupt generation

6.2.2.351 #define DIGYRO_REG_STATUS 0x27

Gyro status register (read only)

6.2.2.352 #define DIGYRO_REG_TEMPAUTO 0xA6

Gyro temperature register - read burst mode (read only)

6.2.2.353 #define DIGYRO_REG_WHOAMI 0x0F

Gyro device identification register (read only)

6.2.2.354 #define DIGYRO_REG_XHIGH 0x29

Gyro x-axis high byte register (read only)

6.2.2.355 #define DIGYRO_REG_XLOW 0x28

Gyro x-axis low byte register (read only)

6.2.2.356 #define DIGYRO_REG_XLOWBURST 0xA8

Gyro x-axis low byte register - read burst mode (read only)

6.2.2.357 #define DIGYRO_REG_YHIGH 0x2B

Gyro y-axis high byte register (read only)

6.2.2.358 #define DIGYRO_REG_YLOW 0x2A

Gyro y-axis low byte register (read only)

6.2.2.359 #define DIGYRO_REG_YLOWBURST 0xAA

Gyro y-axis low byte register - read burst mode (read only)

6.2.2.360 #define DIGYRO_REG_ZHIGH 0x2D

Gyro z-axis high byte register (read only)

6.2.2.361 #define DIGYRO_REG_ZLOW 0x2C

Gyro z-axis low byte register (read only)

6.2.2.362 #define DIGYRO_REG_ZLOWBURST 0xAC

Gyro y-axis low byte register - read burst mode (read only)

6.2.2.363 #define DIGYRO_STATUS_XDATA 0x01

Gyro X-axis new data available

6.2.2.364 #define DIGYRO_STATUS_XOVER 0x10

Gyro X-axis data overrun - new data for the X-axis has overwritten the previous one

6.2.2.365 #define DIGYRO_STATUS_XYZDATA 0x08

Gyro X, Y, or Z-axis new data available - a new set of data is available

6.2.2.366 #define DIGYRO_STATUS_XYZOVER 0x80

Gyro X, Y, or Z-axis data overrun - new data has overwritten the previous one before it was read

6.2.2.367 #define DIGYRO_STATUS_YDATA 0x02

Gyro Y-axis new data available

6.2.2.368 #define DIGYRO_STATUS_YOVER 0x20

Gyro Y-axis data overrun - new data for the Y-axis has overwritten the previous one

6.2.2.369 #define DIGYRO_STATUS_ZDATA 0x04

Gyro Z-axis new data available

6.2.2.370 #define DIGYRO_STATUS_ZOVER 0x40

Gyro Z-axis data overrun - new data for the Z-axis has overwritten the previous one

6.2.2.371 #define DISPLAY_BUSY 0x80

R - Refresh in progress

6.2.2.372 #define DISPLAY_CHAR 0x04

W - draw char (actual font) (CMD,TRUE,X1,Y1,Char,x)

6.2.2.373 #define DISPLAY_CONTRAST_DEFAULT 0x5A

Default display contrast value

6.2.2.374 #define DISPLAY_CONTRAST_MAX 0x7F

Maximum display contrast value

6.2.2.375 #define DISPLAY_ERASE_ALL 0x00

W - erase entire screen (CMD,x,x,x,x,x)

6.2.2.376 #define DISPLAY_ERASE_LINE 0x05

W - erase a single line (CMD,x,LINE,x,x,x)

6.2.2.377 #define DISPLAY_FILL_REGION 0x06

W - fill screen region (CMD,TRUE/FALSE,X1,Y1,X2,Y2)

6.2.2.378 #define DISPLAY_FRAME 0x07

W - draw a frame (on/off) (CMD,TRUE/FALSE,X1,Y1,X2,Y2)

6.2.2.379 #define DISPLAY_HEIGHT 64

The height of the LCD screen in pixels

6.2.2.380 #define DISPLAY_HORIZONTAL_LINE 0x02

W - draw horizontal line (CMD,TRUE/FALSE,X1,Y1,X2,x)

6.2.2.381 #define DISPLAY_MENUICONS_X_DIFF 31**6.2.2.382 #define DISPLAY_MENUICONS_X_OFFSET 7****6.2.2.383 #define DISPLAY_MENUICONS_Y 40****6.2.2.384 #define DISPLAY_ON 0x01**

W - Display on

6.2.2.385 #define DISPLAY_PIXEL 0x01

W - set pixel (on/off) (CMD,TRUE/FALSE,X,Y,x,x)

6.2.2.386 #define DISPLAY_POPUP 0x08

W - Use popup display memory

6.2.2.387 #define DISPLAY_REFRESH 0x02

W - Enable refresh

6.2.2.388 #define DISPLAY_REFRESH_DISABLED 0x40

R - Refresh disabled

6.2.2.389 #define DISPLAY_VERTICAL_LINE 0x03

W - draw vertical line (CMD,TRUE/FALSE,X1,Y1,x,Y2)

6.2.2.390 #define DISPLAY_WIDTH 100

The width of the LCD screen in pixels

6.2.2.391 #define DisplayExecuteFunction 80

Execute one of the Display module's internal functions

6.2.2.392 #define DisplayModuleID 0x000A0001

The display module ID

6.2.2.393 #define DisplayModuleName "Display.mod"

The display module name

6.2.2.394 #define DisplayOffsetContrast 1719

Adjust the display contrast with this field

6.2.2.395 #define DisplayOffsetDisplay 104

Display content copied to physical display every 17 mS

6.2.2.396 #define DisplayOffsetEraseMask 4

Section erase mask (executed first)

6.2.2.397 #define DisplayOffsetFlags 117

Update flags enumerated above

6.2.2.398 #define DisplayOffsetNormal(l, w) (((l)*100)+(w)+119)

Raw display memory for normal screen

6.2.2.399 #define DisplayOffsetPBitmaps(p) (((p)*4)+68)

Pointer to free bitmap files

6.2.2.400 #define DisplayOffsetPFont 12

Pointer to font file

6.2.2.401 #define DisplayOffsetPFunc 0

Simple draw entry

6.2.2.402 #define DisplayOffsetPMenuIcons(p) (((p)*4)+88)

Pointer to menu icon images (NULL == none)

6.2.2.403 #define DisplayOffsetPMenuText 84

Pointer to menu icon text (NULL == none)

6.2.2.404 #define DisplayOffsetPopup(l, w) (((l)*100)+(w)+919)

Raw display memory for popup screen

6.2.2.405 #define DisplayOffsetPScreens(p) (((p)*4)+56)

Pointer to screen bitmap file

6.2.2.406 #define DisplayOffsetPStatusIcons 52

Pointer to status icon collection file

6.2.2.407 #define DisplayOffsetPStatusText 48

Pointer to status text string

6.2.2.408 #define DisplayOffsetPStepIcons 100

Pointer to step icon collection file

6.2.2.409 #define DisplayOffsetPTextLines(p) (((p)*4)+16)

Pointer to text strings

6.2.2.410 #define DisplayOffsetStatusIcons(p) ((p)+108)

Index in status icon collection file (index = 0 -> none)

6.2.2.411 #define DisplayOffsetStepIcons(p) ((p)+112)

Index in step icon collection file (index = 0 -> none)

6.2.2.412 #define DisplayOffsetTextLinesCenterFlags 118

Mask to center TextLines

6.2.2.413 #define DisplayOffsetUpdateMask 8

Section update mask (executed next)

6.2.2.414 #define DIST_CMD_CUSTOM 0x35

Set the DIST-Nx to a custom mode

6.2.2.415 #define DIST_CMD_GP2D12 0x31

Set the DIST-Nx to GP2D12 mode

6.2.2.416 #define DIST_CMD_GP2D120 0x32

Set the DIST-Nx to GP2D120 mode

6.2.2.417 #define DIST_CMD_GP2YA02 0x34

Set the DIST-Nx to GP2YA02 mode

6.2.2.418 #define DIST_CMD_GP2YA21 0x33

Set the DIST-Nx to GP2YA21 mode

6.2.2.419 #define DIST_REG_DIST 0x42

The DIST-Nx distance register

6.2.2.420 #define DIST_REG_DIST1 0x58

The DIST-Nx distance 1 register

6.2.2.421 #define DIST_REG_DIST_MAX 0x54

The DIST-Nx maximum distance register

6.2.2.422 #define DIST_REG_DIST_MIN 0x52

The DIST-Nx minimum distance register

6.2.2.423 #define DIST_REG_MODULE_TYPE 0x50

The DIST-Nx module type register

6.2.2.424 #define DIST_REG_NUM_POINTS 0x51

The DIST-Nx number of data points in Custom curve register

6.2.2.425 #define DIST_REG_VOLT 0x44

The DIST-Nx voltage register

6.2.2.426 #define DIST_REG_VOLT1 0x56

The DIST-Nx voltage 1 register

6.2.2.427 #define DRAW_OPT_CLEAR (0x0004)

Clear pixels while drawing (aka draw in white)

6.2.2.428 #define DRAW_OPT_CLEAR_EXCEPT_STATUS_-SCREEN (0x0002)

Clear the screen except for the status line before drawing

6.2.2.429 #define DRAW_OPT_CLEAR_PIXELS (0x0004)

Clear pixels while drawing (aka draw in white)

6.2.2.430 #define DRAW_OPT_CLEAR_SCREEN_MODES (0x0003)

Bit mask for the clear screen modes

6.2.2.431 #define DRAW_OPT_CLEAR_WHOLE_SCREEN (0x0001)

Clear the entire screen before drawing

6.2.2.432 #define DRAW_OPT_FILL_SHAPE (0x0020)

Fill the shape while drawing (rectangle, circle, ellipses, and polygon)

6.2.2.433 #define DRAW_OPT_FONT_DIR_B2TL (0x0100)

Font bottom to top left align

6.2.2.434 #define DRAW_OPT_FONT_DIR_B2TR (0x0140)

Font bottom to top right align

6.2.2.435 #define DRAW_OPT_FONT_DIR_L2RB (0x0000)

Font left to right bottom align

6.2.2.436 #define DRAW_OPT_FONT_DIR_L2RT (0x0040)

Font left to right top align

6.2.2.437 #define DRAW_OPT_FONT_DIR_R2LB (0x0080)

Font right to left bottom align

6.2.2.438 #define DRAW_OPT_FONT_DIR_R2LT (0x00C0)

Font right to left top align

6.2.2.439 #define DRAW_OPT_FONT_DIR_T2BL (0x0180)

Font top to bottom left align

6.2.2.440 #define DRAW_OPT_FONT_DIR_T2BR (0x01C0)

Font top to bottom right align

6.2.2.441 #define DRAW_OPT_FONT_DIRECTIONS (0x01C0)

Bit mask for the font direction bits

6.2.2.442 #define DRAW_OPT_FONT_WRAP (0x0200)

Option to have text wrap in [FontNumOut](#) and [FontTextOut](#) calls

6.2.2.443 #define DRAW_OPT_INVERT (0x0004)

Invert text or graphics

6.2.2.444 #define DRAW_OPT_LOGICAL_AND (0x0008)

Draw pixels using a logical AND operation

6.2.2.445 #define DRAW_OPT_LOGICAL_COPY (0x0000)

Draw pixels using a logical copy operation

6.2.2.446 #define DRAW_OPT_LOGICAL_OPERATIONS (0x0018)

Bit mask for the logical drawing operations

6.2.2.447 #define DRAW_OPT_LOGICAL_OR (0x0010)

Draw pixels using a logical OR operation

6.2.2.448 #define DRAW_OPT_LOGICAL_XOR (0x0018)

Draw pixels using a logical XOR operation

6.2.2.449 #define DRAW_OPT_NORMAL (0x0000)

Normal drawing

6.2.2.450 #define DRAW_OPT_POLYGON_POLYLINE (0x0400)

When drawing polygons, do not close (i.e., draw a polyline instead)

6.2.2.451 #define DrawCircle 16

Draw a circle on the LCD screen

6.2.2.452 #define DrawEllipse 94

Draw an ellipse on the LCD screen

6.2.2.453 #define DrawFont 95

Draw text using a custom RIC-based font to the LCD screen

6.2.2.454 #define DrawGraphic 18

Draw a graphic image on the LCD screen

6.2.2.455 #define DrawGraphicArray 92

Draw a graphic image from a byte array to the LCD screen

6.2.2.456 #define DrawLine 15

Draw a line on the LCD screen

6.2.2.457 #define DrawPoint 14

Draw a single pixel on the LCD screen

6.2.2.458 #define DrawPolygon 93

Draw a polygon on the LCD screen

6.2.2.459 #define DrawRect 17

Draw a rectangle on the LCD screen

6.2.2.460 #define DrawText 13

Draw text to one of 8 LCD lines

6.2.2.461 #define EMETER_REG_AIN 0x0c

The register address for amps in

6.2.2.462 #define EMETER_REG_AOUT 0x10

The register address for amps out

6.2.2.463 #define EMETER_REG_JOULES 0x12

The register address for joules

6.2.2.464 #define EMETER_REG_VIN 0x0a

The register address for voltage in

6.2.2.465 #define EMETER_REG_VOUT 0x0e

The register address for voltage out

6.2.2.466 #define EMETER_REG_WIN 0x14

The register address for watts in

6.2.2.467 #define EMETER_REG_WOUT 0x16

The register address for watts out

6.2.2.468 #define EOF -1

A constant representing end of file

6.2.2.469 #define ERR_ARG -1

0xFF Bad arguments

6.2.2.470 #define ERR_BAD_POOL_SIZE -10

0xF6 VarsCmd.PoolSize > POOL_MAX_SIZE

6.2.2.471 #define ERR_BAD_PTR -6

0xFA Someone passed us a bad pointer!

6.2.2.472 #define ERR_CLUMP_COUNT -7

0xF9 (FileClumpCount == 0 || FileClumpCount >= NOT_A_CLUMP)

6.2.2.473 #define ERR_COMM_BUFFER_FULL -34

0xDE No room in comm buffer

6.2.2.474 #define ERR_COMM_BUS_ERR -35

0xDD Something went wrong on the communications bus

6.2.2.475 #define ERR_COMM_CHAN_INVALID -33

0xDF Specified channel/connection is not valid

6.2.2.476 #define ERR_COMM_CHAN_NOT_READY -32

0xE0 Specified channel/connection not configured or busy

6.2.2.477 #define ERR_DEFAULT_OFFSETS -14

0xF2 (DefaultsOffset != FileOffsets.DynamicDefaults) || (DefaultsOffset + FileOffsets.DynamicDefaultsSize != FileOffsets.DSDefaultsSize)

6.2.2.478 #define ERR_FILE -3

0xFD Malformed file contents

6.2.2.479 #define ERR_INSANE_OFFSET -9

0xF7 CurrOffset != (DataSize - VarsCmd.CodespaceCount * 2)

6.2.2.480 #define ERR_INSTR -2

0xFE Illegal bytecode instruction

6.2.2.481 #define ERR_INVALID_FIELD -17

0xEF Attempted to access invalid field of a structure

6.2.2.482 #define ERR_INVALID_PORT -16

0xF0 Bad input or output port specified

6.2.2.483 #define ERR_INVALID_QUEUE -18

0xEE Illegal queue ID specified

6.2.2.484 #define ERR_INVALID_SIZE -19

0xED Illegal size specified

6.2.2.485 #define ERR_LOADER_ERR -11

0xF5 LOADER_ERR(LStatus) != SUCCESS || pData == NULL || DataSize == 0

6.2.2.486 #define ERR_MEM -5

0xFB Insufficient memory available

6.2.2.487 #define ERR_MEMMGR_FAIL -15

0xF1 (UBYTE *)VarsCmd.MemMgr.pDopeVectorArray != VarsCmd.pDataspace + DV_ARRAY[0].Offset

6.2.2.488 #define ERR_NO_ACTIVE_CLUMP -13

0xF3 VarsCmd.RunQ.Head == NOT_A_CLUMP

6.2.2.489 #define ERR_NO_CODE -8

0xF8 VarsCmd.CodespaceCount == 0

6.2.2.490 #define ERR_NO_PROG -20

0xEC No active program

6.2.2.491 #define ERR_NON_FATAL -16

Fatal errors are greater than this value

6.2.2.492 #define ERR_RC_BAD_PACKET -65

0xBF Clearly insane packet

6.2.2.493 #define ERR_RC_FAILED -67

0xBD Request failed (i.e. specified file not found)

6.2.2.494 #define ERR_RC_ILLEGAL_VAL -64

0xC0 Data contains out-of-range values

6.2.2.495 #define ERR_RC_UNKNOWN_CMD -66

0xBE Unknown command opcode

6.2.2.496 #define ERR_SPOTCHECK_FAIL -12

0xF4 ((UBYTE*)(VarsCmd.pCodespace) < pData) (c_cmd.c 1893)

6.2.2.497 #define ERR_VER -4

0xFC Version mismatch between firmware and compiler

6.2.2.498 #define FALSE 0

A false value

6.2.2.499 #define FileClose 5

Close the specified file

6.2.2.500 #define FileDelete 8

Delete a file

6.2.2.501 #define FileFindFirst 83

Start a search for a file using a filename pattern

6.2.2.502 #define FileFindNext 84

Continue searching for a file

6.2.2.503 #define FileOpenAppend 2

Open a file for appending to the end of the file

6.2.2.504 #define FileOpenRead 0

Open a file for reading

6.2.2.505 #define FileOpenReadLinear 87

Open a linear file for reading

6.2.2.506 #define FileOpenWrite 1

Open a file for writing (creates a new file)

6.2.2.507 #define FileOpenWriteLinear 85

Open a linear file for writing

6.2.2.508 #define FileOpenWriteNonLinear 86

Open a non-linear file for writing

6.2.2.509 #define FileRead 3

Read from the specified file

6.2.2.510 #define FileRename 7

Rename a file

6.2.2.511 #define FileResize 91

Resize a file (not yet implemented)

6.2.2.512 #define FileResolveHandle 6

Get a file handle for the specified filename if it is already open

6.2.2.513 #define FileSeek 90

Seek to a specific position in an open file

6.2.2.514 #define FileTell 98

Return the current file position in an open file

6.2.2.515 #define FileWrite 4

Write to the specified file

6.2.2.516 #define FRAME_SELECT 0

Center icon select frame

6.2.2.517 #define FREQUENCY_MAX 14080

Maximum frequency [Hz]

6.2.2.518 #define FREQUENCY_MIN 220

Minimum frequency [Hz]

6.2.2.519 #define GetStartTick 25

Get the current system tick count

6.2.2.520 #define GL_CAMERA_DEPTH 3

Set the camera depth.

6.2.2.521 #define GL_CIRCLE 4

Use circle mode.

6.2.2.522 #define GL_CIRCLE_SIZE 1

Set the circle size.

6.2.2.523 #define GL_CULL_BACK 2

Cull lines in back.

6.2.2.524 #define GL_CULL_FRONT 3

Cull lines in front.

6.2.2.525 #define GL_CULL_MODE 2

Set the cull mode.

6.2.2.526 #define GL_CULL_NONE 4

Do not cull any lines.

6.2.2.527 #define GL_LINE 2

Use line mode.

6.2.2.528 #define GL_POINT 3

Use point mode.

6.2.2.529 #define GL_POLYGON 1

Use polygon mode.

6.2.2.530 #define GL_ROTATE_X 4

Rotate around the X axis.

6.2.2.531 #define GL_ROTATE_Y 5

Rotate around the Y axis.

6.2.2.532 #define GL_ROTATE_Z 6

Rotate around the Z axis.

6.2.2.533 #define GL_SCALE_X 7

Scale along the X axis.

6.2.2.534 #define GL_SCALE_Y 8

Scale along the Y axis.

6.2.2.535 #define GL_SCALE_Z 9

Scale along the Z axis.

6.2.2.536 #define GL_TRANSLATE_X 1

Translate along the X axis.

6.2.2.537 #define GL_TRANSLATE_Y 2

Translate along the Y axis.

6.2.2.538 #define GL_TRANSLATE_Z 3

Translate along the Z axis.

6.2.2.539 #define GL_ZOOM_FACTOR 4

Set the zoom factor.

6.2.2.540 #define HS_ADDRESS_1 1

HsAddress device address 1

6.2.2.541 #define HS_ADDRESS_2 2

HsAddress device address 2

6.2.2.542 #define HS_ADDRESS_3 3

HsAddress device address 3

6.2.2.543 #define HS_ADDRESS_4 4

HsAddress device address 4

6.2.2.544 #define HS_ADDRESS_5 5

HsAddress device address 5

6.2.2.545 #define HS_ADDRESS_6 6

HsAddress device address 6

6.2.2.546 #define HS_ADDRESS_7 7

HsAddress device address 7

6.2.2.547 #define HS_ADDRESS_8 8

HsAddress device address 8

6.2.2.548 #define HS_ADDRESS_ALL 0

HsAddress all devices

6.2.2.549 #define HS_BAUD_115200 12

HsSpeed 115200 Baud

6.2.2.550 #define HS_BAUD_1200 0

HsSpeed 1200 Baud

6.2.2.551 #define HS_BAUD_14400 6

HsSpeed 14400 Baud

6.2.2.552 #define HS_BAUD_19200 7

HsSpeed 19200 Baud

6.2.2.553 #define HS_BAUD_230400 13

HsSpeed 230400 Baud

6.2.2.554 #define HS_BAUD_2400 1

HsSpeed 2400 Baud

6.2.2.555 #define HS_BAUD_28800 8

HsSpeed 28800 Baud

6.2.2.556 #define HS_BAUD_3600 2

HsSpeed 3600 Baud

6.2.2.557 #define HS_BAUD_38400 9

HsSpeed 38400 Baud

6.2.2.558 #define HS_BAUD_460800 14

HsSpeed 460800 Baud

6.2.2.559 #define HS_BAUD_4800 3

HsSpeed 4800 Baud

6.2.2.560 #define HS_BAUD_57600 10

HsSpeed 57600 Baud

6.2.2.561 #define HS_BAUD_7200 4

HsSpeed 7200 Baud

6.2.2.562 #define HS_BAUD_76800 11

HsSpeed 76800 Baud

6.2.2.563 #define HS_BAUD_921600 15

HsSpeed 921600 Baud

6.2.2.564 #define HS_BAUD_9600 5

HsSpeed 9600 Baud

6.2.2.565 #define HS_BAUD_DEFAULT 15

HsSpeed default Baud (921600)

6.2.2.566 #define HS_BYTES_REMAINING 16

HsState bytes remaining to be sent

6.2.2.567 #define HS_CMD_READY 0x04

A constant representing high speed direct command

6.2.2.568 #define HS_CTRL_EXIT 2

Ddisable the high speed port

6.2.2.569 #define HS_CTRL_INIT 0

Enable the high speed port

6.2.2.570 #define HS_CTRL_UART 1

Setup the high speed port UART configuration

6.2.2.571 #define HS_DEFAULT 6

HsState default

6.2.2.572 #define HS_DISABLE 4

HsState disable

6.2.2.573 #define HS_ENABLE 5

HsState enable

6.2.2.574 #define HS_INIT_RECEIVER 2

HsState initialize receiver

6.2.2.575 #define HS_INITIALISE 1

HsState initialize

6.2.2.576 #define HS_MODE_10_STOP 0x0000

HsMode 1 stop bit

6.2.2.577 #define HS_MODE_15_STOP 0x1000

HsMode 1.5 stop bits

6.2.2.578 #define HS_MODE_20_STOP 0x2000

HsMode 2 stop bits

6.2.2.579 #define HS_MODE_5_DATA 0x0000

HsMode 5 data bits

6.2.2.580 #define HS_MODE_6_DATA 0x0040

HsMode 6 data bits

6.2.2.581 #define HS_MODE_7_DATA 0x0080

HsMode 7 data bits

**6.2.2.582 #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

6.2.2.583 #define HS_MODE_8_DATA 0x00C0

HsMode 8 data bits

**6.2.2.584 #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

6.2.2.585 #define HS_MODE_DEFAULT HS_MODE_8N1

HsMode default mode (8 data bits, no parity, 1 stop bit)

6.2.2.586 #define HS_MODE_E_PARITY 0x0000

HsMode Even parity

6.2.2.587 #define HS_MODE_M_PARITY 0x0600

HsMode Mark parity

6.2.2.588 #define HS_MODE_MASK 0xFFFF0

HsMode mode mask

6.2.2.589 #define HS_MODE_N_PARITY 0x0800

HsMode No parity

6.2.2.590 #define HS_MODE_O_PARITY 0x0200

HsMode Odd parity

6.2.2.591 #define HS_MODE_S_PARITY 0x0400

HsMode Space parity

6.2.2.592 #define HS_MODE_UART_RS232 0x1

HsMode UART in normal or RS232 mode

6.2.2.593 #define HS_MODE_UART_RS485 0x0

HsMode UART in default or RS485 mode

6.2.2.594 #define HS_SEND_DATA 3

HsState send data

6.2.2.595 #define HS_UART_MASK 0x000F

HsMode UART mask

6.2.2.596 #define HS_UPDATE 1

HsFlags high speed update required

6.2.2.597 #define HT_ADDR_ACCEL 0x02

HiTechnic Accel I2C address

6.2.2.598 #define HT_ADDR_ANGLE 0x02

HiTechnic Angle I2C address

6.2.2.599 #define HT_ADDR_BAROMETRIC 0x02

HiTechnic Barometric I2C address

6.2.2.600 #define HT_ADDR_COLOR 0x02

HiTechnic Color I2C address

6.2.2.601 #define HT_ADDR_COLOR2 0x02

HiTechnic Color2 I2C address

6.2.2.602 #define HT_ADDR_COMPASS 0x02

HiTechnic Compass I2C address

6.2.2.603 #define HT_ADDR_IRLINK 0x02

HiTechnic IRLink I2C address

6.2.2.604 #define HT_ADDR_IRRECEIVER 0x02

HiTechnic IRReceiver I2C address

6.2.2.605 #define HT_ADDR_IRSEEKER 0x02

HiTechnic IRSeeker I2C address

6.2.2.606 #define HT_ADDR_IRSEEKER2 0x10

HiTechnic IRSeeker2 I2C address

6.2.2.607 #define HT_ADDR_PROTOBOARD 0x02

HiTechnic Prototype board I2C address

6.2.2.608 #define HT_ADDR_SUPERPRO 0x10

HiTechnic SuperPro board I2C address

6.2.2.609 #define HT_CH1_A 0

Use IRReceiver channel 1 output A

6.2.2.610 #define HT_CH1_B 1

Use IRReceiver channel 1 output B

6.2.2.611 #define HT_CH2_A 2

Use IRReceiver channel 2 output A

6.2.2.612 #define HT_CH2_B 3

Use IRReceiver channel 2 output B

6.2.2.613 #define HT_CH3_A 4

Use IRReceiver channel 3 output A

6.2.2.614 #define HT_CH3_B 5

Use IRReceiver channel 3 output B

6.2.2.615 #define HT_CH4_A 6

Use IRReceiver channel 4 output A

6.2.2.616 #define HT_CH4_B 7

Use IRReceiver channel 4 output B

6.2.2.617 #define HT_CMD_COLOR2_50HZ 0x35

Set the Color2 sensor to 50Hz mode

6.2.2.618 #define HT_CMD_COLOR2_60HZ 0x36

Set the Color2 sensor to 60Hz mode

6.2.2.619 #define HT_CMD_COLOR2_ACTIVE 0x00

Set the Color2 sensor to active mode

6.2.2.620 #define HT_CMD_COLOR2_BLCAL 0x42

Set the Color2 sensor to black level calibration mode

6.2.2.621 #define HT_CMD_COLOR2_FAR 0x46

Set the Color2 sensor to far mode

6.2.2.622 #define HT_CMD_COLOR2_LED_HI 0x48

Set the Color2 sensor to LED high mode

6.2.2.623 #define HT_CMD_COLOR2_LED_LOW 0x4C

Set the Color2 sensor to LED low mode

6.2.2.624 #define HT_CMD_COLOR2_NEAR 0x4E

Set the Color2 sensor to near mode

6.2.2.625 #define HT_CMD_COLOR2_PASSIVE 0x01

Set the Color2 sensor to passive mode

6.2.2.626 #define HT_CMD_COLOR2_RAW 0x03

Set the Color2 sensor to raw mode

6.2.2.627 #define HT_CMD_COLOR2_WBCAL 0x43

Set the Color2 sensor to white level calibration mode

6.2.2.628 #define HTANGLE_MODE_CALIBRATE 0x43

Resets 0 degree position to current shaft angle

6.2.2.629 #define HTANGLE_MODE_NORMAL 0x00

Normal angle measurement mode

6.2.2.630 #define HTANGLE_MODE_RESET 0x52

Resets the accumulated angle

6.2.2.631 #define HTANGLE_REG_ACDIR 0x49

Angle 16 bit revolutions per minute, low byte register

6.2.2.632 #define HTANGLE_REG_DC01 0x43

Angle current angle (1 degree adder) register

6.2.2.633 #define HTANGLE_REG_DC02 0x44

Angle 32 bit accumulated angle, high byte register

6.2.2.634 #define HTANGLE_REG_DC03 0x45

Angle 32 bit accumulated angle, mid byte register

6.2.2.635 #define HTANGLE_REG_DC04 0x46

Angle 32 bit accumulated angle, mid byte register

6.2.2.636 #define HTANGLE_REG_DC05 0x47

Angle 32 bit accumulated angle, low byte register

6.2.2.637 #define HTANGLE_REG_DCAVG 0x48

Angle 16 bit revolutions per minute, high byte register

6.2.2.638 #define HTANGLE_REG_DCDIR 0x42

Angle current angle (2 degree increments) register

6.2.2.639 #define HTANGLE_REG_MODE 0x41

Angle mode register

6.2.2.640 #define HTBAR_REG_CALIBRATION 0x46

Barometric sensor calibration register (2 bytes msb/lsb)

6.2.2.641 #define HTBAR_REG_COMMAND 0x40

Barometric sensor command register

6.2.2.642 #define HTBAR_REG_PRESSURE 0x44

Barometric sensor pressure register (2 bytes msb/lsb)

6.2.2.643 #define HTBAR_REG_TEMPERATURE 0x42

Barometric sensor temperature register (2 bytes msb/lsb)

6.2.2.644 #define HTIR2_MODE_1200 0

Set IRSeeker2 to 1200 mode

6.2.2.645 #define HTIR2_MODE_600 1

Set IRSeeker2 to 600 mode

6.2.2.646 #define HTIR2_REG_AC01 0x4A

IRSeeker2 AC 01 register

6.2.2.647 #define HTIR2_REG_AC02 0x4B

IRSeeker2 AC 02 register

6.2.2.648 #define HTIR2_REG_AC03 0x4C

IRSeeker2 AC 03 register

6.2.2.649 #define HTIR2_REG_AC04 0x4D

IRSeeker2 AC 04 register

6.2.2.650 #define HTIR2_REG_AC05 0x4E

IRSeeker2 AC 05 register

6.2.2.651 #define HTIR2_REG_ACDIR 0x49

IRSeeker2 AC direction register

6.2.2.652 #define HTIR2_REG_DC01 0x43

IRSeeker2 DC 01 register

6.2.2.653 #define HTIR2_REG_DC02 0x44

IRSeeker2 DC 02 register

6.2.2.654 #define HTIR2_REG_DC03 0x45

IRSeeker2 DC 03 register

6.2.2.655 #define HTIR2_REG_DC04 0x46

IRSeeker2 DC 04 register

6.2.2.656 #define HTIR2_REG_DC05 0x47

IRSeeker2 DC 05 register

6.2.2.657 #define HTIR2_REG_DCAVG 0x48

IRSeeker2 DC average register

6.2.2.658 #define HTIR2_REG_DCDIR 0x42

IRSeeker2 DC direction register

6.2.2.659 #define HTIR2_REG_MODE 0x41

IRSeeker2 mode register

6.2.2.660 #define HTPROTO_A0 0x42

Read Prototype board analog input 0

6.2.2.661 #define HTPROTO_A1 0x44

Read Prototype board analog input 1

6.2.2.662 #define HTPROTO_A2 0x46

Read Prototype board analog input 2

6.2.2.663 #define HTPROTO_A3 0x48

Read Prototype board analog input 3

6.2.2.664 #define HTPROTO_A4 0x4A

Read Prototype board analog input 4

6.2.2.665 #define HTPROTO_REG_A0 0x42

Prototype board analog 0 register (2 bytes msb/lsb)

6.2.2.666 #define HTPROTO_REG_A1 0x44

Prototype board analog 1 register (2 bytes msb/lsb)

6.2.2.667 #define HTPROTO_REG_A2 0x46

Prototype board analog 2 register (2 bytes msb/lsb)

6.2.2.668 #define HTPROTO_REG_A3 0x48

Prototype board analog 3 register (2 bytes msb/lsb)

6.2.2.669 #define HTPROTO_REG_A4 0x4A

Prototype board analog 4 register (2 bytes msb/lsb)

6.2.2.670 #define HTPROTO_REG_DCTRL 0x4E

Prototype board digital pin control register (6 bits)

6.2.2.671 #define HTPROTO_REG_DIN 0x4C

Prototype board digital pin input register (6 bits)

6.2.2.672 #define HTPROTO_REG_DOUT 0x4D

Prototype board digital pin output register (6 bits)

6.2.2.673 #define HTPROTO_REG_SRATE 0x4F

Prototype board sample rate register

6.2.2.674 #define HTSPRO_A0 0x42

Read SuperPro analog input 0

6.2.2.675 #define HTSPRO_A1 0x44

Read SuperPro analog input 1

6.2.2.676 #define HTSPRO_A2 0x46

Read SuperPro analog input 2

6.2.2.677 #define HTSPRO_A3 0x48

Read SuperPro analog input 3

6.2.2.678 #define HTSPRO_DAC0 0x52

Set SuperPro analog output 0 configuration

6.2.2.679 #define HTSPRO_DAC1 0x57

Set SuperPro analog output 1 configuration

6.2.2.680 #define HTSPRO_REG_A0 0x42

SuperPro analog 0 register (10 bits)

6.2.2.681 #define HTSPRO_REG_A1 0x44

SuperPro analog 1 register (10 bits)

6.2.2.682 #define HTSPRO_REG_A2 0x46

SuperPro analog 2 register (10 bits)

6.2.2.683 #define HTSPRO_REG_A3 0x48

SuperPro analog 3 register (10 bits)

6.2.2.684 #define HTSPRO_REG_CTRL 0x40

SuperPro program control register

6.2.2.685 #define HTSPRO_REG_DAC0_FREQ 0x53

SuperPro analog output 0 frequency register (2 bytes msb/lsb)

6.2.2.686 #define HTSPRO_REG_DAC0_MODE 0x52

SuperPro analog output 0 mode register

6.2.2.687 #define HTSPRO_REG_DAC0_VOLTAGE 0x55

SuperPro analog output 0 voltage register (10 bits)

6.2.2.688 #define HTSPRO_REG_DAC1_FREQ 0x58

SuperPro analog output 1 frequency register (2 bytes msb/lsb)

6.2.2.689 #define HTSPRO_REG_DAC1_MODE 0x57

SuperPro analog output 1 mode register

6.2.2.690 #define HTSPRO_REG_DAC1_VOLTAGE 0x5A

SuperPro analog output 1 voltage register (10 bits)

6.2.2.691 #define HTSPRO_REG_DCTRL 0x4E

SuperPro digital pin control register (8 bits)

6.2.2.692 #define HTSPRO_REG_DIN 0x4C

SuperPro digital pin input register (8 bits)

6.2.2.693 #define HTSPRO_REG_DLADDRESS 0x60

SuperPro download address register (2 bytes msb/lsb)

6.2.2.694 #define HTSPRO_REG_DLCHKSUM 0x6A

SuperPro download checksum register

6.2.2.695 #define HTSPRO_REG_DLCONTROL 0x6B

SuperPro download control register

6.2.2.696 #define HTSPRO_REG_DLDATA 0x62

SuperPro download data register (8 bytes)

6.2.2.697 #define HTSPRO_REG_DOUT 0x4D

SuperPro digital pin output register (8 bits)

6.2.2.698 #define HTSPRO_REG_LED 0x51

SuperPro LED control register

6.2.2.699 #define HTSPRO_REG_MEMORY_20 0x80

SuperPro memory address 0x20 register (4 bytes msb/lsb)

6.2.2.700 #define HTSPRO_REG_MEMORY_21 0x84

SuperPro memory address 0x21 register (4 bytes msb/lsb)

6.2.2.701 #define HTSPRO_REG_MEMORY_22 0x88

SuperPro memory address 0x22 register (4 bytes msb/lsb)

6.2.2.702 #define HTSPRO_REG_MEMORY_23 0x8C

SuperPro memory address 0x23 register (4 bytes msb/lsb)

6.2.2.703 #define HTSPRO_REG_MEMORY_24 0x90

SuperPro memory address 0x24 register (4 bytes msb/lsb)

6.2.2.704 #define HTSPRO_REG_MEMORY_25 0x94

SuperPro memory address 0x25 register (4 bytes msb/lsb)

6.2.2.705 #define HTSPRO_REG_MEMORY_26 0x98

SuperPro memory address 0x26 register (4 bytes msb/lsb)

6.2.2.706 #define HTSPRO_REG_MEMORY_27 0x9C

SuperPro memory address 0x27 register (4 bytes msb/lsb)

6.2.2.707 #define HTSPRO_REG_MEMORY_28 0xA0

SuperPro memory address 0x28 register (4 bytes msb/lsb)

6.2.2.708 #define HTSPRO_REG_MEMORY_29 0xA4

SuperPro memory address 0x29 register (4 bytes msb/lsb)

6.2.2.709 #define HTSPRO_REG_MEMORY_2A 0xA8

SuperPro memory address 0x2A register (4 bytes msb/lsb)

6.2.2.710 #define HTSPRO_REG_MEMORY_2B 0xAC

SuperPro memory address 0x2B register (4 bytes msb/lsb)

6.2.2.711 #define HTSPRO_REG_MEMORY_2C 0xB0

SuperPro memory address 0x2C register (4 bytes msb/lsb)

6.2.2.712 #define HTSPRO_REG_MEMORY_2D 0xB4

SuperPro memory address 0x2D register (4 bytes msb/lsb)

6.2.2.713 #define HTSPRO_REG_MEMORY_2E 0xB8

SuperPro memory address 0x2E register (4 bytes msb/lsb)

6.2.2.714 #define HTSPRO_REG_MEMORY_2F 0xBC

SuperPro memory address 0x2F register (4 bytes msb/lsb)

6.2.2.715 #define HTSPRO_REG_MEMORY_30 0xC0

SuperPro memory address 0x30 register (4 bytes msb/lsb)

6.2.2.716 #define HTSPRO_REG_MEMORY_31 0xC4

SuperPro memory address 0x31 register (4 bytes msb/lsb)

6.2.2.717 #define HTSPRO_REG_MEMORY_32 0xC8

SuperPro memory address 0x32 register (4 bytes msb/lsb)

6.2.2.718 #define HTSPRO_REG_MEMORY_33 0xCC

SuperPro memory address 0x33 register (4 bytes msb/lsb)

6.2.2.719 #define HTSPRO_REG_MEMORY_34 0xD0

SuperPro memory address 0x34 register (4 bytes msb/lsb)

6.2.2.720 #define HTSPRO_REG_MEMORY_35 0xD4

SuperPro memory address 0x35 register (4 bytes msb/lsb)

6.2.2.721 #define HTSPRO_REG_MEMORY_36 0xD8

SuperPro memory address 0x36 register (4 bytes msb/lsb)

6.2.2.722 #define HTSPRO_REG_MEMORY_37 0xDC

SuperPro memory address 0x37 register (4 bytes msb/lsb)

6.2.2.723 #define HTSPRO_REG_MEMORY_38 0xE0

SuperPro memory address 0x38 register (4 bytes msb/lsb)

6.2.2.724 #define HTSPRO_REG_MEMORY_39 0xE4

SuperPro memory address 0x39 register (4 bytes msb/lsb)

6.2.2.725 #define HTSPRO_REG_MEMORY_3A 0xE8

SuperPro memory address 0x3A register (4 bytes msb/lsb)

6.2.2.726 #define HTSPRO_REG_MEMORY_3B 0xEC

SuperPro memory address 0x3B register (4 bytes msb/lsb)

6.2.2.727 #define HTSPRO_REG_MEMORY_3C 0xF0

SuperPro memory address 0x3C register (4 bytes msb/lsb)

6.2.2.728 #define HTSPRO_REG_MEMORY_3D 0xF4

SuperPro memory address 0x3D register (4 bytes msb/lsb)

6.2.2.729 #define HTSPRO_REG_MEMORY_3E 0xF8

SuperPro memory address 0x3E register (4 bytes msb/lsb)

6.2.2.730 #define HTSPRO_REG_MEMORY_3F 0xFC

SuperPro memory address 0x3F register (4 bytes msb/lsb)

6.2.2.731 #define HTSPRO_REG_STROBE 0x50

SuperPro strobe control register

6.2.2.732 #define I2C_ADDR_DEFAULT 0x02

Standard NXT I2C device address

6.2.2.733 #define I2C_OPTION_FAST 0x08

Fast I2C speed

6.2.2.734 #define I2C_OPTION_NORESTART 0x04

Use no restart on I2C read

6.2.2.735 #define I2C_OPTION_STANDARD 0x00

Standard I2C speed

6.2.2.736 #define I2C_REG_CMD 0x41

Standard NXT I2C device command register

6.2.2.737 #define I2C_REG_DEVICE_ID 0x10

Standard NXT I2C device ID register

6.2.2.738 #define I2C_REG_VENDOR_ID 0x08

Standard NXT I2C vendor ID register

6.2.2.739 #define I2C_REG_VERSION 0x00

Standard NXT I2C version register

6.2.2.740 #define IN_1 0x00

Input port 1

6.2.2.741 #define IN_2 0x01

Input port 2

6.2.2.742 #define IN_3 0x02

Input port 3

6.2.2.743 #define IN_4 0x03

Input port 4

6.2.2.744 #define IN_MODE_ANGLESTEP 0xE0

RCX rotation sensor (16 ticks per revolution)

6.2.2.745 #define IN_MODE_BOOLEAN 0x20

Boolean value (0 or 1)

6.2.2.746 #define IN_MODE_CELSIUS 0xA0

RCX temperature sensor value in degrees celcius

6.2.2.747 #define IN_MODE_FAHRENHEIT 0xC0

RCX temperature sensor value in degrees fahrenheit

6.2.2.748 #define IN_MODE_MODEMASK 0xE0

Mask for the mode without any slope value

6.2.2.749 #define IN_MODE_PCTFULLSCALE 0x80

Scaled value from 0 to 100

6.2.2.750 #define IN_MODE_PERIODCOUNTER 0x60

Counts the number of boolean periods

6.2.2.751 #define IN_MODE_RAW 0x00

Raw value from 0 to 1023

6.2.2.752 #define IN_MODE_SLOPEMASK 0x1F

Mask for slope parameter added to mode

6.2.2.753 #define IN_MODE_TRANSITIONCNT 0x40

Counts the number of boolean transitions

6.2.2.754 #define IN_TYPE_ANGLE 0x04

RCX rotation sensor

6.2.2.755 #define IN_TYPE_COLORBLUE 0x10

NXT 2.0 color sensor with blue light

6.2.2.756 #define IN_TYPE_COLOREXIT 0x12

NXT 2.0 color sensor internal state

6.2.2.757 #define IN_TYPE_COLORFULL 0x0D

NXT 2.0 color sensor in full color mode

6.2.2.758 #define IN_TYPE_COLORGREEN 0x0F

NXT 2.0 color sensor with green light

6.2.2.759 #define IN_TYPE_COLORNONE 0x11

NXT 2.0 color sensor with no light

6.2.2.760 #define IN_TYPE_COLORRED 0x0E

NXT 2.0 color sensor with red light

6.2.2.761 #define IN_TYPE_CUSTOM 0x09

NXT custom sensor

6.2.2.762 #define IN_TYPE_HISPEED 0x0C

NXT Hi-speed port (only S4)

6.2.2.763 #define IN_TYPE_LIGHT_ACTIVE 0x05

NXT light sensor with light

6.2.2.764 #define IN_TYPE_LIGHT_INACTIVE 0x06

NXT light sensor without light

6.2.2.765 #define IN_TYPE_LOWSPEED 0x0A

NXT I2C digital sensor

6.2.2.766 #define IN_TYPE_LOWSPEED_9V 0x0B

NXT I2C digital sensor with 9V power

6.2.2.767 #define IN_TYPE_NO_SENSOR 0x00

No sensor configured

6.2.2.768 #define IN_TYPE_REFLECTION 0x03

RCX light sensor

6.2.2.769 #define IN_TYPE_SOUND_DB 0x07

NXT sound sensor with dB scaling

6.2.2.770 #define IN_TYPE_SOUND_DBA 0x08

NXT sound sensor with dBA scaling

6.2.2.771 #define IN_TYPE_SWITCH 0x01

NXT or RCX touch sensor

6.2.2.772 #define IN_TYPE_TEMPERATURE 0x02

RCX temperature sensor

6.2.2.773 #define INPUT_BLACKCOLOR 1

The color value is black

6.2.2.774 #define INPUT_BLANK 3

Access the blank value from color sensor value arrays

6.2.2.775 #define INPUT_BLUE 2

Access the blue value from color sensor value arrays

6.2.2.776 #define INPUT_BLUECOLOR 2

The color value is blue

6.2.2.777 #define INPUT_CAL_POINT_0 0

Calibration point 0

6.2.2.778 #define INPUT_CAL_POINT_1 1

Calibration point 1

6.2.2.779 #define INPUT_CAL_POINT_2 2

Calibration point 2

6.2.2.780 #define INPUT_CUSTOM9V 0x01

Custom sensor 9V

6.2.2.781 #define INPUT_CUSTOMACTIVE 0x02

Custom sensor active

6.2.2.782 #define INPUT_CUSTOMINACTIVE 0x00

Custom sensor inactive

6.2.2.783 #define INPUT_DIGI0 0x01

Digital pin 0

6.2.2.784 #define INPUT_DIGI1 0x02

Digital pin 1

6.2.2.785 #define INPUT_GREEN 1

Access the green value from color sensor value arrays

6.2.2.786 #define INPUT_GREENCOLOR 3

The color value is green

6.2.2.787 #define INPUT_INVALID_DATA 0x01

Invalid data flag

6.2.2.788 #define INPUT_NO_OF_COLORS 4

The number of entries in the color sensor value arrays

6.2.2.789 #define INPUT_NO_OF_POINTS 3

The number of calibration points

6.2.2.790 #define INPUT_PINCMD_CLEAR 0x02

Clear digital pin(s)

6.2.2.791 #define INPUT_PINCMD_DIR 0x00

Set digital pin(s) direction

6.2.2.792 #define INPUT_PINCMD_MASK 0x03

Mask for the two bits used by pin function commands

6.2.2.793 #define INPUT_PINCMD_READ 0x03

Read digital pin(s)

6.2.2.794 #define INPUT_PINCMD_SET 0x01

Set digital pin(s)

6.2.2.795 #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

6.2.2.796 #define INPUT_PINDIR_INPUT 0x04

Use with the direction command to set direction to output. OR this with the pin value.

6.2.2.797 #define INPUT_PINDIR_OUTPUT 0x00

Use with the direction command to set direction to input. OR this with the pin value.

6.2.2.798 #define INPUT_RED 0

Access the red value from color sensor value arrays

6.2.2.799 #define INPUT_REDCOLOR 5

The color value is red

6.2.2.800 #define INPUT_RESETCAL 0x80

Unused calibration state constant

6.2.2.801 #define INPUT_RUNNINGCAL 0x20

Unused calibration state constant

6.2.2.802 #define INPUT_SENSORCAL 0x01

The state returned while the color sensor is calibrating

6.2.2.803 #define INPUT_SENSOROFF 0x02

The state returned once calibration has completed

6.2.2.804 #define INPUT_STARTCAL 0x40

Unused calibration state constant

6.2.2.805 #define INPUT_WHITECOLOR 6

The color value is white

6.2.2.806 #define INPUT_YELLOWCOLOR 4

The color value is yellow

6.2.2.807 #define InputModeField 1

Input mode field. Contains one of the sensor mode constants. Read/write.

6.2.2.808 #define InputModuleID 0x00030001

The input module ID

6.2.2.809 #define InputModuleName "Input.mod"

The input module name.

6.2.2.810 #define InputOffsetADRaw(p) (((p)*20)+2)

Read the AD raw sensor value (2 bytes) uword

6.2.2.811 #define InputOffsetColorADRaw(p, nc) (80+((p)*84)+52+((nc)*2))

Read AD raw color sensor values

6.2.2.812 #define InputOffsetColorBoolean(p, nc) (80+((p)*84)+76+((nc)*2))

Read color sensor boolean values

6.2.2.813 #define InputOffsetColorCalibration(p, np, nc) (80+((p)*84)+0+((np)*16)+((nc)*4))

Read/write color calibration point values

6.2.2.814 #define InputOffsetColorCalibrationState(p) (80+((p)*84)+80)

Read color sensor calibration state

6.2.2.815 #define InputOffsetColorCalLimits(p, np) (80+((p)*84)+48+((np)*2))

Read/write color calibration limits

6.2.2.816 #define InputOffsetColorSensorRaw(p, nc) (80+((p)*84)+60+((nc)*2))

Read raw color sensor values

6.2.2.817 #define InputOffsetColorSensorValue(p, nc) (80+((p)*84)+68+((nc)*2))

Read scaled color sensor values

6.2.2.818 #define InputOffsetCustomActiveStatus(p) (((p)*20)+15)

Read/write the active or inactive state of the custom sensor

6.2.2.819 #define InputOffsetCustomPctFullScale(p) (((p)*20)+14)

Read/write the Pct full scale of the custom sensor

6.2.2.820 #define InputOffsetCustomZeroOffset(p) (((p)*20)+0)

Read/write the zero offset of a custom sensor (2 bytes) uword

6.2.2.821 #define InputOffsetDigiPinsDir(p) (((p)*20)+11)

Read/write the direction of the Digital pins (1 is output, 0 is input)

6.2.2.822 #define InputOffsetDigiPinsIn(p) (((p)*20)+12)

Read/write the status of the digital pins

6.2.2.823 #define InputOffsetDigiPinsOut(p) (((p)*20)+13)

Read/write the output level of the digital pins

6.2.2.824 #define InputOffsetInvalidData(p) (((p)*20)+16)

Indicates whether data is invalid (1) or valid (0)

6.2.2.825 #define InputOffsetSensorBoolean(p) (((p)*20)+10)

Read the sensor boolean value

6.2.2.826 #define InputOffsetSensorMode(p) (((p)*20)+9)

Read/write the sensor mode

6.2.2.827 #define InputOffsetSensorRaw(p) (((p)*20)+4)

Read the raw sensor value (2 bytes) uword

6.2.2.828 #define InputOffsetSensorType(p) (((p)*20)+8)

Read/write the sensor type

6.2.2.829 #define InputOffsetSensorValue(p) (((p)*20)+6)

Read/write the scaled sensor value (2 bytes) sword

6.2.2.830 #define InputPinFunction 77

Execute the Input module's pin function

6.2.2.831 #define INT_MAX 32767

The maximum value of the int type

6.2.2.832 #define INT_MIN -32768

The minimum value of the int type

6.2.2.833 #define INTF_BTOFF 13

Turn off the bluetooth radio

6.2.2.834 #define INTF_BTTON 12

Turn on the bluetooth radio

6.2.2.835 #define INTF_CONNECT 3

Connect to one of the known devices

6.2.2.836 #define INTF_CONNECTBYNAME 18

Connect to a bluetooth device by name

6.2.2.837 #define INTF_CONNECTREQ 17

Connection request from another device

6.2.2.838 #define INTF_DISCONNECT 4

Disconnect from one of the connected devices

6.2.2.839 #define INTF_DISCONNECTALL 5

Disconnect all devices

6.2.2.840 #define INTF_EXTREAD 15

External read request

6.2.2.841 #define INTF_FACTORYRESET 11

Reset bluetooth settings to factory values

6.2.2.842 #define INTF_OPENSTREAM 9

Open a bluetooth stream

6.2.2.843 #define INTF_PINREQ 16

Bluetooth PIN request

6.2.2.844 #define INTF_REMOVEDevice 6

Remove a device from the known devices table

6.2.2.845 #define INTF_SEARCH 1

Search for bluetooth devices

6.2.2.846 #define INTF_SENDDATA 10

Send data over a bluetooth connection

6.2.2.847 #define INTF_SENDFILE 0

Send a file via bluetooth to another device

6.2.2.848 #define INTF_SETBTNAME 14

Set the bluetooth name

6.2.2.849 #define INTF_SETCMDMODE 8

Set bluetooth into command mode

6.2.2.850 #define INTF_STOPSEARCH 2

Stop searching for bluetooth devices

6.2.2.851 #define INTF_VISIBILITY 7

Set the bluetooth visibility on or off

6.2.2.852 #define InvalidDataField 5

Invalid data field. Contains a boolean value indicating whether the sensor data is valid or not. Read/write.

6.2.2.853 #define IOCTRL_BOOT 0xA55A

Reboot the NXT into SAMBA mode

6.2.2.854 #define IOCTRL_POWERDOWN 0x5A00

Power down the NXT

6.2.2.855 #define IOCctrlModuleID 0x00060001

The IOCctrl module ID

6.2.2.856 #define IOCtrlModuleName "IOCtrl.mod"

The IOCtrl module name

6.2.2.857 #define IOCtrlOffsetPowerOn 0

Offset to power on field

6.2.2.858 #define IOMapRead 32

Read data from one of the firmware module's IOMap structures using the module's name

6.2.2.859 #define IOMapReadByID 78

Read data from one of the firmware module's IOMap structures using the module's ID

6.2.2.860 #define IOMapWrite 33

Write data to one of the firmware module's IOMap structures using the module's name

6.2.2.861 #define IOMapWriteByID 79

Write data to one of the firmware module's IOMap structures using the module's ID

6.2.2.862 #define KeepAlive 31

Reset the NXT sleep timer

6.2.2.863 #define LCD_LINE1 56

The 1st line of the LCD screen

6.2.2.864 #define LCD_LINE2 48

The 2nd line of the LCD screen

6.2.2.865 #define LCD_LINE3 40

The 3rd line of the LCD screen

6.2.2.866 #define LCD_LINE4 32

The 4th line of the LCD screen

6.2.2.867 #define LCD_LINE5 24

The 5th line of the LCD screen

6.2.2.868 #define LCD_LINE6 16

The 6th line of the LCD screen

6.2.2.869 #define LCD_LINE7 8

The 7th line of the LCD screen

6.2.2.870 #define LCD_LINE8 0

The 8th line of the LCD screen

6.2.2.871 #define LDR_APPENDNOTPOSSIBLE 0x8D00

Only datafiles can be appended to.

6.2.2.872 #define LDR_BTBUSY 0x9400

The bluetooth system is busy.

6.2.2.873 #define LDR_BTCONNECTFAIL 0x9500

Bluetooth connection attempt failed.

6.2.2.874 #define LDR_BTTIMEOUT 0x9600

A timeout in the bluetooth system has occurred.

6.2.2.875 #define LDR_CMD_BOOTCMD 0x97

Reboot the NXT into SAMBA mode

6.2.2.876 #define LDR_CMD_BTFACTORYRESET 0xA4

Reset bluetooth configuration to factory defaults

6.2.2.877 #define LDR_CMD_BTGETADR 0x9A

Get the NXT's bluetooth brick address

6.2.2.878 #define LDR_CMD_CLOSE 0x84

Close a file handle

6.2.2.879 #define LDR_CMD_CLOSEMODHANDLE 0x92

Close a module handle

6.2.2.880 #define LDR_CMD_CROPPDATAFILE 0x8D

Crop a data file to its used space

6.2.2.881 #define LDR_CMD_DELETE 0x85

Delete a file

6.2.2.882 #define LDR_CMD_DELETEUSERFLASH 0xA0

Delete all files from user flash memory

6.2.2.883 #define LDR_CMD_DEVICEINFO 0x9B

Read device information

6.2.2.884 #define LDR_CMD_FINDFIRST 0x86

Find the first file matching the specified pattern

6.2.2.885 #define LDR_CMD_FINDFIRSTMODULE 0x90

Find the first module matching the specified pattern

6.2.2.886 #define LDR_CMD_FINDNEXT 0x87

Find the next file matching the specified pattern

6.2.2.887 #define LDR_CMD_FINDNEXTMODULE 0x91

Find the next module matching the specified pattern

6.2.2.888 #define LDR_CMD_IOMAPREAD 0x94

Read data from a module IOMAP

6.2.2.889 #define LDR_CMD_IOMAPWRITE 0x95

Write data to a module IOMAP

6.2.2.890 #define LDR_CMD_OPENAPPENDDATA 0x8C

Open a data file for appending

6.2.2.891 #define LDR_CMD_OPENREAD 0x80

Open a file for reading

6.2.2.892 #define LDR_CMD_OPENREADLINEAR 0x8A

Open a linear file for reading

6.2.2.893 #define LDR_CMD_OPENWRITE 0x81

Open a file for writing

6.2.2.894 #define LDR_CMD_OPENWRITEDATA 0x8B

Open a data file for writing

6.2.2.895 #define LDR_CMD_OPENWITELINEAR 0x89

Open a linear file for writing

6.2.2.896 #define LDR_CMD_POLLCMD 0xA2

Poll command

6.2.2.897 #define LDR_CMD_POLLCMDLEN 0xA1

Read poll command length

6.2.2.898 #define LDR_CMD_READ 0x82

Read from a file

6.2.2.899 #define LDR_CMD_RENAMEFILE 0xA3

Rename a file

6.2.2.900 #define LDR_CMD_RESIZEDATAFILE 0xD0

Resize a data file

6.2.2.901 #define LDR_CMD_SEEKFROMCURRENT 0xD2

Seek from the current position

6.2.2.902 #define LDR_CMD_SEEKFROMEND 0xD3

Seek from the end of the file

6.2.2.903 #define LDR_CMD_SEEKFROMSTART 0xD1

Seek from the start of the file

6.2.2.904 #define LDR_CMD_SETBRICKNAME 0x98

Set the NXT's brick name

6.2.2.905 #define LDR_CMD_VERSIONS 0x88

Read firmware version information

6.2.2.906 #define LDR_CMD_WRITE 0x83

Write to a file

6.2.2.907 #define LDR_ENDOFFILE 0x8500

The end of the file has been reached.

6.2.2.908 #define LDR_EOFEXPECTED 0x8400

EOF expected.

6.2.2.909 #define LDR_FILEEXISTS 0x8F00

A file with the same name already exists.

6.2.2.910 #define LDR_FILEISBUSY 0x8B00

The file is already being used.

6.2.2.911 #define LDR_FILEISFULL 0x8E00

The allocated file size has been filled.

6.2.2.912 #define LDR_FILENOFOUND 0x8700

No files matched the search criteria.

6.2.2.913 #define LDR_FILETX_CLOSEERROR 0x9B00

Error transmitting file: attempt to close file failed.

6.2.2.914 #define LDR_FILETX_DSEXISTS 0x9800

Error transmitting file: destination file exists.

6.2.2.915 #define LDR_FILETX_SRCMISSING 0x9900

Error transmitting file: source file is missing.

6.2.2.916 #define LDR_FILETX_STREAMERROR 0x9A00

Error transmitting file: a stream error occurred.

6.2.2.917 #define LDR_FILETX_TIMEOUT 0x9700

Error transmitting file: a timeout occurred.

6.2.2.918 #define LDR_HANDLEALREADYCLOSED 0x8800

The file handle has already been closed.

6.2.2.919 #define LDR_ILLEGALFILENAME 0x9200

Filename length to long or attempted open a system file (*.rxe, *.rtm, or *.sys) for writing as a datafile.

6.2.2.920 #define LDR_ILLEGALHANDLE 0x9300

Invalid file handle.

6.2.2.921 #define LDR_INPROGRESS 0x0001

The function is executing but has not yet completed.

6.2.2.922 #define LDR_INVALIDSEEK 0x9C00

Invalid file seek operation.

6.2.2.923 #define LDR_MODULENOTFOUND 0x9000

No modules matched the specified search criteria.

6.2.2.924 #define LDR_NOLINEARSPACE 0x8900

Not enough linear flash memory is available.

6.2.2.925 #define LDR_NMOREFILES 0x8300

The maximum number of files has been reached.

6.2.2.926 #define LDR_NMOREHANDLES 0x8100

All available file handles are in use.

6.2.2.927 #define LDR_NOSPACE 0x8200

Not enough free flash memory for the specified file size.

6.2.2.928 #define LDR_NOTLINEARFILE 0x8600

The specified file is not linear.

6.2.2.929 #define LDR_NOWRITEBUFFERS 0x8C00

No more write buffers are available.

6.2.2.930 #define LDR_OUTOFCOMMAND 0x9100

Specified IOMap offset is outside the bounds of the IOMap.

6.2.2.931 #define LDR_REQPIN 0x0002

A PIN exchange request is in progress.

6.2.2.932 #define LDR_SUCCESS 0x0000

The function completed successfully.

6.2.2.933 #define LDR_UNDEFINEDERROR 0x8A00

An undefined error has occurred.

6.2.2.934 #define LED_BLUE 0x02

Turn on the blue onboard LED.

6.2.2.935 #define LED_NONE 0x00

Turn off the onboard LEDs.

6.2.2.936 #define LED_RED 0x01

Turn on the red onboard LED.

6.2.2.937 #define LEGO_ADDR_EMETER 0x04

The LEGO e-meter sensor's I2C address

6.2.2.938 #define LEGO_ADDR_TEMP 0x98

The LEGO temperature sensor's I2C address

6.2.2.939 #define LEGO_ADDR_US 0x02

The LEGO ultrasonic sensor's I2C address

6.2.2.940 #define ListFiles 47

List files that match the specified filename pattern

6.2.2.941 #define LoaderExecuteFunction 82

Execute one of the Loader module's internal functions

6.2.2.942 #define LoaderModuleID 0x00090001

The Loader module ID

6.2.2.943 #define LoaderModuleName "Loader.mod"

The Loader module name

6.2.2.944 #define LoaderOffsetFreeUserFlash 4

Offset to the amount of free user flash

6.2.2.945 #define LoaderOffsetPFunc 0

Offset to the Loader module function pointer

6.2.2.946 #define LONG_MAX 2147483647

The maximum value of the long type

6.2.2.947 #define LONG_MIN -2147483648

The minimum value of the long type

6.2.2.948 #define LOWSPEED_CH_NOT_READY 1

Lowspeed port is not ready

6.2.2.949 #define LOWSPEED_COMMUNICATING 3

Channel is actively communicating

6.2.2.950 #define LOWSPEED_DATA_RECEIVED 3

Lowspeed port is in data received mode

6.2.2.951 #define LOWSPEED_DONE 5

Channel is done communicating

6.2.2.952 #define LOWSPEED_ERROR 4

Channel is in an error state

6.2.2.953 #define LOWSPEED_IDLE 0

Channel is idle

6.2.2.954 #define LOWSPEED_INIT 1

Channel is being initialized

6.2.2.955 #define LOWSPEED_LOAD_BUFFER 2

Channel buffer is loading

6.2.2.956 #define LOWSPEED_NO_ERROR 0

Lowspeed port has no error

6.2.2.957 #define LOWSPEED RECEIVING 2

Lowspeed port is in receiving mode

6.2.2.958 #define LOWSPEED_RX_ERROR 3

Lowspeed port encountered an error while receiving data

6.2.2.959 #define LOWSPEED_TRANSMITTING 1

Lowspeed port is in transmitting mode

6.2.2.960 #define LOWSPEED_TX_ERROR 2

Lowspeed port encountered an error while transmitting data

6.2.2.961 #define LowSpeedModuleID 0x000B0001

The low speed module ID

6.2.2.962 #define LowSpeedModuleName "Low Speed.mod"

The low speed module name

6.2.2.963 #define LowSpeedOffsetChannelState(p) ((p)+156)

R - Lowspeed channgel state (1 byte)

6.2.2.964 #define LowSpeedOffsetErrorType(p) ((p)+160)

R - Lowspeed port error type (1 byte)

6.2.2.965 #define LowSpeedOffsetInBufBuf(p) (((p)*19)+0)

RW - Input buffer data buffer field offset (16 bytes)

6.2.2.966 #define LowSpeedOffsetInBufBytesToRx(p) (((p)*19)+18)

RW - Input buffer bytes to receive field offset (1 byte)

6.2.2.967 #define LowSpeedOffsetInBufInPtr(p) (((p)*19)+16)

RW - Input buffer in pointer field offset (1 byte)

6.2.2.968 #define LowSpeedOffsetInBufOutPtr(p) (((p)*19)+17)

RW - Input buffer out pointer field offset (1 byte)

6.2.2.969 #define LowSpeedOffsetMode(p) ((p)+152)

R - Lowspeed port mode (1 byte)

6.2.2.970 #define LowSpeedOffsetNoRestartOnRead 166

RW - Lowspeed option for no restart on read (all channels) (NBC/NXC)

6.2.2.971 #define LowSpeedOffsetOutBufBuf(p) (((p)*19)+76)

RW - Output buffer data buffer field offset (16 bytes)

6.2.2.972 #define LowSpeedOffsetOutBufBytesToRx(p) (((p)*19)+94)

RW - Output buffer bytes to receive field offset (1 byte)

6.2.2.973 #define LowSpeedOffsetOutBufInPtr(p) (((p)*19)+92)

RW - Output buffer in pointer field offset (1 byte)

6.2.2.974 #define LowSpeedOffsetOutBufOutPtr(p) (((p)*19)+93)

RW - Output buffer out pointer field offset (1 byte)

6.2.2.975 #define LowSpeedOffsetSpeed 165

R - Lowspeed speed (unused)

6.2.2.976 #define LowSpeedOffsetState 164

R - Lowspeed state (all channels)

6.2.2.977 #define LR_COULD_NOT_SAVE 0x51

Bluetooth list result could not save

6.2.2.978 #define LR_ENTRY_REMOVED 0x53

Bluetooth list result entry removed

6.2.2.979 #define LR_STORE_IS_FULL 0x52

Bluetooth list result store is full

6.2.2.980 #define LR_SUCCESS 0x50

Bluetooth list result success

6.2.2.981 #define LR_UNKNOWN_ADDR 0x54

Bluetooth list result unknown address

6.2.2.982 #define LSREAD_NO_RESTART_1 0x01

No restart on read for channel 1

6.2.2.983 #define LSREAD_NO_RESTART_2 0x02

No restart on read for channel 2

6.2.2.984 #define LSREAD_NO_RESTART_3 0x04

No restart on read for channel 3

6.2.2.985 #define LSREAD_NO_RESTART_4 0x08

No restart on read for channel 4

6.2.2.986 #define LSREAD_NO_RESTART_MASK 0x10

No restart mask

6.2.2.987 #define LSREAD_RESTART_ALL 0x00

Restart on read for all channels (default)

6.2.2.988 #define LSREAD_RESTART_NONE 0x0F

No restart on read for all channels

6.2.2.989 #define MAILBOX1 0

Mailbox number 1

6.2.2.990 #define MAILBOX10 9

Mailbox number 10

6.2.2.991 #define MAILBOX2 1

Mailbox number 2

6.2.2.992 #define MAILBOX3 2

Mailbox number 3

6.2.2.993 #define MAILBOX4 3

Mailbox number 4

6.2.2.994 #define MAILBOX5 4

Mailbox number 5

6.2.2.995 #define MAILBOX6 5

Mailbox number 6

6.2.2.996 #define MAILBOX7 6

Mailbox number 7

6.2.2.997 #define MAILBOX8 7

Mailbox number 8

6.2.2.998 #define MAILBOX9 8

Mailbox number 9

6.2.2.999 #define MAX_BT_MSG_SIZE 60000

Max Bluetooth Message Size

6.2.2.1000 #define MaxAccelerationField 17

MaxAcceleration field. Contains the current max acceleration value. Read/write. Set the maximum acceleration to be used during position regulation.

6.2.2.1001 #define MaxSpeedField 16

MaxSpeed field. Contains the current max speed value. Read/write. Set the maximum speed to be used during position regulation.

6.2.2.1002 #define MemoryManager 96

Read memory manager information, optionally compacting the dataspace first

6.2.2.1003 #define MENUICON_CENTER 1

Center icon

6.2.2.1004 #define MENUICON_LEFT 0

Left icon

6.2.2.1005 #define MENUICON_RIGHT 2

Right icon

6.2.2.1006 #define MENUICONS 3

The number of menu icons

6.2.2.1007 #define MENUTEXT 2

Center icon text

6.2.2.1008 #define MessageRead 27

Read a message from a mailbox

6.2.2.1009 #define MessageWrite 26

Write a message to a mailbox

6.2.2.1010 #define MI_ADDR_XG1300L 0x02

XG1300L I2C address

6.2.2.1011 #define MIN_1 60000

1 minute

6.2.2.1012 #define MS_1 1

1 millisecond

6.2.2.1013 #define MS_10 10

10 milliseconds

6.2.2.1014 #define MS_100 100

100 milliseconds

6.2.2.1015 #define MS_150 150

150 milliseconds

6.2.2.1016 #define MS_2 2

2 milliseconds

6.2.2.1017 #define MS_20 20

20 milliseconds

6.2.2.1018 #define MS_200 200

200 milliseconds

6.2.2.1019 #define MS_250 250

250 milliseconds

6.2.2.1020 #define MS_3 3

3 milliseconds

6.2.2.1021 #define MS_30 30

30 milliseconds

6.2.2.1022 #define MS_300 300

300 milliseconds

6.2.2.1023 #define MS_350 350

350 milliseconds

6.2.2.1024 #define MS_4 4

4 milliseconds

6.2.2.1025 #define MS_40 40

40 milliseconds

6.2.2.1026 #define MS_400 400

400 milliseconds

6.2.2.1027 #define MS_450 450

450 milliseconds

6.2.2.1028 #define MS_5 5

5 milliseconds

6.2.2.1029 #define MS_50 50

50 milliseconds

6.2.2.1030 #define MS_500 500

500 milliseconds

6.2.2.1031 #define MS_6 6

6 milliseconds

6.2.2.1032 #define MS_60 60

60 milliseconds

6.2.2.1033 #define MS_600 600

600 milliseconds

6.2.2.1034 #define MS_7 7

7 milliseconds

6.2.2.1035 #define MS_70 70

70 milliseconds

6.2.2.1036 #define MS_700 700

700 milliseconds

6.2.2.1037 #define MS_8 8

8 milliseconds

6.2.2.1038 #define MS_80 80

80 milliseconds

6.2.2.1039 #define MS_800 800

800 milliseconds

6.2.2.1040 #define MS_9 9

9 milliseconds

6.2.2.1041 #define MS_90 90

90 milliseconds

6.2.2.1042 #define MS_900 900

900 milliseconds

6.2.2.1043 #define MS_ADDR_ACCLNX 0x02

MindSensors ACCL-Nx I2C address

6.2.2.1044 #define MS_ADDR_CMPSNX 0x02

MindSensors CMPS-Nx I2C address

6.2.2.1045 #define MS_ADDR_DISTNX 0x02

MindSensors DIST-Nx I2C address

6.2.2.1046 #define MS_ADDR_IVSENS 0x12

MindSensors IVSens (NXTPowerMeter) I2C address

6.2.2.1047 #define MS_ADDR_LINELDR 0x02

MindSensors LineLdr I2C address

6.2.2.1048 #define MS_ADDR_MTRMUX 0xB4

MindSensors MTRMux I2C address

6.2.2.1049 #define MS_ADDR_NRLINK 0x02

MindSensors NRLink I2C address

6.2.2.1050 #define MS_ADDR_NXTCAM 0x02

MindSensors NXTCam I2C address

6.2.2.1051 #define MS_ADDR_NXTHID 0x04

MindSensors NXTHID I2C address

6.2.2.1052 #define MS_ADDR_NXTMMX 0x06

MindSensors NXTMMX I2C address

6.2.2.1053 #define MS_ADDR_NXTSERVO 0xB0

MindSensors NXTServo I2C address

6.2.2.1054 #define MS_ADDR_NXTSERVO_EM 0x40

MindSensors NXTServo in edit macro mode I2C address

6.2.2.1055 #define MS_ADDR_PFMATE 0x48

MindSensors PFmate I2C address

6.2.2.1056 #define MS_ADDR_PSPNX 0x02

MindSensors PSP-Nx I2C address

6.2.2.1057 #define MS_ADDR_RTCLOCK 0xD0

MindSensors RTClock I2C address

6.2.2.1058 #define MS_ADDR_RXMUX 0x7E

MindSensors RXMux I2C address

6.2.2.1059 #define MS_CMD_ADPA_OFF 0x4F

Turn MindSensors ADPA mode off

6.2.2.1060 #define MS_CMD_ADPA_ON 0x4E

Turn MindSensors ADPA mode on

6.2.2.1061 #define MS_CMD_DEENERGIZED 0x44

De-energize the MindSensors device

6.2.2.1062 #define MS_CMD_ENERGIZED 0x45

Energize the MindSensors device

6.2.2.1063 #define NA 0xFFFF

The specified argument does not apply (aka unwired)

6.2.2.1064 #define NO_ERR 0

Successful execution of the specified command

6.2.2.1065 #define NO_OF_BTNS 4

The number of NXT buttons.

6.2.2.1066 #define NormalizedValueField 3

Normalized value field. Contains the current normalized analog sensor value. Read only.

6.2.2.1067 #define NRLINK_CMD_2400 0x44

Set NRLink to 2400 baud

6.2.2.1068 #define NRLINK_CMD_4800 0x48

Set NRLink to 4800 baud

6.2.2.1069 #define NRLINK_CMD_FLUSH 0x46

Flush the NRLink

6.2.2.1070 #define NRLINK_CMD_IR_LONG 0x4C

Set the NRLink to long range IR

6.2.2.1071 #define NRLINK_CMD_IR_SHORT 0x53

Set the NRLink to short range IR

6.2.2.1072 #define NRLINK_CMD_RUN_MACRO 0x52

Run an NRLink macro

6.2.2.1073 #define NRLINK_CMD_SET_PF 0x50

Set the NRLink to Power Function mode

6.2.2.1074 #define NRLINK_CMD_SET_RCX 0x58

Set the NRLink to RCX mode

6.2.2.1075 #define NRLINK_CMD_SET_TRAIN 0x54

Set the NRLink to IR Train mode

6.2.2.1076 #define NRLINK_CMD_TX_RAW 0x55

Set the NRLink to transmit raw bytes

6.2.2.1077 #define NRLINK_REG_BYTES 0x40

The NRLink bytes register

6.2.2.1078 #define NRLINK_REG_DATA 0x42

The NRLink data register

6.2.2.1079 #define NRLINK_REG_EEPROM 0x50

The NRLink eeprom register

6.2.2.1080 #define NULL 0

A constant representing NULL

6.2.2.1081 #define NXTHID_CMD_ASCII 0x41

Use ASCII data mode. In ASCII mode no non-printable characters can be sent.

6.2.2.1082 #define NXTHID_CMD_DIRECT 0x44

Use direct data mode In direct mode any character can be sent.

6.2.2.1083 #define NXTHID_CMD_TRANSMIT 0x54

Transmit data to the host computer.

6.2.2.1084 #define NXTHID_MOD_LEFT_ALT 0x04

NXTHID left alt modifier.

6.2.2.1085 #define NXTHID_MOD_LEFT_CTRL 0x01

NXTHID left control modifier.

6.2.2.1086 #define NXTHID_MOD_LEFT_GUI 0x08

NXTHID left gui modifier.

6.2.2.1087 #define NXTHID_MOD_LEFT_SHIFT 0x02

NXTHID left shift modifier.

6.2.2.1088 #define NXTHID_MOD_NONE 0x00

NXTHID no modifier.

6.2.2.1089 #define NXTHID_MOD_RIGHT_ALT 0x40

NXTHID right alt modifier.

6.2.2.1090 #define NXTHID_MOD_RIGHT_CTRL 0x10

NXTHID right control modifier.

6.2.2.1091 #define NXTHID_MOD_RIGHT_GUI 0x80

NXTHID right gui modifier.

6.2.2.1092 #define NXTHID_MOD_RIGHT_SHIFT 0x20

NXTHID right shift modifier.

6.2.2.1093 #define NXTHID_REG_CMD 0x41

NXTHID command register. See [MindSensors NXTHID commands](#) group.

6.2.2.1094 #define NXTHID_REG_DATA 0x43

NXTHID data register.

6.2.2.1095 #define NXTHID_REG_MODIFIER 0x42

NXTHID modifier register. See [MindSensors NXTHID modifier keys](#) group.

6.2.2.1096 #define NXTLL_CMD_BLACK 0x42

Black calibration.

6.2.2.1097 #define NXTLL_CMD_EUROPEAN 0x45

European power frequency. (50hz)

6.2.2.1098 #define NXTLL_CMD_INVERT 0x49

Invert color.

6.2.2.1099 #define NXTLL_CMD_POWERDOWN 0x44

Power down the device.

6.2.2.1100 #define NXTLL_CMD_POWERUP 0x50

Power up the device.

6.2.2.1101 #define NXTLL_CMD_RESET 0x52

Reset inversion.

6.2.2.1102 #define NXTLL_CMD_SNAPSHOT 0x53

Setpoint based on snapshot (automatically sets invert if needed).

6.2.2.1103 #define NXTLL_CMD_UNIVERSAL 0x55

Universal power frequency. The sensor auto adjusts for any frequency. This is the default mode.

6.2.2.1104 #define NXTLL_CMD_USA 0x41

USA power frequency. (60hz)

6.2.2.1105 #define NXTLL_CMD_WHITE 0x57

White balance calibration.

6.2.2.1106 #define NXTLL_REG_AVERAGE 0x43

NXTLineLeader average result register.

6.2.2.1107 #define NXTLL_REG_BLACKDATA 0x6C

NXTLineLeader black calibration data registers. 8 bytes.

6.2.2.1108 #define NXTLL_REG_BLACKLIMITS 0x59

NXTLineLeader black limit registers. 8 bytes.

6.2.2.1109 #define NXTLL_REG_CALIBRATED 0x49

NXTLineLeader calibrated sensor reading registers. 8 bytes.

6.2.2.1110 #define NXTLL_REG_CMD 0x41

NXTLineLeader command register. See the [MindSensors NXTLineLeader commands](#) group.

6.2.2.1111 #define NXTLL_REG_KD_FACTOR 0x63

NXTLineLeader Kd factor register. Default = 32.

6.2.2.1112 #define NXTLL_REG_KD_VALUE 0x48

NXTLineLeader Kd value register. Default = 8.

6.2.2.1113 #define NXTLL_REG_KI_FACTOR 0x62

NXTLineLeader Ki factor register. Default = 32.

6.2.2.1114 #define NXTLL_REG_KI_VALUE 0x47

NXTLineLeader Ki value register. Default = 0.

6.2.2.1115 #define NXTLL_REG_KP_FACTOR 0x61

NXTLineLeader Kp factor register. Default = 32.

6.2.2.1116 #define NXTLL_REG_KP_VALUE 0x46

NXTLineLeader Kp value register. Default = 25.

6.2.2.1117 #define NXTLL_REG_RAWVOLTAGE 0x74

NXTLineLeader uncalibrated sensor voltage registers. 16 bytes.

6.2.2.1118 #define NXTLL_REG_RESULT 0x44

NXTLineLeader result register (sensor bit values).

6.2.2.1119 #define NXTLL_REG_SETPOINT 0x45

NXTLineLeader user settable average (setpoint) register. Default = 45.

6.2.2.1120 #define NXTLL_REG_STEERING 0x42

NXTLineLeader steering register.

6.2.2.1121 #define NXTLL_REG_WHITEDATA 0x64

NXTLineLeader white calibration data registers. 8 bytes.

6.2.2.1122 #define NXTLL_REG_WHITELIMITS 0x51

NXTLineLeader white limit registers. 8 bytes.

6.2.2.1123 #define NXTPM_CMD_RESET 0x52

Reset counters.

6.2.2.1124 #define NXTPM_REG_CAPACITY 0x46

NXTPowerMeter capacity used since last reset register. (2 bytes)

6.2.2.1125 #define NXTPM_REG_CMD 0x41

NXTPowerMeter command register. See the [MindSensors NXTPowerMeter commands](#) group.

6.2.2.1126 #define NXTPM_REG_CURRENT 0x42

NXTPowerMeter present current in mA register. (2 bytes)

6.2.2.1127 #define NXTPM_REG_ERRORCOUNT 0x5F

NXTPowerMeter error count register. (2 bytes)

6.2.2.1128 #define NXTPM_REG_GAIN 0x5E

NXTPowerMeter gain register. (1 byte)

6.2.2.1129 #define NXTPM_REG_MAXCURRENT 0x4E

NXTPowerMeter max current register. (2 bytes)

6.2.2.1130 #define NXTPM_REG_MAXVOLTAGE 0x52

NXTPowerMeter max voltage register. (2 bytes)

6.2.2.1131 #define NXTPM_REG_MINCURRENT 0x50

NXTPowerMeter min current register. (2 bytes)

6.2.2.1132 #define NXTPM_REG_MINVOLTAGE 0x54

NXTPowerMeter min voltage register. (2 bytes)

6.2.2.1133 #define NXTPM_REG_POWER 0x48

NXTPowerMeter present power register. (2 bytes)

6.2.2.1134 #define NXTPM_REG_TIME 0x56

NXTPowerMeter time register. (4 bytes)

6.2.2.1135 #define NXTPM_REG_TOTALPOWER 0x4A

NXTPowerMeter total power consumed since last reset register. (4 bytes)

6.2.2.1136 #define NXTPM_REG_USERGAIN 0x5A

NXTPowerMeter user gain register. Not yet implemented. (4 bytes)

6.2.2.1137 #define NXTPM_REG_VOLTAGE 0x44

NXTPowerMeter present voltage in mV register. (2 bytes)

6.2.2.1138 #define NXTSE_ZONE_FRONT 1

Obstacle zone front.

6.2.2.1139 #define NXTSE_ZONE_LEFT 2

Obstacle zone left.

6.2.2.1140 #define NXTSE_ZONE_NONE 0

Obstacle zone none.

6.2.2.1141 #define NXTSE_ZONE_RIGHT 3

Obstacle zone right.

6.2.2.1142 #define NXTSERVO_CMD_EDIT1 0x45

Edit Macro (part 1 of 2 character command sequence)

6.2.2.1143 #define NXTSERVO_CMD_EDIT2 0x4D

Edit Macro (part 2 of 2 character command sequence)

6.2.2.1144 #define NXTSERVO_CMD_GOTO 0x47

Goto EEPROM position x. This command re-initializes the macro environment.

6.2.2.1145 #define NXTSERVO_CMD_HALT 0x48

Halt Macro. This command re-initializes the macro environment.

6.2.2.1146 #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.

6.2.2.1147 #define NXTSERVO_CMD_PAUSE 0x50

Pause Macro. This command will pause the macro, and save the environment for subsequent resumption.

6.2.2.1148 #define NXTSERVO_CMD_RESET 0x53

Reset servo properties to factory default. Initial Position of servos to 1500, and speed to 0.

6.2.2.1149 #define NXTSERVO_CMD_RESUME 0x52

Resume macro Execution. This command resumes macro where it was paused last, using the same environment.

6.2.2.1150 #define NXTSERVO_EM_CMD_QUIT 0x51

Exit edit macro mode

6.2.2.1151 #define NXTSERVO_EM_REG_CMD 0x00

NXTServo in macro edit mode command register.

6.2.2.1152 #define NXTSERVO_EM_REG_EEPROM_END 0xFF

NXTServo in macro edit mode EEPROM end register.

6.2.2.1153 #define NXTSERVO_EM_REG_EEPROM_START 0x21

NXTServo in macro edit mode EEPROM start register.

6.2.2.1154 #define NXTSERVO_POS_CENTER 1500

Center position for 1500us servos.

6.2.2.1155 #define NXTSERVO_POS_MAX 2500

Maximum position for 1500us servos.

6.2.2.1156 #define NXTSERVO_POS_MIN 500

Minimum position for 1500us servos.

6.2.2.1157 #define NXTSERVO_QPOS_CENTER 150

Center quick position for 1500us servos.

6.2.2.1158 #define NXTSERVO_QPOS_MAX 250

Maximum quick position for 1500us servos.

6.2.2.1159 #define NXTSERVO_QPOS_MIN 50

Minimum quick position for 1500us servos.

6.2.2.1160 #define NXTSERVO_REG_CMD 0x41

NXTServo command register. See [MindSensors NXTServo commands](#) group. (write only)

6.2.2.1161 #define NXTSERVO_REG_S1_POS 0x42

NXTServo servo 1 position register.

6.2.2.1162 #define NXTSERVO_REG_S1_QPOS 0x5A

NXTServo servo 1 quick position register. (write only)

6.2.2.1163 #define NXTSERVO_REG_S1_SPEED 0x52

NXTServo servo 1 speed register.

6.2.2.1164 #define NXTSERVO_REG_S2_POS 0x44

NXTServo servo 2 position register.

6.2.2.1165 #define NXTSERVO_REG_S2_QPOS 0x5B

NXTServo servo 2 quick position register. (write only)

6.2.2.1166 #define NXTSERVO_REG_S2_SPEED 0x53

NXTServo servo 2 speed register.

6.2.2.1167 #define NXTSERVO_REG_S3_POS 0x46

NXTServo servo 3 position register.

6.2.2.1168 #define NXTSERVO_REG_S3_QPOS 0x5C

NXTServo servo 3 quick position register. (write only)

6.2.2.1169 #define NXTSERVO_REG_S3_SPEED 0x54

NXTServo servo 3 speed register.

6.2.2.1170 #define NXTSERVO_REG_S4_POS 0x48

NXTServo servo 4 position register.

6.2.2.1171 #define NXTSERVO_REG_S4_QPOS 0x5D

NXTServo servo 4 quick position register. (write only)

6.2.2.1172 #define NXTSERVO_REG_S4_SPEED 0x55

NXTServo servo 4 speed register.

6.2.2.1173 #define NXTSERVO_REG_S5_POS 0x4A

NXTServo servo 5 position register.

6.2.2.1174 #define NXTSERVO_REG_S5_QPOS 0x5E

NXTServo servo 5 quick position register. (write only)

6.2.2.1175 #define NXTSERVO_REG_S5_SPEED 0x56

NXTServo servo 5 speed register.

6.2.2.1176 #define NXTSERVO_REG_S6_POS 0x4C

NXTServo servo 6 position register.

6.2.2.1177 #define NXTSERVO_REG_S6_QPOS 0x5F

NXTServo servo 6 quick position register. (write only)

6.2.2.1178 #define NXTSERVO_REG_S6_SPEED 0x57

NXTServo servo 6 speed register.

6.2.2.1179 #define NXTSERVO_REG_S7_POS 0x4E

NXTServo servo 7 position register.

6.2.2.1180 #define NXTSERVO_REG_S7_QPOS 0x60

NXTServo servo 7 quick position register. (write only)

6.2.2.1181 #define NXTSERVO_REG_S7_SPEED 0x58

NXTServo servo 7 speed register.

6.2.2.1182 #define NXTSERVO_REG_S8_POS 0x50

NXTServo servo 8 position register.

6.2.2.1183 #define NXTSERVO_REG_S8_QPOS 0x61

NXTServo servo 8 quick position register. (write only)

6.2.2.1184 #define NXTSERVO_REG_S8_SPEED 0x59

NXTServo servo 8 speed register.

6.2.2.1185 #define NXTSERVO_REG_VOLTAGE 0x41

Battery voltage register. (read only)

6.2.2.1186 #define NXTSERVO_SERVO_1 0

NXTServo server number 1.

6.2.2.1187 #define NXTSERVO_SERVO_2 1

NXTServo server number 2.

6.2.2.1188 #define NXTSERVO_SERVO_3 2

NXTServo server number 3.

6.2.2.1189 #define NXTSERVO_SERVO_4 3

NXTServo server number 4.

6.2.2.1190 #define NXTSERVO_SERVO_5 4

NXTServo server number 5.

6.2.2.1191 #define NXTSERVO_SERVO_6 5

NXTServo server number 6.

6.2.2.1192 #define NXTSERVO_SERVO_7 6

NXTServo server number 7.

6.2.2.1193 #define NXTSERVO_SERVO_8 7

NXTServo server number 8.

6.2.2.1194 #define OPARR_MAX 0x05

Calculate the maximum value of the elements in the numeric input array

6.2.2.1195 #define OPARR_MEAN 0x01

Calculate the mean value for the elements in the numeric input array

6.2.2.1196 #define OPARR_MIN 0x04

Calculate the minimum value of the elements in the numeric input array

6.2.2.1197 #define OPARR_SORT 0x06

Sort the elements in the numeric input array

6.2.2.1198 #define OPARR_STD 0x03

Calculate the standard deviation of the elements in the numeric input array

6.2.2.1199 #define OPARR_SUM 0x00

Calculate the sum of the elements in the numeric input array

6.2.2.1200 #define OPARR_SUMSQR 0x02

Calculate the sum of the squares of the elements in the numeric input array

6.2.2.1201 #define OUT_A 0x00

Output port A

6.2.2.1202 #define OUT_AB 0x03

Output ports A and B

6.2.2.1203 #define OUT_ABC 0x06

Output ports A, B, and C

6.2.2.1204 #define OUT_AC 0x04

Output ports A and C

6.2.2.1205 #define OUT_B 0x01

Output port B

6.2.2.1206 #define OUT_BC 0x05

Output ports B and C

6.2.2.1207 #define OUT_C 0x02

Output port C

6.2.2.1208 #define OUT_MODE_BRAKE 0x02

Uses electronic braking to outputs

6.2.2.1209 #define OUT_MODE_COAST 0x00

No power and no braking so motors rotate freely.

6.2.2.1210 #define OUT_MODE_MOTORON 0x01

Enables PWM power to the outputs given the power setting

6.2.2.1211 #define OUT_MODE_REGMETHOD 0xF0

Mask for unimplemented regulation mode

6.2.2.1212 #define OUT_MODE_REGULATED 0x04

Enables active power regulation using the regulation mode value

6.2.2.1213 #define OUT_OPTION_HOLDATLIMIT 0x10

Option to have the firmware hold the motor when it reaches the tachometer limit

6.2.2.1214 #define OUT_OPTION_RAMPDOWNTOLIMIT 0x20

Option to have the firmware rampdown the motor power as it approaches the tachometer limit

6.2.2.1215 #define OUT_REGMODE_IDLE 0

No motor regulation.

6.2.2.1216 #define OUT_REGMODE_POS 4

Regulate a motor's position.

6.2.2.1217 #define OUT_REGMODE_SPEED 1

Regulate a motor's speed (aka power).

6.2.2.1218 #define OUT_REGMODE_SYNC 2

Synchronize the rotation of two motors.

6.2.2.1219 #define OUT_REGOPTION_NO_SATURATION 0x01

Do not limit intermediary regulation results

6.2.2.1220 #define OUT_RUNSTATE_HOLD 0x60

Set motor run state to hold at the current position.

6.2.2.1221 #define OUT_RUNSTATE_IDLE 0x00

Disable all power to motors.

6.2.2.1222 #define OUT_RUNSTATE_RAMPDOWN 0x40

Enable ramping down from a current power to a new (lower) power over a specified [TachoLimitField](#) goal.

6.2.2.1223 #define OUT_RUNSTATE_RAMPUP 0x10

Enable ramping up from a current power to a new (higher) power over a specified [TachoLimitField](#) goal.

6.2.2.1224 #define OUT_RUNSTATE_RUNNING 0x20

Enable power to motors at the specified power level.

6.2.2.1225 #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.

6.2.2.1226 #define OutputModuleID 0x00020001

The output module ID

6.2.2.1227 #define OutputModuleName "Output.mod"

The output module name

6.2.2.1228 #define OutputOffsetActualSpeed(p) (((p)*32)+21)

R - Holds the current motor speed (1 byte) sbyte

6.2.2.1229 #define OutputOffsetBlockTachoCount(p) (((p)*32)+4)

R - Holds current number of counts for the current output block (4 bytes) slong

6.2.2.1230 #define OutputOffsetFlags(p) (((p)*32)+18)

RW - Holds flags for which data should be updated (1 byte) ubyte

6.2.2.1231 #define OutputOffsetMaxAccel(p) (((p)*32)+31)

RW - holds the maximum acceleration for position regulation (1 byte) sbyte (NBC/NXC)

6.2.2.1232 #define OutputOffsetMaxSpeed(p) (((p)*32)+30)

RW - holds the maximum speed for position regulation (1 byte) sbyte (NBC/NXC)

6.2.2.1233 #define OutputOffsetMode(p) (((p)*32)+19)

RW - Holds motor mode: Run, Break, regulated, ... (1 byte) ubyte

6.2.2.1234 #define OutputOffsetMotorRPM(p) (((p)*32)+16)

Not updated, will be removed later !! (2 bytes) sword

6.2.2.1235 #define OutputOffsetOptions(p) (((p)*32)+29)

RW - holds extra motor options related to the tachometer limit (1 byte) ubyte (NBC/NXC)

6.2.2.1236 #define OutputOffsetOverloaded(p) (((p)*32)+27)

R - True if the motor has been overloaded within speed control regulation (1 byte) ubyte

6.2.2.1237 #define OutputOffsetRegDParameter(p) (((p)*32)+24)

RW - Holds the D-constant used in the regulation (1 byte) ubyte

6.2.2.1238 #define OutputOffsetRegIParameter(p) (((p)*32)+23)

RW - Holds the I-constant used in the regulation (1 byte) ubyte

6.2.2.1239 #define OutputOffsetRegMode(p) (((p)*32)+26)

RW - Tells which regulation mode should be used (1 byte) ubyte

6.2.2.1240 #define OutputOffsetRegPParameter(p) (((p)*32)+22)

RW - Holds the P-constant used in the regulation (1 byte) ubyte

6.2.2.1241 #define OutputOffsetRegulationOptions 97

use for position regulation options (1 byte) ubyte (NBC/NXC)

6.2.2.1242 #define OutputOffsetRegulationTime 96

use for frequency of checking regulation mode (1 byte) ubyte (NBC/NXC)

6.2.2.1243 #define OutputOffsetRotationCount(p) (((p)*32)+8)

R - Holds current number of counts for the rotation counter to the output (4 bytes)
slong

6.2.2.1244 #define OutputOffsetRunState(p) (((p)*32)+25)

RW - Holds the current motor run state in the output module (1 byte) ubyte

6.2.2.1245 #define OutputOffsetSpeed(p) (((p)*32)+20)

RW - Holds the wanted speed (1 byte) sbyte

6.2.2.1246 #define OutputOffsetSyncTurnParameter(p) (((p)*32)+28)

RW - Holds the turning parameter need within MoveBlock (1 byte) sbyte

6.2.2.1247 #define OutputOffsetTachoCount(p) (((p)*32)+0)

R - Holds current number of counts, since last reset, updated every 1 mS (4 bytes)
slong

6.2.2.1248 #define OutputOffsetTachoLimit(p) (((p)*32)+12)

RW - Holds number of counts to travel, 0 => Run forever (4 bytes) ulong

6.2.2.1249 #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.

6.2.2.1250 #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](#).

6.2.2.1251 #define PF_CHANNEL_1 0

Power function channel 1

6.2.2.1252 #define PF_CHANNEL_2 1

Power function channel 2

6.2.2.1253 #define PF_CHANNEL_3 2

Power function channel 3

6.2.2.1254 #define PF_CHANNEL_4 3

Power function channel 4

6.2.2.1255 #define PF_CMD_BRAKE 3

Power function command brake

6.2.2.1256 #define PF_CMD_FLOAT 0

Power function command float (same as stop)

6.2.2.1257 #define PF_CMD_FWD 1

Power function command forward

6.2.2.1258 #define PF_CMD_REV 2

Power function command reverse

6.2.2.1259 #define PF_CMD_STOP 0

Power function command stop

6.2.2.1260 #define PF_CST_CLEAR1_CLEAR2 0

Power function CST clear 1 and clear 2

6.2.2.1261 #define PF_CST_CLEAR1_SET2 2

Power function CST clear 1 and set 2

6.2.2.1262 #define PF_CST_DECREMENT_PWM 5

Power function CST decrement PWM

6.2.2.1263 #define PF_CST_FULL_FWD 6

Power function CST full forward

6.2.2.1264 #define PF_CST_FULL_REV 7

Power function CST full reverse

6.2.2.1265 #define PF_CST_INCREMENT_PWM 4

Power function CST increment PWM

6.2.2.1266 #define PF_CST_SET1_CLEAR2 1

Power function CST set 1 and clear 2

6.2.2.1267 #define PF_CST_SET1_SET2 3

Power function CST set 1 and set 2

6.2.2.1268 #define PF_CST_TOGGLE_DIR 8

Power function CST toggle direction

6.2.2.1269 #define PF_FUNC_CLEAR 1

Power function single pin - clear

6.2.2.1270 #define PF_FUNC_NOCHANGE 0

Power function single pin - no change

6.2.2.1271 #define PF_FUNC_SET 2

Power function single pin - set

6.2.2.1272 #define PF_FUNC_TOGGLE 3

Power function single pin - toggle

6.2.2.1273 #define PF_MODE_COMBO_DIRECT 1

Power function mode combo direct

6.2.2.1274 #define PF_MODE_COMBO_PWM 4

Power function mode combo pulse width modulation (PWM)

6.2.2.1275 #define PF_MODE_SINGLE_OUTPUT_CST 6

Power function mode single output clear, set, toggle (CST)

6.2.2.1276 #define PF_MODE_SINGLE_OUTPUT_PWM 4

Power function mode single output pulse width modulation (PWM)

6.2.2.1277 #define PF_MODE_SINGLE_PIN_CONT 2

Power function mode single pin continuous

6.2.2.1278 #define PF_MODE_SINGLE_PIN_TIME 3

Power function mode single pin timed

6.2.2.1279 #define PF_MODE_TRAIN 0

Power function mode IR Train

6.2.2.1280 #define PF_OUT_A 0

Power function output A

6.2.2.1281 #define PF_OUT_B 1

Power function output B

6.2.2.1282 #define PF_PIN_C1 0

Power function pin C1

6.2.2.1283 #define PF_PIN_C2 1

Power function pin C2

6.2.2.1284 #define PF_PWM_BRAKE 8

Power function PWM brake

6.2.2.1285 #define PF_PWM_FLOAT 0

Power function PWM float

6.2.2.1286 #define PF_PWM_FWD1 1

Power function PWM foward level 1

6.2.2.1287 #define PF_PWM_FWD2 2

Power function PWM foward level 2

6.2.2.1288 #define PF_PWM_FWD3 3

Power function PWM foward level 3

6.2.2.1289 #define PF_PWM_FWD4 4

Power function PWM foward level 4

6.2.2.1290 #define PF_PWM_FWD5 5

Power function PWM foward level 5

6.2.2.1291 #define PF_PWM_FWD6 6

Power function PWM foward level 6

6.2.2.1292 #define PF_PWM_FWD7 7

Power function PWM foward level 7

6.2.2.1293 #define PF_PWM_REV1 15

Power function PWM reverse level 1

6.2.2.1294 #define PF_PWM_REV2 14

Power function PWM reverse level 2

6.2.2.1295 #define PF_PWM_REV3 13

Power function PWM reverse level 3

6.2.2.1296 #define PF_PWM_REV4 12

Power function PWM reverse level 4

6.2.2.1297 #define PF_PWM_REV5 11

Power function PWM reverse level 5

6.2.2.1298 #define PF_PWM_REV6 10

Power function PWM reverse level 6

6.2.2.1299 #define PF_PWM_REV7 9

Power function PWM reverse level 7

6.2.2.1300 #define PFMATE_CHANNEL_1 1

Power function channel 1

6.2.2.1301 #define PFMATE_CHANNEL_2 2

Power function channel 2

6.2.2.1302 #define PFMATE_CHANNEL_3 3

Power function channel 3

6.2.2.1303 #define PFMATE_CHANNEL_4 4

Power function channel 4

6.2.2.1304 #define PFMATE_CMD_GO 0x47

Send IR signal to IR receiver

6.2.2.1305 #define PFMATE_CMD_RAW 0x52

Send raw IR signal to IR receiver

6.2.2.1306 #define PFMATE_MOTORS_A 0x01

Control only motor A

6.2.2.1307 #define PFMATE_MOTORS_B 0x02

Control only motor B

6.2.2.1308 #define PFMATE_MOTORS_BOTH 0x00

Control both motors

6.2.2.1309 #define PFMATE_REG_A_CMD 0x44

PF command for motor A? (PF_CMD_FLOAT, PF_CMD_FWD, PF_CMD_REV,
PF_CMD_BRAKE)

6.2.2.1310 #define PFMATE_REG_A_SPEED 0x45

PF speed for motor A? (0-7)

6.2.2.1311 #define PFMATE_REG_B_CMD 0x46

PF command for motor B? (PF_CMD_FLOAT, PF_CMD_FWD, PF_CMD_REV,
PF_CMD_BRAKE)

6.2.2.1312 #define PFMATE_REG_B_SPEED 0x47

PF speed for motor B? (0-7)

6.2.2.1313 #define PFMATE_REG_CHANNEL 0x42

PF channel? 1, 2, 3, or 4

6.2.2.1314 #define PFMATE_REG_CMD 0x41

PFMate command

6.2.2.1315 #define PFMATE_REG_MOTORS 0x43

PF motors? (0 = both, 1 = A, 2 = B)

6.2.2.1316 #define PI 3.141593

A constant for PI

6.2.2.1317 #define PID_0 0

PID zero

6.2.2.1318 #define PID_1 32

PID one

6.2.2.1319 #define PID_2 64

PID two

6.2.2.1320 #define PID_3 96

PID three

6.2.2.1321 #define PID_4 128

PID four

6.2.2.1322 #define PID_5 160

PID five

6.2.2.1323 #define PID_6 192

PID six

6.2.2.1324 #define PID_7 224

PID seven

6.2.2.1325 #define POOL_MAX_SIZE 32768

Maximum size of memory pool, in bytes

6.2.2.1326 #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.

6.2.2.1327 #define PROG_ABORT 4

Program has been aborted

6.2.2.1328 #define PROG_ERROR 3

A program error has occurred

6.2.2.1329 #define PROG_IDLE 0

Program state is idle

6.2.2.1330 #define PROG_OK 1

Program state is okay

6.2.2.1331 #define PROG_RESET 5

Program has been reset

6.2.2.1332 #define PROG_RUNNING 2

Program is running

6.2.2.1333 #define PSP_BTNSET1_DOWN 0x40

The PSP-Nx button set 1 down arrow

6.2.2.1334 #define PSP_BTNSET1_L3 0x02

The PSP-Nx button set 1 L3

6.2.2.1335 #define PSP_BTNSET1_LEFT 0x80

The PSP-Nx button set 1 left arrow

6.2.2.1336 #define PSP_BTNSET1_R3 0x04

The PSP-Nx button set 1 R3

6.2.2.1337 #define PSP_BTNSET1_RIGHT 0x20

The PSP-Nx button set 1 right arrow

6.2.2.1338 #define PSP_BTNSET1_SELECT 0x01

The PSP-Nx button set 1 select

6.2.2.1339 #define PSP_BTNSET1_START 0x08

The PSP-Nx button set 1 start

6.2.2.1340 #define PSP_BTNSET1_UP 0x10

The PSP-Nx button set 1 up arrow

6.2.2.1341 #define PSP_BTNSET2_CIRCLE 0x20

The PSP-Nx button set 2 circle

6.2.2.1342 #define PSP_BTNSET2_CROSS 0x40

The PSP-Nx button set 2 cross

6.2.2.1343 #define PSP_BTNSET2_L1 0x04

The PSP-Nx button set 2 L1

6.2.2.1344 #define PSP_BTNSET2_L2 0x01

The PSP-Nx button set 2 L2

6.2.2.1345 #define PSP_BTNSET2_R1 0x08

The PSP-Nx button set 2 R1

6.2.2.1346 #define PSP_BTNSET2_R2 0x02

The PSP-Nx button set 2 R2

6.2.2.1347 #define PSP_BTNSET2_SQUARE 0x80

The PSP-Nx button set 2 square

6.2.2.1348 #define PSP_BTNSET2_TRIANGLE 0x10

The PSP-Nx button set 2 triangle

6.2.2.1349 #define PSP_CMD_ANALOG 0x73

Set the PSP-Nx to analog mode

6.2.2.1350 #define PSP_CMD_DIGITAL 0x41

Set the PSP-Nx to digital mode

6.2.2.1351 #define PSP_REG_BTNSET1 0x42

The PSP-Nx button set 1 register

6.2.2.1352 #define PSP_REG_BTNSET2 0x43

The PSP-Nx button set 2 register

6.2.2.1353 #define PSP_REG_XLEFT 0x44

The PSP-Nx X left register

6.2.2.1354 #define PSP_REG_XRIGHT 0x46

The PSP-Nx X right register

6.2.2.1355 #define PSP_REG_YLEFT 0x45

The PSP-Nx Y left register

6.2.2.1356 #define PSP_REG_YRIGHT 0x47

The PSP-Nx Y right register

6.2.2.1357 #define RADIANS_PER_DEGREE PI/180

Used for converting from degrees to radians

6.2.2.1358 #define RAND_MAX 2147483646

The maximum long random number returned by rand

6.2.2.1359 #define RandomEx 99

Generate a random number or seed the RNG.

6.2.2.1360 #define RandomNumber 24

Generate a random number

6.2.2.1361 #define RawValueField 2

Raw value field. Contains the current raw analog sensor value. Read only.

6.2.2.1362 #define RC_PROP_BTTONOFF 0x0

Set/get whether bluetooth is on or off

6.2.2.1363 #define RC_PROP_DEBUGGING 0xF

Set/get enhanced firmware debugging information

6.2.2.1364 #define RC_PROP_SLEEP_TIMEOUT 0x2

Set/get the NXT sleep timeout value (times 60000)

6.2.2.1365 #define RC_PROP_SOUND_LEVEL 0x1

Set/get the NXT sound level

6.2.2.1366 #define RCX_AbsVarOp 0x74

Absolute value function

6.2.2.1367 #define RCX_AndVarOp 0x84

AND function

6.2.2.1368 #define RCX_AutoOffOp 0xb1

Set auto off timer

6.2.2.1369 #define RCX_BatteryLevelOp 0x30

Read the battery level

6.2.2.1370 #define RCX_BatteryLevelSrc 34

The RCX battery level source

6.2.2.1371 #define RCX_BootModeOp 0x65

Set into book mode

6.2.2.1372 #define RCX_CalibrateEventOp 0x04

Calibrate event

6.2.2.1373 #define RCX_ClearAllEventsOp 0x06

Clear all events

6.2.2.1374 #define RCX_ClearCounterOp 0xb7

Clear a counter

6.2.2.1375 #define RCX_ClearMsgOp 0x90

Clear message

6.2.2.1376 #define RCX_ClearSensorOp 0xd1

Clear a sensor

6.2.2.1377 #define RCX_ClearSoundOp 0x80

Clear sound

6.2.2.1378 #define RCX_ClearTimerOp 0xa1

Clear a timer

6.2.2.1379 #define RCX_ClickCounterSrc 27

The RCX event click counter source

6.2.2.1380 #define RCX_ConstantSrc 2

The RCX constant value source

6.2.2.1381 #define RCX_CounterSrc 21

The RCX counter source

6.2.2.1382 #define RCX_DatalogOp 0x62

Datalog the specified source/value

6.2.2.1383 #define RCX_DatalogRawDirectSrc 42

The RCX direct datalog raw source

6.2.2.1384 #define RCX_DatalogRawIndirectSrc 41

The RCX indirect datalog raw source

6.2.2.1385 #define RCX_DatalogSrcDirectSrc 38

The RCX direct datalog source source

6.2.2.1386 #define RCX_DatalogSrcIndirectSrc 37

The RCX indirect datalog source source

6.2.2.1387 #define RCX_DatalogValueDirectSrc 40

The RCX direct datalog value source

6.2.2.1388 #define RCX_DatalogValueIndirectSrc 39

The RCX indirect datalog value source

6.2.2.1389 #define RCX_DecCounterOp 0xa7

Decrement a counter

6.2.2.1390 #define RCX_DeleteSubOp 0xc1

Delete a subroutine

6.2.2.1391 #define RCX_DeleteSubsOp 0x70

Delete subroutines

6.2.2.1392 #define RCX_DeleteTaskOp 0x61

Delete a task

6.2.2.1393 #define RCX_DeleteTasksOp 0x40

Delete tasks

6.2.2.1394 #define RCX_DirectEventOp 0x03

Fire an event

6.2.2.1395 #define RCX_DisplayOp 0x33

Set LCD display value

6.2.2.1396 #define RCX_DivVarOp 0x44

Divide function

6.2.2.1397 #define RCX_DurationSrc 31

The RCX event duration source

6.2.2.1398 #define RCX_EventStateSrc 25

The RCX event static source

6.2.2.1399 #define RCX_FirmwareVersionSrc 35

The RCX firmware version source

6.2.2.1400 #define RCX_GlobalMotorStatusSrc 17

The RCX global motor status source

6.2.2.1401 #define RCX_GOutputDirOp 0x77

Set global motor direction

6.2.2.1402 #define RCX_GOutputModeOp 0x67

Set global motor mode

6.2.2.1403 #define RCX_GOutputPowerOp 0xa3

Set global motor power levels

6.2.2.1404 #define RCX_HysteresisSrc 30

The RCX event hysteresis source

6.2.2.1405 #define RCX_IncCounterOp 0x97

Increment a counter

6.2.2.1406 #define RCX_IndirectVarSrc 36

The RCX indirect variable source

6.2.2.1407 #define RCX_InputBooleanSrc 13

The RCX input boolean source

6.2.2.1408 #define RCX_InputModeOp 0x42

Set the input mode

6.2.2.1409 #define RCX_InputModeSrc 11

The RCX input mode source

6.2.2.1410 #define RCX_InputRawSrc 12

The RCX input raw source

6.2.2.1411 #define RCX_InputTypeOp 0x32

Set the input type

6.2.2.1412 #define RCX_InputTypeSrc 10

The RCX input type source

6.2.2.1413 #define RCX_InputValueSrc 9

The RCX input value source

6.2.2.1414 #define RCX_IRModeOp 0x31

Set the IR transmit mode

6.2.2.1415 #define RCX_LightOp 0x87

Light opcode

6.2.2.1416 #define RCX_LowerThresholdSrc 29

The RCX event lower threshold source

6.2.2.1417 #define RCX_LSBlinkTimeOp 0xe3

Set the light sensor blink time

6.2.2.1418 #define RCX_LSCalibrateOp 0xc0

Calibrate the light sensor

6.2.2.1419 #define RCX_LSHysteresisOp 0xd3

Set the light sensor hysteresis

6.2.2.1420 #define RCX_LSLowerThreshOp 0xc3

Set the light sensor lower threshold

6.2.2.1421 #define RCX_LSSupperThreshOp 0xb3

Set the light sensor upper threshold

6.2.2.1422 #define RCX_MessageOp 0xf7

Set message

6.2.2.1423 #define RCX_MessageSrc 15

The RCX message source

6.2.2.1424 #define RCX_MulVarOp 0x54

Multiply function

6.2.2.1425 #define RCX_MuteSoundOp 0xd0

Mute sound

6.2.2.1426 #define RCX_OnOffFloatOp 0x21

Control motor state - on, off, float

6.2.2.1427 #define RCX_OrVarOp 0x94

OR function

6.2.2.1428 #define RCX_OUT_A 0x01

RCX Output A

6.2.2.1429 #define RCX_OUT_AB 0x03

RCX Outputs A and B

6.2.2.1430 #define RCX_OUT_ABC 0x07

RCX Outputs A, B, and C

6.2.2.1431 #define RCX_OUT_AC 0x05

RCX Outputs A and C

6.2.2.1432 #define RCX_OUT_B 0x02

RCX Output B

6.2.2.1433 #define RCX_OUT_BC 0x04

RCX Outputs B and C

6.2.2.1434 #define RCX_OUT_C 0x04

RCX Output C

6.2.2.1435 #define RCX_OUT_FLOAT 0

Set RCX output to float

6.2.2.1436 #define RCX_OUT_FULL 7

Set RCX output power level to full

6.2.2.1437 #define RCX_OUT_FWD 0x80

Set RCX output direction to forward

6.2.2.1438 #define RCX_OUT_HALF 3

Set RCX output power level to half

6.2.2.1439 #define RCX_OUT_LOW 0

Set RCX output power level to low

6.2.2.1440 #define RCX_OUT_OFF 0x40

Set RCX output to off

6.2.2.1441 #define RCX_OUT_ON 0x80

Set RCX output to on

6.2.2.1442 #define RCX_OUT_REV 0

Set RCX output direction to reverse

6.2.2.1443 #define RCX_OUT_TOGGLE 0x40

Set RCX output direction to toggle

6.2.2.1444 #define RCX_OutputDirOp 0xe1

Set the motor direction

6.2.2.1445 #define RCX_OutputPowerOp 0x13

Set the motor power level

6.2.2.1446 #define RCX_OutputStatusSrc 3

The RCX output status source

6.2.2.1447 #define RCX_PBTurnOffOp 0x60

Turn off the brick

6.2.2.1448 #define RCX_PingOp 0x10

Ping the brick

6.2.2.1449 #define RCX_PlaySoundOp 0x51

Play a sound

6.2.2.1450 #define RCX_PlayToneOp 0x23

Play a tone

6.2.2.1451 #define RCX_PlayToneVarOp 0x02

Play a tone using a variable

6.2.2.1452 #define RCX_PollMemoryOp 0x63

Poll a memory location

6.2.2.1453 #define RCX_PollOp 0x12

Poll a source/value combination

6.2.2.1454 #define RCX_ProgramSlotSrc 8

The RCX program slot source

6.2.2.1455 #define RCX_RandomSrc 4

The RCX random number source

6.2.2.1456 #define RCX_RemoteKeysReleased 0x0000

All remote keys have been released

6.2.2.1457 #define RCX_RemoteOp 0xd2

Execute simulated remote control buttons

6.2.2.1458 #define RCX_RemoteOutABackward 0x4000

Set output A backward

6.2.2.1459 #define RCX_RemoteOutAForward 0x0800

Set output A forward

6.2.2.1460 #define RCX_RemoteOutBBackward 0x8000

Set output B backward

6.2.2.1461 #define RCX_RemoteOutBForward 0x1000

Set output B forward

6.2.2.1462 #define RCX_RemoteOutCBackward 0x0001

Set output C backward

6.2.2.1463 #define RCX_RemoteOutCForward 0x2000

Set output C forward

6.2.2.1464 #define RCX_RemotePBMessage1 0x0100

Send PB message 1

6.2.2.1465 #define RCX_RemotePBMessage2 0x0200

Send PB message 2

6.2.2.1466 #define RCX_RemotePBMessage3 0x0400

Send PB message 3

6.2.2.1467 #define RCX_RemotePlayASound 0x0080

Play a sound

6.2.2.1468 #define RCX_RemoteSelProgram1 0x0002

Select program 1

6.2.2.1469 #define RCX_RemoteSelProgram2 0x0004

Select program 2

6.2.2.1470 #define RCX_RemoteSelProgram3 0x0008

Select program 3

6.2.2.1471 #define RCX_RemoteSelProgram4 0x0010

Select program 4

6.2.2.1472 #define RCX_RemoteSelProgram5 0x0020

Select program 5

6.2.2.1473 #define RCX_RemoteStopOutOff 0x0040

Stop and turn off outputs

6.2.2.1474 #define RCX_ScoutCounterLimitSrc 22

The Scout counter limit source

6.2.2.1475 #define RCX_ScoutEventFBSrc 24

The Scout event feedback source

6.2.2.1476 #define RCX_ScoutLightParamsSrc 19

The Scout light parameters source

6.2.2.1477 #define RCX_ScoutOp 0x47

Scout opcode

6.2.2.1478 #define RCX_ScoutRulesOp 0xd5

Set Scout rules

6.2.2.1479 #define RCX_ScoutRulesSrc 18

The Scout rules source

6.2.2.1480 #define RCX_ScoutTimerLimitSrc 20

The Scout timer limit source

6.2.2.1481 #define RCX_SelectProgramOp 0x91

Select a program slot

6.2.2.1482 #define RCX_SendUARTDataOp 0xc2

Send data via IR using UART settings

6.2.2.1483 #define RCX_SetCounterOp 0xd4

Set counter value

6.2.2.1484 #define RCX_SetDatalogOp 0x52

Set the datalog size

6.2.2.1485 #define RCX_SetEventOp 0x93

Set an event

6.2.2.1486 #define RCX_SetFeedbackOp 0x83

Set Scout feedback

6.2.2.1487 #define RCX_SetPriorityOp 0xd7

Set task priority

6.2.2.1488 #define RCX_SetSourceValueOp 0x05

Set a source/value

6.2.2.1489 #define RCX_SetTimerLimitOp 0xc4

Set timer limit

6.2.2.1490 #define RCX_SetVarOp 0x14

Set function

6.2.2.1491 #define RCX_SetWatchOp 0x22

Set the watch source/value

6.2.2.1492 #define RCX_SgnVarOp 0x64

Sign function

6.2.2.1493 #define RCX_SoundOp 0x57

Sound opcode

6.2.2.1494 #define RCX_StartTaskOp 0x71

Start a task

6.2.2.1495 #define RCX_StopAllTasksOp 0x50

Stop all tasks

6.2.2.1496 #define RCX_StopTaskOp 0x81

Stop a task

6.2.2.1497 #define RCX_SubVarOp 0x34

Subtract function

6.2.2.1498 #define RCX_SumVarOp 0x24

Sum function

6.2.2.1499 #define RCX_TaskEventsSrc 23

The RCX task events source

6.2.2.1500 #define RCX_TenMSTimerSrc 26

The RCX 10ms timer source

6.2.2.1501 #define RCX_TimerSrc 1

The RCX timer source

6.2.2.1502 #define RCX_UARTSetupSrc 33

The RCX UART setup source

6.2.2.1503 #define RCX_UnlockFirmOp 0xa5

Unlock the firmware

6.2.2.1504 #define RCX_UnlockOp 0x15

Unlock the brick

6.2.2.1505 #define RCX_UnmuteSoundOp 0xe0

Unmute sound

6.2.2.1506 #define RCX_UploadDatalogOp 0xa4

Upload datalog contents

6.2.2.1507 #define RCX_UpperThresholdSrc 28

The RCX event upper threshold source

6.2.2.1508 #define RCX_VariableSrc 0

The RCX variable source

6.2.2.1509 #define RCX_ViewSourceValOp 0xe5

View a source/value

6.2.2.1510 #define RCX_VLLOp 0xe2

Send visual light link (VLL) data

6.2.2.1511 #define RCX_WatchSrc 14

The RCX watch source

6.2.2.1512 #define ReadButton 20

Read the current button state

6.2.2.1513 #define ReadLastResponse 97

Read the last response packet received by the NXT. Optionally clear the value after reading it.

6.2.2.1514 #define ReadSemData 40

Read motor semaphore data

6.2.2.1515 #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.

6.2.2.1516 #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.

6.2.2.1517 #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.

6.2.2.1518 #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.

6.2.2.1519 #define RESET_ALL 0x68

Reset all three tachometer counters

6.2.2.1520 #define RESET_BLOCK_COUNT 0x20

Reset the NXT-G block tachometer counter

6.2.2.1521 #define RESET_BLOCKANDTACHO 0x28

Reset both the internal counter and the NXT-G block counter

6.2.2.1522 #define RESET_COUNT 0x08

Reset the internal tachometer counter

6.2.2.1523 #define RESET_NONE 0x00

No counters will be reset

6.2.2.1524 #define RESET_ROTATION_COUNT 0x40

Reset the rotation counter

6.2.2.1525 #define RFID_MODE_CONTINUOUS 2

Configure the RFID device for continuous reading

6.2.2.1526 #define RFID_MODE_SINGLE 1

Configure the RFID device for a single reading

6.2.2.1527 #define RFID_MODE_STOP 0

Stop the RFID device

6.2.2.1528 #define RICArg(_arg) ((_arg)|0x1000)

Output an RIC parameterized argument.

Parameters:

_arg The argument that you want to parameterize.

6.2.2.1529 #define RICImgPoint(_X, _Y) (_X)&0xFF, (_X)>>8, (_Y)&0xFF, (_Y)>>8

Output an RIC ImgPoint structure.

Parameters:

_X The X coordinate.

_Y The Y coordinate.

6.2.2.1530 #define RICImgRect(_Pt, _W, _H) _Pt, (_W)&0xFF, (_W)>>8, (_H)&0xFF, (_H)>>8

Output an RIC ImgRect structure.

Parameters:

_Pt An ImgPoint. See [RICImgPoint](#).

_W The rectangle width.

_H The rectangle height.

6.2.2.1531 #define RICMapArg(_mapidx, _arg) ((*_arg*)|0x1000|(((*_mapidx*)&0xF)<<8))

Output an RIC parameterized and mapped argument.

Parameters:

_mapidx The varmap data address.

_arg The parameterized argument you want to pass through a varmap.

**6.2.2.1532 #define RICMapElement(_Domain, _Range) (_Domain)&0xFF,
 (_Domain)>>8, (_Range)&0xFF, (_Range)>>8**

Output an RIC map element.

Parameters:

_Domain The map element domain.

_Range The map element range.

**6.2.2.1533 #define RICMapFunction(_MapElement, ...) _MapElement,
 __VA_ARGS__**

Output an RIC VarMap function.

Parameters:

_MapElement An entry in the varmap function. At least 2 elements are required.
See [RICMapElement](#).

**6.2.2.1534 #define RICOpCircle(_CopyOptions, _Point, _Radius) 10,
 0, 7, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point,
 (_Radius)&0xFF, (_Radius)>>8**

Output an RIC Circle opcode.

Parameters:

_CopyOptions Circle copy options. See [Drawing option constants](#).

_Point The circle's center point. See [RICImgPoint](#).

_Radius The circle's radius.

**6.2.2.1535 #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:

_CopyOptions CopyBits copy options. See [Drawing option constants](#).

_DataAddr The address of the sprite from which to copy data.

_SrcRect The rectangular portion of the sprite to copy. See [RICImgRect](#).

_DstPoint The LCD coordinate to which to copy the data. See [RICImgPoint](#).

**6.2.2.1536 #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:

_Options RIC options.

_Width The total RIC width.

_Height The total RIC height.

**6.2.2.1537 #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:

_CopyOptions Ellipse copy options. See [Drawing option constants](#).

_Point The center of the ellipse. See [RICImgPoint](#).

_RadiusX The x-axis radius of the ellipse.

_RadiusY The y-axis radius of the ellipse.

**6.2.2.1538 #define RICOpLine(_CopyOptions, _Point1, _Point2) 12, 0, 5, 0,
 (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point1, _Point2**

Output an RIC Line opcode.

Parameters:

_CopyOptions Line copy options. See [Drawing option constants](#).

_Point1 The starting point of the line. See [RICImgPoint](#).

_Point2 The ending point of the line. See [RICImgPoint](#).

**6.2.2.1539 #define RICOpNumBox(_CopyOptions, _Point, _Value) 10,
 0, 8, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point,
 (_Value)&0xFF, (_Value)>>8**

Output an RIC NumBox opcode.

Parameters:

_CopyOptions NumBox copy options. See [Drawing option constants](#).

_Point The numbox bottom left corner. See [RICImgPoint](#).

_Value The number to draw.

**6.2.2.1540 #define RICOpPixel(_CopyOptions, _Point, _Value) 10, 0,
 4, 0, (_CopyOptions)&0xFF, (_CopyOptions)>>8, _Point,
 (_Value)&0xFF, (_Value)>>8**

Output an RIC Pixel opcode.

Parameters:

_CopyOptions Pixel copy options. See [Drawing option constants](#).

_Point The pixel coordinate. See [RICImgPoint](#).

_Value The pixel value (unused).

```
6.2.2.1541 #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:

_CopyOptions Polygon copy options. See [Drawing option constants](#).

_Count The number of points in the polygon.

_ThePoints The list of polygon points. See [RICPolygonPoints](#).

```
6.2.2.1542 #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:

_CopyOptions Rect copy options. See [Drawing option constants](#).

_Point The rectangle's top left corner. See [RICImgPoint](#).

_Width The rectangle's width.

_Height The rectangle's height.

```
6.2.2.1543 #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:

- _DataAddr* The address of the sprite.
- _Rows* The number of rows of data.
- _BytesPerRow* The number of bytes per row.
- _SpriteData* The actual sprite data. See [RICSpriteData](#).

**6.2.2.1544 #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:

- _DataAddr* The address of the varmap.
- _MapCount* The number of points in the function.
- _MapFunction* The definition of the varmap function. See [RICMapFunction](#).

**6.2.2.1545 #define RICPolygonPoints(_pPoint1, _pPoint2, ...) _pPoint1,
_pPoint2, __VA_ARGS__**

Output RIC polygon points.

Parameters:

- _pPoint1* The first polygon point. See [RICImgPoint](#).
- _pPoint2* The second polygon point (at least 3 points are required). See [RICImgPoint](#).

6.2.2.1546 #define RICSpriteData(...) __VA_ARGS__

Output RIC sprite data.

6.2.2.1547 #define ROTATE_QUEUE 5

VM should rotate queue

6.2.2.1548 #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.

6.2.2.1549 #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](#).

6.2.2.1550 #define SAMPLERATE_DEFAULT 8000

Default sample rate [sps]

6.2.2.1551 #define SAMPLERATE_MAX 16000

Max sample rate [sps]

6.2.2.1552 #define SAMPLERATE_MIN 2000

Min sample rate [sps]

6.2.2.1553 #define ScaledValueField 4

Scaled value field. Contains the current scaled analog sensor value. Read/write.

6.2.2.1554 #define SCHAR_MAX 127

The maximum value of the signed char type

6.2.2.1555 #define SCHAR_MIN -128

The minimum value of the signed char type

6.2.2.1556 #define SCOUT_FXR_ALARM 2

Alarm special effects

6.2.2.1557 #define SCOUT_FXR_BUG 1

Bug special effects

6.2.2.1558 #define SCOUT_FXR_NONE 0

No special effects

6.2.2.1559 #define SCOUT_FXR_RANDOM 3

Random special effects

6.2.2.1560 #define SCOUT_FXR_SCIENCE 4

Science special effects

6.2.2.1561 #define SCOUT_LIGHT_OFF 0

Turn off the scout light

6.2.2.1562 #define SCOUT_LIGHT_ON 0x80

Turn on the scout light

6.2.2.1563 #define SCOUT_LR_AVOID 3

Light rule avoid

6.2.2.1564 #define SCOUT_LR_IGNORE 0

Light rule ignore

6.2.2.1565 #define SCOUT_LR_OFF_WHEN 5

Light rule off when

6.2.2.1566 #define SCOUT_LR_SEEK_DARK 2

Light rule seek dark

6.2.2.1567 #define SCOUT_LR_SEEK_LIGHT 1

Light rule seek light

6.2.2.1568 #define SCOUT_LR_WAIT_FOR 4

Light rule wait for

6.2.2.1569 #define SCOUT_MODE_POWER 1

Enter power mode

6.2.2.1570 #define SCOUT_MODE_STANDALONE 0

Enter stand alone mode

6.2.2.1571 #define SCOUT_MR_CIRCLE_LEFT 4

Motion rule circle left

6.2.2.1572 #define SCOUT_MR_CIRCLE_RIGHT 3

Motion rule circle right

6.2.2.1573 #define SCOUT_MR_FORWARD 1

Motion rule forward

6.2.2.1574 #define SCOUT_MR_LOOP_A 5

Motion rule loop A

6.2.2.1575 #define SCOUT_MR_LOOP_AB 7

Motion rule loop A then B

6.2.2.1576 #define SCOUT_MR_LOOP_B 6

Motion rule loop B

6.2.2.1577 #define SCOUT_MR_NO_MOTION 0

Motion rule none

6.2.2.1578 #define SCOUT_MR_ZIGZAG 2

Motion rule zigzag

6.2.2.1579 #define SCOUT SNDSET_ALARM 3

Set sound set to alarm

6.2.2.1580 #define SCOUT SNDSET_BASIC 1

Set sound set to basic

6.2.2.1581 #define SCOUT SNDSET_BUG 2

Set sound set to bug

6.2.2.1582 #define SCOUT SNDSET_NONE 0

Set sound set to none

6.2.2.1583 #define SCOUT_SNDSET_RANDOM 4

Set sound set to random

6.2.2.1584 #define SCOUT_SNDSET_SCIENCE 5

Set sound set to science

6.2.2.1585 #define SCOUT_SOUND_1_BLINK 17

Play the Scout 1 blink sound

6.2.2.1586 #define SCOUT_SOUND_2_BLINK 18

Play the Scout 2 blink sound

6.2.2.1587 #define SCOUT_SOUND_COUNTER1 19

Play the Scout counter 1 sound

6.2.2.1588 #define SCOUT_SOUND_COUNTER2 20

Play the Scout counter 2 sound

6.2.2.1589 #define SCOUT_SOUND_ENTER_BRIGHT 14

Play the Scout enter bright sound

6.2.2.1590 #define SCOUT_SOUND_ENTER_DARK 16

Play the Scout enter dark sound

6.2.2.1591 #define SCOUT_SOUND_ENTER_NORMAL 15

Play the Scout enter normal sound

6.2.2.1592 #define SCOUT_SOUND_ENTERSA 7

Play the Scout enter standalone sound

6.2.2.1593 #define SCOUT_SOUND_KEYERROR 8

Play the Scout key error sound

6.2.2.1594 #define SCOUT_SOUND_MAIL_RECEIVED 24

Play the Scout mail received sound

6.2.2.1595 #define SCOUT_SOUND_NONE 9

Play the Scout none sound

6.2.2.1596 #define SCOUT_SOUND_REMOTE 6

Play the Scout remote sound

6.2.2.1597 #define SCOUT_SOUND_SPECIAL1 25

Play the Scout special 1 sound

6.2.2.1598 #define SCOUT_SOUND_SPECIAL2 26

Play the Scout special 2 sound

6.2.2.1599 #define SCOUT_SOUND_SPECIAL3 27

Play the Scout special 3 sound

6.2.2.1600 #define SCOUT_SOUND_TIMER1 21

Play the Scout timer 1 sound

6.2.2.1601 #define SCOUT_SOUND_TIMER2 22

Play the Scout timer 2 sound

6.2.2.1602 #define SCOUT_SOUND_TIMER3 23

Play the Scout timer 3 sound

6.2.2.1603 #define SCOUT_SOUND_TOUCH1_PRES 10

Play the Scout touch 1 pressed sound

6.2.2.1604 #define SCOUT_SOUND_TOUCH1_REL 11

Play the Scout touch 1 released sound

6.2.2.1605 #define SCOUT_SOUND_TOUCH2_PRES 12

Play the Scout touch 2 pressed sound

6.2.2.1606 #define SCOUT_SOUND_TOUCH2_REL 13

Play the Scout touch 2 released sound

6.2.2.1607 #define SCOUT_TGS_LONG 2

Transmit level long

6.2.2.1608 #define SCOUT_TGS_MEDIUM 1

Transmit level medium

6.2.2.1609 #define SCOUT_TGS_SHORT 0

Transmit level short

6.2.2.1610 #define SCOUT_TR_AVOID 2

Touch rule avoid

6.2.2.1611 #define SCOUT_TR_IGNORE 0

Touch rule ignore

6.2.2.1612 #define SCOUT_TR_OFF_WHEN 4

Touch rule off when

6.2.2.1613 #define SCOUT_TR_REVERSE 1

Touch rule reverse

6.2.2.1614 #define SCOUT_TR_WAIT_FOR 3

Touch rule wait for

6.2.2.1615 #define SCREEN_BACKGROUND 0

Entire screen

6.2.2.1616 #define SCREEN_LARGE 1

Entire screen except status line

6.2.2.1617 #define SCREEN_MODE_CLEAR 0x01

Clear the screen

See also:

[SetScreenMode\(\)](#)

6.2.2.1618 #define SCREEN_MODE_RESTORE 0x00

Restore the screen

See also:

[SetScreenMode\(\)](#)

6.2.2.1619 #define SCREEN_SMALL 2

Screen between menu icons and status line

6.2.2.1620 #define SCREENS 3

The number of screen bits

6.2.2.1621 #define SEC_1 1000

1 second

6.2.2.1622 #define SEC_10 10000

10 seconds

6.2.2.1623 #define SEC_15 15000

15 seconds

6.2.2.1624 #define SEC_2 2000

2 seconds

6.2.2.1625 #define SEC_20 20000

20 seconds

6.2.2.1626 #define SEC_3 3000

3 seconds

6.2.2.1627 #define SEC_30 30000

30 seconds

6.2.2.1628 #define SEC_4 4000

4 seconds

6.2.2.1629 #define SEC_5 5000

5 seconds

6.2.2.1630 #define SEC_6 6000

6 seconds

6.2.2.1631 #define SEC_7 7000

7 seconds

6.2.2.1632 #define SEC_8 8000

8 seconds

6.2.2.1633 #define SEC_9 9000

9 seconds

6.2.2.1634 #define SetScreenMode 19

Set the screen mode

6.2.2.1635 #define SetSleepTimeoutVal 46

Set the NXT sleep timeout value

6.2.2.1636 #define SHRT_MAX 32767

The maximum value of the short type

6.2.2.1637 #define SHRT_MIN -32768

The minimum value of the short type

6.2.2.1638 #define SIZE_OF_BDADDR 7

Size of Bluetooth Address

6.2.2.1639 #define SIZE_OF_BRICK_NAME 8

Size of NXT Brick name

6.2.2.1640 #define SIZE_OF_BT_CONNECT_TABLE 4

Size of Bluetooth connection table -- Index 0 is always incoming connection

6.2.2.1641 #define SIZE_OF_BT_DEVICE_TABLE 30

Size of Bluetooth device table

6.2.2.1642 #define SIZE_OF_BT_NAME 16

Size of Bluetooth name

6.2.2.1643 #define SIZE_OF_BT_PINCODE 16

Size of Bluetooth PIN

6.2.2.1644 #define SIZE_OF_BTBUF 128

Size of Bluetooth buffer

6.2.2.1645 #define SIZE_OF_CLASS_OF_DEVICE 4

Size of class of device

6.2.2.1646 #define SIZE_OF_HSBUF 128

Size of High Speed Port 4 buffer

6.2.2.1647 #define SIZE_OF_USBBUF 64

Size of USB Buffer in bytes

6.2.2.1648 #define SIZE_OF_USBDATA 62

Size of USB Buffer available for data

6.2.2.1649 #define SOUND_CLICK 0

Play the standard key click sound

6.2.2.1650 #define SOUND_DOUBLE_BEEP 1

Play the standard double beep sound

6.2.2.1651 #define SOUND_DOWN 2

Play the standard sweep down sound

6.2.2.1652 #define SOUND_FAST_UP 5

Play the standard fast up sound

6.2.2.1653 #define SOUND_FLAGS_IDLE 0x00

R - Sound is idle

6.2.2.1654 #define SOUND_FLAGS_RUNNING 0x02

R - Currently processing a tone or file

6.2.2.1655 #define SOUND_FLAGS_UPDATE 0x01

W - Make changes take effect

6.2.2.1656 #define SOUND_LOW_BEEP 4

Play the standard low beep sound

6.2.2.1657 #define SOUND_MODE_LOOP 0x01

W - Play file until writing SOUND_STATE_STOP into SoundState

6.2.2.1658 #define SOUND_MODE_ONCE 0x00

W - Only play file once

6.2.2.1659 #define SOUND_MODE_TONE 0x02

W - Play tone specified in Frequency for Duration ms

6.2.2.1660 #define SOUND_STATE_FILE 0x02

R - Processing a file of sound/melody data

6.2.2.1661 #define SOUND_STATE_IDLE 0x00

R - Idle, ready for start sound (SOUND_UPDATE)

6.2.2.1662 #define SOUND_STATE_STOP 0x04

W - Stop sound immediately and close hardware

6.2.2.1663 #define SOUND_STATE_TONE 0x03

R - Processing a play tone request

6.2.2.1664 #define SOUND_UP 3

Play the standard sweep up sound

6.2.2.1665 #define SoundGetState 11

Get the current sound module state

6.2.2.1666 #define SoundModuleID 0x00080001

The sound module ID

6.2.2.1667 #define SoundModuleName "Sound.mod"

The sound module name

6.2.2.1668 #define SoundOffsetDuration 2

RW - Tone duration [mS] (2 bytes)

6.2.2.1669 #define SoundOffsetFlags 26

RW - Play flag - described above (1 byte) [SoundFlags constants](#)

6.2.2.1670 #define SoundOffsetFreq 0

RW - Tone frequency [Hz] (2 bytes)

6.2.2.1671 #define SoundOffsetMode 28

RW - Play mode - described above (1 byte) [SoundMode constants](#)

6.2.2.1672 #define SoundOffsetSampleRate 4

RW - Sound file sample rate [2000..16000] (2 bytes)

6.2.2.1673 #define SoundOffsetSoundFilename 6

RW - Sound/melody filename (20 bytes)

6.2.2.1674 #define SoundOffsetState 27

RW - Play state - described above (1 byte) [SoundState constants](#)

6.2.2.1675 #define SoundOffsetVolume 29

RW - Sound/melody volume [0..4] 0 = off (1 byte)

6.2.2.1676 #define SoundPlayFile 9

Play a sound or melody file

6.2.2.1677 #define SoundPlayTone 10

Play a simple tone with the specified frequency and duration

6.2.2.1678 #define SoundSetState 12

Set the sound module state

6.2.2.1679 #define SPECIALS 5

The number of special bit values

6.2.2.1680 #define STAT_COMM_PENDING 32

Pending setup operation in progress

6.2.2.1681 #define STAT_MSG_EMPTY_MAILBOX 64

Specified mailbox contains no new messages

6.2.2.1682 #define STATUSICON_BATTERY 3

Battery status icon collection

6.2.2.1683 #define STATUSICON_BLUETOOTH 0

BlueTooth status icon collection

6.2.2.1684 #define STATUSICON_USB 1

USB status icon collection

6.2.2.1685 #define STATUSICON_VM 2

VM status icon collection

6.2.2.1686 #define STATUSICONS 4

The number of status icons

6.2.2.1687 #define STATUSTEXT 1

Status text (BT name)

6.2.2.1688 #define STEPICON_1 0

Left most step icon

6.2.2.1689 #define STEPICON_2 1

6.2.2.1690 #define STEPICON_3 2

6.2.2.1691 #define STEPICON_4 3

Right most step icon

6.2.2.1692 #define STEPICON_5 4

6.2.2.1693 #define STEPICONS 5

6.2.2.1694 #define STEPLINE 3

Step collection lines

6.2.2.1695 #define STOP_REQ 4

VM should stop executing program

6.2.2.1696 #define STROBE_READ 0x10

Access read pin (RD)

6.2.2.1697 #define STROBE_S0 0x01

Access strobe 0 pin (S0)

6.2.2.1698 #define STROBE_S1 0x02

Access strobe 1 pin (S1)

6.2.2.1699 #define STROBE_S2 0x04

Access strobe 2 pin (S2)

6.2.2.1700 #define STROBE_S3 0x08

Access strobe 3 pin (S3)

6.2.2.1701 #define STROBE_WRITE 0x20

Access write pin (WR)

6.2.2.1702 #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.

6.2.2.1703 #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.

6.2.2.1704 #define TEMP_FQ_1 0x00

Set fault queue to 1 fault before alert

6.2.2.1705 #define TEMP_FQ_2 0x08

Set fault queue to 2 faults before alert

6.2.2.1706 #define TEMP_FQ_4 0x10

Set fault queue to 4 faults before alert

6.2.2.1707 #define TEMP_FQ_6 0x18

Set fault queue to 6 faults before alert

6.2.2.1708 #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.

6.2.2.1709 #define TEMP_POL_HIGH 0x04

Set polarity of ALERT pin to be active HIGH

6.2.2.1710 #define TEMP_POL_LOW 0x00

Set polarity of ALERT pin to be active LOW

6.2.2.1711 #define TEMP_REG_CONFIG 0x01

The register for reading/writing sensor configuration values

6.2.2.1712 #define TEMP_REG_TEMP 0x00

The register where temperature values can be read

6.2.2.1713 #define TEMP_REG_THIGH 0x03

The register for reading/writing a user-defined high temperature limit

6.2.2.1714 #define TEMP_REG_TLOW 0x02

The register for reading/writing a user-defined low temperature limit

6.2.2.1715 #define TEMP_RES_10BIT 0x20

Set the temperature conversion resolution to 10 bit

6.2.2.1716 #define TEMP_RES_11BIT 0x40

Set the temperature conversion resolution to 11 bit

6.2.2.1717 #define TEMP_RES_12BIT 0x60

Set the temperature conversion resolution to 12 bit

6.2.2.1718 #define TEMP_RES_9BIT 0x00

Set the temperature conversion resolution to 9 bit

6.2.2.1719 #define TEMP_SD_CONTINUOUS 0x00

Set the sensor mode to continuous

6.2.2.1720 #define TEMP_SD_SHUTDOWN 0x01

Set the sensor mode to shutdown. The device will shut down after the current conversion is completed.

6.2.2.1721 #define TEMP_TM_COMPARATOR 0x00

Set the thermostat mode to comparator

6.2.2.1722 #define TEMP_TM_INTERRUPT 0x02

Set the thermostat mode to interrupt

6.2.2.1723 #define TEXTLINE_1 0

Text line 1

6.2.2.1724 #define TEXTLINE_2 1

Text line 2

6.2.2.1725 #define TEXTLINE_3 2

Text line 3

6.2.2.1726 #define TEXTLINE_4 3

Text line 4

6.2.2.1727 #define TEXTLINE_5 4

Text line 5

6.2.2.1728 #define TEXTLINE_6 5

Text line 6

6.2.2.1729 #define TEXTLINE_7 6

Text line 7

6.2.2.1730 #define TEXTLINE_8 7

Text line 8

6.2.2.1731 #define TEXTLINES 8

The number of text lines on the LCD

6.2.2.1732 #define TIMES_UP 6

VM time is up

6.2.2.1733 #define TONE_A3 220

Third octave A

6.2.2.1734 #define TONE_A4 440

Fourth octave A

6.2.2.1735 #define TONE_A5 880

Fifth octave A

6.2.2.1736 #define TONE_A6 1760

Sixth octave A

6.2.2.1737 #define TONE_A7 3520

Seventh octave A

6.2.2.1738 #define TONE_AS3 233

Third octave A sharp

6.2.2.1739 #define TONE_AS4 466

Fourth octave A sharp

6.2.2.1740 #define TONE_AS5 932

Fifth octave A sharp

6.2.2.1741 #define TONE_AS6 1865

Sixth octave A sharp

6.2.2.1742 #define TONE_AS7 3729

Seventh octave A sharp

6.2.2.1743 #define TONE_B3 247

Third octave B

6.2.2.1744 #define TONE_B4 494

Fourth octave B

6.2.2.1745 #define TONE_B5 988

Fifth octave B

6.2.2.1746 #define TONE_B6 1976

Sixth octave B

6.2.2.1747 #define TONE_B7 3951

Seventh octave B

6.2.2.1748 #define TONE_C4 262

Fourth octave C

6.2.2.1749 #define TONE_C5 523

Fifth octave C

6.2.2.1750 #define TONE_C6 1047

Sixth octave C

6.2.2.1751 #define TONE_C7 2093

Seventh octave C

6.2.2.1752 #define TONE_CS4 277

Fourth octave C sharp

6.2.2.1753 #define TONE_CS5 554

Fifth octave C sharp

6.2.2.1754 #define TONE_CS6 1109

Sixth octave C sharp

6.2.2.1755 #define TONE_CS7 2217

Seventh octave C sharp

6.2.2.1756 #define TONE_D4 294

Fourth octave D

6.2.2.1757 #define TONE_D5 587

Fifth octave D

6.2.2.1758 #define TONE_D6 1175

Sixth octave D

6.2.2.1759 #define TONE_D7 2349

Seventh octave D

6.2.2.1760 #define TONE_DS4 311

Fourth octave D sharp

6.2.2.1761 #define TONE_DS5 622

Fifth octave D sharp

6.2.2.1762 #define TONE_DS6 1245

Sixth octave D sharp

6.2.2.1763 #define TONE_DS7 2489

Seventh octave D sharp

6.2.2.1764 #define TONE_E4 330

Fourth octave E

6.2.2.1765 #define TONE_E5 659

Fifth octave E

6.2.2.1766 #define TONE_E6 1319

Sixth octave E

6.2.2.1767 #define TONE_E7 2637

Seventh octave E

6.2.2.1768 #define TONE_F4 349

Fourth octave F

6.2.2.1769 #define TONE_F5 698

Fifth octave F

6.2.2.1770 #define TONE_F6 1397

Sixth octave F

6.2.2.1771 #define TONE_F7 2794

Seventh octave F

6.2.2.1772 #define TONE_FS4 370

Fourth octave F sharp

6.2.2.1773 #define TONE_FS5 740

Fifth octave F sharp

6.2.2.1774 #define TONE_FS6 1480

Sixth octave F sharp

6.2.2.1775 #define TONE_FS7 2960

Seventh octave F sharp

6.2.2.1776 #define TONE_G4 392

Fourth octave G

6.2.2.1777 #define TONE_G5 784

Fifth octave G

6.2.2.1778 #define TONE_G6 1568

Sixth octave G

6.2.2.1779 #define TONE_G7 3136

Seventh octave G

6.2.2.1780 #define TONE_GS4 415

Fourth octave G sharp

6.2.2.1781 #define TONE_GS5 831

Fifth octave G sharp

6.2.2.1782 #define TONE_GS6 1661

Sixth octave G sharp

6.2.2.1783 #define TONE_GS7 3322

Seventh octave G sharp

6.2.2.1784 #define TOPLINE 4

Top status underline

6.2.2.1785 #define TRAIN_CHANNEL_1 0

IR Train channel 1

6.2.2.1786 #define TRAIN_CHANNEL_2 1

IR Train channel 2

6.2.2.1787 #define TRAIN_CHANNEL_3 2

IR Train channel 3

6.2.2.1788 #define TRAIN_CHANNEL_ALL 3

IR Train channel all

6.2.2.1789 #define TRAIN_FUNC_DECR_SPEED 2

PF/IR Train function decrement speed

6.2.2.1790 #define TRAIN_FUNC_INCR_SPEED 1

PF/IR Train function increment speed

6.2.2.1791 #define TRAIN_FUNC_STOP 0

PF/IR Train function stop

6.2.2.1792 #define TRAIN_FUNC_TOGGLE_LIGHT 4

PF/IR Train function toggle light

6.2.2.1793 #define TRUE 1

A true value

6.2.2.1794 #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.

6.2.2.1795 #define TypeField 0

Type field. Contains one of the sensor type constants. Read/write.

6.2.2.1796 #define UCHAR_MAX 255

The maximum value of the unsigned char type

6.2.2.1797 #define UF_PENDING_UPDATES 0x80

Are there any pending motor updates?

6.2.2.1798 #define UF_UPDATE_MODE 0x01

Commits changes to the [OutputModeField](#) output property

6.2.2.1799 #define UF_UPDATE_PID_VALUES 0x10

Commits changes to the PID motor regulation properties

6.2.2.1800 #define UF_UPDATE_RESET_BLOCK_COUNT 0x20

Resets the NXT-G block-relative rotation counter

6.2.2.1801 #define UF_UPDATE_RESET_COUNT 0x08

Resets all rotation counters, cancels the current goal, and resets the rotation error-correction system

6.2.2.1802 #define UF_UPDATE_RESET_ROTATION_COUNT 0x40

Resets the program-relative (user) rotation counter

6.2.2.1803 #define UF_UPDATE_SPEED 0x02

Commits changes to the [PowerField](#) output property

6.2.2.1804 #define UF_UPDATE_TACHO_LIMIT 0x04

Commits changes to the [TachoLimitField](#) output property

6.2.2.1805 #define UI_BT_CONNECT_REQUEST 0x40

RW - BT get connect accept in progress

6.2.2.1806 #define UI_BT_ERROR_ATTENTION 0x08

W - BT error attention

6.2.2.1807 #define UI_BT_PIN_REQUEST 0x80

RW - BT get pin code

6.2.2.1808 #define UI_BT_STATE_CONNECTED 0x02

RW - BT connected to something

6.2.2.1809 #define UI_BT_STATE_OFF 0x04

RW - BT power off

6.2.2.1810 #define UI_BT_STATE_VISIBLE 0x01

RW - BT visible

6.2.2.1811 #define UI_BUTTON_ENTER 2

W - Insert enter button

6.2.2.1812 #define UI_BUTTON_EXIT 4

W - Insert exit button

6.2.2.1813 #define UI_BUTTON_LEFT 1

W - Insert left arrow button

6.2.2.1814 #define UI_BUTTON_NONE 0

R - Button inserted are executed

6.2.2.1815 #define UI_BUTTON_RIGHT 3

W - Insert right arrow button

6.2.2.1816 #define UI_FLAGS_BUSY 0x40

R - UI busy running or datalogging (popup disabled)

6.2.2.1817 #define UI_FLAGS_DISABLE_EXIT 0x04

RW - Disable exit button

6.2.2.1818 #define UI_FLAGS_DISABLE_LEFT_RIGHT_ENTER 0x02

RW - Disable left, right and enter button

6.2.2.1819 #define UI_FLAGS_ENABLE_STATUS_UPDATE 0x80

W - Enable status line to be updated

6.2.2.1820 #define UI_FLAGS_EXECUTE_LMS_FILE 0x20

W - Execute LMS file in "LMSfilename" (Try It)

6.2.2.1821 #define UI_FLAGS_REDRAW_STATUS 0x08

W - Redraw entire status line

6.2.2.1822 #define UI_FLAGS_RESET_SLEEP_TIMER 0x10

W - Reset sleep timeout timer

6.2.2.1823 #define UI_FLAGS_UPDATE 0x01

W - Make changes take effect

6.2.2.1824 #define UI_STATE_BT_ERROR 16

R - BT error

6.2.2.1825 #define UI_STATE_CONNECT_REQUEST 12

RW - Request for connection accept

6.2.2.1826 #define UI_STATE_DRAW_MENU 6

RW - Execute function and draw menu icons

6.2.2.1827 #define UI_STATE_ENTER_PRESSED 10

RW - Load selected function and next menu id

6.2.2.1828 #define UI_STATE_EXECUTE_FILE 13

RW - Execute file in "LMSfilename"

6.2.2.1829 #define UI_STATE_EXECUTING_FILE 14

R - Executing file in "LMSfilename"

6.2.2.1830 #define UI_STATE_EXIT_PRESSED 11

RW - Load selected function and next menu id

6.2.2.1831 #define UI_STATE_INIT_DISPLAY 0

RW - Init display and load font, menu etc.

6.2.2.1832 #define UI_STATE_INIT_INTRO 2

R - Display intro

6.2.2.1833 #define UI_STATE_INIT_LOW_BATTERY 1

R - Low battery voltage at power on

6.2.2.1834 #define UI_STATE_INIT_MENU 4

RW - Init menu system

6.2.2.1835 #define UI_STATE_INIT_WAIT 3

RW - Wait for initialization end

6.2.2.1836 #define UI_STATE_LEFT_PRESSED 8

RW - Load selected function and next menu id

6.2.2.1837 #define UI_STATE_LOW_BATTERY 15

R - Low battery at runtime

6.2.2.1838 #define UI_STATE_NEXT_MENU 5

RW - Next menu icons ready for drawing

6.2.2.1839 #define UI_STATE_RIGHT_PRESSED 9

RW - Load selected function and next menu id

6.2.2.1840 #define UI_STATE_TEST_BUTTONS 7

RW - Wait for buttons to be pressed

6.2.2.1841 #define UI_VM_IDLE 0

VM_IDLE: Just sitting around. Request to run program will lead to ONE of the VM_RUN* states.

6.2.2.1842 #define UI_VM_RESET1 4

VM_RESET1: Initialize state variables and some I/O devices -- executed when programs end

6.2.2.1843 #define UI_VM_RESET2 5

VM_RESET2: Final clean up and return to IDLE

6.2.2.1844 #define UI_VM_RUN_FREE 1

VM_RUN_FREE: Attempt to run as many instructions as possible within our timeslice

6.2.2.1845 #define UI_VM_RUN_PAUSE 3

VM_RUN_PAUSE: Program still "active", but someone has asked us to pause

6.2.2.1846 #define UI_VM_RUN_SINGLE 2

VM_RUN_SINGLE: Run exactly one instruction per timeslice

6.2.2.1847 #define UIModuleID 0x000C0001

The Ui module ID

6.2.2.1848 #define UIModuleName "Ui.mod"

The Ui module name

6.2.2.1849 #define UINT_MAX 65535

The maximum value of the unsigned int type

6.2.2.1850 #define UIOffsetAbortFlag 40

RW - Long Abort (true == use long press to abort) (1 byte)

6.2.2.1851 #define UIOffsetBatteryState 30

W - Battery state (0..4 capacity) (1 byte)

6.2.2.1852 #define UIOffsetBatteryVoltage 4

R - Battery voltage in millivolts (2 bytes)

6.2.2.1853 #define UIOffsetBluetoothState 31

W - Bluetooth state (0=on, 1=visible, 2=conn, 3=conn.visible, 4=off, 5=dfu) (1 byte)

6.2.2.1854 #define UIOffsetButton 28

RW - Insert button (buttons enumerated above) (1 byte)

6.2.2.1855 #define UIOffsetError 37

W - Error code (1 byte)

6.2.2.1856 #define UIOffsetFlags 26

RW - Update command flags (flags enumerated above) (1 byte)

6.2.2.1857 #define UIOffsetForceOff 39

W - Force off (> 0 = off) (1 byte)

6.2.2.1858 #define UIOffsetLMSfilename 6

W - LMS filename to execute (Try It) (20 bytes)

6.2.2.1859 #define UIOffsetOBPPointer 38

W - Actual OBP step (0 - 4) (1 byte)

6.2.2.1860 #define UIOffsetPMenu 0

W - Pointer to menu file (4 bytes)

6.2.2.1861 #define UIOffsetRechargeable 35

R - Rechargeable battery (0 = no, 1 = yes) (1 byte)

6.2.2.1862 #define UIOffsetRunState 29

W - VM Run state (0 = stopped, 1 = running) (1 byte)

6.2.2.1863 #define UIOffsetSleepTimeout 33

RW - Sleep timeout time (min) (1 byte)

6.2.2.1864 #define UIOffsetSleepTimer 34

RW - Sleep timer (min) (1 byte)

6.2.2.1865 #define UIOffsetState 27

RW - UI state (states enumerated above) (1 byte)

6.2.2.1866 #define UIOffsetUsbState 32

W - Usb state (0=disconnected, 1=connected, 2=working) (1 byte)

6.2.2.1867 #define UIOffsetVolume 36

RW - Volume used in UI (0 - 4) (1 byte)

6.2.2.1868 #define ULONG_MAX 4294967295

The maximum value of the unsigned long type

6.2.2.1869 #define UpdateCalibCacheInfo 43

Update sensor calibration cache information

6.2.2.1870 #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.

6.2.2.1871 #define US_CMD_CONTINUOUS 0x02

Command to put the ultrasonic sensor into continuous polling mode (default)

6.2.2.1872 #define US_CMD_EVENTCAPTURE 0x03

Command to put the ultrasonic sensor into event capture mode

6.2.2.1873 #define US_CMD_OFF 0x00

Command to turn off the ultrasonic sensor

6.2.2.1874 #define US_CMD_SINGLESHTOT 0x01

Command to put the ultrasonic sensor into single shot mode

6.2.2.1875 #define US_CMD_WARMRESET 0x04

Command to warm reset the ultrasonic sensor

6.2.2.1876 #define US_REG_ACTUAL_ZERO 0x50

The register address used to store the actual zero value

6.2.2.1877 #define US_REG_CM_INTERVAL 0x40

The register address used to store the CM interval

6.2.2.1878 #define US_REG_FACTORY_ACTUAL_ZERO 0x11

The register address containing the factory setting for the actual zero value

6.2.2.1879 #define US_REG_FACTORY_SCALE_DIVISOR 0x13

The register address containing the factory setting for the scale divisor value

6.2.2.1880 #define US_REG_FACTORY_SCALE_FACTOR 0x12

The register address containing the factory setting for the scale factor value

6.2.2.1881 #define US_REG_MEASUREMENT_UNITS 0x14

The register address containing the measurement units (degrees C or F)

6.2.2.1882 #define US_REG_SCALE_DIVISOR 0x52

The register address used to store the scale divisor value

6.2.2.1883 #define US_REG_SCALE_FACTOR 0x51

The register address used to store the scale factor value

6.2.2.1884 #define USB_CMD_READY 0x01

A constant representing usb direct command

6.2.2.1885 #define USB_PROTOCOL_OVERHEAD 2

Size of USB Overhead in bytes -- Command type byte + Command

6.2.2.1886 #define USHRT_MAX 65535

The maximum value of the unsigned short type

6.2.2.1887 #define WriteSemData 41

Write motor semaphore data

6.2.2.1888 #define XG1300L_REG_2G 0x61

Select +/- 2G accelerometer range.

6.2.2.1889 #define XG1300L_REG_4G 0x62

Select +/- 4G accelerometer range.

6.2.2.1890 #define XG1300L_REG_8G 0x63

Select +/- 8G accelerometer range.

6.2.2.1891 #define XG1300L_REG_ANGLE 0x42

Read accumulated angle (2 bytes little endian) in 1/100s of degrees.

6.2.2.1892 #define XG1300L_REG_RESET 0x60

Reset the XG1300L device.

6.2.2.1893 #define XG1300L_REG_TURNRATE 0x44

Read rate of turn (2 bytes little endian) in 1/100s of degrees/second.

6.2.2.1894 #define XG1300L_REG_XAXIS 0x46

Read x-axis acceleration (2 bytes little endian) in m/s² scaled by 100/ACC_RANGE*2, where ACC_RANGE is 2, 4, or 8.

6.2.2.1895 #define XG1300L_REG_YAXIS 0x48

Read y-axis acceleration (2 bytes little endian) in m/s² scaled by 100/ACC_RANGE*2, where ACC_RANGE is 2, 4, or 8.

6.2.2.1896 #define XG1300L_REG_ZAXIS 0x4A

Read z-axis acceleration (2 bytes little endian) in m/s² scaled by 100/ACC_RANGE*2, where ACC_RANGE is 2, 4, or 8.

6.2.2.1897 #define XG1300L_SCALE_2G 0x01

Select +/- 2G accelerometer range.

6.2.2.1898 #define XG1300L_SCALE_4G 0x02

Select +/- 4G accelerometer range.

6.2.2.1899 #define XG1300L_SCALE_8G 0x04

Select +/- 8G accelerometer range.

Index

- A simple 3D graphics library, 268
- ACCL_CMD_RESET_CAL
 - MSACCLNx, 724
 - NBCCCommon.h, 838
- ACCL_CMD_X_CAL
 - MSACCLNx, 724
 - NBCCCommon.h, 838
- ACCL_CMD_X_CAL_END
 - MSACCLNx, 725
 - NBCCCommon.h, 839
- ACCL_CMD_Y_CAL
 - MSACCLNx, 725
 - NBCCCommon.h, 839
- ACCL_CMD_Y_CAL_END
 - MSACCLNx, 725
 - NBCCCommon.h, 839
- ACCL_CMD_Z_CAL
 - MSACCLNx, 725
 - NBCCCommon.h, 839
- ACCL_CMD_Z_CAL_END
 - MSACCLNx, 725
 - NBCCCommon.h, 839
- ACCL_REG_SENS_LVL
 - MSACCLNx, 725
 - NBCCCommon.h, 839
- ACCL_REG_X_ACCEL
 - MSACCLNx, 725
 - NBCCCommon.h, 839
- ACCL_REG_X_OFFSET
 - MSACCLNx, 725
 - NBCCCommon.h, 839
- ACCL_REG_X_RANGE
 - MSACCLNx, 725
 - NBCCCommon.h, 839
- ACCL_REG_X_TILT
 - MSACCLNx, 725
 - NBCCCommon.h, 839
- ACCL_REG_Y_ACCEL
 - MSACCLNx, 726
 - NBCCCommon.h, 840
- ACCL_REG_Y_OFFSET
 - MSACCLNx, 726
- NBCCCommon.h, 840
- ACCL_REG_Y_RANGE
 - MSACCLNx, 726
 - NBCCCommon.h, 840
- ACCL_REG_Y_TILT
 - MSACCLNx, 726
 - NBCCCommon.h, 840
- ACCL_REG_Z_ACCEL
 - MSACCLNx, 726
 - NBCCCommon.h, 840
- ACCL_REG_Z_OFFSET
 - MSACCLNx, 726
 - NBCCCommon.h, 840
- ACCL_REG_Z_RANGE
 - MSACCLNx, 726
 - NBCCCommon.h, 840
- ACCL_REG_Z_TILT
 - MSACCLNx, 726
 - NBCCCommon.h, 840
- ACCL_SENSITIVITY_LEVEL_1
 - MSACCLNxSLevel, 727
 - NBCCCommon.h, 840
- ACCL_SENSITIVITY_LEVEL_2
 - MSACCLNxSLevel, 727
 - NBCCCommon.h, 840
- ACCL_SENSITIVITY_LEVEL_3
 - MSACCLNxSLevel, 727
 - NBCCCommon.h, 841
- ACCL_SENSITIVITY_LEVEL_4
 - MSACCLNxSLevel, 727
 - NBCCCommon.h, 841
- ACCLNxCalibrateX
 - MindSensorsAPI, 167
- ACCLNxCalibrateXEnd
 - MindSensorsAPI, 167
- ACCLNxCalibrateY
 - MindSensorsAPI, 167
- ACCLNxCalibrateYEnd
 - MindSensorsAPI, 168
- ACCLNxCalibrateZ
 - MindSensorsAPI, 168
- ACCLNxCalibrateZEnd

MindSensorsAPI, 168
ACCLNxResetCalibration
 MindSensorsAPI, 169
ActualSpeedField
 NBCCCommon.h, 841
 OutputFieldConstants, 570
Array operation constants, 476
ArrayOpConstants
 OPARR_MAX, 476
 OPARR_MEAN, 476
 OPARR_MIN, 477
 OPARR_SORT, 477
 OPARR_STD, 477
 OPARR_SUM, 477
 OPARR_SUMSQR, 477

bcd2dec
 cmathAPI, 475
BITMAP_1
 DisplayModuleConstants, 597
 NBCCCommon.h, 841
BITMAP_2
 DisplayModuleConstants, 597
 NBCCCommon.h, 841
BITMAP_3
 DisplayModuleConstants, 597
 NBCCCommon.h, 841
BITMAP_4
 DisplayModuleConstants, 597
 NBCCCommon.h, 841
BITMAPS
 DisplayModuleConstants, 597
 NBCCCommon.h, 841
BlockTachoCountField
 NBCCCommon.h, 841
 OutputFieldConstants, 570
Bluetooth hardware status constants, 621
Bluetooth State constants, 616
Bluetooth state status constants, 618
BluetoothState constants, 539
BluetoothStatus
 CommModuleFunctions, 400
BluetoothWrite
 CommModuleFunctions, 400
BREAKOUT_REQ
 CommandVMState, 496

NBCCCommon.h, 842
BT_ARM_CMD_MODE
 CommBtStateConstants, 616
 NBCCCommon.h, 842
BT_ARM_DATA_MODE
 CommBtStateConstants, 616
 NBCCCommon.h, 842
BT_ARM_OFF
 CommBtStateConstants, 616
 NBCCCommon.h, 842
BT_BRICK_PORT_OPEN
 CommBtStateStatusConstants, 618
 NBCCCommon.h, 842
BT_BRICK_VISIBILITY
 CommBtStateStatusConstants, 618
 NBCCCommon.h, 842
BT_CMD_BYTE
 CommMiscConstants, 614
 NBCCCommon.h, 842
BT_CMD_READY
 CommStatusCodesConstants, 637
 NBCCCommon.h, 842
BT_CONNECTION_0_ENABLE
 CommBtStateStatusConstants, 618
 NBCCCommon.h, 843
BT_CONNECTION_1_ENABLE
 CommBtStateStatusConstants, 618
 NBCCCommon.h, 843
BT_CONNECTION_2_ENABLE
 CommBtStateStatusConstants, 618
 NBCCCommon.h, 843
BT_CONNECTION_3_ENABLE
 CommBtStateStatusConstants, 618
 NBCCCommon.h, 843
BT_DEFAULT_INQUIRY_MAX
 CommMiscConstants, 614
 NBCCCommon.h, 843
BT_DEFAULT_INQUIRY_TIMEOUT_-
 LO
 CommMiscConstants, 614
 NBCCCommon.h, 843
BT_DEVICE_AWAY
 CommDeviceStatusConstants, 633
 NBCCCommon.h, 843
BT_DEVICE_EMPTY
 CommDeviceStatusConstants, 633

NBCCCommon.h, 843
BT_DEVICE_KNOWN
 CommDeviceStatusConstants, 634
 NBCCCommon.h, 843
BT_DEVICE_NAME
 CommDeviceStatusConstants, 634
 NBCCCommon.h, 843
BT_DEVICE_UNKNOWN
 CommDeviceStatusConstants, 634
 NBCCCommon.h, 844
BT_DISABLE
 CommBtHwStatusConstants, 621
 NBCCCommon.h, 844
BT_ENABLE
 CommBtHwStatusConstants, 621
 NBCCCommon.h, 844
BTN1
 ButtonNameConstants, 530
 NBCCCommon.h, 844
BTN2
 ButtonNameConstants, 530
 NBCCCommon.h, 844
BTN3
 ButtonNameConstants, 530
 NBCCCommon.h, 844
BTN4
 ButtonNameConstants, 530
 NBCCCommon.h, 844
BTNCENTER
 ButtonNameConstants, 530
 NBCCCommon.h, 844
BTNEXTIT
 ButtonNameConstants, 530
 NBCCCommon.h, 844
BTNLEFT
 ButtonNameConstants, 530
 NBCCCommon.h, 844
BTNRIGHT
 ButtonNameConstants, 530
 NBCCCommon.h, 845
BTNSTATE_LONG_PRESSED_EV
 ButtonStateConstants, 531
 NBCCCommon.h, 845
BTNSTATE_LONG_RELEASED_EV
 ButtonStateConstants, 531
 NBCCCommon.h, 845
BTNSTATE_NONE
 ButtonStateConstants, 531
 NBCCCommon.h, 845
BTNSTATE_PRESSED_EV
 ButtonStateConstants, 531
 NBCCCommon.h, 845
BTNSTATE_PRESSED_STATE
 ButtonStateConstants, 532
 NBCCCommon.h, 845
BTNSTATE_SHORT_RELEASED_EV
 ButtonStateConstants, 532
 NBCCCommon.h, 845
Button module, 84
Button module constants, 529
Button module functions, 377
Button module IOMAP offsets, 532
Button name constants, 529
ButtonIOMAP
 ButtonOffsetLongPressCnt, 532
 ButtonOffsetLongRelCnt, 532
 ButtonOffsetPressedCnt, 532
 ButtonOffsetRelCnt, 533
 ButtonOffsetShortRelCnt, 533
 ButtonOffsetState, 533
ButtonModuleFunctions
 GetButtonLongPressCount, 379
 GetButtonLongReleaseCount, 379
 GetButtonPressCount, 379
 GetButtonReleaseCount, 379
 GetButtonShortReleaseCount, 380
 GetButtonState, 380
 ReadButtonEx, 380
 SetButtonLongPressCount, 381
 SetButtonLongReleaseCount, 381
 SetButtonPressCount, 381
 SetButtonReleaseCount, 381
 SetButtonShortReleaseCount, 382
 SetButtonState, 382
ButtonModuleID
 ModuleIDConstants, 264
 NBCCCommon.h, 845
ButtonModuleName
 ModuleNameConstants, 262
 NBCCCommon.h, 845
ButtonNameConstants
 BTN1, 530

BTN2, 530
BTN3, 530
BTN4, 530
BTNCENTER, 530
BTNEXIT, 530
BTNLEFT, 530
BTNRIGHT, 530
NO_OF_BTNS, 530
ButtonOffsetLongPressCnt
 ButtonIOMAP, 532
 NBCCommon.h, 845
ButtonOffsetLongRelCnt
 ButtonIOMAP, 532
 NBCCommon.h, 846
ButtonOffsetPressedCnt
 ButtonIOMAP, 532
 NBCCommon.h, 846
ButtonOffsetRelCnt
 ButtonIOMAP, 533
 NBCCommon.h, 846
ButtonOffsetShortRelCnt
 ButtonIOMAP, 533
 NBCCommon.h, 846
ButtonOffsetState
 ButtonIOMAP, 533
 NBCCommon.h, 846
ButtonState constants, 531
ButtonStateConstants
 BTNSTATE_LONG_PRESSED_-
 EV, 531
 BTNSTATE_LONG_RELEASED_-
 EV, 531
 BTNSTATE_NONE, 531
 BTNSTATE_PRESSED_EV, 531
 BTNSTATE_PRESSED_STATE,
 532
 BTNSTATE_SHORT_-
 RELEASED_EV, 532
CHAR_BIT
 NBCCommon.h, 846
 NXTLimits, 785
CHAR_MAX
 NBCCommon.h, 846
 NXTLimits, 785
CHAR_MIN
 NBCCommon.h, 846
 NXTLimits, 785
 NXTLimits, 785
CircleOut
 DisplayModuleFunctions, 333
CircleOutEx
 DisplayModuleFunctions, 333
ClearLine
 DisplayModuleFunctions, 334
ClearScreen
 DisplayModuleFunctions, 334
ClearSensor
 InputModuleFunctions, 300
CloseFile
 LoaderModuleFunctions, 464
CLUMP_DONE
 CommandVMState, 496
 NBCCommon.h, 846
CLUMP_SUSPEND
 CommandVMState, 496
 NBCCommon.h, 846
cmath API, 474
cmathAPI
 bcd2dec, 475
cmpconst
 EQ, 73
 GT, 73
 GTEQ, 74
 LT, 74
 LTEQ, 74
 NEQ, 74
Coast
 OutputModuleFunctions, 279
CoastEx
 OutputModuleFunctions, 280
Codatex API Functions, 236
Codatex device constants, 751
Codatex RFID sensor constants, 751
Codatex RFID sensor modes, 752
CodatexAPI
 RFIDInit, 237
 RFIDMode, 237
 RFIDRead, 238
 RFIDReadContinuous, 238
 RFIDReadSingle, 238
 RFIDStatus, 239
 RFIDStop, 239

Color calibration constants, 554
Color calibration state constants, 553
Color sensor array indices, 551
Color values, 552
ColorSensorRead
 NBCCCommon.h, 847
 SysCallConstants, 479
COM_CHANNEL_FOUR_ACTIVE
 LowSpeedStateConstants, 580
 NBCCCommon.h, 847
COM_CHANNEL_NONE_ACTIVE
 LowSpeedStateConstants, 580
 NBCCCommon.h, 847
COM_CHANNEL_ONE_ACTIVE
 LowSpeedStateConstants, 580
 NBCCCommon.h, 847
COM_CHANNEL_THREE_ACTIVE
 LowSpeedStateConstants, 580
 NBCCCommon.h, 847
COM_CHANNEL_TWO_ACTIVE
 LowSpeedStateConstants, 580
 NBCCCommon.h, 847
Comm module, 83
Comm module constants, 612
Comm module functions, 391
Comm module interface function constants, 634
Comm module IOMAP offsets, 638
Comm module status code constants, 637
Command module, 80
Command module constants, 81
Command module functions, 357
Command module IOMAP offsets, 503
CommandCommErrors
 ERR_COMM_BUFFER_FULL, 501
 ERR_COMM_BUS_ERR, 501
 ERR_COMM_CHAN_INVALID, 501
 ERR_COMM_CHAN_NOT_READY, 501
CommandFatalErrors
 ERR_ARG, 497
 ERR_BAD_POOL_SIZE, 497
 ERR_BAD_PTR, 498
 ERR_CLUMP_COUNT, 498
ERR_DEFAULT_OFFSETS, 498
ERR_FILE, 498
ERR_INSANE_OFFSET, 498
ERR_INSTR, 498
ERR_LOADER_ERR, 498
ERR_MEM, 498
ERR_MEMMGR_FAIL, 498
ERR_NO_ACTIVE_CLUMP, 499
ERR_NO_CODE, 499
ERR_NON_FATAL, 499
ERR_SPOTCHECK_FAIL, 499
ERR_VER, 499
CommandFlags constants, 534
CommandGenErrors
 ERR_INVALID_FIELD, 500
 ERR_INVALID_PORT, 500
 ERR_INVALID_QUEUE, 500
 ERR_INVALID_SIZE, 500
 ERR_NO_PROG, 500
CommandIOMAP
 CommandOffsetActivateFlag, 504
 CommandOffsetAwake, 504
 CommandOffsetDeactivateFlag, 504
 CommandOffsetFileName, 504
 CommandOffsetFormatString, 504
 CommandOffsetMemoryPool, 504
 CommandOffsetOffsetDS, 504
 CommandOffsetOffsetDVA, 504
 CommandOffsetPRCHandler, 504
 CommandOffsetProgStatus, 504
 CommandOffsetSyncTick, 505
 CommandOffsetSyncTime, 505
 CommandOffsetTick, 505
CommandModuleConstants
 NO_ERR, 82
 POOL_MAX_SIZE, 82
 STAT_COMM_PENDING, 82
 STAT_MSG_EMPTY_MAILBOX, 82
CommandModuleFunctions
 GetButtonModuleValue, 361
 GetCommandModuleBytes, 361
 GetCommandModuleValue, 362
 GetCommModuleBytes, 362
 GetCommModuleValue, 362
 GetDisplayModuleBytes, 363

GetDisplayModuleValue, 363
GetFirstTick, 363
GetInputModuleValue, 364
GetIOMapBytes, 364
GetIOMapBytesByID, 364
GetIOMapValue, 365
GetIOMapValueByID, 365
GetLastResponseInfo, 366
GetLoaderModuleValue, 366
GetLowSpeedModuleBytes, 367
GetLowSpeedModuleValue, 367
GetMemoryInfo, 367
GetOutputModuleValue, 368
GetSoundModuleValue, 368
GetUIModuleValue, 369
ResetSleepTimer, 369
SetButtonModuleValue, 369
SetCommandModuleBytes, 370
SetCommandModuleValue, 370
SetCommModuleBytes, 370
SetCommModuleValue, 371
SetDisplayModuleBytes, 371
SetDisplayModuleValue, 372
SetInputModuleValue, 372
SetIOCctrlModuleValue, 372
SetIOMapBytes, 373
SetIOMapBytesByID, 373
SetIOMapValue, 374
SetIOMapValueByID, 374
SetLoaderModuleValue, 375
SetLowSpeedModuleBytes, 375
SetLowSpeedModuleValue, 375
SetOutputModuleValue, 376
SetSoundModuleValue, 376
SetUIModuleValue, 377
Wait, 377

CommandModuleID
 ModuleIDConstants, 264
 NBCCCommon.h, 847

CommandModuleName
 ModuleNameConstants, 262
 NBCCCommon.h, 847

CommandOffsetActivateFlag
 CommandIOMAP, 504
 NBCCCommon.h, 847

CommandOffsetAwake

CommandIOMAP, 504
NBCCCommon.h, 847

CommandOffsetDeactivateFlag
 CommandIOMAP, 504
 NBCCCommon.h, 848

CommandOffsetFileName
 CommandIOMAP, 504
 NBCCCommon.h, 848

CommandOffsetFormatString
 CommandIOMAP, 504
 NBCCCommon.h, 848

CommandOffsetMemoryPool
 CommandIOMAP, 504
 NBCCCommon.h, 848

CommandOffsetOffsetDS
 CommandIOMAP, 504
 NBCCCommon.h, 848

CommandOffsetOffsetDVA
 CommandIOMAP, 504
 NBCCCommon.h, 848

CommandOffsetPRCHandler
 CommandIOMAP, 504
 NBCCCommon.h, 848

CommandOffsetProgStatus
 CommandIOMAP, 504
 NBCCCommon.h, 848

CommandOffsetSyncTick
 CommandIOMAP, 505
 NBCCCommon.h, 848

CommandOffsetSyncTime
 CommandIOMAP, 505
 NBCCCommon.h, 848

CommandOffsetTick
 CommandIOMAP, 505
 NBCCCommon.h, 849

CommandProgStatus
 PROG_ABORT, 502
 PROG_ERROR, 502
 PROG_IDLE, 502
 PROG_OK, 503
 PROG_RESET, 503
 PROG_RUNNING, 503

CommandRCErrors
 ERR_RC_BAD_PACKET, 501
 ERR_RC_FAILED, 501
 ERR_RC_ILLEGAL_VAL, 502

ERR_RC_UNKNOWN_CMD, [502](#)
CommandVMState
 BREAKOUT_REQ, [496](#)
 CLUMP_DONE, [496](#)
 CLUMP_SUSPEND, [496](#)
 ROTATE_QUEUE, [496](#)
 STOP_REQ, [496](#)
 TIMES_UP, [497](#)
CommBTCheckStatus
 NBCCommon.h, [849](#)
 SysCallConstants, [479](#)
CommBTConnection
 NBCCommon.h, [849](#)
 SysCallConstants, [479](#)
CommBtHwStatusConstants
 BT_DISABLE, [621](#)
 BT_ENABLE, [621](#)
CommBTOnOff
 NBCCommon.h, [849](#)
 SysCallConstants, [479](#)
CommBTRead
 NBCCommon.h, [849](#)
 SysCallConstants, [479](#)
CommBtStateConstants
 BT_ARM_CMD_MODE, [616](#)
 BT_ARM_DATA_MODE, [616](#)
 BT_ARM_OFF, [616](#)
CommBtStateStatusConstants
 BT_BRICK_PORT_OPEN, [618](#)
 BT_BRICK_VISIBILITY, [618](#)
 BT_CONNECTION_0_ENABLE, [618](#)
 BT_CONNECTION_1_ENABLE, [618](#)
 BT_CONNECTION_2_ENABLE, [618](#)
 BT_CONNECTION_3_ENABLE, [618](#)
CommBTWrite
 NBCCommon.h, [849](#)
 SysCallConstants, [480](#)
CommConnectionConstants
 CONN_BT0, [619](#)
 CONN_BT1, [619](#)
 CONN_BT2, [619](#)
 CONN_BT3, [620](#)
 CONN_HS4, [620](#)
 CONN_HS_1, [620](#)
 CONN_HS_2, [620](#)
 CONN_HS_3, [620](#)
 CONN_HS_4, [620](#)
 CONN_HS_5, [620](#)
 CONN_HS_6, [620](#)
 CONN_HS_7, [620](#)
 CONN_HS_8, [620](#)
 CONN_HS_ALL, [621](#)
CommDataModeConstants
 DATA_MODE_GPS, [617](#)
 DATA_MODE_MASK, [617](#)
 DATA_MODE_NXT, [617](#)
 DATA_MODE_RAW, [617](#)
 DATA_MODE_UPDATE, [617](#)
CommDeviceStatusConstants
 BT_DEVICE_AWAY, [633](#)
 BT_DEVICE_EMPTY, [633](#)
 BT_DEVICE_KNOWN, [634](#)
 BT_DEVICE_NAME, [634](#)
 BT_DEVICE_UNKNOWN, [634](#)
CommExecuteFunction
 NBCCommon.h, [849](#)
 SysCallConstants, [480](#)
CommHiSpeedAddressConstants
 HS_ADDRESS_1, [632](#)
 HS_ADDRESS_2, [632](#)
 HS_ADDRESS_3, [632](#)
 HS_ADDRESS_4, [632](#)
 HS_ADDRESS_5, [632](#)
 HS_ADDRESS_6, [632](#)
 HS_ADDRESS_7, [633](#)
 HS_ADDRESS_8, [633](#)
 HS_ADDRESS_ALL, [633](#)
CommHiSpeedBaudConstants
 HS_BAUD_115200, [625](#)
 HS_BAUD_1200, [625](#)
 HS_BAUD_14400, [625](#)
 HS_BAUD_19200, [625](#)
 HS_BAUD_230400, [626](#)
 HS_BAUD_2400, [626](#)
 HS_BAUD_28800, [626](#)
 HS_BAUD_3600, [626](#)
 HS_BAUD_38400, [626](#)
 HS_BAUD_460800, [626](#)

HS_BAUD_4800, [626](#)
HS_BAUD_57600, [626](#)
HS_BAUD_7200, [626](#)
HS_BAUD_76800, [626](#)
HS_BAUD_921600, [627](#)
HS_BAUD_9600, [627](#)
HS_BAUD_DEFAULT, [627](#)
CommHiSpeedCombinedConstants
 HS_MODE_7E1, [631](#)
 HS_MODE_8N1, [631](#)
CommHiSpeedCtrlConstants
 HS_CTRL_EXIT, [624](#)
 HS_CTRL_INIT, [624](#)
 HS_CTRL_UART, [624](#)
CommHiSpeedDataBitsConstants
 HS_MODE_5_DATA, [629](#)
 HS_MODE_6_DATA, [629](#)
 HS_MODE_7_DATA, [629](#)
 HS_MODE_8_DATA, [629](#)
CommHiSpeedFlagsConstants
 HS_UPDATE, [622](#)
CommHiSpeedModeConstants
 HS_MODE_DEFAULT, [628](#)
 HS_MODE_MASK, [628](#)
 HS_MODE_UART_RS232, [628](#)
 HS_MODE_UART_RS485, [628](#)
 HS_UART_MASK, [628](#)
CommHiSpeedParityConstants
 HS_MODE_E_PARITY, [630](#)
 HS_MODE_M_PARITY, [630](#)
 HS_MODE_N_PARITY, [630](#)
 HS_MODE_O_PARITY, [631](#)
 HS_MODE_S_PARITY, [631](#)
CommHiSpeedStateConstants
 HS_BYTES_REMAINING, [623](#)
 HS_DEFAULT, [623](#)
 HS_DISABLE, [623](#)
 HS_ENABLE, [623](#)
 HS_INIT_RECEIVER, [623](#)
 HS_INITIALISE, [623](#)
 HS_SEND_DATA, [624](#)
CommHiSpeedStopBitsConstants
 HS_MODE_10_STOP, [630](#)
 HS_MODE_15_STOP, [630](#)
 HS_MODE_20_STOP, [630](#)
CommHSCheckStatus
NBCCCommon.h, [849](#)
SysCallConstants, [480](#)
CommHSControl
 NBCCCommon.h, [849](#)
 SysCallConstants, [480](#)
CommHSRead
 NBCCCommon.h, [849](#)
 SysCallConstants, [480](#)
CommHSWrite
 NBCCCommon.h, [850](#)
 SysCallConstants, [480](#)
CommInterfaceConstants
 INTF_BTOFF, [635](#)
 INTF_BTON, [635](#)
 INTF_CONNECT, [635](#)
 INTF_CONNECTBYNAME, [635](#)
 INTF_CONNECTREQ, [635](#)
 INTF_DISCONNECT, [635](#)
 INTF_DISCONNECTALL, [635](#)
 INTF_EXTREAD, [635](#)
 INTF_FACTORYRESET, [636](#)
 INTF_OPENSTREAM, [636](#)
 INTF_PINREQ, [636](#)
 INTF_REMOVEDEVICE, [636](#)
 INTF_SEARCH, [636](#)
 INTF_SENDDATA, [636](#)
 INTF_SENDFILE, [636](#)
 INTF_SETBTNAME, [636](#)
 INTF_SETCMDMODE, [636](#)
 INTF_STOPSEARCH, [636](#)
 INTF_VISIBILITY, [637](#)
ComMIOMAP
 CommOffsetBrickDataBdAddr, [640](#)
 CommOffsetBrickDataBluecoreVersion, [640](#)
 CommOffsetBrickDataBtHwStatus, [640](#)
 CommOffsetBrickDataBtStateStatus, [640](#)
 CommOffsetBrickDataName, [640](#)
 CommOffsetBrickDataTimeOutValue, [640](#)
 CommOffsetBtConnectTableBdAddr, [640](#)
 CommOffsetBtConnectTableClassOfDevice, [640](#)

CommOffsetBtConnectTableHandleNr, 640
CommOffsetBtConnectTableLinkQuality, 640
CommOffsetBtConnectTableName, 641
CommOffsetBtConnectTablePinCode, 641
CommOffsetBtConnectTableStreamStatus, 641
CommOffsetBtDataMode, 641
CommOffsetBtDeviceCnt, 641
CommOffsetBtDeviceNameCnt, 641
CommOffsetBtDeviceTableBdAddr, 641
CommOffsetBtDeviceTableClassOfDevice, 641
CommOffsetBtDeviceTableDeviceStatus, 641
CommOffsetBtDeviceTableName, 641
CommOffsetBtInBufBuf, 642
CommOffsetBtInBufInPtr, 642
CommOffsetBtInBufOutPtr, 642
CommOffsetBtOutBufBuf, 642
CommOffsetBtOutBufInPtr, 642
CommOffsetBtOutBufOutPtr, 642
CommOffsetHsAddress, 642
CommOffsetHsDataMode, 642
CommOffsetHsFlags, 642
CommOffsetHsInBufBuf, 642
CommOffsetHsInBufInPtr, 643
CommOffsetHsInBufOutPtr, 643
CommOffsetHsMode, 643
CommOffsetHsOutBufBuf, 643
CommOffsetHsOutBufInPtr, 643
CommOffsetHsOutBufOutPtr, 643
CommOffsetHsSpeed, 643
CommOffsetHsState, 643
CommOffsetPFunc, 643
CommOffsetPFuncTwo, 643
CommOffsetUsbInBufBuf, 644
CommOffsetUsbInBufInPtr, 644
CommOffsetUsbInBufOutPtr, 644
CommOffsetUsbOutBufBuf, 644
CommOffsetUsbOutBufInPtr, 644
CommOffsetUsbOutBufOutPtr, 644
CommOffsetUsbPollBufBuf, 644
CommOffsetUsbPollBufInPtr, 644
CommOffsetUsbPollBufOutPtr, 644
CommOffsetUsbState, 644
CommLSCheckStatus
 NBCCCommon.h, 850
 SysCallConstants, 480
CommLSRead
 NBCCCommon.h, 850
 SysCallConstants, 480
CommLSWrite
 NBCCCommon.h, 850
 SysCallConstants, 480
CommLSWriteEx
 NBCCCommon.h, 850
 SysCallConstants, 480
CommMiscConstants
 BT_CMD_BYTE, 614
 BT_DEFAULT_INQUIRY_MAX, 614
 BT_DEFAULT_INQUIRY_TIMEOUT_LO, 614
 MAX_BT_MSG_SIZE, 614
 SIZE_OF_BDADDR, 615
 SIZE_OF_BRICK_NAME, 615
 SIZE_OF_BT_CONNECT_TABLE, 615
 SIZE_OF_BT_DEVICE_TABLE, 615
 SIZE_OF_BT_NAME, 615
 SIZE_OF_BT_PINCODE, 615
 SIZE_OF_BTBUF, 615
 SIZE_OF_CLASS_OF_DEVICE, 615
 SIZE_OF_HSBUF, 615
 SIZE_OF_USBBUF, 615
 SIZE_OF_USBDATA, 616
 USB_PROTOCOL_OVERHEAD, 616
CommModuleDCFunctions
 RemoteDatalogRead, 436
 RemoteDatalogSetTimes, 436
 RemoteGetBatteryLevel, 436
 RemoteGetConnectionCount, 437
 RemoteGetConnectionName, 437

RemoteGetContactCount, 438
RemoteGetContactName, 438
RemoteGetCurrentProgramName,
 438
RemoteGetInputValues, 439
RemoteGetOutputState, 439
RemoteGetProperty, 439
RemoteKeepAlive, 440
RemoteLowspeedGetStatus, 440
RemoteLowspeedRead, 441
RemoteLowspeedWrite, 441
RemoteMessageRead, 441
RemoteMessageWrite, 442
RemotePlaySoundFile, 442
RemotePlayTone, 443
RemoteResetMotorPosition, 443
RemoteResetScaledValue, 443
RemoteResetTachoCount, 444
RemoteSetInputModule, 444
RemoteSetOutputState, 445
Remote SetProperty, 445
RemoteStartProgram, 446
RemoteStopProgram, 446
RemoteStopSound, 447
CommModuleFunctions
 BluetoothStatus, 400
 BluetoothWrite, 400
 GetBrickDataAddress, 401
 GetBrickDataBluecoreVersion, 401
 GetBrickDataBtHardwareStatus,
 401
 GetBrickDataBtStateStatus, 401
 GetBrickDataName, 402
 GetBrickDataTimeoutValue, 402
 GetBTConnectionAddress, 402
 GetBTConnectionClass, 403
 GetBTConnectionHandleNum, 403
 GetBTConnectionLinkQuality, 403
 GetBTConnectionName, 403
 GetBTConnectionPinCode, 404
 GetBTConnectionStreamStatus, 404
 GetBTDataMode, 404
 GetBTDeviceAddress, 405
 GetBTDeviceClass, 405
 GetBTDeviceCount, 405
 GetBTDeviceName, 405
 GetBTDeviceNameCount, 406
 GetBTDeviceStatus, 406
 GetBTInputBuffer, 406
 GetBTInputBufferInPtr, 407
 GetBTInputBufferOutPtr, 407
 GetBTOutputBuffer, 407
 GetBTOutputBufferInPtr, 408
 GetBTOutputBufferOutPtr, 408
 GetHSAddress, 408
 GetHSDataMode, 408
 GetHSFlags, 409
 GetHSInputBuffer, 409
 GetHSInputBufferInPtr, 409
 GetHSInputBufferOutPtr, 410
 GetHSMode, 410
 GetHSOutputBuffer, 410
 GetHSOutputBufferInPtr, 411
 GetHSOutputBufferOutPtr, 411
 GetHSSpeed, 411
 GetHSState, 412
 GetUSBInputBuffer, 412
 GetUSBInputBufferInPtr, 412
 GetUSBInputBufferOutPtr, 413
 GetUSBOutputBuffer, 413
 GetUSBOutputBufferInPtr, 413
 GetUSBOutputBufferOutPtr, 414
 GetUSBPollBuffer, 414
 GetUSBPollBufferInPtr, 414
 GetUSBPollBufferOutPtr, 415
 GetUSBState, 415
 ReceiveMessage, 415
 ReceiveRemoteBool, 416
 ReceiveRemoteMessageEx, 416
 ReceiveRemoteNumber, 417
 ReceiveRemoteString, 417
 RemoteConnectionIdle, 418
 RemoteConnectionWrite, 418
 RS485Control, 418
 RS485Disable, 419
 RS485Enable, 419
 RS485Initialize, 420
 RS485Read, 420
 RS485ReadEx, 420
 RS485Status, 421
 RS485Uart, 421
 RS485Write, 422

SendMessage, 422
SendRemoteBool, 422
SendRemoteNumber, 423
SendRemoteString, 423
SendResponseBool, 423
SendResponseNumber, 424
SendResponseString, 424
SendRS485Bool, 424
SendRS485Number, 425
SendRS485String, 425
SetBTDataMode, 425
SetBTInputBuffer, 426
SetBTInputBufferInPtr, 426
SetBTInputBufferOutPtr, 426
SetBTOutputBuffer, 426
SetBTOutputBufferInPtr, 427
SetBTOutputBufferOutPtr, 427
SetHSAddress, 427
SetHSDataMode, 427
SetHSFlags, 428
SetHSInputBuffer, 428
SetHSInputBufferInPtr, 428
SetHSInputBufferOutPtr, 428
SetHSMode, 429
SetHSOutputBuffer, 429
SetHSOutputBufferInPtr, 429
SetHSOutputBufferOutPtr, 430
SetHSSpeed, 430
SetHSState, 430
SetUSBInputBuffer, 430
SetUSBInputBufferInPtr, 431
SetUSBInputBufferOutPtr, 431
SetUSBOutputBuffer, 431
SetUSBOutputBufferInPtr, 431
SetUSBOutputBufferOutPtr, 432
SetUSBPollBuffer, 432
SetUSBPollBufferInPtr, 432
SetUSBPollBufferOutPtr, 432
SetUSBState, 432
UseRS485, 433
CommModuleID
 ModuleIDConstants, 264
 NBCCCommon.h, 850
CommModuleName
 ModuleNameConstants, 262
 NBCCCommon.h, 850

CommModuleSCFunctions
 RemoteBluetoothFactoryReset, 450
 RemoteCloseFile, 450
 RemoteDeleteFile, 450
 RemoteDeleteUserFlash, 451
 RemoteFindFirstFile, 451
 RemoteFindNextFile, 452
 RemoteGetBluetoothAddress, 452
 RemoteGetDeviceInfo, 453
 RemoteGetFirmwareVersion, 453
 RemoteIOMapRead, 454
 RemoteIOMapWriteBytes, 454
 RemoteIOMapWriteValue, 455
 RemoteOpenAppendData, 455
 RemoteOpenRead, 456
 RemoteOpenWrite, 456
 RemoteOpenWriteData, 457
 RemoteOpenWriteLinear, 457
 RemotePollCommand, 458
 RemotePollCommandLength, 458
 RemoteRead, 459
 RemoteRenameFile, 459
 RemoteSetBrickName, 460
 RemoteWrite, 460

CommOffsetBrickDataBdAddr
 CommIOMAP, 640
 NBCCCommon.h, 850

CommOffsetBrickDataBluecoreVersion
 CommIOMAP, 640
 NBCCCommon.h, 850

CommOffsetBrickDataBtHwStatus
 CommIOMAP, 640
 NBCCCommon.h, 850

CommOffsetBrickDataBtStateStatus
 CommIOMAP, 640
 NBCCCommon.h, 851

CommOffsetBrickDataName
 CommIOMAP, 640
 NBCCCommon.h, 851

CommOffsetBrickDataTimeOutValue
 CommIOMAP, 640
 NBCCCommon.h, 851

CommOffsetBtConnectTableBdAddr
 CommIOMAP, 640
 NBCCCommon.h, 851

CommOffsetBtConnectTableClassOfDevice

CommIOMAP, 640
NBCCCommon.h, 851

CommOffsetBtConnectTableHandleNr
 CommIOMAP, 640
 NBCCCommon.h, 851

CommOffsetBtConnectTableLinkQuality
 CommIOMAP, 640
 NBCCCommon.h, 851

CommOffsetBtConnectTableName
 CommIOMAP, 641
 NBCCCommon.h, 851

CommOffsetBtConnectTablePinCode
 CommIOMAP, 641
 NBCCCommon.h, 851

CommOffsetBtConnectTableStreamStatus
 CommIOMAP, 641
 NBCCCommon.h, 851

CommOffsetBtDataMode
 CommIOMAP, 641
 NBCCCommon.h, 852

CommOffsetBtDeviceCnt
 CommIOMAP, 641
 NBCCCommon.h, 852

CommOffsetBtDeviceNameCnt
 CommIOMAP, 641
 NBCCCommon.h, 852

CommOffsetBtDeviceTableBdAddr
 CommIOMAP, 641
 NBCCCommon.h, 852

CommOffsetBtDeviceTableClassOfDevice
 CommIOMAP, 641
 NBCCCommon.h, 852

CommOffsetBtDeviceTableDeviceStatus
 CommIOMAP, 641
 NBCCCommon.h, 852

CommOffsetBtDeviceTableName
 CommIOMAP, 641
 NBCCCommon.h, 852

CommOffsetBtInBufBuf
 CommIOMAP, 642
 NBCCCommon.h, 852

CommOffsetBtInBufInPtr
 CommIOMAP, 642
 NBCCCommon.h, 852

CommOffsetBtInBufOutPtr
 CommIOMAP, 642

NBCCCommon.h, 852

CommOffsetBtOutBufBuf

 CommIOMAP, 642
 NBCCCommon.h, 853

CommOffsetBtOutBufInPtr

 CommIOMAP, 642
 NBCCCommon.h, 853

CommOffsetBtOutBufOutPtr

 CommIOMAP, 642
 NBCCCommon.h, 853

CommOffsetHsAddress
 CommIOMAP, 642
 NBCCCommon.h, 853

CommOffsetHsDataMode
 CommIOMAP, 642
 NBCCCommon.h, 853

CommOffsetHsFlags
 CommIOMAP, 642
 NBCCCommon.h, 853

CommOffsetHsInBufBuf

 CommIOMAP, 642
 NBCCCommon.h, 853

CommOffsetHsInBufInPtr

 CommIOMAP, 643
 NBCCCommon.h, 853

CommOffsetHsInBufOutPtr

 CommIOMAP, 643
 NBCCCommon.h, 853

CommOffsetHsMode
 CommIOMAP, 643
 NBCCCommon.h, 853

CommOffsetHsOutBufBuf

 CommIOMAP, 643
 NBCCCommon.h, 854

CommOffsetHsOutBufInPtr

 CommIOMAP, 643
 NBCCCommon.h, 854

CommOffsetHsOutBufOutPtr

 CommIOMAP, 643
 NBCCCommon.h, 854

CommOffsetHsSpeed
 CommIOMAP, 643
 NBCCCommon.h, 854

CommOffsetHsState
 CommIOMAP, 643
 NBCCCommon.h, 854

CommOffsetPFunc
 CommIOMAP, 643
 NBCCCommon.h, 854

CommOffsetPFuncTwo
 CommIOMAP, 643
 NBCCCommon.h, 854

CommOffsetUsbInBufBuf
 CommIOMAP, 644
 NBCCCommon.h, 854

CommOffsetUsbInBufInPtr
 CommIOMAP, 644
 NBCCCommon.h, 854

CommOffsetUsbInBufOutPtr
 CommIOMAP, 644
 NBCCCommon.h, 854

CommOffsetUsbOutBufBuf
 CommIOMAP, 644
 NBCCCommon.h, 855

CommOffsetUsbOutBufInPtr
 CommIOMAP, 644
 NBCCCommon.h, 855

CommOffsetUsbOutBufOutPtr
 CommIOMAP, 644
 NBCCCommon.h, 855

CommOffsetUsbPollBufBuf
 CommIOMAP, 644
 NBCCCommon.h, 855

CommOffsetUsbPollBufInPtr
 CommIOMAP, 644
 NBCCCommon.h, 855

CommOffsetUsbPollBufOutPtr
 CommIOMAP, 644
 NBCCCommon.h, 855

CommOffsetUsbState
 CommIOMAP, 644
 NBCCCommon.h, 855

CommStatusCodesConstants
 BT_CMD_READY, 637
 HS_CMD_READY, 637
 LR_COULD_NOT_SAVE, 637
 LR_ENTRY_REMOVED, 637
 LR_STORE_IS_FULL, 638
 LR_SUCCESS, 638
 LR_UNKNOWN_ADDR, 638
 USB_CMD_READY, 638

Communications specific errors, 500

Comparison Constants, 73

ComputeCalibValue
 NBCCCommon.h, 855
 SysCallConstants, 481

ConfigureTemperatureSensor
 LowSpeedModuleFunctions, 315

CONN_BT0
 CommConnectionConstants, 619
 NBCCCommon.h, 855

CONN_BT1
 CommConnectionConstants, 619
 NBCCCommon.h, 855

CONN_BT2
 CommConnectionConstants, 619
 NBCCCommon.h, 856

CONN_BT3
 CommConnectionConstants, 620
 NBCCCommon.h, 856

CONN_HS4
 CommConnectionConstants, 620
 NBCCCommon.h, 856

CONN_HS_1
 CommConnectionConstants, 620
 NBCCCommon.h, 856

CONN_HS_2
 CommConnectionConstants, 620
 NBCCCommon.h, 856

CONN_HS_3
 CommConnectionConstants, 620
 NBCCCommon.h, 856

CONN_HS_4
 CommConnectionConstants, 620
 NBCCCommon.h, 856

CONN_HS_5
 CommConnectionConstants, 620
 NBCCCommon.h, 856

CONN_HS_6
 CommConnectionConstants, 620
 NBCCCommon.h, 856

CONN_HS_7
 CommConnectionConstants, 620
 NBCCCommon.h, 856

CONN_HS_8
 CommConnectionConstants, 620
 NBCCCommon.h, 857

CONN_HS_ALL

CommConnectionConstants, 621
NBCCCommon.h, 857

Constants to use with the Input module's Pin function, 557

CreateFile
 LoaderModuleFunctions, 464

CreateFileLinear
 LoaderModuleFunctions, 465

CreateFileNonLinear
 LoaderModuleFunctions, 465

cstdlib API, 473

cstdlibAPI
 Random, 474
 SignedRandom, 474

CT_ADDR_RFID
 CTRFIDConstants, 752
 NBCCCommon.h, 857

CT_REG_DATA
 CTRFIDConstants, 752
 NBCCCommon.h, 857

CT_REG_MODE
 CTRFIDConstants, 752
 NBCCCommon.h, 857

CT_REG_STATUS
 CTRFIDConstants, 752
 NBCCCommon.h, 857

CTRFIDConstants
 CT_ADDR_RFID, 752
 CT_REG_DATA, 752
 CT_REG_MODE, 752
 CT_REG_STATUS, 752

CTRFIDModeConstants
 RFID_MODE_CONTINUOUS, 753
 RFID_MODE_SINGLE, 753
 RFID_MODE_STOP, 753

DAC_MODE_DCOUT
 DacModeConstants, 143
 NBCCCommon.h, 857

DAC_MODE_PWMVOLTAGE
 DacModeConstants, 143
 NBCCCommon.h, 857

DAC_MODE_SAWNEGWAVE
 DacModeConstants, 143
 NBCCCommon.h, 857

DAC_MODE_SAWPOSWAVE

DacModeConstants, 143
NBCCCommon.h, 857

DAC_MODE_SINEWAVE
 DacModeConstants, 143
 NBCCCommon.h, 858

DAC_MODE_SQUAREWAVE
 DacModeConstants, 144
 NBCCCommon.h, 858

DAC_MODE_TRIANGLEWAVE
 DacModeConstants, 144
 NBCCCommon.h, 858

DacModeConstants
 DAC_MODE_DCOUT, 143
 DAC_MODE_PWMVOLTAGE,
 143
 DAC_MODE_SAWNEGWAVE,
 143
 DAC_MODE_SAWPOSWAVE, 143
 DAC_MODE_SINEWAVE, 143
 DAC_MODE_SQUAREWAVE, 144
 DAC_MODE_TRIANGLEWAVE,
 144

Data mode constants, 617

Data type limits, 784

DATA_MODE_GPS
 CommDataModeConstants, 617
 NBCCCommon.h, 858

DATA_MODE_MASK
 CommDataModeConstants, 617
 NBCCCommon.h, 858

DATA_MODE_NXT
 CommDataModeConstants, 617
 NBCCCommon.h, 858

DATA_MODE_RAW
 CommDataModeConstants, 617
 NBCCCommon.h, 858

DATA_MODE_UPDATE
 CommDataModeConstants, 617
 NBCCCommon.h, 858

DatalogGetTimes
 NBCCCommon.h, 858
 SysCallConstants, 481

DatalogWrite
 NBCCCommon.h, 858
 SysCallConstants, 481

DEGREES_PER_RADIAN

MiscConstants, 266
NBCCCommon.h, 859

DeleteFile
 LoaderModuleFunctions, 466

Device status constants, 633

Dexter Industries API Functions, 239

Dexter Industries device constants, 753

Dexter Industries GPS sensor constants, 754

Dexter Industries IMU Accelerometer control register 1 constants, 779

Dexter Industries IMU Accelerometer control register 2 constants, 780

Dexter Industries IMU Accelerometer interrupt latch reset register constants, 778

Dexter Industries IMU Accelerometer mode control register constants, 777

Dexter Industries IMU Accelerometer register constants, 772

Dexter Industries IMU Accelerometer status register constants, 776

Dexter Industries IMU Gyro control register 1 constants, 762

Dexter Industries IMU Gyro control register 2 constants, 764

Dexter Industries IMU Gyro control register 3 constants, 766

Dexter Industries IMU Gyro control register 4 constants, 767

Dexter Industries IMU Gyro control register 5 constants, 768

Dexter Industries IMU Gyro FIFO control register constants, 770

Dexter Industries IMU Gyro register constants, 757

Dexter Industries IMU Gyro status register constants, 771

Dexter Industries IMU sensor constants, 755

DexterIndustriesAPI
 ReadSensorDIAccl, 242
 ReadSensorDIAccl8, 242
 ReadSensorDIAccl8Raw, 243
 ReadSensorDIAcclDrift, 243

ReadSensorDIAcclRaw, 244
ReadSensorDIAcclStatus, 244
ReadSensorDIGPSDistanceToWaypoint, 244
ReadSensorDIGPSHeading, 245
ReadSensorDIGPSHeadingToWaypoint, 245
ReadSensorDIGPSLatitude, 245
ReadSensorDIGPSLongitude, 245
ReadSensorDIGPSRelativeHeading, 246
ReadSensorDIGPSStatus, 246
ReadSensorDIGPSTime, 246
ReadSensorDIGPSVelocity, 247
ReadSensorDIGyro, 247
ReadSensorDIGyroRaw, 247
ReadSensorDIGyroStatus, 248
ReadSensorDIGyroTemperature, 248
SetSensorDIAccl, 248
SetSensorDIAcclDrift, 249
SetSensorDIAcclEx, 249
SetSensorDIGPSWaypoint, 250
SetSensorDIGyro, 250
SetSensorDIGyroEx, 250

DGPS_REG_DISTANCE
 DIGPSConstants, 754
 NBCCCommon.h, 859

DGPS_REG_HEADING
 DIGPSConstants, 754
 NBCCCommon.h, 859

DGPS_REG_LASTANGLE
 DIGPSConstants, 754
 NBCCCommon.h, 859

DGPS_REG_LATITUDE
 DIGPSConstants, 754
 NBCCCommon.h, 859

DGPS_REG_LONGITUDE
 DIGPSConstants, 754
 NBCCCommon.h, 859

DGPS_REG_SETLATITUDE
 DIGPSConstants, 755
 NBCCCommon.h, 859

DGPS_REG_SETLONGITUDE
 DIGPSConstants, 755
 NBCCCommon.h, 859

DGPS_REG_STATUS
DIGPSConstants, 755
NBCCCommon.h, 859

DGPS_REG_TIME
DIGPSConstants, 755
NBCCCommon.h, 859

DGPS_REG_VELOCITY
DIGPSConstants, 755
NBCCCommon.h, 860

DGPS_REG_WAYANGLE
DIGPSConstants, 755
NBCCCommon.h, 860

DI_ADDR_ACCL
DIIMUConstants, 757
NBCCCommon.h, 860

DI_ADDR_DGPS
DIGPSConstants, 755
NBCCCommon.h, 860

DI_ADDR_GYRO
DIIMUConstants, 757
NBCCCommon.h, 860

DIACCL_CTRL1_FILT_BW125
DIIMUAccelCtrl1Constants, 779
NBCCCommon.h, 860

DIACCL_CTRL1_INT2TOINT1
DIIMUAccelCtrl1Constants, 779
NBCCCommon.h, 860

DIACCL_CTRL1_LEVELPULSE
DIIMUAccelCtrl1Constants, 780
NBCCCommon.h, 860

DIACCL_CTRL1_NO_XDETECT
DIIMUAccelCtrl1Constants, 780
NBCCCommon.h, 860

DIACCL_CTRL1_NO_YDETECT
DIIMUAccelCtrl1Constants, 780
NBCCCommon.h, 860

DIACCL_CTRL1_NO_ZDETECT
DIIMUAccelCtrl1Constants, 780
NBCCCommon.h, 861

DIACCL_CTRL1_PULSELEVEL
DIIMUAccelCtrl1Constants, 780
NBCCCommon.h, 861

DIACCL_CTRL1_PULSEPULSE
DIIMUAccelCtrl1Constants, 780
NBCCCommon.h, 861

DIACCL_CTRL1_THRESH_INT
DIIMUAccelCtrl1Constants, 780
NBCCCommon.h, 861

DIACCL_CTRL2_DETPOL_NEGAND
DIIMUAccelCtrl2Constants, 781
NBCCCommon.h, 861

DIACCL_CTRL2_DRIVE_STRONG
DIIMUAccelCtrl2Constants, 781
NBCCCommon.h, 861

DIACCL_CTRL2_LVLPOLE_NEGAND
DIIMUAccelCtrl2Constants, 781
NBCCCommon.h, 861

DIACCL_INTERRUPT_LATCH_-
CLEAR1
DIIMUAccelInterruptLatchCon-
stants, 779
NBCCCommon.h, 861

DIACCL_INTERRUPT_LATCH_-
CLEAR2
DIIMUAccelInterruptLatchCon-
stants, 779
NBCCCommon.h, 861

DIACCL_MODE_GLVL2
DIIMUAccelModeConstants, 777
NBCCCommon.h, 862

DIACCL_MODE_GLVL4
DIIMUAccelModeConstants, 777
NBCCCommon.h, 862

DIACCL_MODE_GLVL8
DIIMUAccelModeConstants, 778
NBCCCommon.h, 862

DIACCL_MODE_LVLDETECT
DIIMUAccelModeConstants, 778
NBCCCommon.h, 862

DIACCL_MODE_MEASURE
DIIMUAccelModeConstants, 778
NBCCCommon.h, 862

DIACCL_MODE_PLSDETECT
DIIMUAccelModeConstants, 778
NBCCCommon.h, 862

DIACCL_MODE_STANDBY
DIIMUAccelModeConstants, 778
NBCCCommon.h, 862

DIACCL_REG_CTRL1
DIIMUAccelRegisterConstants, 773
NBCCCommon.h, 862

DIACCL_REG_CTRL2

DIIMUAcclRegisterConstants, 773
NBCCCommon.h, 862

DIACCL_REG_DETECTSRC
DIIMUAcclRegisterConstants, 773
NBCCCommon.h, 862

DIACCL_REG_I2CADDR
DIIMUAcclRegisterConstants, 773
NBCCCommon.h, 863

DIACCL_REG_INTLATCH
DIIMUAcclRegisterConstants, 773
NBCCCommon.h, 863

DIACCL_REG_LATENCYTM
DIIMUAcclRegisterConstants, 774
NBCCCommon.h, 863

DIACCL_REG_LVLDETTHR
DIIMUAcclRegisterConstants, 774
NBCCCommon.h, 863

DIACCL_REG_MODECTRL
DIIMUAcclRegisterConstants, 774
NBCCCommon.h, 863

DIACCL_REG_OUTTEMP
DIIMUAcclRegisterConstants, 774
NBCCCommon.h, 863

DIACCL_REG_PLSDETTHR
DIIMUAcclRegisterConstants, 774
NBCCCommon.h, 863

DIACCL_REG_PLSDURVAL
DIIMUAcclRegisterConstants, 774
NBCCCommon.h, 863

DIACCL_REG_STATUS
DIIMUAcclRegisterConstants, 774
NBCCCommon.h, 863

DIACCL_REG_TIMEWINDOW
DIIMUAcclRegisterConstants, 774
NBCCCommon.h, 863

DIACCL_REG_USERINFO
DIIMUAcclRegisterConstants, 774
NBCCCommon.h, 864

DIACCL_REG_WHOAMI
DIIMUAcclRegisterConstants, 774
NBCCCommon.h, 864

DIACCL_REG_X8
DIIMUAcclRegisterConstants, 775
NBCCCommon.h, 864

DIACCL_REG_XHIGH
DIIMUAcclRegisterConstants, 775

NBCCCommon.h, 864

DIACCL_REG_XHIGHDRIFT
DIIMUAcclRegisterConstants, 775
NBCCCommon.h, 864

DIACCL_REG_XLOW
DIIMUAcclRegisterConstants, 775
NBCCCommon.h, 864

DIACCL_REG_XLOWDRIFT
DIIMUAcclRegisterConstants, 775
NBCCCommon.h, 864

DIACCL_REG_Y8
DIIMUAcclRegisterConstants, 775
NBCCCommon.h, 864

DIACCL_REG_YHIGH
DIIMUAcclRegisterConstants, 775
NBCCCommon.h, 864

DIACCL_REG_YHIGHDRIFT
DIIMUAcclRegisterConstants, 775
NBCCCommon.h, 864

DIACCL_REG_YLOW
DIIMUAcclRegisterConstants, 775
NBCCCommon.h, 865

DIACCL_REG_YLOWDRIFT
DIIMUAcclRegisterConstants, 775
NBCCCommon.h, 865

DIACCL_REG_Z8
DIIMUAcclRegisterConstants, 776
NBCCCommon.h, 865

DIACCL_REG_ZHIGH
DIIMUAcclRegisterConstants, 776
NBCCCommon.h, 865

DIACCL_REG_ZHIGHDRIFT
DIIMUAcclRegisterConstants, 776
NBCCCommon.h, 865

DIACCL_REG_ZLOW
DIIMUAcclRegisterConstants, 776
NBCCCommon.h, 865

DIACCL_REG_ZLOWDRIFT
DIIMUAcclRegisterConstants, 776
NBCCCommon.h, 865

DIACCL_STATUS_DATAOVER
DIIMUAccelStatusConstants, 777
NBCCCommon.h, 865

DIACCL_STATUS_DATAREADY
DIIMUAccelStatusConstants, 777
NBCCCommon.h, 865

DIACCL_STATUS_PARITYERR
 DIIMUAccelStatusConstants, [777](#)
 NBCCommon.h, [865](#)

DIGI_PIN0
 DigitalPinConstants, [145](#)
 NBCCommon.h, [866](#)

DIGI_PIN1
 DigitalPinConstants, [145](#)
 NBCCommon.h, [866](#)

DIGI_PIN2
 DigitalPinConstants, [145](#)
 NBCCommon.h, [866](#)

DIGI_PIN3
 DigitalPinConstants, [145](#)
 NBCCommon.h, [866](#)

DIGI_PIN4
 DigitalPinConstants, [145](#)
 NBCCommon.h, [866](#)

DIGI_PIN5
 DigitalPinConstants, [146](#)
 NBCCommon.h, [866](#)

DIGI_PIN6
 DigitalPinConstants, [146](#)
 NBCCommon.h, [866](#)

DIGI_PIN7
 DigitalPinConstants, [146](#)
 NBCCommon.h, [866](#)

DigitalPinConstants
 DIGI_PIN0, [145](#)
 DIGI_PIN1, [145](#)
 DIGI_PIN2, [145](#)
 DIGI_PIN3, [145](#)
 DIGI_PIN4, [145](#)
 DIGI_PIN5, [146](#)
 DIGI_PIN6, [146](#)
 DIGI_PIN7, [146](#)

DIGPSConstants
 DGPS_REG_DISTANCE, [754](#)
 DGPS_REG_HEADING, [754](#)
 DGPS_REG_LASTANGLE, [754](#)
 DGPS_REG_LATITUDE, [754](#)
 DGPS_REG_LONGITUDE, [754](#)
 DGPS_REG_SETLATITUDE, [755](#)
 DGPS_REG_SETLONGITUDE, [755](#)
 DGPS_REG_STATUS, [755](#)

DGPS_REG_TIME, [755](#)
DGPS_REG_VELOCITY, [755](#)
DGPS_REG_WAYANGLE, [755](#)

DI_ADDR_DGPS, [755](#)

DIGYRO_CTRL1_BANDWIDTH_1
 DIIMUGyroCtrl1Constants, [762](#)
 NBCCommon.h, [866](#)

DIGYRO_CTRL1_BANDWIDTH_2
 DIIMUGyroCtrl1Constants, [762](#)
 NBCCommon.h, [866](#)

DIGYRO_CTRL1_BANDWIDTH_3
 DIIMUGyroCtrl1Constants, [762](#)
 NBCCommon.h, [867](#)

DIGYRO_CTRL1_BANDWIDTH_4
 DIIMUGyroCtrl1Constants, [762](#)
 NBCCommon.h, [867](#)

DIGYRO_CTRL1_DATARATE_100
 DIIMUGyroCtrl1Constants, [763](#)
 NBCCommon.h, [867](#)

DIGYRO_CTRL1_DATARATE_200
 DIIMUGyroCtrl1Constants, [763](#)
 NBCCommon.h, [867](#)

DIGYRO_CTRL1_DATARATE_400
 DIIMUGyroCtrl1Constants, [763](#)
 NBCCommon.h, [867](#)

DIGYRO_CTRL1_DATARATE_800
 DIIMUGyroCtrl1Constants, [763](#)
 NBCCommon.h, [867](#)

DIGYRO_CTRL1_NORMAL
 DIIMUGyroCtrl1Constants, [763](#)
 NBCCommon.h, [867](#)

DIGYRO_CTRL1_POWERDOWN
 DIIMUGyroCtrl1Constants, [763](#)
 NBCCommon.h, [867](#)

DIGYRO_CTRL1_XENABLE
 DIIMUGyroCtrl1Constants, [763](#)
 NBCCommon.h, [867](#)

DIGYRO_CTRL1_YENABLE
 DIIMUGyroCtrl1Constants, [763](#)
 NBCCommon.h, [867](#)

DIGYRO_CTRL1_ZENABLE
 DIIMUGyroCtrl1Constants, [763](#)
 NBCCommon.h, [868](#)

DIGYRO_CTRL2_CUTOFF_FREQ_-
 001
 DIIMUGyroCtrl2Constants, [764](#)

NBCCCommon.h, 868
DIGYRO_CTRL2_CUTOFF_FREQ_-
 002
 DIIMUGyroCtrl2Constants, 764
 NBCCCommon.h, 868
DIGYRO_CTRL2_CUTOFF_FREQ_-
 005
 DIIMUGyroCtrl2Constants, 764
 NBCCCommon.h, 868
DIGYRO_CTRL2_CUTOFF_FREQ_01
 DIIMUGyroCtrl2Constants, 764
 NBCCCommon.h, 868
DIGYRO_CTRL2_CUTOFF_FREQ_02
 DIIMUGyroCtrl2Constants, 765
 NBCCCommon.h, 868
DIGYRO_CTRL2_CUTOFF_FREQ_05
 DIIMUGyroCtrl2Constants, 765
 NBCCCommon.h, 868
DIGYRO_CTRL2_CUTOFF_FREQ_1
 DIIMUGyroCtrl2Constants, 765
 NBCCCommon.h, 868
DIGYRO_CTRL2_CUTOFF_FREQ_2
 DIIMUGyroCtrl2Constants, 765
 NBCCCommon.h, 868
DIGYRO_CTRL2_CUTOFF_FREQ_4
 DIIMUGyroCtrl2Constants, 765
 NBCCCommon.h, 868
DIGYRO_CTRL2_CUTOFF_FREQ_8
 DIIMUGyroCtrl2Constants, 765
 NBCCCommon.h, 869
DIGYRO_CTRL2_HPMODE_-
 AUTOINT
 DIIMUGyroCtrl2Constants, 765
 NBCCCommon.h, 869
DIGYRO_CTRL2_HPMODE_-
 NORMAL
 DIIMUGyroCtrl2Constants, 765
 NBCCCommon.h, 869
DIGYRO_CTRL2_HPMODE_REFSIG
 DIIMUGyroCtrl2Constants, 765
 NBCCCommon.h, 869
DIGYRO_CTRL2_HPMODE_RESET
 DIIMUGyroCtrl2Constants, 765
 NBCCCommon.h, 869
DIGYRO_CTRL3_INT1_BOOT
 DIIMUGyroCtrl3Constants, 766
NBCCCommon.h, 869
DIGYRO_CTRL3_INT1_ENABLE
 DIIMUGyroCtrl3Constants, 766
 NBCCCommon.h, 869
DIGYRO_CTRL3_INT1_LOWACTIVE
 DIIMUGyroCtrl3Constants, 766
 NBCCCommon.h, 869
DIGYRO_CTRL3_INT2_DATAREADY
 DIIMUGyroCtrl3Constants, 766
 NBCCCommon.h, 869
DIGYRO_CTRL3_INT2_EMPTY
 DIIMUGyroCtrl3Constants, 767
 NBCCCommon.h, 869
DIGYRO_CTRL3_INT2_OVERRUN
 DIIMUGyroCtrl3Constants, 767
 NBCCCommon.h, 870
DIGYRO_CTRL3_INT2_-
 WATERMARK
 DIIMUGyroCtrl3Constants, 767
 NBCCCommon.h, 870
DIGYRO_CTRL3_OPENDRAIN
 DIIMUGyroCtrl3Constants, 767
 NBCCCommon.h, 870
DIGYRO_CTRL4_BIGENDIAN
 DIIMUGyroCtrl4Constants, 768
 NBCCCommon.h, 870
DIGYRO_CTRL4_BLOCKDATA
 DIIMUGyroCtrl4Constants, 768
 NBCCCommon.h, 870
DIGYRO_CTRL4_SCALE_2000
 DIIMUGyroCtrl4Constants, 768
 NBCCCommon.h, 870
DIGYRO_CTRL4_SCALE_250
 DIIMUGyroCtrl4Constants, 768
 NBCCCommon.h, 870
DIGYRO_CTRL4_SCALE_500
 DIIMUGyroCtrl4Constants, 768
 NBCCCommon.h, 870
DIGYRO_CTRL5_FIFOENABLE
 DIIMUGyroCtrl5Constants, 769
 NBCCCommon.h, 870
DIGYRO_CTRL5_HPENABLE
 DIIMUGyroCtrl5Constants, 769
 NBCCCommon.h, 870
DIGYRO_CTRL5_INT1_SEL_1
 DIIMUGyroCtrl5Constants, 769

NBCCCommon.h, 871
DIGYRO_CTRL5_INT1_SEL_2
 DIIMUGyroCtrl5Constants, 769
 NBCCCommon.h, 871
DIGYRO_CTRL5_INT1_SEL_3
 DIIMUGyroCtrl5Constants, 769
 NBCCCommon.h, 871
DIGYRO_CTRL5_OUT_SEL_1
 DIIMUGyroCtrl5Constants, 769
 NBCCCommon.h, 871
DIGYRO_CTRL5_OUT_SEL_2
 DIIMUGyroCtrl5Constants, 769
 NBCCCommon.h, 871
DIGYRO_CTRL5_OUT_SEL_3
 DIIMUGyroCtrl5Constants, 769
 NBCCCommon.h, 871
DIGYRO_CTRL5_REBOOTMEM
 DIIMUGyroCtrl5Constants, 769
 NBCCCommon.h, 871
DIGYRO_FIFOCTRL_BYPASS
 DIIMUGyroFifoCtrlConstants, 770
 NBCCCommon.h, 871
DIGYRO_FIFOCTRL_-
 BYPASS2STREAM
 DIIMUGyroFifoCtrlConstants, 770
 NBCCCommon.h, 871
DIGYRO_FIFOCTRL_FIFO
 DIIMUGyroFifoCtrlConstants, 770
 NBCCCommon.h, 871
DIGYRO_FIFOCTRL_STREAM
 DIIMUGyroFifoCtrlConstants, 770
 NBCCCommon.h, 872
DIGYRO_FIFOCTRL_STREAM2FIFO
 DIIMUGyroFifoCtrlConstants, 770
 NBCCCommon.h, 872
DIGYRO_FIFOCTRL_WATERMARK_-
 MASK
 DIIMUGyroFifoCtrlConstants, 771
 NBCCCommon.h, 872
DIGYRO_REG_CTRL1
 DIIMUGyroRegisterConstants, 758
 NBCCCommon.h, 872
DIGYRO_REG_CTRL1AUTO
 DIIMUGyroRegisterConstants, 758
 NBCCCommon.h, 872
DIGYRO_REG_CTRL2
 DIIMUGyroRegisterConstants, 758
 NBCCCommon.h, 872
DIGYRO_REG_CTRL3
 DIIMUGyroRegisterConstants, 759
 NBCCCommon.h, 872
DIGYRO_REG_CTRL4
 DIIMUGyroRegisterConstants, 759
 NBCCCommon.h, 872
DIGYRO_REG_CTRL5
 DIIMUGyroRegisterConstants, 759
 NBCCCommon.h, 872
DIGYRO_REG_FIFOCTRL
 DIIMUGyroRegisterConstants, 759
 NBCCCommon.h, 872
DIGYRO_REG_FIFOSRC
 DIIMUGyroRegisterConstants, 759
 NBCCCommon.h, 873
DIGYRO_REG_INT1_CFG
 DIIMUGyroRegisterConstants, 759
 NBCCCommon.h, 873
DIGYRO_REG_INT1_DUR
 DIIMUGyroRegisterConstants, 759
 NBCCCommon.h, 873
DIGYRO_REG_INT1_SRC
 DIIMUGyroRegisterConstants, 759
 NBCCCommon.h, 873
DIGYRO_REG_INT1_XHI
 DIIMUGyroRegisterConstants, 759
 NBCCCommon.h, 873
DIGYRO_REG_INT1_XLO
 DIIMUGyroRegisterConstants, 759
 NBCCCommon.h, 873
DIGYRO_REG_INT1_YHI
 DIIMUGyroRegisterConstants, 760
 NBCCCommon.h, 873
DIGYRO_REG_INT1_YLO
 DIIMUGyroRegisterConstants, 760
 NBCCCommon.h, 873
DIGYRO_REG_INT1_ZHI
 DIIMUGyroRegisterConstants, 760
 NBCCCommon.h, 873
DIGYRO_REG_INT1_ZLO
 DIIMUGyroRegisterConstants, 760
 NBCCCommon.h, 873
DIGYRO_REG_OUTTEMP
 DIIMUGyroRegisterConstants, 760

NBCCCommon.h, 874
DIGYRO_REG_REFERENCE
 DIIMUGyroRegisterConstants, 760
 NBCCCommon.h, 874
DIGYRO_REG_STATUS
 DIIMUGyroRegisterConstants, 760
 NBCCCommon.h, 874
DIGYRO_REG_TEMPAUTO
 DIIMUGyroRegisterConstants, 760
 NBCCCommon.h, 874
DIGYRO_REG_WHOAMI
 DIIMUGyroRegisterConstants, 760
 NBCCCommon.h, 874
DIGYRO_REG_XHIGH
 DIIMUGyroRegisterConstants, 760
 NBCCCommon.h, 874
DIGYRO_REG_XLOW
 DIIMUGyroRegisterConstants, 761
 NBCCCommon.h, 874
DIGYRO_REG_XLOWBURST
 DIIMUGyroRegisterConstants, 761
 NBCCCommon.h, 874
DIGYRO_REG_YHIGH
 DIIMUGyroRegisterConstants, 761
 NBCCCommon.h, 874
DIGYRO_REG_YLOW
 DIIMUGyroRegisterConstants, 761
 NBCCCommon.h, 874
DIGYRO_REG_YLOWBURST
 DIIMUGyroRegisterConstants, 761
 NBCCCommon.h, 875
DIGYRO_REG_ZHIGH
 DIIMUGyroRegisterConstants, 761
 NBCCCommon.h, 875
DIGYRO_REG_ZLOW
 DIIMUGyroRegisterConstants, 761
 NBCCCommon.h, 875
DIGYRO_REG_ZLOWBURST
 DIIMUGyroRegisterConstants, 761
 NBCCCommon.h, 875
DIGYRO_STATUS_XDATA
 DIIMUGyroStatusConstants, 771
 NBCCCommon.h, 875
DIGYRO_STATUS_XOVER
 DIIMUGyroStatusConstants, 771
 NBCCCommon.h, 875
DIGYRO_STATUS_XYZDATA
 DIIMUGyroStatusConstants, 771
 NBCCCommon.h, 875
DIGYRO_STATUS_XZOVER
 DIIMUGyroStatusConstants, 771
 NBCCCommon.h, 875
DIGYRO_STATUS_YDATA
 DIIMUGyroStatusConstants, 772
 NBCCCommon.h, 875
DIGYRO_STATUS_YOVER
 DIIMUGyroStatusConstants, 772
 NBCCCommon.h, 875
DIGYRO_STATUS_ZDATA
 DIIMUGyroStatusConstants, 772
 NBCCCommon.h, 876
DIIMUAccelCtrl1Constants
 DIACCL_CTRL1_filt_BW125,
 779
 DIACCL_CTRL1_INT2TOINT1,
 779
 DIACCL_CTRL1_LEVELPULSE,
 780
 DIACCL_CTRL1_NO_XDETECT,
 780
 DIACCL_CTRL1_NO_YDETECT,
 780
 DIACCL_CTRL1_NO_ZDETECT,
 780
 DIACCL_CTRL1_PULSELEVEL,
 780
 DIACCL_CTRL1_PULSEPULSE,
 780
 DIACCL_CTRL1_THRESH_INT,
 780
DIIMUAccelCtrl2Constants
 DIACCL_CTRL2_DET POL_-
 NEGAND, 781
 DIACCL_CTRL2_DRIVE_-
 STRONG, 781
 DIACCL_CTRL2_LVLPOL_-
 NEGAND, 781
DIIMUAccelInterruptLatchConstants

DIACCL_INTERRUPT_LATCH_-
CLEAR1, 779
DIACCL_INTERRUPT_LATCH_-
CLEAR2, 779
DIIMUAccelModeConstants
 DIACCL_MODE_GLVL2, 777
 DIACCL_MODE_GLVL4, 777
 DIACCL_MODE_GLVL8, 778
 DIACCL_MODE_LVLDETECT,
 778
 DIACCL_MODE_MEASURE, 778
 DIACCL_MODE_PLSDETECT,
 778
 DIACCL_MODE_STANDBY, 778
DIIMUAccelStatusConstants
 DIACCL_STATUS_DATAOVER,
 777
 DIACCL_STATUS_DATAREADY,
 777
 DIACCL_STATUS_PARITYERR,
 777
DIIMUAcclRegisterConstants
 DIACCL_REG_CTRL1, 773
 DIACCL_REG_CTRL2, 773
 DIACCL_REG_DETECTSRC, 773
 DIACCL_REG_I2CADDR, 773
 DIACCL_REG_INTLATCH, 773
 DIACCL_REG_LATENCYTM, 774
 DIACCL_REG_LVLDETTHR, 774
 DIACCL_REG_MODECTRL, 774
 DIACCL_REG_OUTTEMP, 774
 DIACCL_REG_PLSDETTHR, 774
 DIACCL_REG_PLSDURVAL, 774
 DIACCL_REG_STATUS, 774
 DIACCL_REG_TIMEWINDOW,
 774
 DIACCL_REG_USERINFO, 774
 DIACCL_REG_WHOAMI, 774
 DIACCL_REG_X8, 775
 DIACCL_REG_XHIGH, 775
 DIACCL_REG_XHIGHDRIFT, 775
 DIACCL_REG_XLOW, 775
 DIACCL_REG_XLOWDRIFT, 775
 DIACCL_REG_Y8, 775
 DIACCL_REG_YHIGH, 775
 DIACCL_REG_YHIGHDRIFT, 775
 DIACCL_REG_YLOW, 775
 DIACCL_REG_YLOWDRIFT, 775
 DIACCL_REG_Z8, 776
 DIACCL_REG_ZHIGH, 776
 DIACCL_REG_ZHIGHDRIFT, 776
 DIACCL_REG_ZLOW, 776
 DIACCL_REG_ZLOWDRIFT, 776
DIIMUConstants
 DI_ADDR_ACCL, 757
 DI_ADDR_GYRO, 757
DIIMUGyroCtrl1Constants
 DIGYRO_CTRL1_-
 BANDWIDTH_1, 762
 DIGYRO_CTRL1_-
 BANDWIDTH_2, 762
 DIGYRO_CTRL1_-
 BANDWIDTH_3, 762
 DIGYRO_CTRL1_-
 BANDWIDTH_4, 762
 DIGYRO_CTRL1_DATARATE_-
 100, 763
 DIGYRO_CTRL1_DATARATE_-
 200, 763
 DIGYRO_CTRL1_DATARATE_-
 400, 763
 DIGYRO_CTRL1_DATARATE_-
 800, 763
 DIGYRO_CTRL1_NORMAL, 763
 DIGYRO_CTRL1_POWERDOWN,
 763
 DIGYRO_CTRL1_XENABLE, 763
 DIGYRO_CTRL1_YENABLE, 763
 DIGYRO_CTRL1_ZENABLE, 763
DIIMUGyroCtrl2Constants
 DIGYRO_CTRL2_CUTOFF_-
 FREQ_001, 764
 DIGYRO_CTRL2_CUTOFF_-
 FREQ_002, 764
 DIGYRO_CTRL2_CUTOFF_-
 FREQ_005, 764
 DIGYRO_CTRL2_CUTOFF_-
 FREQ_01, 764
 DIGYRO_CTRL2_CUTOFF_-
 FREQ_02, 765
 DIGYRO_CTRL2_CUTOFF_-
 FREQ_05, 765

DIGYRO_CTRL2_CUTOFF_-
 FREQ_1, 765
DIGYRO_CTRL2_CUTOFF_-
 FREQ_2, 765
DIGYRO_CTRL2_CUTOFF_-
 FREQ_4, 765
DIGYRO_CTRL2_CUTOFF_-
 FREQ_8, 765
DIGYRO_CTRL2_HPMODE_-
 AUToint, 765
DIGYRO_CTRL2_HPMODE_-
 NORMAL, 765
DIGYRO_CTRL2_HPMODE_-
 REFSIG, 765
DIGYRO_CTRL2_HPMODE_-
 RESET, 765

DIIMUGyroCtrl3Constants
 DIGYRO_CTRL3_INT1_BOOT, 766
 DIGYRO_CTRL3_INT1_ENABLE, 766
 DIGYRO_CTRL3_INT1_-
 LOWACTIVE, 766
 DIGYRO_CTRL3_INT2_-
 DATAREADY, 766
 DIGYRO_CTRL3_INT2_EMPTY, 767
 DIGYRO_CTRL3_INT2_-
 OVERRUN, 767
 DIGYRO_CTRL3_INT2_-
 WATERMARK, 767
 DIGYRO_CTRL3_OPENDRAIN, 767

DIIMUGyroCtrl4Constants
 DIGYRO_CTRL4_BIGENDIAN, 768
 DIGYRO_CTRL4_BLOCKDATA, 768
 DIGYRO_CTRL4_SCALE_2000, 768
 DIGYRO_CTRL4_SCALE_250, 768
 DIGYRO_CTRL4_SCALE_500, 768

DIIMUGyroCtrl5Constants
 DIGYRO_CTRL5_FIFOENABLE, 769
 DIGYRO_CTRL5_HPENABLE, 769
 DIGYRO_CTRL5_INT1_SEL_1, 769
 DIGYRO_CTRL5_INT1_SEL_2, 769
 DIGYRO_CTRL5_INT1_SEL_3, 769
 DIGYRO_CTRL5_OUT_SEL_1, 769
 DIGYRO_CTRL5_OUT_SEL_2, 769
 DIGYRO_CTRL5_OUT_SEL_3, 769
 DIGYRO_CTRL5_REBOOTMEM, 769

DIIMUGyroFifoCtrlConstants
 DIGYRO_FIFOCTRL_BYPASS, 770
 DIGYRO_FIFOCTRL_-
 BYPASS2STREAM, 770
 DIGYRO_FIFOCTRL_FIFO, 770
 DIGYRO_FIFOCTRL_STREAM, 770
 DIGYRO_FIFOCTRL_-
 STREAM2FIFO, 770
 DIGYRO_FIFOCTRL_-
 WATERMARK_MASK, 771

DIIMUGyroRegisterConstants
 DIGYRO_REG_CTRL1, 758
 DIGYRO_REG_CTRL1AUTO, 758
 DIGYRO_REG_CTRL2, 758
 DIGYRO_REG_CTRL3, 759
 DIGYRO_REG_CTRL4, 759
 DIGYRO_REG_CTRL5, 759
 DIGYRO_REG_FIFOCTRL, 759
 DIGYRO_REG_FIFOSRC, 759
 DIGYRO_REG_INT1_CFG, 759
 DIGYRO_REG_INT1_DUR, 759
 DIGYRO_REG_INT1_SRC, 759
 DIGYRO_REG_INT1_XHI, 759
 DIGYRO_REG_INT1_XLO, 759
 DIGYRO_REG_INT1_YHI, 760
 DIGYRO_REG_INT1_YLO, 760

DIGYRO_REG_INT1_ZHI, 760
DIGYRO_REG_INT1_ZLO, 760
DIGYRO_REG_OUTTEMP, 760
DIGYRO_REG_REFERENCE, 760
DIGYRO_REG_STATUS, 760
DIGYRO_REG_TEMPAUTO, 760
DIGYRO_REG_WHOAMI, 760
DIGYRO_REG_XHIGH, 760
DIGYRO_REG_XLOW, 761
DIGYRO_REG_XLOWBURST,
 761
DIGYRO_REG_YHIGH, 761
DIGYRO_REG_YLOW, 761
DIGYRO_REG_YLOWBURST,
 761
DIGYRO_REG_ZHIGH, 761
DIGYRO_REG_ZLOW, 761
DIGYRO_REG_ZLOWBURST,
 761
DIIMUGyroStatusConstants
 DIGYRO_STATUS_XDATA, 771
 DIGYRO_STATUS_XOVER, 771
 DIGYRO_STATUS_XYZDATA,
 771
 DIGYRO_STATUS_XYZOVER,
 771
 DIGYRO_STATUS_YDATA, 772
 DIGYRO_STATUS_YOVER, 772
 DIGYRO_STATUS_ZDATA, 772
 DIGYRO_STATUS_ZOVER, 772
Direct Command functions, 433
Display contrast constants, 607
Display flags, 606
Display module, 88
Display module constants, 595
Display module functions, 329
Display module IOMAP offsets, 610
DISPLAY_BUSY
 DisplayFlagsGroup, 607
 NBCCCommon.h, 876
DISPLAY_CHAR
 DisplayExecuteFunctionConstants,
 601
 NBCCCommon.h, 876
DISPLAY_CONTRAST_DEFAULT
 DisplayContrastConstants, 608
NBCCCommon.h, 876
DISPLAY_CONTRAST_MAX
 DisplayContrastConstants, 608
 NBCCCommon.h, 876
DISPLAY_ERASE_ALL
 DisplayExecuteFunctionConstants,
 601
 NBCCCommon.h, 876
DISPLAY_ERASE_LINE
 DisplayExecuteFunctionConstants,
 601
 NBCCCommon.h, 876
DISPLAY_FILL_REGION
 DisplayExecuteFunctionConstants,
 601
 NBCCCommon.h, 876
DISPLAY_FRAME
 DisplayExecuteFunctionConstants,
 601
 NBCCCommon.h, 876
DISPLAY_HEIGHT
 DisplayModuleConstants, 597
 NBCCCommon.h, 877
DISPLAY_HORIZONTAL_LINE
 DisplayExecuteFunctionConstants,
 602
 NBCCCommon.h, 877
DISPLAY_MENUICONS_X_DIFF
 DisplayModuleConstants, 597
 NBCCCommon.h, 877
DISPLAY_MENUICONS_X_OFFSETS
 DisplayModuleConstants, 597
 NBCCCommon.h, 877
DISPLAY_MENUICONS_Y
 DisplayModuleConstants, 597
 NBCCCommon.h, 877
DISPLAY_ON
 DisplayFlagsGroup, 607
 NBCCCommon.h, 877
DISPLAY_PIXEL
 DisplayExecuteFunctionConstants,
 602
 NBCCCommon.h, 877
DISPLAY_POPUP
 DisplayFlagsGroup, 607
 NBCCCommon.h, 877

DISPLAY_REFRESH
 DisplayFlagsGroup, [607](#)
 NBCCommon.h, [877](#)

DISPLAY_REFRESH_DISABLED
 DisplayFlagsGroup, [607](#)
 NBCCommon.h, [877](#)

DISPLAY_VERTICAL_LINE
 DisplayExecuteFunctionConstants,
 [602](#)
 NBCCommon.h, [878](#)

DISPLAY_WIDTH
 DisplayModuleConstants, [598](#)
 NBCCommon.h, [878](#)

DisplayContrastConstants
 DISPLAY_CONTRAST_-
 DEFAULT, [608](#)
 DISPLAY_CONTRAST_MAX, [608](#)

DisplayDrawOptionConstants
 DRAW_OPT_CLEAR, [603](#)
 DRAW_OPT_CLEAR_EXCEPT_-
 STATUS_SCREEN, [603](#)
 DRAW_OPT_CLEAR_PIXELS,
 [603](#)
 DRAW_OPT_CLEAR_SCREEN_-
 MODES, [603](#)
 DRAW_OPT_CLEAR_WHOLE_-
 SCREEN, [603](#)
 DRAW_OPT_FILL_SHAPE, [604](#)
 DRAW_OPT_INVERT, [604](#)
 DRAW_OPT_LOGICAL_AND,
 [604](#)
 DRAW_OPT_LOGICAL_COPY,
 [604](#)
 DRAW_OPT_LOGICAL_-
 OPERATIONS, [604](#)
 DRAW_OPT_LOGICAL_OR, [604](#)
 DRAW_OPT_LOGICAL_XOR,
 [604](#)
 DRAW_OPT_NORMAL, [604](#)
 DRAW_OPT_POLYGON_-
 POLYLINE, [604](#)

DisplayExecuteFunction
 NBCCommon.h, [878](#)
 SysCallConstants, [481](#)

DisplayExecuteFunction constants, [601](#)

DisplayExecuteFunctionConstants

DISPLAY_CHAR, [601](#)

DISPLAY_ERASE_ALL, [601](#)

DISPLAY_ERASE_LINE, [601](#)

DISPLAY_FILL_REGION, [601](#)

DISPLAY_FRAME, [601](#)

DISPLAY_HORIZONTAL_LINE,
 [602](#)

DISPLAY_PIXEL, [602](#)

DISPLAY_VERTICAL_LINE, [602](#)

DisplayFlagsGroup
 DISPLAY_BUSY, [607](#)
 DISPLAY_ON, [607](#)
 DISPLAY_POPUP, [607](#)
 DISPLAY_REFRESH, [607](#)
 DISPLAY_REFRESH_DISABLED,
 [607](#)

DisplayFontDrawOptionConstants
 DRAW_OPT_FONT_DIR_B2TL,
 [605](#)
 DRAW_OPT_FONT_DIR_B2TR,
 [605](#)
 DRAW_OPT_FONT_DIR_L2RB,
 [605](#)
 DRAW_OPT_FONT_DIR_L2RT,
 [605](#)
 DRAW_OPT_FONT_DIR_R2LB,
 [606](#)
 DRAW_OPT_FONT_DIR_R2LT,
 [606](#)
 DRAW_OPT_FONT_DIR_T2BL,
 [606](#)
 DRAW_OPT_FONT_DIR_T2BR,
 [606](#)
 DRAW_OPT_FONT_-
 DIRECTIONS, [606](#)
 DRAW_OPT_FONT_WRAP, [606](#)

DisplayIOMAP
 DisplayOffsetContrast, [610](#)
 DisplayOffsetDisplay, [610](#)
 DisplayOffsetEraseMask, [610](#)
 DisplayOffsetFlags, [611](#)
 DisplayOffsetNormal, [611](#)
 DisplayOffsetPBitmaps, [611](#)
 DisplayOffsetPFont, [611](#)
 DisplayOffsetPFunc, [611](#)
 DisplayOffsetPMenuIcons, [611](#)

DisplayOffsetPMenuText, 611
DisplayOffsetPopup, 611
DisplayOffsetPScreens, 611
DisplayOffsetPStatusIcons, 611
DisplayOffsetPStatusText, 612
DisplayOffsetPStepIcons, 612
DisplayOffsetPTextLines, 612
DisplayOffsetStatusIcons, 612
DisplayOffsetStepIcons, 612
DisplayOffsetTextLinesCenterFlags,
 612
DisplayOffsetUpdateMask, 612
DisplayModuleConstants
 BITMAP_1, 597
 BITMAP_2, 597
 BITMAP_3, 597
 BITMAP_4, 597
 BITMAPS, 597
 DISPLAY_HEIGHT, 597
 DISPLAY_MENUICONS_X_DIFF,
 597
 DISPLAY_MENUICONS_X_-
 OFFS, 597
 DISPLAY_MENUICONS_Y, 597
 DISPLAY_WIDTH, 598
 FRAME_SELECT, 598
 MENUICON_CENTER, 598
 MENUICON_LEFT, 598
 MENUICON_RIGHT, 598
 MENUICONS, 598
 MENUTEXT, 598
 SCREEN_BACKGROUND, 598
 SCREEN_LARGE, 598
 SCREEN_MODE_CLEAR, 598
 SCREEN_MODE_RESTORE, 599
 SCREEN_SMALL, 599
 SCREENS, 599
 SPECIALS, 599
 STATUSICON_BATTERY, 599
 STATUSICON_BLUETOOTH, 599
 STATUSICON_USB, 599
 STATUSICON_VM, 599
 STATUSICONS, 600
 STATUSTEXT, 600
 STEPICON_1, 600
 STEPICON_2, 600
 STEPICON_3, 600
 STEPICON_4, 600
 STEPICON_5, 600
 STEPICONS, 600
 STEPLINE, 600
 TOPLINE, 600
DisplayModuleFunctions
 CircleOut, 333
 CircleOutEx, 333
 ClearLine, 334
 ClearScreen, 334
 EllipseOut, 334
 EllipseOutEx, 335
 FontNumOut, 335
 FontNumOutEx, 336
 FontTextOut, 336
 FontTextOutEx, 337
 GetDisplayContrast, 337
 GetDisplayDisplay, 338
 GetDisplayEraseMask, 338
 GetDisplayFlags, 338
 GetDisplayFont, 338
 GetDisplayNormal, 339
 GetDisplayPopup, 339
 GetDisplayTextLinesCenterFlags,
 339
 GetDisplayUpdateMask, 340
 GraphicArrayOut, 340
 GraphicArrayOutEx, 340
 GraphicOut, 341
 GraphicOutEx, 341
 LineOut, 342
 LineOutEx, 342
 NumOut, 343
 NumOutEx, 343
 PointOut, 343
 PointOutEx, 344
 PolyOut, 344
 PolyOutEx, 345
 RectOut, 345
 RectOutEx, 345
 SetDisplayContrast, 346
 SetDisplayDisplay, 346
 SetDisplayEraseMask, 346
 SetDisplayFlags, 347
 SetDisplayFont, 347

SetDisplayNormal, [347](#)
SetDisplayPopup, [348](#)
SetDisplayTextLinesCenterFlags,
 [348](#)
SetDisplayUpdateMask, [348](#)
TextOut, [348](#)
 TextOutEx, [349](#)
DisplayModuleID
 ModuleIDConstants, [264](#)
 NBCCCommon.h, [878](#)
DisplayModuleName
 ModuleNameConstants, [262](#)
 NBCCCommon.h, [878](#)
DisplayOffsetContrast
 DisplayIOMAP, [610](#)
 NBCCCommon.h, [878](#)
DisplayOffsetDisplay
 DisplayIOMAP, [610](#)
 NBCCCommon.h, [878](#)
DisplayOffsetEraseMask
 DisplayIOMAP, [610](#)
 NBCCCommon.h, [878](#)
DisplayOffsetFlags
 DisplayIOMAP, [611](#)
 NBCCCommon.h, [878](#)
DisplayOffsetNormal
 DisplayIOMAP, [611](#)
 NBCCCommon.h, [878](#)
DisplayOffsetPBitmaps
 DisplayIOMAP, [611](#)
 NBCCCommon.h, [879](#)
DisplayOffsetPFont
 DisplayIOMAP, [611](#)
 NBCCCommon.h, [879](#)
DisplayOffsetPFunc
 DisplayIOMAP, [611](#)
 NBCCCommon.h, [879](#)
DisplayOffsetPMenuIcons
 DisplayIOMAP, [611](#)
 NBCCCommon.h, [879](#)
DisplayOffsetPMenuText
 DisplayIOMAP, [611](#)
 NBCCCommon.h, [879](#)
DisplayOffsetPopup
 DisplayIOMAP, [611](#)
 NBCCCommon.h, [879](#)
DisplayOffsetPScreens
 DisplayIOMAP, [611](#)
 NBCCCommon.h, [879](#)
DisplayOffsetPStatusIcons
 DisplayIOMAP, [611](#)
 NBCCCommon.h, [879](#)
DisplayOffsetPStatusText
 DisplayIOMAP, [612](#)
 NBCCCommon.h, [879](#)
DisplayOffsetPStepIcons
 DisplayIOMAP, [612](#)
 NBCCCommon.h, [879](#)
DisplayOffsetPTextLines
 DisplayIOMAP, [612](#)
 NBCCCommon.h, [880](#)
DisplayOffsetStatusIcons
 DisplayIOMAP, [612](#)
 NBCCCommon.h, [880](#)
DisplayOffsetStepIcons
 DisplayIOMAP, [612](#)
 NBCCCommon.h, [880](#)
DisplayOffsetTextLinesCenterFlags
 DisplayIOMAP, [612](#)
 NBCCCommon.h, [880](#)
DisplayOffsetUpdateMask
 DisplayIOMAP, [612](#)
 NBCCCommon.h, [880](#)
DisplayTextLineConstants
 TEXTLINE_1, [609](#)
 TEXTLINE_2, [609](#)
 TEXTLINE_3, [609](#)
 TEXTLINE_4, [609](#)
 TEXTLINE_5, [609](#)
 TEXTLINE_6, [609](#)
 TEXTLINE_7, [609](#)
 TEXTLINE_8, [609](#)
 TEXTLINES, [609](#)
DIST_CMD_CUSTOM
 MSDistNX, [716](#)
 NBCCCommon.h, [880](#)
DIST_CMD_GP2D12
 MSDistNX, [716](#)
 NBCCCommon.h, [880](#)
DIST_CMD_GP2D120
 MSDistNX, [716](#)
 NBCCCommon.h, [880](#)

DIST_CMD_GP2YA02
 MSDistNX, 716
 NBCCommon.h, 880

DIST_CMD_GP2YA21
 MSDistNX, 716
 NBCCommon.h, 880

DIST_REG_DIST
 MSDistNX, 716
 NBCCommon.h, 881

DIST_REG_DIST1
 MSDistNX, 717
 NBCCommon.h, 881

DIST_REG_DIST_MAX
 MSDistNX, 717
 NBCCommon.h, 881

DIST_REG_DIST_MIN
 MSDistNX, 717
 NBCCommon.h, 881

DIST_REG_MODULE_TYPE
 MSDistNX, 717
 NBCCommon.h, 881

DIST_REG_NUM_POINTS
 MSDistNX, 717
 NBCCommon.h, 881

DIST_REG_VOLT
 MSDistNX, 717
 NBCCommon.h, 881

DIST_REG_VOLT1
 MSDistNX, 717
 NBCCommon.h, 881

DISTNxGP2D12
 MindSensorsAPI, 169

DISTNxGP2D120
 MindSensorsAPI, 169

DISTNxGP2YA02
 MindSensorsAPI, 170

DISTNxGP2YA21
 MindSensorsAPI, 170

DRAW_OPT_CLEAR
 DisplayDrawOptionConstants, 603
 NBCCommon.h, 881

DRAW_OPT_CLEAR_EXCEPT_-
 STATUS_SCREEN
 DisplayDrawOptionConstants, 603
 NBCCommon.h, 881

DRAW_OPT_CLEAR_PIXELS

 DisplayDrawOptionConstants, 603
 NBCCommon.h, 882

DRAW_OPT_CLEAR_SCREEN_-
 MODES
 DisplayDrawOptionConstants, 603
 NBCCommon.h, 882

DRAW_OPT_CLEAR_WHOLE_-
 SCREEN
 DisplayDrawOptionConstants, 603
 NBCCommon.h, 882

DRAW_OPT_FILL_SHAPE
 DisplayDrawOptionConstants, 604
 NBCCommon.h, 882

DRAW_OPT_FONT_DIR_B2TL
 DisplayFontDrawOptionConstants,
 605
 NBCCommon.h, 882

DRAW_OPT_FONT_DIR_B2TR
 DisplayFontDrawOptionConstants,
 605
 NBCCommon.h, 882

DRAW_OPT_FONT_DIR_L2RB
 DisplayFontDrawOptionConstants,
 605
 NBCCommon.h, 882

DRAW_OPT_FONT_DIR_L2RT
 DisplayFontDrawOptionConstants,
 605
 NBCCommon.h, 882

DRAW_OPT_FONT_DIR_R2LB
 DisplayFontDrawOptionConstants,
 606
 NBCCommon.h, 882

DRAW_OPT_FONT_DIR_R2LT
 DisplayFontDrawOptionConstants,
 606
 NBCCommon.h, 882

DRAW_OPT_FONT_DIR_T2BL
 DisplayFontDrawOptionConstants,
 606
 NBCCommon.h, 883

DRAW_OPT_FONT_DIR_T2BR
 DisplayFontDrawOptionConstants,
 606
 NBCCommon.h, 883

DRAW_OPT_FONT_DIRECTIONS

DisplayFontDrawOptionConstants,
 606
 NBCCCommon.h, [883](#)

DRAW_OPT_FONT_WRAP
 DisplayFontDrawOptionConstants,
 606
 NBCCCommon.h, [883](#)

DRAW_OPT_INVERT
 DisplayDrawOptionConstants, [604](#)
 NBCCCommon.h, [883](#)

DRAW_OPT_LOGICAL_AND
 DisplayDrawOptionConstants, [604](#)
 NBCCCommon.h, [883](#)

DRAW_OPT_LOGICAL_COPY
 DisplayDrawOptionConstants, [604](#)
 NBCCCommon.h, [883](#)

DRAW_OPT_LOGICAL_-
 OPERATIONS
 DisplayDrawOptionConstants, [604](#)
 NBCCCommon.h, [883](#)

DRAW_OPT_LOGICAL_OR
 DisplayDrawOptionConstants, [604](#)
 NBCCCommon.h, [883](#)

DRAW_OPT_LOGICAL_XOR
 DisplayDrawOptionConstants, [604](#)
 NBCCCommon.h, [883](#)

DRAW_OPT_NORMAL
 DisplayDrawOptionConstants, [604](#)
 NBCCCommon.h, [884](#)

DRAW_OPT_POLYGON_POLYLINE
 DisplayDrawOptionConstants, [604](#)
 NBCCCommon.h, [884](#)

DrawCircle
 NBCCCommon.h, [884](#)
 SysCallConstants, [481](#)

DrawEllipse
 NBCCCommon.h, [884](#)
 SysCallConstants, [481](#)

DrawFont
 NBCCCommon.h, [884](#)
 SysCallConstants, [481](#)

DrawGraphic
 NBCCCommon.h, [884](#)
 SysCallConstants, [481](#)

DrawGraphicArray
 NBCCCommon.h, [884](#)

SysCallConstants, [481](#)

Drawing option constants, [602](#)

DrawLine
 NBCCCommon.h, [884](#)
 SysCallConstants, [481](#)

DrawPoint
 NBCCCommon.h, [884](#)
 SysCallConstants, [482](#)

DrawPolygon
 NBCCCommon.h, [884](#)
 SysCallConstants, [482](#)

DrawRect
 NBCCCommon.h, [885](#)
 SysCallConstants, [482](#)

DrawText
 NBCCCommon.h, [885](#)
 SysCallConstants, [482](#)

E-Meter sensor constants, [593](#)

EllipseOut
 DisplayModuleFunctions, [334](#)

EllipseOutEx
 DisplayModuleFunctions, [335](#)

EMETER_REG_AIN
 EMeterI2CConstants, [593](#)
 NBCCCommon.h, [885](#)

EMETER_REG_AOUT
 EMeterI2CConstants, [593](#)
 NBCCCommon.h, [885](#)

EMETER_REG_JOULES
 EMeterI2CConstants, [594](#)
 NBCCCommon.h, [885](#)

EMETER_REG_VIN
 EMeterI2CConstants, [594](#)
 NBCCCommon.h, [885](#)

EMETER_REG_VOUT
 EMeterI2CConstants, [594](#)
 NBCCCommon.h, [885](#)

EMETER_REG_WIN
 EMeterI2CConstants, [594](#)
 NBCCCommon.h, [885](#)

EMETER_REG_WOUT
 EMeterI2CConstants, [594](#)
 NBCCCommon.h, [885](#)

EMeterI2CConstants
 EMETER_REG_AIN, [593](#)

EMETER_REG_AOUT, 593
EMETER_REG_JOULES, 594
EMETER_REG_VIN, 594
EMETER_REG_VOUT, 594
EMETER_REG_WIN, 594
EMETER_REG_WOUT, 594
EOF
 LoaderModuleConstants, 507
 NBCCCommon.h, 885
EQ
 cmpconst, 73
ERR_ARG
 CommandFatalErrors, 497
 NBCCCommon.h, 886
ERR_BAD_POOL_SIZE
 CommandFatalErrors, 497
 NBCCCommon.h, 886
ERR_BAD_PTR
 CommandFatalErrors, 498
 NBCCCommon.h, 886
ERR_CLUMP_COUNT
 CommandFatalErrors, 498
 NBCCCommon.h, 886
ERR_COMM_BUFFER_FULL
 CommandCommErrors, 501
 NBCCCommon.h, 886
ERR_COMM_BUS_ERR
 CommandCommErrors, 501
 NBCCCommon.h, 886
ERR_COMM_CHAN_INVALID
 CommandCommErrors, 501
 NBCCCommon.h, 886
ERR_COMM_CHAN_NOT_READY
 CommandCommErrors, 501
 NBCCCommon.h, 886
ERR_DEFAULT_OFFSETS
 CommandFatalErrors, 498
 NBCCCommon.h, 886
ERR_FILE
 CommandFatalErrors, 498
 NBCCCommon.h, 886
ERR_INSANE_OFFSET
 CommandFatalErrors, 498
 NBCCCommon.h, 887
ERR_INSTR
 CommandFatalErrors, 498
NBCCCommon.h, 887
ERR_INVALID_FIELD
 CommandGenErrors, 500
 NBCCCommon.h, 887
ERR_INVALID_PORT
 CommandGenErrors, 500
 NBCCCommon.h, 887
ERR_INVALID_QUEUE
 CommandGenErrors, 500
 NBCCCommon.h, 887
ERR_INVALID_SIZE
 CommandGenErrors, 500
 NBCCCommon.h, 887
ERR_LOADER_ERR
 CommandFatalErrors, 498
 NBCCCommon.h, 887
ERR_MEM
 CommandFatalErrors, 498
 NBCCCommon.h, 887
ERR_MEMMGR_FAIL
 CommandFatalErrors, 498
 NBCCCommon.h, 887
ERR_NO_ACTIVE_CLUMP
 CommandFatalErrors, 499
 NBCCCommon.h, 887
ERR_NO_CODE
 CommandFatalErrors, 499
 NBCCCommon.h, 888
ERR_NO_PROG
 CommandGenErrors, 500
 NBCCCommon.h, 888
ERR_NON_FATAL
 CommandFatalErrors, 499
 NBCCCommon.h, 888
ERR_RC_BAD_PACKET
 CommandRCErrors, 501
 NBCCCommon.h, 888
ERR_RC_FAILED
 CommandRCErrors, 501
 NBCCCommon.h, 888
ERR_RC_ILLEGAL_VAL
 CommandRCErrors, 502
 NBCCCommon.h, 888
ERR_RC_UNKNOWN_CMD
 CommandRCErrors, 502
 NBCCCommon.h, 888

ERR_SPOTCHECK_FAIL
 CommandFatalErrors, 499
 NBCCCommon.h, 888

ERR_VER
 CommandFatalErrors, 499
 NBCCCommon.h, 888

FALSE
 MiscConstants, 266
 NBCCCommon.h, 888

Fatal errors, 497

FileClose
 NBCCCommon.h, 889
 SysCallConstants, 482

FileDelete
 NBCCCommon.h, 889
 SysCallConstants, 482

FileFindFirst
 NBCCCommon.h, 889
 SysCallConstants, 482

FileFindNext
 NBCCCommon.h, 889
 SysCallConstants, 482

FileOpenAppend
 NBCCCommon.h, 889
 SysCallConstants, 482

FileOpenRead
 NBCCCommon.h, 889
 SysCallConstants, 482

FileOpenReadLinear
 NBCCCommon.h, 889
 SysCallConstants, 483

FileOpenWrite
 NBCCCommon.h, 889
 SysCallConstants, 483

FileOpenWriteLinear
 NBCCCommon.h, 889
 SysCallConstants, 483

FileOpenWriteNonLinear
 NBCCCommon.h, 889
 SysCallConstants, 483

FileRead
 NBCCCommon.h, 890
 SysCallConstants, 483

FileRename
 NBCCCommon.h, 890

SysCallConstants, 483

FileResize
 NBCCCommon.h, 890
 SysCallConstants, 483

FileResolveHandle
 NBCCCommon.h, 890
 SysCallConstants, 483

FileSeek
 NBCCCommon.h, 890
 SysCallConstants, 483

FileTell
 NBCCCommon.h, 890
 SysCallConstants, 483

FileWrite
 NBCCCommon.h, 890
 SysCallConstants, 484

FindFirstFile
 LoaderModuleFunctions, 466

FindNextFile
 LoaderModuleFunctions, 466

Float
 OutputModuleFunctions, 280

Font drawing option constants, 605

FontNumOut
 DisplayModuleFunctions, 335

FontNumOutEx
 DisplayModuleFunctions, 336

FontTextOut
 DisplayModuleFunctions, 336

FontTextOutEx
 DisplayModuleFunctions, 337

ForceOff
 UiModuleFunctions, 385

FRAME_SELECT
 DisplayModuleConstants, 598
 NBCCCommon.h, 890

FREQUENCY_MAX
 NBCCCommon.h, 890
 SoundMisc, 521

FREQUENCY_MIN
 NBCCCommon.h, 890
 SoundMisc, 521

General errors, 499

GenericI2CConstants
 I2C_ADDR_DEFAULT, 587

I2C_REG_CMD, 587
I2C_REG_DEVICE_ID, 587
I2C_REG_VENDOR_ID, 587
I2C_REG_VERSION, 587
GetAbortFlag
 UiModuleFunctions, 385
GetBatteryLevel
 UiModuleFunctions, 385
GetBatteryState
 UiModuleFunctions, 385
GetBluetoothState
 UiModuleFunctions, 385
GetBrickDataAddress
 CommModuleFunctions, 401
GetBrickDataBluecoreVersion
 CommModuleFunctions, 401
GetBrickDataBtHardwareStatus
 CommModuleFunctions, 401
GetBrickDataBtStateStatus
 CommModuleFunctions, 401
GetBrickDataName
 CommModuleFunctions, 402
GetBrickDataTimeoutValue
 CommModuleFunctions, 402
GetBTConnectionAddress
 CommModuleFunctions, 402
GetBTConnectionClass
 CommModuleFunctions, 403
GetBTConnectionHandleNum
 CommModuleFunctions, 403
GetBTConnectionLinkQuality
 CommModuleFunctions, 403
GetBTConnectionName
 CommModuleFunctions, 403
GetBTConnectionPinCode
 CommModuleFunctions, 404
GetBTConnectionStreamStatus
 CommModuleFunctions, 404
GetBTDataMode
 CommModuleFunctions, 404
GetBTDeviceAddress
 CommModuleFunctions, 405
GetBTDeviceClass
 CommModuleFunctions, 405
GetBTDeviceCount
 CommModuleFunctions, 405
GetBTDeviceName
 CommModuleFunctions, 405
GetBTDeviceNameCount
 CommModuleFunctions, 406
GetBTDeviceStatus
 CommModuleFunctions, 406
GetBTInputBuffer
 CommModuleFunctions, 406
GetBTInputBufferInPtr
 CommModuleFunctions, 407
GetBTInputBufferOutPtr
 CommModuleFunctions, 407
GetBTOutputBuffer
 CommModuleFunctions, 407
GetBTOutputBufferInPtr
 CommModuleFunctions, 408
GetBTOutputBufferOutPtr
 CommModuleFunctions, 408
GetButtonLongPressCount
 ButtonModuleFunctions, 379
GetButtonLongReleaseCount
 ButtonModuleFunctions, 379
GetButtonModuleValue
 CommandModuleFunctions, 361
GetButtonPressCount
 ButtonModuleFunctions, 379
GetButtonReleaseCount
 ButtonModuleFunctions, 379
GetButtonShortReleaseCount
 ButtonModuleFunctions, 380
GetButtonState
 ButtonModuleFunctions, 380
GetCommandFlags
 UiModuleFunctions, 386
GetCommandModuleBytes
 CommandModuleFunctions, 361
GetCommandModuleValue
 CommandModuleFunctions, 362
GetCommModuleBytes
 CommandModuleFunctions, 362
GetCommModuleValue
 CommandModuleFunctions, 362
GetDisplayContrast
 DisplayModuleFunctions, 337
GetDisplayDisplay
 DisplayModuleFunctions, 338

GetDisplayEraseMask
 DisplayModuleFunctions, 338
GetDisplayFlags
 DisplayModuleFunctions, 338
GetDisplayFont
 DisplayModuleFunctions, 338
GetDisplayModuleBytes
 CommandModuleFunctions, 363
GetDisplayModuleValue
 CommandModuleFunctions, 363
GetDisplayNormal
 DisplayModuleFunctions, 339
GetDisplayPopup
 DisplayModuleFunctions, 339
GetDisplayTextLinesCenterFlags
 DisplayModuleFunctions, 339
GetDisplayUpdateMask
 DisplayModuleFunctions, 340
GetFirstTick
 CommandModuleFunctions, 363
GetFreeMemory
 LoaderModuleFunctions, 467
GetHSAddress
 CommModuleFunctions, 408
GetHSDaDataMode
 CommModuleFunctions, 408
GetHSFlags
 CommModuleFunctions, 409
GetHSInputBuffer
 CommModuleFunctions, 409
GetHSInputBufferInPtr
 CommModuleFunctions, 409
GetHSInputBufferOutPtr
 CommModuleFunctions, 410
GetHSMode
 CommModuleFunctions, 410
GetHSOutputBuffer
 CommModuleFunctions, 410
GetHSOutputBufferInPtr
 CommModuleFunctions, 411
GetHSOutputBufferOutPtr
 CommModuleFunctions, 411
GetHSSpeed
 CommModuleFunctions, 411
GetHSState
 CommModuleFunctions, 412
GetInColorADRaw
 InputModuleFunctions, 300
GetInColorBoolean
 InputModuleFunctions, 300
GetInColorCalibration
 InputModuleFunctions, 301
GetInColorCalibrationState
 InputModuleFunctions, 301
GetInColorCalLimits
 InputModuleFunctions, 302
GetInColorSensorRaw
 InputModuleFunctions, 302
GetInColorSensorValue
 InputModuleFunctions, 302
GetInCustomActiveStatus
 InputModuleFunctions, 303
GetInCustomPercentFullScale
 InputModuleFunctions, 303
GetInCustomZeroOffset
 InputModuleFunctions, 303
GetInDigiPinsDirection
 InputModuleFunctions, 304
GetInDigiPinsOutputLevel
 InputModuleFunctions, 304
GetInDigiPinsStatus
 InputModuleFunctions, 304
GetInputModuleValue
 CommandModuleFunctions, 364
GetInSensorBoolean
 InputModuleFunctions, 305
GetIOMapBytes
 CommandModuleFunctions, 364
GetIOMapBytesByID
 CommandModuleFunctions, 364
GetIOMapValue
 CommandModuleFunctions, 365
GetIOMapValueByID
 CommandModuleFunctions, 365
GetLastResponseInfo
 CommandModuleFunctions, 366
GetLoaderModuleValue
 CommandModuleFunctions, 366
GetLowSpeedModuleBytes
 CommandModuleFunctions, 367
GetLowSpeedModuleValue
 CommandModuleFunctions, 367

GetLSChannelState
 LowLevelLowSpeedModuleFunctions, 325

GetLSErrorType
 LowLevelLowSpeedModuleFunctions, 326

GetLSInputBuffer
 LowLevelLowSpeedModuleFunctions, 326

GetLSInputBufferBytesToRx
 LowLevelLowSpeedModuleFunctions, 326

GetLSInputBufferInPtr
 LowLevelLowSpeedModuleFunctions, 326

GetLSInputBufferOutPtr
 LowLevelLowSpeedModuleFunctions, 327

GetLSMode
 LowLevelLowSpeedModuleFunctions, 327

GetLSNoRestartOnRead
 LowLevelLowSpeedModuleFunctions, 327

GetLSSOutputBuffer
 LowLevelLowSpeedModuleFunctions, 327

GetLSSOutputBufferBytesToRx
 LowLevelLowSpeedModuleFunctions, 328

GetLSSOutputBufferInPtr
 LowLevelLowSpeedModuleFunctions, 328

GetLSSOutputBufferOutPtr
 LowLevelLowSpeedModuleFunctions, 328

GetLSSpeed
 LowLevelLowSpeedModuleFunctions, 329

GetLSSState
 LowLevelLowSpeedModuleFunctions, 329

GetMemoryInfo
 CommandModuleFunctions, 367

GetOnBrickProgramPointer
 UiModuleFunctions, 386

GetOutputModuleValue
 CommandModuleFunctions, 368

GetOutPwnFreq
 OutputModuleFunctions, 280

GetOutRegulationOptions
 OutputModuleFunctions, 280

GetOutRegulationTime
 OutputModuleFunctions, 281

GetRechargeableBattery
 UiModuleFunctions, 386

GetSleepTimeout
 UiModuleFunctions, 386

GetSleepTimer
 UiModuleFunctions, 387

GetSoundDuration
 SoundModuleFunctions, 351

GetSoundFrequency
 SoundModuleFunctions, 351

GetSoundMode
 SoundModuleFunctions, 352

GetSoundModuleValue
 CommandModuleFunctions, 368

GetSoundSampleRate
 SoundModuleFunctions, 352

GetSoundState
 SoundModuleFunctions, 352

GetSoundVolume
 SoundModuleFunctions, 352

GetStartTick
 NBCCCommon.h, 891
 SysCallConstants, 484

GetUIButton
 UiModuleFunctions, 387

GetUIModuleValue
 CommandModuleFunctions, 369

GetUIState
 UiModuleFunctions, 387

GetUSBInputBuffer
 CommModuleFunctions, 412

GetUSBInputBufferInPtr
 CommModuleFunctions, 412

GetUSBInputBufferOutPtr
 CommModuleFunctions, 413

GetUSBOutputBuffer
 CommModuleFunctions, 413

GetUSBOutputBufferInPtr

CommModuleFunctions, 413
GetUSBOutputBufferOutPtr
 CommModuleFunctions, 414
GetUSBPollBuffer
 CommModuleFunctions, 414
GetUSBPollBufferInPtr
 CommModuleFunctions, 414
GetUSBPollBufferOutPtr
 CommModuleFunctions, 415
GetUSBState
 CommModuleFunctions, 415
GetUsbState
 UiModuleFunctions, 387
GetVMRunState
 UiModuleFunctions, 388
GetVolume
 UiModuleFunctions, 388
GL_CAMERA_DEPTH
 GLConstantsSettings, 789
 NBCCCommon.h, 891
GL_CIRCLE
 GLConstantsBeginModes, 787
 NBCCCommon.h, 891
GL_CIRCLE_SIZE
 GLConstantsSettings, 789
 NBCCCommon.h, 891
GL_CULL_BACK
 GLConstantsCullMode, 790
 NBCCCommon.h, 891
GL_CULL_FRONT
 GLConstantsCullMode, 790
 NBCCCommon.h, 891
GL_CULL_MODE
 GLConstantsSettings, 789
 NBCCCommon.h, 891
GL_CULL_NONE
 GLConstantsCullMode, 790
 NBCCCommon.h, 891
GL_LINE
 GLConstantsBeginModes, 787
 NBCCCommon.h, 891
GL_POINT
 GLConstantsBeginModes, 787
 NBCCCommon.h, 891
GL_POLYGON
 GLConstantsBeginModes, 787
NBCCCommon.h, 892
GL_ROTATE_X
 GLConstantsActions, 788
 NBCCCommon.h, 892
GL_ROTATE_Y
 GLConstantsActions, 788
 NBCCCommon.h, 892
GL_ROTATE_Z
 GLConstantsActions, 788
 NBCCCommon.h, 892
GL_SCALE_X
 GLConstantsActions, 788
 NBCCCommon.h, 892
GL_SCALE_Y
 GLConstantsActions, 788
 NBCCCommon.h, 892
GL_SCALE_Z
 GLConstantsActions, 788
 NBCCCommon.h, 892
GL_TRANSLATE_X
 GLConstantsActions, 788
 NBCCCommon.h, 892
GL_TRANSLATE_Y
 GLConstantsActions, 788
 NBCCCommon.h, 892
GL_TRANSLATE_Z
 GLConstantsActions, 789
 NBCCCommon.h, 892
GL_ZOOM_FACTOR
 GLConstantsSettings, 789
 NBCCCommon.h, 893
glAddToAngleX
 GraphicsLibrary, 270
glAddToAngleY
 GraphicsLibrary, 270
glAddToAngleZ
 GraphicsLibrary, 270
glAddVertex
 GraphicsLibrary, 270
glBegin
 GraphicsLibrary, 271
glBeginObject
 GraphicsLibrary, 271
glBeginRender
 GraphicsLibrary, 271
glBox

GraphicsLibrary, 271
glCallObject
 GraphicsLibrary, 272
GLConstantsActions
 GL_ROTATE_X, 788
 GL_ROTATE_Y, 788
 GL_ROTATE_Z, 788
 GL_SCALE_X, 788
 GL_SCALE_Y, 788
 GL_SCALE_Z, 788
 GL_TRANSLATE_X, 788
 GL_TRANSLATE_Y, 788
 GL_TRANSLATE_Z, 789
GLConstantsBeginModes
 GL_CIRCLE, 787
 GL_LINE, 787
 GL_POINT, 787
 GL_POLYGON, 787
GLConstantsCullMode
 GL_CULL_BACK, 790
 GL_CULL_FRONT, 790
 GL_CULL_NONE, 790
GLConstantsSettings
 GL_CAMERA_DEPTH, 789
 GL_CIRCLE_SIZE, 789
 GL_CULL_MODE, 789
 GL_ZOOM_FACTOR, 789
glCos32768
 GraphicsLibrary, 272
glCube
 GraphicsLibrary, 272
glEnd
 GraphicsLibrary, 273
glEndObject
 GraphicsLibrary, 273
glFinishRender
 GraphicsLibrary, 273
glInit
 GraphicsLibrary, 273
glObjectAction
 GraphicsLibrary, 273
glPyramid
 GraphicsLibrary, 274
glSet
 GraphicsLibrary, 274
glSetAngleX
 GraphicsLibrary, 274
 glSetAngleY
 GraphicsLibrary, 275
 glSetAngleZ
 GraphicsLibrary, 275
 glSin32768
 GraphicsLibrary, 275
GraphicArrayOut
 DisplayModuleFunctions, 340
GraphicArrayOutEx
 DisplayModuleFunctions, 340
GraphicOut
 DisplayModuleFunctions, 341
GraphicOutEx
 DisplayModuleFunctions, 341
Graphics library actions, 787
Graphics library begin modes, 786
Graphics library cull mode, 790
Graphics library settings, 789
GraphicsLibrary
 glAddToAngleX, 270
 glAddToAngleY, 270
 glAddToAngleZ, 270
 glAddVertex, 270
 glBegin, 271
 glBeginObject, 271
 glBeginRender, 271
 glBox, 271
 glCallObject, 272
 glCos32768, 272
 glCube, 272
 glEnd, 273
 glEndObject, 273
 glFinishRender, 273
 glInit, 273
 glObjectAction, 273
 glPyramid, 274
 glSet, 274
 glSetAngleX, 274
 glSetAngleY, 275
 glSetAngleZ, 275
 glSin32768, 275
GT
 cmpconst, 73
GTEQ
 cmpconst, 74

- Hi-speed port address constants, 632
- Hi-speed port baud rate constants, 625
- Hi-speed port combined UART constants, 631
 - Hi-speed port constants, 621
 - Hi-speed port data bits constants, 628
 - Hi-speed port flags constants, 622
 - Hi-speed port parity constants, 630
 - Hi-speed port state constants, 623
 - Hi-speed port stop bits constants, 629
 - Hi-speed port SysCommHSControl constants, 624
 - Hi-speed port UART mode constants, 627
- HiTechnic Angle sensor constants, 698
- HiTechnic API Functions, 88
- HiTechnic Barometric sensor constants, 700
 - HiTechnic Color2 constants, 696
 - HiTechnic device constants, 690
 - HiTechnic IRReceiver constants, 695
 - HiTechnic IRSeeker2 constants, 692
 - HiTechnic Prototype board analog input constants, 702
 - HiTechnic Prototype board constants, 700
 - HiTechnic SuperPro analog input index constants, 710
 - HiTechnic SuperPro analog output index constants, 711
 - HiTechnic SuperPro constants, 703
- HiTechnic/mindsensors Power Function/IR Train constants, 679
- HiTechnicAPI
 - HTIRTrain, 101
 - HTPFComboDirect, 101
 - HTPFComboPWM, 101
 - HTPFRawOutput, 102
 - HTPFRepeat, 102
 - HTPFSingleOutputCST, 103
 - HTPFSingleOutputPWM, 103
 - HTPFSinglePin, 104
 - HTPFTTrain, 105
 - HTPowerFunctionCommand, 105
 - HTRCXAddToDatalog, 106
 - HTRCXBatteryLevel, 106
 - HTRCXClearAllEvents, 106
 - HTRCXClearCounter, 106
 - HTRCXClearMsg, 107
 - HTRCXClearSensor, 107
 - HTRCXClearSound, 107
 - HTRCXClearTimer, 107
 - HTRCXCreateDatalog, 107
 - HTRCXDecCounter, 108
 - HTRCXDeleteSub, 108
 - HTRCXDeleteSubs, 108
 - HTRCXDeleteTask, 108
 - HTRCXDeleteTasks, 108
 - HTRCXDisableOutput, 109
 - HTRCXEnableOutput, 109
 - HTRCXEvent, 109
 - HTRCXFloat, 109
 - HTRCXFwd, 110
 - HTRCXIncCounter, 110
 - HTRCXInvertOutput, 110
 - HTRCXMuteSound, 110
 - HTRCXObvertOutput, 110
 - HTRCXOff, 111
 - HTRCXOn, 111
 - HTRCXOnFor, 111
 - HTRCXOnFwd, 111
 - HTRCXOnRev, 112
 - HTRCXPBTurnOff, 112
 - HTRCXPing, 112
 - HTRCXPlaySound, 112
 - HTRCXPlayTone, 112
 - HTRCXPlayToneVar, 113
 - HTRCXPoll, 113
 - HTRCXPollMemory, 113
 - HTRCXRemote, 114
 - HTRCXRev, 114
 - HTRCXSelectDisplay, 114
 - HTRCXSelectProgram, 114
 - HTRCXSendSerial, 115
 - HTRCXSetDirection, 115
 - HTRCXSetEvent, 115
 - HTRCXSetGlobalDirection, 115
 - HTRCXSetGlobalOutput, 116
 - HTRCXSetIRLinkPort, 116
 - HTRCXSetMaxPower, 116
 - HTRCXSetMessage, 117
 - HTRCXSetOutput, 117
 - HTRCXSetPower, 117
 - HTRCXSetPriority, 117

HTRCXSetSensorMode, 118
HTRCXSetSensorType, 118
HTRCXSetSleepTime, 118
HTRCXSetTxPower, 118
HTRCXSetWatch, 119
HTRCXStartTask, 119
HTRCXStopAllTasks, 119
HTRCXStopTask, 119
HTRCXToggle, 120
HTRCXUnmuteSound, 120
HTScoutCalibrateSensor, 120
HTScoutMuteSound, 120
HTScoutSelectSounds, 120
HTScoutSendVLL, 120
HTScoutSetEventFeedback, 121
HTScoutSetLight, 121
HTScoutSetScoutMode, 121
HTScoutSetSensorClickTime, 121
HTScoutSetSensorHysteresis, 122
HTScoutSetSensorLowerLimit, 122
HTScoutSetSensorUpperLimit, 122
HTScoutUnmuteSound, 123
ReadSensorHTAccel, 123
ReadSensorHTAngle, 123
ReadSensorHTBarometric, 124
ReadSensorHTColor, 124
ReadSensorHTColor2Active, 124
ReadSensorHTColorNum, 125
ReadSensorHTCompass, 125
ReadSensorHTEOPD, 126
ReadSensorHTGyro, 126
ReadSensorHTIRReceiver, 126
ReadSensorHTIRReceiverEx, 127
ReadSensorHTIRSeeker, 127
ReadSensorHTIRSeeker2AC, 127
ReadSensorHTIRSeeker2Addr, 128
ReadSensorHTIRSeeker2DC, 128
ReadSensorHTIRSeekerDir, 129
ReadSensorHTMagnet, 129
ReadSensorHTNormalizedColor,
 130
ReadSensorHTNormalized-
 Color2Active, 130
ReadSensorHTProtoAllAnalog, 130
ReadSensorHTProtoAnalog, 131
ReadSensorHTProtoDigital, 131
ReadSensorHTProtoDigitalControl,
 132
ReadSensorHTRawColor, 132
ReadSensorHTRawColor2, 133
ReadSensorHTSuperProAllAnalog,
 133
ReadSensorHTSuperProAnalog, 134
ReadSensorHTSuperProAnalogOut,
 134
ReadSensorHTSuperProDigital, 135
ReadSensorHTSuperProDigitalCon-
 trol, 135
ReadSensorHTSuperProLED, 135
ReadSensorHTSuperProProgram-
 Control, 136
ReadSensorHTSuperProStrobe, 136
ReadSensorHTTouchMultiplexer,
 136
ResetHTBarometricCalibration, 137
ResetSensorHTAngle, 137
SetHTBarometricCalibration, 137
SetHTColor2Mode, 138
SetHTIRSeeker2Mode, 138
SetSensorHTEOPD, 138
SetSensorHTGyro, 139
SetSensorHTMagnet, 139
SetSensorHTProtoDigital, 139
SetSensorHTProtoDigitalControl,
 140
SetSensorHTSuperProAnalogOut,
 140
SetSensorHTSuperProDigital, 140
SetSensorHTSuperProDigitalCon-
 trol, 141
SetSensorHTSuperProLED, 141
SetSensorHTSuperProProgramCon-
 trol, 142
SetSensorHTSuperProStrobe, 142
HiTechnicConstants
 HT_ADDR_ACCEL, 691
 HT_ADDR_ANGLE, 691
 HT_ADDR_BAROMETRIC, 691
 HT_ADDR_COLOR, 691
 HT_ADDR_COLOR2, 692
 HT_ADDR_COMPASS, 692
 HT_ADDR_IRLINK, 692

HT_ADDR_IRRECEIVER, [692](#)
HT_ADDR_IRSEEKER, [692](#)
HT_ADDR_IRSEEKER2, [692](#)
HT_ADDR_PROTOBOARD, [692](#)
HT_ADDR_SUPERPRO, [692](#)
HS_ADDRESS_1
 CommHiSpeedAddressConstants,
 [632](#)
 NBCCCommon.h, [893](#)
HS_ADDRESS_2
 CommHiSpeedAddressConstants,
 [632](#)
 NBCCCommon.h, [893](#)
HS_ADDRESS_3
 CommHiSpeedAddressConstants,
 [632](#)
 NBCCCommon.h, [893](#)
HS_ADDRESS_4
 CommHiSpeedAddressConstants,
 [632](#)
 NBCCCommon.h, [893](#)
HS_ADDRESS_5
 CommHiSpeedAddressConstants,
 [632](#)
 NBCCCommon.h, [893](#)
HS_ADDRESS_6
 CommHiSpeedAddressConstants,
 [632](#)
 NBCCCommon.h, [893](#)
HS_ADDRESS_7
 CommHiSpeedAddressConstants,
 [633](#)
 NBCCCommon.h, [893](#)
HS_ADDRESS_8
 CommHiSpeedAddressConstants,
 [633](#)
 NBCCCommon.h, [893](#)
HS_ADDRESS_ALL
 CommHiSpeedAddressConstants,
 [633](#)
 NBCCCommon.h, [893](#)
HS_BAUD_115200
 CommHiSpeedBaudConstants, [625](#)
 NBCCCommon.h, [894](#)
HS_BAUD_1200
 CommHiSpeedBaudConstants, [625](#)
NBCCCommon.h, [894](#)
HS_BAUD_14400
 CommHiSpeedBaudConstants, [625](#)
 NBCCCommon.h, [894](#)
HS_BAUD_19200
 CommHiSpeedBaudConstants, [625](#)
 NBCCCommon.h, [894](#)
HS_BAUD_230400
 CommHiSpeedBaudConstants, [626](#)
 NBCCCommon.h, [894](#)
HS_BAUD_2400
 CommHiSpeedBaudConstants, [626](#)
 NBCCCommon.h, [894](#)
HS_BAUD_28800
 CommHiSpeedBaudConstants, [626](#)
 NBCCCommon.h, [894](#)
HS_BAUD_3600
 CommHiSpeedBaudConstants, [626](#)
 NBCCCommon.h, [894](#)
HS_BAUD_38400
 CommHiSpeedBaudConstants, [626](#)
 NBCCCommon.h, [894](#)
HS_BAUD_460800
 CommHiSpeedBaudConstants, [626](#)
 NBCCCommon.h, [894](#)
HS_BAUD_4800
 CommHiSpeedBaudConstants, [626](#)
 NBCCCommon.h, [895](#)
HS_BAUD_57600
 CommHiSpeedBaudConstants, [626](#)
 NBCCCommon.h, [895](#)
HS_BAUD_7200
 CommHiSpeedBaudConstants, [626](#)
 NBCCCommon.h, [895](#)
HS_BAUD_76800
 CommHiSpeedBaudConstants, [626](#)
 NBCCCommon.h, [895](#)
HS_BAUD_921600
 CommHiSpeedBaudConstants, [627](#)
 NBCCCommon.h, [895](#)
HS_BAUD_9600
 CommHiSpeedBaudConstants, [627](#)
 NBCCCommon.h, [895](#)
HS_BAUD_DEFAULT
 CommHiSpeedBaudConstants, [627](#)
 NBCCCommon.h, [895](#)

HS_BYTES_REMAINING
 CommHiSpeedStateConstants, 623
 NBCCommon.h, 895

HS_CMD_READY
 CommStatusCodesConstants, 637
 NBCCommon.h, 895

HS_CTRL_EXIT
 CommHiSpeedCtrlConstants, 624
 NBCCommon.h, 895

HS_CTRL_INIT
 CommHiSpeedCtrlConstants, 624
 NBCCommon.h, 896

HS_CTRL_UART
 CommHiSpeedCtrlConstants, 624
 NBCCommon.h, 896

HS_DEFAULT
 CommHiSpeedStateConstants, 623
 NBCCommon.h, 896

HS_DISABLE
 CommHiSpeedStateConstants, 623
 NBCCommon.h, 896

HS_ENABLE
 CommHiSpeedStateConstants, 623
 NBCCommon.h, 896

HS_INIT_RECEIVER
 CommHiSpeedStateConstants, 623
 NBCCommon.h, 896

HS_INITIALISE
 CommHiSpeedStateConstants, 623
 NBCCommon.h, 896

HS_MODE_10_STOP
 CommHiSpeedStopBitsConstants,
 630
 NBCCommon.h, 896

HS_MODE_15_STOP
 CommHiSpeedStopBitsConstants,
 630
 NBCCommon.h, 896

HS_MODE_20_STOP
 CommHiSpeedStopBitsConstants,
 630
 NBCCommon.h, 896

HS_MODE_5_DATA
 CommHiSpeedDataBitsConstants,
 629
 NBCCommon.h, 897

HS_MODE_6_DATA
 CommHiSpeedDataBitsConstants,
 629
 NBCCommon.h, 897

HS_MODE_7_DATA
 CommHiSpeedDataBitsConstants,
 629
 NBCCommon.h, 897

HS_MODE_7E1
 CommHiSpeedCombinedConstants,
 631
 NBCCommon.h, 897

HS_MODE_8_DATA
 CommHiSpeedDataBitsConstants,
 629
 NBCCommon.h, 897

HS_MODE_8N1
 CommHiSpeedCombinedConstants,
 631
 NBCCommon.h, 897

HS_MODE_DEFAULT
 CommHiSpeedModeConstants, 628
 NBCCommon.h, 897

HS_MODE_E_PARITY
 CommHiSpeedParityConstants, 630
 NBCCommon.h, 897

HS_MODE_M_PARITY
 CommHiSpeedParityConstants, 630
 NBCCommon.h, 897

HS_MODE_MASK
 CommHiSpeedModeConstants, 628
 NBCCommon.h, 898

HS_MODE_N_PARITY
 CommHiSpeedParityConstants, 630
 NBCCommon.h, 898

HS_MODE_O_PARITY
 CommHiSpeedParityConstants, 631
 NBCCommon.h, 898

HS_MODE_S_PARITY
 CommHiSpeedParityConstants, 631
 NBCCommon.h, 898

HS_MODE_UART_RS232
 CommHiSpeedModeConstants, 628
 NBCCommon.h, 898

HS_MODE_UART_RS485
 CommHiSpeedModeConstants, 628

NBCCCommon.h, 898
HS_SEND_DATA
 CommHiSpeedStateConstants, 624
 NBCCCommon.h, 898
HS_UART_MASK
 CommHiSpeedModeConstants, 628
 NBCCCommon.h, 898
HS_UPDATE
 CommHiSpeedFlagsConstants, 622
 NBCCCommon.h, 898
HT_ADDR_ACCEL
 HiTechnicConstants, 691
 NBCCCommon.h, 898
HT_ADDR_ANGLE
 HiTechnicConstants, 691
 NBCCCommon.h, 899
HT_ADDR_BAROMETRIC
 HiTechnicConstants, 691
 NBCCCommon.h, 899
HT_ADDR_COLOR
 HiTechnicConstants, 691
 NBCCCommon.h, 899
HT_ADDR_COLOR2
 HiTechnicConstants, 692
 NBCCCommon.h, 899
HT_ADDR_COMPASS
 HiTechnicConstants, 692
 NBCCCommon.h, 899
HT_ADDR_IRLINK
 HiTechnicConstants, 692
 NBCCCommon.h, 899
HT_ADDR_IRRECEIVER
 HiTechnicConstants, 692
 NBCCCommon.h, 899
HT_ADDR_IRSEEKER
 HiTechnicConstants, 692
 NBCCCommon.h, 899
HT_ADDR_IRSEEKER2
 HiTechnicConstants, 692
 NBCCCommon.h, 899
HT_ADDR_PROTOBOARD
 HiTechnicConstants, 692
 NBCCCommon.h, 899
HT_ADDR_SUPERPRO
 HiTechnicConstants, 692
 NBCCCommon.h, 900
HT_CH1_A
 HTIRReceiverConstants, 695
 NBCCCommon.h, 900
HT_CH1_B
 HTIRReceiverConstants, 695
 NBCCCommon.h, 900
HT_CH2_A
 HTIRReceiverConstants, 695
 NBCCCommon.h, 900
HT_CH2_B
 HTIRReceiverConstants, 695
 NBCCCommon.h, 900
HT_CH3_A
 HTIRReceiverConstants, 696
 NBCCCommon.h, 900
HT_CH3_B
 HTIRReceiverConstants, 696
 NBCCCommon.h, 900
HT_CH4_A
 HTIRReceiverConstants, 696
 NBCCCommon.h, 900
HT_CH4_B
 HTIRReceiverConstants, 696
 NBCCCommon.h, 900
HT_CMD_COLOR2_50HZ
 HTColor2Constants, 697
 NBCCCommon.h, 900
HT_CMD_COLOR2_60HZ
 HTColor2Constants, 697
 NBCCCommon.h, 901
HT_CMD_COLOR2_ACTIVE
 HTColor2Constants, 697
 NBCCCommon.h, 901
HT_CMD_COLOR2_BLCAL
 HTColor2Constants, 697
 NBCCCommon.h, 901
HT_CMD_COLOR2_FAR
 HTColor2Constants, 697
 NBCCCommon.h, 901
HT_CMD_COLOR2_LED_HI
 HTColor2Constants, 697
 NBCCCommon.h, 901
HT_CMD_COLOR2_LED_LOW
 HTColor2Constants, 697
 NBCCCommon.h, 901
HT_CMD_COLOR2_NEAR

HTColor2Constants, 697
NBCCCommon.h, 901

HT_CMD_COLOR2_PASSIVE
HTColor2Constants, 697
NBCCCommon.h, 901

HT_CMD_COLOR2_RAW
HTColor2Constants, 697
NBCCCommon.h, 901

HT_CMD_COLOR2_WBCAL
HTColor2Constants, 698
NBCCCommon.h, 901

HTANGLE_MODE_CALIBRATE
HTAngleConstants, 698
NBCCCommon.h, 902

HTANGLE_MODE_NORMAL
HTAngleConstants, 698
NBCCCommon.h, 902

HTANGLE_MODE_RESET
HTAngleConstants, 698
NBCCCommon.h, 902

HTANGLE_REG_ACDIR
HTAngleConstants, 699
NBCCCommon.h, 902

HTANGLE_REG_DC01
HTAngleConstants, 699
NBCCCommon.h, 902

HTANGLE_REG_DC02
HTAngleConstants, 699
NBCCCommon.h, 902

HTANGLE_REG_DC03
HTAngleConstants, 699
NBCCCommon.h, 902

HTANGLE_REG_DC04
HTAngleConstants, 699
NBCCCommon.h, 902

HTANGLE_REG_DC05
HTAngleConstants, 699
NBCCCommon.h, 902

HTANGLE_REG_DCAVG
HTAngleConstants, 699
NBCCCommon.h, 902

HTANGLE_REG_DCDIR
HTAngleConstants, 699
NBCCCommon.h, 903

HTANGLE_REG_MODE
HTAngleConstants, 699

NBCCCommon.h, 903

HTANGLE_MODE_CALIBRATE, 698

HTANGLE_MODE_NORMAL, 698

HTANGLE_MODE_RESET, 698

HTANGLE_REG_ACDIR, 699

HTANGLE_REG_DC01, 699

HTANGLE_REG_DC02, 699

HTANGLE_REG_DC03, 699

HTANGLE_REG_DC04, 699

HTANGLE_REG_DC05, 699

HTANGLE_REG_DCAVG, 699

HTANGLE_REG_DCDIR, 699

HTANGLE_REG_MODE, 699

HTBAR_REG_CALIBRATION
HTBarometricConstants, 700
NBCCCommon.h, 903

HTBAR_REG_COMMAND
HTBarometricConstants, 700
NBCCCommon.h, 903

HTBAR_REG_PRESSURE
HTBarometricConstants, 700
NBCCCommon.h, 903

HTBAR_REG_TEMPERATURE
HTBarometricConstants, 700
NBCCCommon.h, 903

HTBarometricConstants
HTBAR_REG_CALIBRATION, 700
HTBAR_REG_COMMAND, 700
HTBAR_REG_PRESSURE, 700
HTBAR_REG_TEMPERATURE, 700

HTColor2Constants
HT_CMD_COLOR2_50HZ, 697
HT_CMD_COLOR2_60HZ, 697
HT_CMD_COLOR2_ACTIVE, 697
HT_CMD_COLOR2_BLCAL, 697
HT_CMD_COLOR2_FAR, 697
HT_CMD_COLOR2_LED_HI, 697
HT_CMD_COLOR2_LED_LOW, 697
HT_CMD_COLOR2_NEAR, 697

HT_CMD_COLOR2_PASSIVE,
 [697](#)
HT_CMD_COLOR2_RAW, [697](#)
HT_CMD_COLOR2_WBCAL, [698](#)

HTIR2_MODE_1200
 HTIRSeeker2Constants, [693](#)
 NBCCommon.h, [903](#)

HTIR2_MODE_600
 HTIRSeeker2Constants, [693](#)
 NBCCommon.h, [903](#)

HTIR2_REG_AC01
 HTIRSeeker2Constants, [693](#)
 NBCCommon.h, [903](#)

HTIR2_REG_AC02
 HTIRSeeker2Constants, [693](#)
 NBCCommon.h, [903](#)

HTIR2_REG_AC03
 HTIRSeeker2Constants, [693](#)
 NBCCommon.h, [904](#)

HTIR2_REG_AC04
 HTIRSeeker2Constants, [694](#)
 NBCCommon.h, [904](#)

HTIR2_REG_AC05
 HTIRSeeker2Constants, [694](#)
 NBCCommon.h, [904](#)

HTIR2_REG_ACDIR
 HTIRSeeker2Constants, [694](#)
 NBCCommon.h, [904](#)

HTIR2_REG_DC01
 HTIRSeeker2Constants, [694](#)
 NBCCommon.h, [904](#)

HTIR2_REG_DC02
 HTIRSeeker2Constants, [694](#)
 NBCCommon.h, [904](#)

HTIR2_REG_DC03
 HTIRSeeker2Constants, [694](#)
 NBCCommon.h, [904](#)

HTIR2_REG_DC04
 HTIRSeeker2Constants, [694](#)
 NBCCommon.h, [904](#)

HTIR2_REG_DC05
 HTIRSeeker2Constants, [694](#)
 NBCCommon.h, [904](#)

HTIR2_REG_DCAVG
 HTIRSeeker2Constants, [694](#)
 NBCCommon.h, [904](#)

HTIR2_REG_DCDIR
 HTIRSeeker2Constants, [694](#)
 NBCCommon.h, [905](#)

HTIR2_REG_MODE
 HTIRSeeker2Constants, [695](#)
 NBCCommon.h, [905](#)

HTIRRReceiverConstants
 HT_CH1_A, [695](#)
 HT_CH1_B, [695](#)
 HT_CH2_A, [695](#)
 HT_CH2_B, [695](#)
 HT_CH3_A, [696](#)
 HT_CH3_B, [696](#)
 HT_CH4_A, [696](#)
 HT_CH4_B, [696](#)

HTIRSeeker2Constants
 HTIR2_MODE_1200, [693](#)
 HTIR2_MODE_600, [693](#)
 HTIR2_REG_AC01, [693](#)
 HTIR2_REG_AC02, [693](#)
 HTIR2_REG_AC03, [693](#)
 HTIR2_REG_AC04, [694](#)
 HTIR2_REG_AC05, [694](#)
 HTIR2_REG_ACDIR, [694](#)
 HTIR2_REG_DC01, [694](#)
 HTIR2_REG_DC02, [694](#)
 HTIR2_REG_DC03, [694](#)
 HTIR2_REG_DC04, [694](#)
 HTIR2_REG_DC05, [694](#)
 HTIR2_REG_DCAVG, [694](#)
 HTIR2_REG_DCDIR, [694](#)
 HTIR2_REG_MODE, [695](#)

HTIRTrain
 HiTechnicAPI, [101](#)

HTPFCComboDirect
 HiTechnicAPI, [101](#)

HTPFCComboPWM
 HiTechnicAPI, [101](#)

HTPFRawOutput
 HiTechnicAPI, [102](#)

HTPFRRepeat
 HiTechnicAPI, [102](#)

HTPPFSingleOutputCST
 HiTechnicAPI, [103](#)

HTPPFSingleOutputPWM
 HiTechnicAPI, [103](#)

HTPFSinglePin
 HiTechnicAPI, 104

HTPFTrain
 HiTechnicAPI, 105

HTPowerFunctionCommand
 HiTechnicAPI, 105

HTPROTO_A0
 HTProtoAnalogInputConstants, 703
 NBCCommon.h, 905

HTPROTO_A1
 HTProtoAnalogInputConstants, 703
 NBCCommon.h, 905

HTPROTO_A2
 HTProtoAnalogInputConstants, 703
 NBCCommon.h, 905

HTPROTO_A3
 HTProtoAnalogInputConstants, 703
 NBCCommon.h, 905

HTPROTO_A4
 HTProtoAnalogInputConstants, 703
 NBCCommon.h, 905

HTPROTO_REG_A0
 HTProtoConstants, 701
 NBCCommon.h, 905

HTPROTO_REG_A1
 HTProtoConstants, 701
 NBCCommon.h, 905

HTPROTO_REG_A2
 HTProtoConstants, 701
 NBCCommon.h, 905

HTPROTO_REG_A3
 HTProtoConstants, 701
 NBCCommon.h, 906

HTPROTO_REG_A4
 HTProtoConstants, 701
 NBCCommon.h, 906

HTPROTO_REG_DCTRL
 HTProtoConstants, 702
 NBCCommon.h, 906

HTPROTO_REG_DIN
 HTProtoConstants, 702
 NBCCommon.h, 906

HTPROTO_REG_DOUT
 HTProtoConstants, 702
 NBCCommon.h, 906

HTPROTO_REG_SRATE
 HTProtoConstants, 702
 NBCCommon.h, 906

HTProtoConstants, 702
NBCCommon.h, 906

HTProtoAnalogInputConstants
 HTPROTO_A0, 703
 HTPROTO_A1, 703
 HTPROTO_A2, 703
 HTPROTO_A3, 703
 HTPROTO_A4, 703

HTProtoConstants
 HTPROTO_REG_A0, 701
 HTPROTO_REG_A1, 701
 HTPROTO_REG_A2, 701
 HTPROTO_REG_A3, 701
 HTPROTO_REG_A4, 701
 HTPROTO_REG_DCTRL, 702
 HTPROTO_REG_DIN, 702
 HTPROTO_REG_DOUT, 702
 HTPROTO_REG_SRATE, 702

HTRCXAddToDatalog
 HiTechnicAPI, 106

HTRCXBatteryLevel
 HiTechnicAPI, 106

HTRCXClearAllEvents
 HiTechnicAPI, 106

HTRCXClearCounter
 HiTechnicAPI, 106

HTRCXClearMsg
 HiTechnicAPI, 107

HTRCXClearSensor
 HiTechnicAPI, 107

HTRCXClearSound
 HiTechnicAPI, 107

HTRCXClearTimer
 HiTechnicAPI, 107

HTRCXCreateDatalog
 HiTechnicAPI, 107

HTRCXDecCounter
 HiTechnicAPI, 108

HTRCXDeleteSub
 HiTechnicAPI, 108

HTRCXDeleteSubs
 HiTechnicAPI, 108

HTRCXDeleteTask
 HiTechnicAPI, 108

HTRCXDeleteTasks
 HiTechnicAPI, 108

HTRCXDisableOutput
 HiTechnicAPI, 109
HTRCXEnableOutput
 HiTechnicAPI, 109
HTRCXEvent
 HiTechnicAPI, 109
HTRCXFloat
 HiTechnicAPI, 109
HTRCXFwd
 HiTechnicAPI, 110
HTRCXIncCounter
 HiTechnicAPI, 110
HTRCXInvertOutput
 HiTechnicAPI, 110
HTRCXMuteSound
 HiTechnicAPI, 110
HTRCXObvertOutput
 HiTechnicAPI, 110
HTRCXOff
 HiTechnicAPI, 111
HTRCXOn
 HiTechnicAPI, 111
HTRCXOnFor
 HiTechnicAPI, 111
HTRCXOnFwd
 HiTechnicAPI, 111
HTRCXOnRev
 HiTechnicAPI, 112
HTRCXPBTurnOff
 HiTechnicAPI, 112
HTRCXPing
 HiTechnicAPI, 112
HTRCXPlaySound
 HiTechnicAPI, 112
HTRCXPlayTone
 HiTechnicAPI, 112
HTRCXPlayToneVar
 HiTechnicAPI, 113
HTRCXPoll
 HiTechnicAPI, 113
HTRCXPollMemory
 HiTechnicAPI, 113
HTRCXRemote
 HiTechnicAPI, 114
HTRCXRev
 HiTechnicAPI, 114
HTRCXSelectDisplay
 HiTechnicAPI, 114
HTRCXSelectProgram
 HiTechnicAPI, 114
HTRCXSendSerial
 HiTechnicAPI, 115
HTRCXSetDirection
 HiTechnicAPI, 115
HTRCXSetEvent
 HiTechnicAPI, 115
HTRCXSetGlobalDirection
 HiTechnicAPI, 115
HTRCXSetGlobalOutput
 HiTechnicAPI, 116
HTRCXSetIRLinkPort
 HiTechnicAPI, 116
HTRCXSetMaxPower
 HiTechnicAPI, 116
HTRCXSetMessage
 HiTechnicAPI, 117
HTRCXSetOutput
 HiTechnicAPI, 117
HTRCXSetPower
 HiTechnicAPI, 117
HTRCXSetPriority
 HiTechnicAPI, 117
HTRCXSetSensorMode
 HiTechnicAPI, 118
HTRCXSetSensorType
 HiTechnicAPI, 118
HTRCXSetSleepTime
 HiTechnicAPI, 118
HTRCXSetTxPower
 HiTechnicAPI, 118
HTRCXSetWatch
 HiTechnicAPI, 119
HTRCXStartTask
 HiTechnicAPI, 119
HTRCXStopAllTasks
 HiTechnicAPI, 119
HTRCXStopTask
 HiTechnicAPI, 119
HTRCXToggle
 HiTechnicAPI, 120
HTRCXUnmuteSound
 HiTechnicAPI, 120

HTScoutCalibrateSensor
 HiTechnicAPI, 120

HTScoutMuteSound
 HiTechnicAPI, 120

HTScoutSelectSounds
 HiTechnicAPI, 120

HTScoutSendVLL
 HiTechnicAPI, 120

HTScoutSetEventFeedback
 HiTechnicAPI, 121

HTScoutSetLight
 HiTechnicAPI, 121

HTScoutSetScoutMode
 HiTechnicAPI, 121

HTScoutSetSensorClickTime
 HiTechnicAPI, 121

HTScoutSetSensorHysteresis
 HiTechnicAPI, 122

HTScoutSetSensorLowerLimit
 HiTechnicAPI, 122

HTScoutSetSensorUpperLimit
 HiTechnicAPI, 122

HTScoutUnmuteSound
 HiTechnicAPI, 123

HTSPRO_A0
 HTSProAnalogInputConstants, 711
 NBCCCommon.h, 906

HTSPRO_A1
 HTSProAnalogInputConstants, 711
 NBCCCommon.h, 906

HTSPRO_A2
 HTSProAnalogInputConstants, 711
 NBCCCommon.h, 906

HTSPRO_A3
 HTSProAnalogInputConstants, 711
 NBCCCommon.h, 906

HTSPRO_DAC0
 HTSProDACIndexConstants, 712
 NBCCCommon.h, 907

HTSPRO_DAC1
 HTSProDACIndexConstants, 712
 NBCCCommon.h, 907

HTSPRO_REG_A0
 HTSuperProConstants, 705
 NBCCCommon.h, 907

HTSPRO_REG_A1
 HTSuperProConstants, 705
 NBCCCommon.h, 907

HTSPRO_REG_A2
 HTSuperProConstants, 705
 NBCCCommon.h, 907

HTSPRO_REG_A3
 HTSuperProConstants, 705
 NBCCCommon.h, 907

HTSPRO_REG_CTRL
 HTSuperProConstants, 705
 NBCCCommon.h, 907

HTSPRO_REG_DAC0_FREQ
 HTSuperProConstants, 706
 NBCCCommon.h, 907

HTSPRO_REG_DAC0_MODE
 HTSuperProConstants, 706
 NBCCCommon.h, 907

HTSPRO_REG_DAC0_VOLTAGE
 HTSuperProConstants, 706
 NBCCCommon.h, 907

HTSPRO_REG_DAC1_FREQ
 HTSuperProConstants, 706
 NBCCCommon.h, 908

HTSPRO_REG_DAC1_MODE
 HTSuperProConstants, 706
 NBCCCommon.h, 908

HTSPRO_REG_DAC1_VOLTAGE
 HTSuperProConstants, 706
 NBCCCommon.h, 908

HTSPRO_REG_DCTRL
 HTSuperProConstants, 706
 NBCCCommon.h, 908

HTSPRO_REG_DIN
 HTSuperProConstants, 706
 NBCCCommon.h, 908

HTSPRO_REG_DLADDRESS
 HTSuperProConstants, 706
 NBCCCommon.h, 908

HTSPRO_REG_DLCKSUM
 HTSuperProConstants, 706
 NBCCCommon.h, 908

HTSPRO_REG_DLCONTROL
 HTSuperProConstants, 707
 NBCCCommon.h, 908

HTSPRO_REG_DLDATA
 HTSuperProConstants, 707

NBCCCommon.h, 908
HTSPRO_REG_DOUT
 HTSuperProConstants, 707
 NBCCCommon.h, 908
HTSPRO_REG_LED
 HTSuperProConstants, 707
 NBCCCommon.h, 909
HTSPRO_REG_MEMORY_20
 HTSuperProConstants, 707
 NBCCCommon.h, 909
HTSPRO_REG_MEMORY_21
 HTSuperProConstants, 707
 NBCCCommon.h, 909
HTSPRO_REG_MEMORY_22
 HTSuperProConstants, 707
 NBCCCommon.h, 909
HTSPRO_REG_MEMORY_23
 HTSuperProConstants, 707
 NBCCCommon.h, 909
HTSPRO_REG_MEMORY_24
 HTSuperProConstants, 707
 NBCCCommon.h, 909
HTSPRO_REG_MEMORY_25
 HTSuperProConstants, 707
 NBCCCommon.h, 909
HTSPRO_REG_MEMORY_26
 HTSuperProConstants, 708
 NBCCCommon.h, 909
HTSPRO_REG_MEMORY_27
 HTSuperProConstants, 708
 NBCCCommon.h, 909
HTSPRO_REG_MEMORY_28
 HTSuperProConstants, 708
 NBCCCommon.h, 909
HTSPRO_REG_MEMORY_29
 HTSuperProConstants, 708
 NBCCCommon.h, 910
HTSPRO_REG_MEMORY_2A
 HTSuperProConstants, 708
 NBCCCommon.h, 910
HTSPRO_REG_MEMORY_2B
 HTSuperProConstants, 708
 NBCCCommon.h, 910
HTSPRO_REG_MEMORY_2C
 HTSuperProConstants, 708
 NBCCCommon.h, 910
HTSPRO_REG_MEMORY_2D
 HTSuperProConstants, 708
 NBCCCommon.h, 910
HTSPRO_REG_MEMORY_2E
 HTSuperProConstants, 708
 NBCCCommon.h, 910
HTSPRO_REG_MEMORY_2F
 HTSuperProConstants, 708
 NBCCCommon.h, 910
HTSPRO_REG_MEMORY_30
 HTSuperProConstants, 709
 NBCCCommon.h, 910
HTSPRO_REG_MEMORY_31
 HTSuperProConstants, 709
 NBCCCommon.h, 910
HTSPRO_REG_MEMORY_32
 HTSuperProConstants, 709
 NBCCCommon.h, 910
HTSPRO_REG_MEMORY_33
 HTSuperProConstants, 709
 NBCCCommon.h, 911
HTSPRO_REG_MEMORY_34
 HTSuperProConstants, 709
 NBCCCommon.h, 911
HTSPRO_REG_MEMORY_35
 HTSuperProConstants, 709
 NBCCCommon.h, 911
HTSPRO_REG_MEMORY_36
 HTSuperProConstants, 709
 NBCCCommon.h, 911
HTSPRO_REG_MEMORY_37
 HTSuperProConstants, 709
 NBCCCommon.h, 911
HTSPRO_REG_MEMORY_38
 HTSuperProConstants, 709
 NBCCCommon.h, 911
HTSPRO_REG_MEMORY_39
 HTSuperProConstants, 709
 NBCCCommon.h, 911
HTSPRO_REG_MEMORY_3A
 HTSuperProConstants, 710
 NBCCCommon.h, 911
HTSPRO_REG_MEMORY_3B
 HTSuperProConstants, 710
 NBCCCommon.h, 911
HTSPRO_REG_MEMORY_3C

HTSuperProConstants, 710
NBCCCommon.h, 911

HTSPRO_REG_MEMORY_3D
HTSuperProConstants, 710
NBCCCommon.h, 912

HTSPRO_REG_MEMORY_3E
HTSuperProConstants, 710
NBCCCommon.h, 912

HTSPRO_REG_MEMORY_3F
HTSuperProConstants, 710
NBCCCommon.h, 912

HTSPRO_REG_STROBE
HTSuperProConstants, 710
NBCCCommon.h, 912

HTSProAnalogInputConstants
HTSPRO_A0, 711
HTSPRO_A1, 711
HTSPRO_A2, 711
HTSPRO_A3, 711

HTSProDACIndexConstants
HTSPRO_DAC0, 712
HTSPRO_DAC1, 712

HTSuperProConstants
HTSPRO_REG_A0, 705
HTSPRO_REG_A1, 705
HTSPRO_REG_A2, 705
HTSPRO_REG_A3, 705
HTSPRO_REG_CTRL, 705
HTSPRO_REG_DAC0_FREQ, 706
HTSPRO_REG_DAC0_MODE, 706
HTSPRO_REG_DAC0_VOLTAGE, 706
HTSPRO_REG_DAC1_FREQ, 706
HTSPRO_REG_DAC1_MODE, 706
HTSPRO_REG_DAC1_VOLTAGE, 706
HTSPRO_REG_DCTRL, 706
HTSPRO_REG_DIN, 706
HTSPRO_REG_DLADDRESS, 706
HTSPRO_REG_DLCHKSUM, 706
HTSPRO_REG_DLCONTROL, 707
HTSPRO_REG_DLDATA, 707
HTSPRO_REG_DOUT, 707
HTSPRO_REG_LED, 707
HTSPRO_REG_MEMORY_20, 707
HTSPRO_REG_MEMORY_21, 707
HTSPRO_REG_MEMORY_22, 707
HTSPRO_REG_MEMORY_23, 707
HTSPRO_REG_MEMORY_24, 707
HTSPRO_REG_MEMORY_25, 707
HTSPRO_REG_MEMORY_26, 708
HTSPRO_REG_MEMORY_27, 708
HTSPRO_REG_MEMORY_28, 708
HTSPRO_REG_MEMORY_29, 708
HTSPRO_REG_MEMORY_2A, 708
HTSPRO_REG_MEMORY_2B, 708
HTSPRO_REG_MEMORY_2C, 708
HTSPRO_REG_MEMORY_2D, 708
HTSPRO_REG_MEMORY_2E, 708
HTSPRO_REG_MEMORY_2F, 708
HTSPRO_REG_MEMORY_30, 709
HTSPRO_REG_MEMORY_31, 709
HTSPRO_REG_MEMORY_32, 709
HTSPRO_REG_MEMORY_33, 709
HTSPRO_REG_MEMORY_34, 709
HTSPRO_REG_MEMORY_35, 709
HTSPRO_REG_MEMORY_36, 709
HTSPRO_REG_MEMORY_37, 709
HTSPRO_REG_MEMORY_38, 709
HTSPRO_REG_MEMORY_39, 709
HTSPRO_REG_MEMORY_3A, 710
HTSPRO_REG_MEMORY_3B, 710
HTSPRO_REG_MEMORY_3C, 710
HTSPRO_REG_MEMORY_3D, 710
HTSPRO_REG_MEMORY_3E, 710
HTSPRO_REG_MEMORY_3F, 710
HTSPRO_REG_STROBE, 710

I2C option constants, 594
I2C_ADDR_DEFAULT
GenericI2CConstants, 587
NBCCCommon.h, 912

I2C_OPTION_FAST
 I2COptionConstants, 595
 NBCCCommon.h, 912

I2C_OPTION_NORESTART
 I2COptionConstants, 595
 NBCCCommon.h, 912

I2C_OPTION_STANDARD
 I2COptionConstants, 595
 NBCCCommon.h, 912

I2C_REG_CMD
 GenericI2CConstants, 587
 NBCCCommon.h, 912

I2C_REG_DEVICE_ID
 GenericI2CConstants, 587
 NBCCCommon.h, 912

I2C_REG_VENDOR_ID
 GenericI2CConstants, 587
 NBCCCommon.h, 913

I2C_REG_VERSION
 GenericI2CConstants, 587
 NBCCCommon.h, 913

I2COptionConstants
 I2C_OPTION_FAST, 595
 I2C_OPTION_NORESTART, 595
 I2C_OPTION_STANDARD, 595

I2CSendCommand
 LowSpeedModuleFunctions, 315

IN_1
 NBCCCommon.h, 913
 NBCInputPortConstants, 544

IN_2
 NBCCCommon.h, 913
 NBCInputPortConstants, 544

IN_3
 NBCCCommon.h, 913
 NBCInputPortConstants, 544

IN_4
 NBCCCommon.h, 913
 NBCInputPortConstants, 544

IN_MODE_ANGLESTEP
 NBCCCommon.h, 913
 NBCSensorModeConstants, 548

IN_MODE_BOOLEAN
 NBCCCommon.h, 913
 NBCSensorModeConstants, 548

IN_MODE_CELSIUS
 NBCCCommon.h, 913
 NBCCCommon.h, 912

IN_MODE_FAHRENHEIT
 NBCCCommon.h, 913
 NBCSensorModeConstants, 548

IN_MODE_MODEMASK
 NBCCCommon.h, 914
 NBCSensorModeConstants, 548

IN_MODE_PCTFULLSCALE
 NBCCCommon.h, 914
 NBCSensorModeConstants, 548

IN_MODE_PERIODCOUNTER
 NBCCCommon.h, 914
 NBCSensorModeConstants, 548

IN_MODE_RAW
 NBCCCommon.h, 914
 NBCSensorModeConstants, 549

IN_MODE_SLOPEMASK
 NBCCCommon.h, 914
 NBCSensorModeConstants, 549

IN_MODE_TRANSITIONCNT
 NBCCCommon.h, 914
 NBCSensorModeConstants, 549

IN_TYPE_ANGLE
 NBCCCommon.h, 914
 NBCSensorTypeConstants, 545

IN_TYPE_COLORBLUE
 NBCCCommon.h, 914
 NBCSensorTypeConstants, 545

IN_TYPE_COLOREXIT
 NBCCCommon.h, 914
 NBCSensorTypeConstants, 545

IN_TYPE_COLORFULL
 NBCCCommon.h, 914
 NBCSensorTypeConstants, 546

IN_TYPE_COLORGREEN
 NBCCCommon.h, 915
 NBCSensorTypeConstants, 546

IN_TYPE_COLORNONE
 NBCCCommon.h, 915
 NBCSensorTypeConstants, 546

IN_TYPE_COLORRED
 NBCCCommon.h, 915
 NBCSensorTypeConstants, 546

IN_TYPE_CUSTOM
 NBCCCommon.h, 915

NBCSensorTypeConstants, 546
IN_TYPE_HISPEED
 NBCCommon.h, 915
 NBCSensorTypeConstants, 546
IN_TYPE_LIGHT_ACTIVE
 NBCCommon.h, 915
 NBCSensorTypeConstants, 546
IN_TYPE_LIGHT_INACTIVE
 NBCCommon.h, 915
 NBCSensorTypeConstants, 546
IN_TYPE_LOWSPEED
 NBCCommon.h, 915
 NBCSensorTypeConstants, 546
IN_TYPE_LOWSPEED_9V
 NBCCommon.h, 915
 NBCSensorTypeConstants, 546
IN_TYPE_NO_SENSOR
 NBCCommon.h, 915
 NBCSensorTypeConstants, 547
IN_TYPE_REFLECTION
 NBCCommon.h, 916
 NBCSensorTypeConstants, 547
IN_TYPE_SOUND_DB
 NBCCommon.h, 916
 NBCSensorTypeConstants, 547
IN_TYPE_SOUND_DBA
 NBCCommon.h, 916
 NBCSensorTypeConstants, 547
IN_TYPE_SWITCH
 NBCCommon.h, 916
 NBCSensorTypeConstants, 547
IN_TYPE_TEMPERATURE
 NBCCommon.h, 916
 NBCSensorTypeConstants, 547
Input field constants, 549
Input module, 75
Input module constants, 76
Input module functions, 296
Input module IOMAP offsets, 555
Input port digital pin constants, 550
INPUT_BLACKCOLOR
 InputColorValueConstants, 552
 NBCCommon.h, 916
INPUT_BLANK
 InputColorIdxConstants, 551
 NBCCommon.h, 916
INPUT_BLUE
 InputColorIdxConstants, 551
 NBCCommon.h, 916
INPUT_BLUECOLOR
 InputColorValueConstants, 552
 NBCCommon.h, 916
INPUT_CAL_POINT_0
 InputColorCalibrationConstants,
 554
 NBCCommon.h, 916
INPUT_CAL_POINT_1
 InputColorCalibrationConstants,
 554
 NBCCommon.h, 917
INPUT_CAL_POINT_2
 InputColorCalibrationConstants,
 554
 NBCCommon.h, 917
INPUT_CUSTOM9V
 InputModuleConstants, 77
 NBCCommon.h, 917
INPUT_CUSTOMACTIVE
 InputModuleConstants, 77
 NBCCommon.h, 917
INPUT_CUSTOMINACTIVE
 InputModuleConstants, 77
 NBCCommon.h, 917
INPUT_DIGI0
 InputDigiPinConstants, 550
 NBCCommon.h, 917
INPUT_DIGI1
 InputDigiPinConstants, 550
 NBCCommon.h, 917
INPUT_GREEN
 InputColorIdxConstants, 551
 NBCCommon.h, 917
INPUT_GREENCOLOR
 InputColorValueConstants, 552
 NBCCommon.h, 917
INPUT_INVALID_DATA
 InputModuleConstants, 77
 NBCCommon.h, 917
INPUT_NO_OF_COLORS
 InputColorIdxConstants, 551
 NBCCommon.h, 918
INPUT_NO_OF_POINTS

InputColorCalibrationConstants, 554
NBCCCommon.h, 918

INPUT_PINCMD_CLEAR
InputPinFuncConstants, 558
NBCCCommon.h, 918

INPUT_PINCMD_DIR
InputPinFuncConstants, 558
NBCCCommon.h, 918

INPUT_PINCMD_MASK
InputPinFuncConstants, 558
NBCCCommon.h, 918

INPUT_PINCMD_READ
InputPinFuncConstants, 558
NBCCCommon.h, 918

INPUT_PINCMD_SET
InputPinFuncConstants, 558
NBCCCommon.h, 918

INPUT_PINCMD_WAIT
InputPinFuncConstants, 558
NBCCCommon.h, 918

INPUT_PINDIR_INPUT
InputPinFuncConstants, 559
NBCCCommon.h, 918

INPUT_PINDIR_OUTPUT
InputPinFuncConstants, 559
NBCCCommon.h, 919

INPUT_RED
InputColorIdxConstants, 551
NBCCCommon.h, 919

INPUT_REDCOLOR
InputColorValueConstants, 552
NBCCCommon.h, 919

INPUT_RESETCAL
InputColorCalibrationStateConstants, 553
NBCCCommon.h, 919

INPUT_RUNNINGCAL
InputColorCalibrationStateConstants, 553
NBCCCommon.h, 919

INPUT_SENSORCAL
InputColorCalibrationStateConstants, 553
NBCCCommon.h, 919

INPUT_SENSOROFF

InputColorCalibrationStateConstants, 553
NBCCCommon.h, 919

INPUT_STARTCAL
InputColorCalibrationStateConstants, 554
NBCCCommon.h, 919

INPUT_WHITECOLOR
InputColorValueConstants, 552
NBCCCommon.h, 919

INPUT_YELLOWCOLOR
InputColorValueConstants, 553
NBCCCommon.h, 919

InputColorCalibrationConstants
INPUT_CAL_POINT_0, 554
INPUT_CAL_POINT_1, 554
INPUT_CAL_POINT_2, 554
INPUT_NO_OF_POINTS, 554

InputColorCalibrationStateConstants
INPUT_RESETCAL, 553
INPUT_RUNNINGCAL, 553
INPUT_SENSORCAL, 553
INPUT_SENSOROFF, 553
INPUT_STARTCAL, 554

InputColorIdxConstants
INPUT_BLANK, 551
INPUT_BLUE, 551
INPUT_GREEN, 551
INPUT_NO_OF_COLORS, 551
INPUT_RED, 551

InputColorValueConstants
INPUT_BLACKCOLOR, 552
INPUT_BLUECOLOR, 552
INPUT_GREENCOLOR, 552
INPUT_REDCOLOR, 552
INPUT_WHITECOLOR, 552
INPUT_YELLOWCOLOR, 553

InputDigiPinConstants
INPUT_DIGI0, 550
INPUT_DIGI1, 550

InputFieldConstants
InputModeField, 549
InvalidDataField, 549
NormalizedValueField, 550
RawValueField, 550
ScaledValueField, 550

TypeField, 550
InputIOMAP
 InputOffsetADRaw, 555
 InputOffsetColorADRaw, 555
 InputOffsetColorBoolean, 556
 InputOffsetColorCalibration, 556
 InputOffsetColorCalibrationState, 556
 InputOffsetColorCalLimits, 556
 InputOffsetColorSensorRaw, 556
 InputOffsetColorSensorValue, 556
 InputOffsetCustomActiveStatus, 556
 InputOffsetCustomPctFullScale, 556
 InputOffsetCustomZeroOffset, 556
 InputOffsetDigiPinsDir, 556
 InputOffsetDigiPinsIn, 557
 InputOffsetDigiPinsOut, 557
 InputOffsetInvalidData, 557
 InputOffsetSensorBoolean, 557
 InputOffsetSensorMode, 557
 InputOffsetSensorRaw, 557
 InputOffsetSensorType, 557
 InputOffsetSensorValue, 557
InputModeField
 InputFieldConstants, 549
 NBCCommon.h, 920
InputModuleConstants
 INPUT_CUSTOM9V, 77
 INPUT_CUSTOMACTIVE, 77
 INPUT_CUSTOMINACTIVE, 77
 INPUT_INVALID_DATA, 77
InputModuleFunctions
 ClearSensor, 300
 GetInColorADRaw, 300
 GetInColorBoolean, 300
 GetInColorCalibration, 301
 GetInColorCalibrationState, 301
 GetInColorCallLimits, 302
 GetInColorSensorRaw, 302
 GetInColorSensorValue, 302
 GetInCustomActiveStatus, 303
 GetInCustomPercentFullScale, 303
 GetInCustomZeroOffset, 303
 GetInDigiPinsDirection, 304
 GetInDigiPinsOutputLevel, 304
 GetInDigiPinsStatus, 304
 GetInSensorBoolean, 305
 ReadSensor, 305
 ReadSensorColorEx, 305
 ReadSensorColorRaw, 306
 ResetSensor, 306
 SetInCustomActiveStatus, 306
 SetInCustomPercentFullScale, 307
 SetInCustomZeroOffset, 307
 SetInDigiPinsDirection, 307
 SetInDigiPinsOutputLevel, 308
 SetInDigiPinsStatus, 308
 SetInSensorBoolean, 308
 SetSensorColorBlue, 308
 SetSensorColorFull, 309
 SetSensorColorGreen, 309
 SetSensorColorNone, 309
 SetSensorColorRed, 310
 SetSensorEMeter, 310
 SetSensorLight, 310
 SetSensorLowspeed, 311
 SetSensorMode, 311
 SetSensorSound, 311
 SetSensorTemperature, 311
 SetSensorTouch, 312
 SetSensorType, 312
 SetSensorUltrasonic, 312
InputModuleID
 ModuleIDConstants, 264
 NBCCommon.h, 920
InputModuleName
 ModuleNameConstants, 262
 NBCCommon.h, 920
InputOffsetADRaw
 InputIOMAP, 555
 NBCCommon.h, 920
InputOffsetColorADRaw
 InputIOMAP, 555
 NBCCommon.h, 920
InputOffsetColorBoolean
 InputIOMAP, 556
 NBCCommon.h, 920
InputOffsetColorCalibration
 InputIOMAP, 556
 NBCCommon.h, 920
InputOffsetColorCalibrationState
 InputIOMAP, 556

NBCCCommon.h, 920
InputOffsetColorCalLimits
 InputIOMAP, 556
 NBCCCommon.h, 920
InputOffsetColorSensorRaw
 InputIOMAP, 556
 NBCCCommon.h, 920
InputOffsetColorSensorValue
 InputIOMAP, 556
 NBCCCommon.h, 921
InputOffsetCustomActiveStatus
 InputIOMAP, 556
 NBCCCommon.h, 921
InputOffsetCustomPctFullScale
 InputIOMAP, 556
 NBCCCommon.h, 921
InputOffsetCustomZeroOffset
 InputIOMAP, 556
 NBCCCommon.h, 921
InputOffsetDigiPinsDir
 InputIOMAP, 556
 NBCCCommon.h, 921
InputOffsetDigiPinsIn
 InputIOMAP, 557
 NBCCCommon.h, 921
InputOffsetDigiPinsOut
 InputIOMAP, 557
 NBCCCommon.h, 921
InputOffsetInvalidData
 InputIOMAP, 557
 NBCCCommon.h, 921
InputOffsetSensorBoolean
 InputIOMAP, 557
 NBCCCommon.h, 921
InputOffsetSensorMode
 InputIOMAP, 557
 NBCCCommon.h, 921
InputOffsetSensorRaw
 InputIOMAP, 557
 NBCCCommon.h, 922
InputOffsetSensorType
 InputIOMAP, 557
 NBCCCommon.h, 922
InputOffsetSensorValue
 InputIOMAP, 557
 NBCCCommon.h, 922

InputPinFuncConstants
 INPUT_PINCMD_CLEAR, 558
 INPUT_PINCMD_DIR, 558
 INPUT_PINCMD_MASK, 558
 INPUT_PINCMD_READ, 558
 INPUT_PINCMD_SET, 558
 INPUT_PINCMD_WAIT, 558
 INPUT_PINDIR_INPUT, 559
 INPUT_PINDIR_OUTPUT, 559

InputPinFunction
 NBCCCommon.h, 922
 SysCallConstants, 484

INT_MAX
 NBCCCommon.h, 922
 NXTLimits, 785

INT_MIN
 NBCCCommon.h, 922
 NXTLimits, 785

INTF_BTOFF
 CommInterfaceConstants, 635
 NBCCCommon.h, 922

INTF_BTON
 CommInterfaceConstants, 635
 NBCCCommon.h, 922

INTF_CONNECT
 CommInterfaceConstants, 635
 NBCCCommon.h, 922

INTF_CONNECTBYNAME
 CommInterfaceConstants, 635
 NBCCCommon.h, 922

INTF_CONNECTREQ
 CommInterfaceConstants, 635
 NBCCCommon.h, 923

INTF_DISCONNECT
 CommInterfaceConstants, 635
 NBCCCommon.h, 923

INTF_DISCONNECTALL
 CommInterfaceConstants, 635
 NBCCCommon.h, 923

INTF_EXTREAD
 CommInterfaceConstants, 635
 NBCCCommon.h, 923

INTF_FACTORYRESET
 CommInterfaceConstants, 636
 NBCCCommon.h, 923

INTF_OPENSTREAM

CommInterfaceConstants, 636
NBCCCommon.h, 923

INTF_PINREQ
CommInterfaceConstants, 636
NBCCCommon.h, 923

INTF_REMOVEDevice
CommInterfaceConstants, 636
NBCCCommon.h, 923

INTF_SEARCH
CommInterfaceConstants, 636
NBCCCommon.h, 923

INTF_SENDDATA
CommInterfaceConstants, 636
NBCCCommon.h, 923

INTF_SENDFILE
CommInterfaceConstants, 636
NBCCCommon.h, 924

INTF_SETBTNAME
CommInterfaceConstants, 636
NBCCCommon.h, 924

INTF_SETCMDMODE
CommInterfaceConstants, 636
NBCCCommon.h, 924

INTF_STOPSEARCH
CommInterfaceConstants, 636
NBCCCommon.h, 924

INTF_VISIBILITY
CommInterfaceConstants, 637
NBCCCommon.h, 924

InvalidDataField
InputFieldConstants, 549
NBCCCommon.h, 924

IOCtrl module, 84

IOCtrl module constants, 505

IOCtrl module functions, 461

IOCtrl module IOMAP offsets, 506

IOCTRL_BOOT
IOCtrlPO, 506
NBCCCommon.h, 924

IOCTRL_POWERDOWN
IOCtrlPO, 506
NBCCCommon.h, 924

IOCrlIOMAP
IOCrlOffsetPowerOn, 506

IOCrlModuleFunctions
PowerDown, 461

RebootInFirmwareMode, 461

IOCrlModuleID
ModuleIDConstants, 264
NBCCCommon.h, 924

IOCrlModuleName
ModuleNameConstants, 262
NBCCCommon.h, 924

IOCrlOffsetPowerOn
IOCrlIOMAP, 506
NBCCCommon.h, 925

IOCrlPO
IOCTRL_BOOT, 506
IOCTRL_POWERDOWN, 506

IOMapRead
NBCCCommon.h, 925
SysCallConstants, 484

IOMapReadByID
NBCCCommon.h, 925
SysCallConstants, 484

IOMapWrite
NBCCCommon.h, 925
SysCallConstants, 484

IOMapWriteByID
NBCCCommon.h, 925
SysCallConstants, 484

IR Train channel constants, 683

IRTrainChannels
TRAIN_CHANNEL_1, 684
TRAIN_CHANNEL_2, 684
TRAIN_CHANNEL_3, 684
TRAIN_CHANNEL_ALL, 684

IRTrainFuncs
TRAIN_FUNC_DECR_SPEED, 683
TRAIN_FUNC_INCR_SPEED, 683
TRAIN_FUNC_STOP, 683
TRAIN_FUNC_TOGGLE_LIGHT, 683

KeepAlive
NBCCCommon.h, 925
SysCallConstants, 484

LCD_LINE1
LineConstants, 487
NBCCCommon.h, 925

LCD_LINE2
LineConstants, 487
NBCCCommon.h, 925

LCD_LINE3
LineConstants, 487
NBCCCommon.h, 925

LCD_LINE4
LineConstants, 487
NBCCCommon.h, 926

LCD_LINE5
LineConstants, 487
NBCCCommon.h, 926

LCD_LINE6
LineConstants, 487
NBCCCommon.h, 926

LCD_LINE7
LineConstants, 488
NBCCCommon.h, 926

LCD_LINE8
LineConstants, 488
NBCCCommon.h, 926

LDR_APPENDNOTPOSSIBLE
LoaderErrors, 509
NBCCCommon.h, 926

LDR_BTBUSY
LoaderErrors, 509
NBCCCommon.h, 926

LDR_BTCONNECTFAIL
LoaderErrors, 509
NBCCCommon.h, 926

LDR_BTTIMEOUT
LoaderErrors, 509
NBCCCommon.h, 926

LDR_CMD_BOOTCMD
LoaderFunctionConstants, 513
NBCCCommon.h, 926

LDR_CMD_BTFACTORYRESET
LoaderFunctionConstants, 513
NBCCCommon.h, 927

LDR_CMD_BTGETADR
LoaderFunctionConstants, 513
NBCCCommon.h, 927

LDR_CMD_CLOSE
LoaderFunctionConstants, 513
NBCCCommon.h, 927

LDR_CMD_CLOSEMODHANDLE
LoaderFunctionConstants, 514
NBCCCommon.h, 927

LDR_CMD_CROPDATAFILE
LoaderFunctionConstants, 514
NBCCCommon.h, 927

LDR_CMD_DELETE
LoaderFunctionConstants, 514
NBCCCommon.h, 927

LDR_CMD_DELETEUSERFLASH
LoaderFunctionConstants, 514
NBCCCommon.h, 927

LDR_CMD_DEVICEINFO
LoaderFunctionConstants, 514
NBCCCommon.h, 927

LDR_CMD_FINDFIRST
LoaderFunctionConstants, 514
NBCCCommon.h, 927

LDR_CMD_FINDFIRSTMODULE
LoaderFunctionConstants, 514
NBCCCommon.h, 927

LDR_CMD_FINDNEXT
LoaderFunctionConstants, 514
NBCCCommon.h, 928

LDR_CMD_FINDNEXTMODULE
LoaderFunctionConstants, 514
NBCCCommon.h, 928

LDR_CMD_IOMAPREAD
LoaderFunctionConstants, 514
NBCCCommon.h, 928

LDR_CMD_IOMAPWRITE
LoaderFunctionConstants, 515
NBCCCommon.h, 928

LDR_CMD_OPENAPPENDDATA
LoaderFunctionConstants, 515
NBCCCommon.h, 928

LDR_CMD_OPENREAD
LoaderFunctionConstants, 515
NBCCCommon.h, 928

LDR_CMD_OPENREADLINEAR
LoaderFunctionConstants, 515
NBCCCommon.h, 928

LDR_CMD_OPENWRITE
LoaderFunctionConstants, 515
NBCCCommon.h, 928

LDR_CMD_OPENWRITEDATA
LoaderFunctionConstants, 515

NBCCCommon.h, 928
LDR_CMD_OPENWITELINEAR
 LoaderFunctionConstants, 515
 NBCCCommon.h, 928
LDR_CMD_POLLCMD
 LoaderFunctionConstants, 515
 NBCCCommon.h, 929
LDR_CMD_POLLCMDLEN
 LoaderFunctionConstants, 515
 NBCCCommon.h, 929
LDR_CMD_READ
 LoaderFunctionConstants, 515
 NBCCCommon.h, 929
LDR_CMD_RENAMEFILE
 LoaderFunctionConstants, 516
 NBCCCommon.h, 929
LDR_CMD_RESIZEDATAFILE
 LoaderFunctionConstants, 516
 NBCCCommon.h, 929
LDR_CMD_SEEKFROMCURRENT
 LoaderFunctionConstants, 516
 NBCCCommon.h, 929
LDR_CMD_SEEKFROMEND
 LoaderFunctionConstants, 516
 NBCCCommon.h, 929
LDR_CMD_SEEKFROMSTART
 LoaderFunctionConstants, 516
 NBCCCommon.h, 929
LDR_CMD_SETBRICKNAME
 LoaderFunctionConstants, 516
 NBCCCommon.h, 929
LDR_CMD_VERSIONS
 LoaderFunctionConstants, 516
 NBCCCommon.h, 929
LDR_CMD_WRITE
 LoaderFunctionConstants, 516
 NBCCCommon.h, 930
LDR_ENDOFFILE
 LoaderErrors, 509
 NBCCCommon.h, 930
LDR_EOFEXPECTED
 LoaderErrors, 509
 NBCCCommon.h, 930
LDR_FILEEXISTS
 LoaderErrors, 510
 NBCCCommon.h, 930
LDR_FILEISBUSY
 LoaderErrors, 510
 NBCCCommon.h, 930
LDR_FILEISFULL
 LoaderErrors, 510
 NBCCCommon.h, 930
LDR_FILENOFOUND
 LoaderErrors, 510
 NBCCCommon.h, 930
LDR_FILETX_CLOSEERROR
 LoaderErrors, 510
 NBCCCommon.h, 930
LDR_FILETX_DSTEXISTS
 LoaderErrors, 510
 NBCCCommon.h, 930
LDR_FILETX_SRCMISSING
 LoaderErrors, 510
 NBCCCommon.h, 930
LDR_FILETX_STREAMERROR
 LoaderErrors, 510
 NBCCCommon.h, 931
LDR_FILETX_TIMEOUT
 LoaderErrors, 510
 NBCCCommon.h, 931
LDR_HANDLEALREADYCLOSED
 LoaderErrors, 510
 NBCCCommon.h, 931
LDR_ILLEGALFILENAME
 LoaderErrors, 511
 NBCCCommon.h, 931
LDR_ILLEGALHANDLE
 LoaderErrors, 511
 NBCCCommon.h, 931
LDR_INPROGRESS
 LoaderErrors, 511
 NBCCCommon.h, 931
LDR_INVALIDSEEK
 LoaderErrors, 511
 NBCCCommon.h, 931
LDR_MODULENOTFOUND
 LoaderErrors, 511
 NBCCCommon.h, 931
LDR_NOLINEARSPACE
 LoaderErrors, 511
 NBCCCommon.h, 931
LDR_NOMOREFILES

LoaderErrors, 511
 NBCCCommon.h, 931

LDR_NOMOREHANDLES
 LoaderErrors, 511
 NBCCCommon.h, 932

LDR_NOSPACE
 LoaderErrors, 511
 NBCCCommon.h, 932

LDR_NOTLINEARFILE
 LoaderErrors, 511
 NBCCCommon.h, 932

LDR_NOWRITEBUFFERS
 LoaderErrors, 512
 NBCCCommon.h, 932

LDR_OUTOFCBOUNDARY
 LoaderErrors, 512
 NBCCCommon.h, 932

LDR_REQPIN
 LoaderErrors, 512
 NBCCCommon.h, 932

LDR_SUCCESS
 LoaderErrors, 512
 NBCCCommon.h, 932

LDR_UNDEFINEDERROR
 LoaderErrors, 512
 NBCCCommon.h, 932

LED_BLUE
 LEDCtrlConstants, 144
 NBCCCommon.h, 932

LED_NONE
 LEDCtrlConstants, 144
 NBCCCommon.h, 932

LED_RED
 LEDCtrlConstants, 144
 NBCCCommon.h, 933

LEDCtrlConstants
 LED_BLUE, 144
 LED_NONE, 144
 LED_RED, 144

LEGO I2C address constants, 588

LEGO temperature sensor constants, 590

LEGO_ADDR_EMETER
 LEGOI2CAddressConstants, 588
 NBCCCommon.h, 933

LEGO_ADDR_TEMP
 LEGOI2CAddressConstants, 588

 NBCCCommon.h, 933

LEGO_ADDR_US
 LEGOI2CAddressConstants, 588
 NBCCCommon.h, 933

LEGOI2CAddressConstants
 LEGO_ADDR_EMETER, 588
 LEGO_ADDR_TEMP, 588
 LEGO_ADDR_US, 588

Line number constants, 486

LineConstants
 LCD_LINE1, 487
 LCD_LINE2, 487
 LCD_LINE3, 487
 LCD_LINE4, 487
 LCD_LINE5, 487
 LCD_LINE6, 487
 LCD_LINE7, 488
 LCD_LINE8, 488

LineOut
 DisplayModuleFunctions, 342

LineOutEx
 DisplayModuleFunctions, 342

ListFiles
 NBCCCommon.h, 933
 SysCallConstants, 484

Loader module, 84

Loader module constants, 507

Loader module error codes, 508

Loader module function constants, 512

Loader module functions, 461

Loader module IOMAP offsets, 507

LoaderErrors
 LDR_APPENDNOTPOSSIBLE, 509
 LDR_BTBUSY, 509
 LDR_BTCONNECTFAIL, 509
 LDR_BTTIMEOUT, 509
 LDR_ENDOFFILE, 509
 LDR_EOFEXPECTED, 509
 LDR_FILEEXISTS, 510
 LDR_FILEISBUSY, 510
 LDR_FILEISFULL, 510
 LDR_FILENOFOUND, 510
 LDR_FILETX_CLOSEERROR, 510
 LDR_FILETX_DSEXISTS, 510

LDR_FILETX_SRCMISSING, 510
LDR_FILETX_STREAMERROR,
 510
LDR_FILETX_TIMEOUT, 510
LDR_-
 HANDLEALREADYCLOSED,
 510
 LDR_ILLEGALFILENAME, 511
 LDR_ILLEGALHANDLE, 511
 LDR_INPROGRESS, 511
 LDR_INVALIDSEEK, 511
 LDR_MODULENOTFOUND, 511
 LDR_NOLINEARSPACE, 511
 LDR_NOMOREFILES, 511
 LDR_NOMOREHANDLES, 511
 LDR_NOSPACE, 511
 LDR_NOTLINEARFILE, 511
 LDR_NOWRITEBUFFERS, 512
 LDR_OUTOFBOUNDARY, 512
 LDR_REQPIN, 512
 LDR_SUCCESS, 512
 LDR_UNDEFINEDERROR, 512
LoaderExecuteFunction
 NBCCommon.h, 933
 SysCallConstants, 485
LoaderFunctionConstants
 LDR_CMD_BOOTCMD, 513
 LDR_CMD_BTFACTORYRESET,
 513
 LDR_CMD_BTGETADR, 513
 LDR_CMD_CLOSE, 513
 LDR_CMD_-
 CLOSEMODHANDLE,
 514
 LDR_CMD_CROPDATAFILE, 514
 LDR_CMD_DELETE, 514
 LDR_CMD_-
 DELETEUSERFLASH, 514
 LDR_CMD_DEVICEINFO, 514
 LDR_CMD_FINDFIRST, 514
 LDR_CMD_-
 FINDFIRSTMODULE, 514
 LDR_CMD_FINDNEXT, 514
 LDR_CMD_-
 FINDNEXTMODULE, 514
 LDR_CMD_IOMAPREAD, 514
LDR_CMD_IOMAPWRITE, 515
LDR_CMD_OPENAPPENDDATA,
 515
LDR_CMD_OPENREAD, 515
LDR_CMD_OPENREADLINEAR,
 515
LDR_CMD_OPENWRITE, 515
LDR_CMD_OPENWRITEDATA,
 515
LDR_CMD_-
 OPENWRITELINEAR, 515
LDR_CMD_POLLCMD, 515
LDR_CMD_POLLCMDLEN, 515
LDR_CMD_READ, 515
LDR_CMD_RENAMEFILE, 516
LDR_CMD_RESIZEDATAFILE,
 516
LDR_CMD_-
 SEEKFROMCURRENT,
 516
LDR_CMD_SEEKFROMEND, 516
LDR_CMD_SEEKFROMSTART,
 516
LDR_CMD_SETBRICKNAME,
 516
LDR_CMD_VERSIONS, 516
LDR_CMD_WRITE, 516
LoaderIOMAP
 LoaderOffsetFreeUserFlash, 508
 LoaderOffsetPFunc, 508
LoaderModuleConstants
 EOF, 507
 NULL, 507
LoaderModuleFunctions
 CloseFile, 464
 CreateFile, 464
 CreateFileLinear, 465
 CreateFileNonLinear, 465
 DeleteFile, 466
 FindFirstFile, 466
 FindNextFile, 466
 GetFreeMemory, 467
 OpenFileAppend, 467
 OpenFileRead, 467
 OpenFileReadLinear, 468
 Read, 468

ReadBytes, 469
 ReadLn, 469
 ReadLnString, 469
 RenameFile, 470
 ResizeFile, 470
 ResolveHandle, 470
 SizeOf, 471
 Write, 471
 WriteBytes, 471
 WriteBytesEx, 472
 WriteLn, 472
 WriteLnString, 473
 WriteString, 473

 LoaderModuleID
 ModuleIDConstants, 264
 NBCCommon.h, 933

 LoaderModuleName
 ModuleNameConstants, 263
 NBCCommon.h, 933

 LoaderOffsetFreeUserFlash
 LoaderIOMAP, 508
 NBCCommon.h, 933

 LoaderOffsetPFunc
 LoaderIOMAP, 508
 NBCCommon.h, 933

 LONG_MAX
 NBCCommon.h, 934
 NXTLimits, 785

 LONG_MIN
 NBCCommon.h, 934
 NXTLimits, 785

 Low level LowSpeed module functions, 324

 Low Speed module, 86

 Low speed module IOMAP offsets, 583

 LowLevelLowSpeedModuleFunctions
 GetLSCannelState, 325
 GetLSErrorType, 326
 GetLSInputBuffer, 326
 GetLSInputBufferBytesToRx, 326
 GetLSInputBufferInPtr, 326
 GetLSInputBufferOutPtr, 327
 GetLSMode, 327
 GetLSNoRestartOnRead, 327
 GetLSOutputBuffer, 327
 GetLSOutputBufferBytesToRx, 328

 GetLSOutputBufferInPtr, 328
 GetLSOutputBufferOutPtr, 328
 GetLSSpeed, 329
 GetLSState, 329

 LowSpeed module constants, 578

 LowSpeed module functions, 313

 LOWSPEED_CH_NOT_READY
 LowSpeedErrorTypeConstants, 583
 NBCCommon.h, 934

 LOWSPEED_COMMUNICATING
 LowSpeedChannelStateConstants, 581
 NBCCommon.h, 934

 LOWSPEED_DATA_RECEIVED
 LowSpeedModeConstants, 582
 NBCCommon.h, 934

 LOWSPEED_DONE
 LowSpeedChannelStateConstants, 581
 NBCCommon.h, 934

 LOWSPEED_ERROR
 LowSpeedChannelStateConstants, 581
 NBCCommon.h, 934

 LOWSPEED_IDLE
 LowSpeedChannelStateConstants, 581
 NBCCommon.h, 934

 LOWSPEED_INIT
 LowSpeedChannelStateConstants, 581
 NBCCommon.h, 934

 LOWSPEED_LOAD_BUFFER
 LowSpeedChannelStateConstants, 581
 NBCCommon.h, 934

 LOWSPEED_NO_ERROR
 LowSpeedErrorTypeConstants, 583
 NBCCommon.h, 935

 LOWSPEED RECEIVING
 LowSpeedModeConstants, 582
 NBCCommon.h, 935

 LOWSPEED_RX_ERROR
 LowSpeedErrorTypeConstants, 583
 NBCCommon.h, 935

 LOWSPEED_TRANSMITTING

LowSpeedModeConstants, 582
NBCCCommon.h, 935

LOWSPEED_TX_ERROR
LowSpeedErrorTypeConstants, 583
NBCCCommon.h, 935

LowspeedBytesReady
LowSpeedModuleFunctions, 316

LowSpeedChannelStateConstants
LOWSPEED_COMMUNICATING, 581
LOWSPEED_DONE, 581
LOWSPEED_ERROR, 581
LOWSPEED_IDLE, 581
LOWSPEED_INIT, 581
LOWSPEED_LOAD_BUFFER, 581

LowspeedCheckStatus
LowSpeedModuleFunctions, 316

LowSpeedErrorTypeConstants
LOWSPEED_CH_NOT_READY, 583
LOWSPEED_NO_ERROR, 583
LOWSPEED_RX_ERROR, 583
LOWSPEED_TX_ERROR, 583

LowSpeedIOMAP
LowSpeedOffsetChannelState, 584
LowSpeedOffsetErrorType, 584
LowSpeedOffsetInBufBuf, 584
LowSpeedOffsetInBufBytesToRx, 584
LowSpeedOffsetInBufInPtr, 584
LowSpeedOffsetInBufOutPtr, 584
LowSpeedOffsetMode, 584
LowSpeedOffsetNoRestartOnRead, 584
LowSpeedOffsetOutBufBuf, 585
LowSpeedOffsetOutBufBytesToRx, 585
LowSpeedOffsetOutBufInPtr, 585
LowSpeedOffsetOutBufOutPtr, 585
LowSpeedOffsetSpeed, 585
LowSpeedOffsetState, 585

LowSpeedModeConstants
LOWSPEED_DATA_RECEIVED, 582
LOWSPEED RECEIVING, 582

LOWSPEED_TRANSMITTING, 582

LowSpeedModuleFunctions
ConfigureTemperatureSensor, 315
I2CSendCommand, 315
LowspeedBytesReady, 316
LowspeedCheckStatus, 316
LowspeedRead, 317
LowspeedStatus, 317
LowspeedWrite, 318
ReadI2CBytes, 319
ReadI2CDeviceId, 319
ReadI2CDeviceInfo, 320
ReadI2CRegister, 320
ReadI2CVendorId, 321
ReadI2CVersion, 321
ReadSensorEMeter, 321
ReadSensorTemperature, 322
ReadSensorUS, 322
ReadSensorUSEx, 323
SetI2COptions, 323
WriteI2CRegister, 323

LowSpeedModuleID
ModuleIDConstants, 264
NBCCCommon.h, 935

LowSpeedModuleName
ModuleNameConstants, 263
NBCCCommon.h, 935

LowSpeedNoRestartConstants
LSREAD_NO_RESTART_1, 586
LSREAD_NO_RESTART_2, 586
LSREAD_NO_RESTART_3, 586
LSREAD_NO_RESTART_4, 586
LSREAD_NO_RESTART_MASK, 586
LSREAD_RESTART_ALL, 586
LSREAD_RESTART_NONE, 586

LowSpeedOffsetChannelState
LowSpeedIOMAP, 584
NBCCCommon.h, 935

LowSpeedOffsetErrorType
LowSpeedIOMAP, 584
NBCCCommon.h, 935

LowSpeedOffsetInBufBuf
LowSpeedIOMAP, 584
NBCCCommon.h, 935

LowSpeedOffsetInBufBytesToRx
 LowSpeedIOMAP, 584
 NBCCommon.h, 936

LowSpeedOffsetInBufInPtr
 LowSpeedIOMAP, 584
 NBCCommon.h, 936

LowSpeedOffsetInBufOutPtr
 LowSpeedIOMAP, 584
 NBCCommon.h, 936

LowSpeedOffsetMode
 LowSpeedIOMAP, 584
 NBCCommon.h, 936

LowSpeedOffsetNoRestartOnRead
 LowSpeedIOMAP, 584
 NBCCommon.h, 936

LowSpeedOffsetOutBufBuf
 LowSpeedIOMAP, 585
 NBCCommon.h, 936

LowSpeedOffsetOutBufBytesToRx
 LowSpeedIOMAP, 585
 NBCCommon.h, 936

LowSpeedOffsetOutBufInPtr
 LowSpeedIOMAP, 585
 NBCCommon.h, 936

LowSpeedOffsetOutBufOutPtr
 LowSpeedIOMAP, 585
 NBCCommon.h, 936

LowSpeedOffsetSpeed
 LowSpeedIOMAP, 585
 NBCCommon.h, 936

LowSpeedOffsetState
 LowSpeedIOMAP, 585
 NBCCommon.h, 937

LowspeedRead
 LowSpeedModuleFunctions, 317

LowSpeedStateConstants
 COM_CHANNEL_FOUR_-
 ACTIVE, 580
 COM_CHANNEL_NONE_-
 ACTIVE, 580
 COM_CHANNEL_ONE_ACTIVE,
 580
 COM_CHANNEL_THREE_-
 ACTIVE, 580
 COM_CHANNEL_TWO_ACTIVE,
 580

LowspeedStatus
 LowSpeedModuleFunctions, 317

LowspeedWrite
 LowSpeedModuleFunctions, 318

LR_COULD_NOT_SAVE
 CommStatusCodesConstants, 637
 NBCCommon.h, 937

LR_ENTRY_REMOVED
 CommStatusCodesConstants, 637
 NBCCommon.h, 937

LR_STORE_IS_FULL
 CommStatusCodesConstants, 638
 NBCCommon.h, 937

LR_SUCCESS
 CommStatusCodesConstants, 638
 NBCCommon.h, 937

LR_UNKNOWN_ADDR
 CommStatusCodesConstants, 638
 NBCCommon.h, 937

LSChannelState constants, 580

LSErrorType constants, 582

LSMode constants, 581

LSNoRestartOnRead constants, 585

LSREAD_NO_RESTART_1
 LowSpeedNoRestartConstants, 586
 NBCCommon.h, 937

LSREAD_NO_RESTART_2
 LowSpeedNoRestartConstants, 586
 NBCCommon.h, 937

LSREAD_NO_RESTART_3
 LowSpeedNoRestartConstants, 586
 NBCCommon.h, 937

LSREAD_NO_RESTART_4
 LowSpeedNoRestartConstants, 586
 NBCCommon.h, 937

LSREAD_NO_RESTART_MASK
 LowSpeedNoRestartConstants, 586
 NBCCommon.h, 938

LSREAD_RESTART_ALL
 LowSpeedNoRestartConstants, 586
 NBCCommon.h, 938

LSREAD_RESTART_NONE
 LowSpeedNoRestartConstants, 586
 NBCCommon.h, 938

LSState constants, 579

LT

cmpconst, 74
LTEQ
 cmpconst, 74

Mailbox constants, 494
MAILBOX1
 MailboxConstants, 495
 NBCCCommon.h, 938
MAILBOX10
 MailboxConstants, 495
 NBCCCommon.h, 938
MAILBOX2
 MailboxConstants, 495
 NBCCCommon.h, 938
MAILBOX3
 MailboxConstants, 495
 NBCCCommon.h, 938
MAILBOX4
 MailboxConstants, 495
 NBCCCommon.h, 938
MAILBOX5
 MailboxConstants, 495
 NBCCCommon.h, 938
MAILBOX6
 MailboxConstants, 495
 NBCCCommon.h, 938
MAILBOX7
 MailboxConstants, 495
 NBCCCommon.h, 939
MAILBOX8
 MailboxConstants, 495
 NBCCCommon.h, 939
MAILBOX9
 MailboxConstants, 495
 NBCCCommon.h, 939
MailboxConstants
 MAILBOX1, 495
 MAILBOX10, 495
 MAILBOX2, 495
 MAILBOX3, 495
 MAILBOX4, 495
 MAILBOX5, 495
 MAILBOX6, 495
 MAILBOX7, 495
 MAILBOX8, 495
 MAILBOX9, 495

 MAX_BT_MSG_SIZE
 CommMiscConstants, 614
 NBCCCommon.h, 939
 MaxAccelerationField
 NBCCCommon.h, 939
 OutputFieldConstants, 571
 MaxSpeedField
 NBCCCommon.h, 939
 OutputFieldConstants, 571
 MemoryManager
 NBCCCommon.h, 939
 SysCallConstants, 485
MENUICON_CENTER
 DisplayModuleConstants, 598
 NBCCCommon.h, 939
MENUICON_LEFT
 DisplayModuleConstants, 598
 NBCCCommon.h, 939
MENUICON_RIGHT
 DisplayModuleConstants, 598
 NBCCCommon.h, 940
MENUICONS
 DisplayModuleConstants, 598
 NBCCCommon.h, 940
MENUTEXT
 DisplayModuleConstants, 598
 NBCCCommon.h, 940
MessageRead
 NBCCCommon.h, 940
 SysCallConstants, 485
MessageWrite
 NBCCCommon.h, 940
 SysCallConstants, 485
MI_ADDR_XG1300L
 NBCCCommon.h, 940
 XG1300LConstants, 782
Microinfinity API Functions, 251
Microinfinity CruizCore XG1300L, 783
Microinfinity CruizCore XG1300L sensor
 constants, 782
Microinfinity device constants, 781
MicroinfinityAPI
 ReadSensorMIXG1300L, 252
 ReadSensorMIXG1300LScale, 252
 ResetMIXG1300L, 252
 SetSensorMIXG1300LScale, 253

MIN_1
 NBCCCommon.h, 940
 TimeConstants, 489

MindSensors ACCL-Nx constants, 724

MindSensors ACCL-Nx sensitivity level constants, 726

MindSensors API Functions, 147

MindSensors device constants, 712

MindSensors DIST-Nx constants, 715

MindSensors nRLink constants, 722

MindSensors NXTHID commands, 742

MindSensors NXTHID constants, 739

MindSensors NXTHID modifier keys, 741

MindSensors NXTHID registers, 740

MindSensors NXTLineLeader commands, 749

MindSensors NXTLineLeader constants, 747

MindSensors NXTLineLeader registers, 747

MindSensors NXTPowerMeter commands, 745

MindSensors NXTPowerMeter constants, 743

MindSensors NXTPowerMeter registers, 743

MindSensors NXTServo commands, 738

MindSensors NXTServo constants, 730

MindSensors NXTServo position constants, 735

MindSensors NXTServo quick position constants, 736

MindSensors NXTServo registers, 731

MindSensors NXTServo servo numbers, 736

MindSensors NXTSumoEyes constants, 746

MindSensors PFMate constants, 727

MindSensors PSP-Nx button set 1 constants, 719

MindSensors PSP-Nx button set 2 constants, 720

MindSensors PSP-Nx constants, 717

MindSensorsAPI
 ACCLNxCalibrateX, 167

ACCLNxCalibrateXEnd, 167

ACCLNxCalibrateY, 167

ACCLNxCalibrateYEnd, 168

ACCLNxCalibrateZ, 168

ACCLNxCalibrateZEnd, 168

ACCLNxResetCalibration, 169

DISTNxGP2D12, 169

DISTNxGP2D120, 169

DISTNxGP2YA02, 170

DISTNxGP2YA21, 170

MSADPAOff, 170

MSADPAOn, 171

MSDeenergize, 171

MSEnergize, 171

MSIRTrain, 172

MSPFComboDirect, 172

MSPFComboPWM, 173

MSPFRawOutput, 173

MSPFRepeat, 174

MSPFSingleOutputCST, 174

MSPFSingleOutputPWM, 175

MSPFSinglePin, 176

MSPFTrain, 176

MSRCXAbsVar, 177

MSRCXAddToDatalog, 177

MSRCXAndVar, 177

MSRCXBatteryLevel, 178

MSRCXBoot, 178

MSRCXCalibrateEvent, 178

MSRCXClearAllEvents, 178

MSRCXClearCounter, 179

MSRCXClearMsg, 179

MSRCXClearSensor, 179

MSRCXClearSound, 179

MSRCXClearTimer, 179

MSRCXCreateDatalog, 180

MSRCXDecCounter, 180

MSRCXDeleteSub, 180

MSRCXDeleteSubs, 180

MSRCXDeleteTask, 180

MSRCXDeleteTasks, 181

MSRCXDisableOutput, 181

MSRCXDivVar, 181

MSRCXEnableOutput, 181

MSRCXEvent, 182

MSRCXFloat, 182

MSRCXFwd, 182
MSRCXIncCounter, 182
MSRCXIvertOutput, 183
MSRCXMulVar, 183
MSRCXMuteSound, 183
MSRCXObvertOutput, 183
MSRCXOff, 184
MSRCXOn, 184
MSRCXOnFor, 184
MSRCXOnFwd, 184
MSRCXOnRev, 185
MSRCXOrVar, 185
MSRCXPBTurnOff, 185
MSRCXPing, 185
MSRCXPlaySound, 186
MSRCXPlayTone, 186
MSRCXPlayToneVar, 186
MSRCXPoll, 186
MSRCXPollMemory, 187
MSRCXRemote, 187
MSRCXReset, 187
MSRCXRev, 187
MSRCXSelectDisplay, 188
MSRCXSelectProgram, 188
MSRCXSendSerial, 188
MSRCXSet, 188
MSRCXSetDirection, 189
MSRCXSetEvent, 189
MSRCXSetGlobalDirection, 189
MSRCXSetGlobalOutput, 190
MSRCXSetMaxPower, 190
MSRCXSetMessage, 190
MSRCXSetNRLinkPort, 191
MSRCXSetOutput, 191
MSRCXSetPower, 191
MSRCXSetPriority, 191
MSRCXSetSensorMode, 192
MSRCXSetSensorType, 192
MSRCXSetSleepTime, 192
MSRCXSetTxPower, 192
MSRCXSetUserDisplay, 193
MSRCXSetVar, 193
MSRCXSetWatch, 193
MSRCXSgnVar, 194
MSRCXStartTask, 194
MSRCXStopAllTasks, 194
MSRCXStopTask, 194
MSRCXSubVar, 194
MSRCXSumVar, 195
MSRCXToggle, 195
MSRCXUnlock, 195
MSRCXUnmuteSound, 195
MSReadValue, 196
MSScoutCalibrateSensor, 196
MSScoutMuteSound, 196
MSScoutSelectSounds, 196
MSScoutSendVLL, 197
MSScoutSetCounterLimit, 197
MSScoutSetEventFeedback, 197
MSScoutSetLight, 197
MSScoutSetScoutMode, 198
MSScoutSetScoutRules, 198
MSScoutSetSensorClickTime, 198
MSScoutSetSensorHysteresis, 199
MSScoutSetSensorLowerLimit, 199
MSScoutSetSensorUpperLimit, 199
MSScoutSetTimerLimit, 199
MSScoutUnmuteSound, 200
NRLink2400, 200
NRLink4800, 200
NRLinkFlush, 201
NRLinkIRLong, 201
NRLinkIRShort, 201
NRLinkSetPF, 202
NRLinkSetRCX, 202
NRLinkSetTrain, 202
NRLinkTxRaw, 203
NXTHIDAsciiMode, 203
NXTHIDDirectMode, 203
NXTHIDLoadCharacter, 204
NXTHIDTransmit, 204
NXTLineLeaderCalibrateBlack, 204
NXTLineLeaderCalibrateWhite, 205
NXTLineLeaderInvert, 205
NXTLineLeaderPowerDown, 206
NXTLineLeaderPowerUp, 206
NXTLineLeaderReset, 206
NXTLineLeaderSnapshot, 207
NXTPowerMeterResetCounters,
 207
NXTServoEditMacro, 208
NXTServoGotoMacroAddress, 208

NXTServoHaltMacro, 208
NXTServoInit, 209
NXTServoPauseMacro, 209
NXTServoQuitEdit, 210
NXTServoReset, 210
NXTServoResumeMacro, 210
PFMateSend, 211
PFMateSendRaw, 211
PSPNxAnalog, 212
PSPNxDigital, 212
ReadACCLNxSensitivity, 213
ReadACCLNxXOffset, 213
ReadACCLNxXRange, 213
ReadACCLNxYOffset, 214
ReadACCLNxYRange, 214
ReadACCLNxZOffset, 215
ReadACCLNxZRange, 215
ReadDISTNxDistance, 215
ReadDISTNxMaxDistance, 216
ReadDISTNxMinDistance, 216
ReadDISTNxModuleType, 216
ReadDISTNxNumPoints, 217
ReadDISTNxVoltage, 217
ReadNRLLinkBytes, 218
ReadNRLLinkStatus, 218
ReadNXTLineLeaderAverage, 218
ReadNXTLineLeaderResult, 219
ReadNXTLineLeaderSteering, 219
ReadNXTPowerMeterCapacityUsed, 220
ReadNXTPowerMeterElapsed Time, 220
ReadNXTPowerMeterErrorCount, 220
ReadNXTPowerMeterMaxCurrent, 221
ReadNXTPowerMeterMaxVoltage, 221
ReadNXTPowerMeterMinCurrent, 222
ReadNXTPowerMeterMinVoltage, 222
ReadNXTPowerMeterPresentCurrent, 222
ReadNXTPowerMeterPresentPower, 223
ReadNXTPowerMeterPresentVoltage, 223
ReadNXTPowerMeterTotalPowerConsumed, 224
ReadNXTServoBatteryVoltage, 224
ReadNXTServoPosition, 224
ReadNXTServoSpeed, 225
ReadSensorMSAccel, 225
ReadSensorMSCompass, 226
ReadSensorMSDROD, 226
ReadSensorMSPlayStation, 226
ReadSensorMSPressure, 227
ReadSensorMSPressureRaw, 227
ReadSensorMSRTClock, 228
ReadSensorMSTilt, 228
ReadSensorNXTSumoEyes, 229
RunNRLLinkMacro, 229
SetACCLNxSensitivity, 229
SetNXTLineLeaderKdFactor, 230
SetNXTLineLeaderKdValue, 230
SetNXTLineLeaderKiFactor, 231
SetNXTLineLeaderKiValue, 231
SetNXTLineLeaderKpFactor, 231
SetNXTLineLeaderKpValue, 232
SetNXTLineLeaderSetpoint, 232
SetNXTServoPosition, 233
SetNXTServoQuickPosition, 233
SetNXTServoSpeed, 234
SetSensorMSDRODActive, 234
SetSensorMSDRODInactive, 234
SetSensorMSPressure, 235
SetSensorMSTouchMux, 235
SetSensorNXTSumoEyesLong, 235
SetSensorNXTSumoEyesShort, 235
WriteNRLLinkBytes, 236

MindSensorsConstants

MS_ADDR_ACCLNX, 713
MS_ADDR_CMPSNX, 713
MS_ADDR_DISTNX, 714
MS_ADDR_IVSENS, 714
MS_ADDR_LINELDR, 714
MS_ADDR_MTRMUX, 714
MS_ADDR_NRLINK, 714
MS_ADDR_NXTCAM, 714
MS_ADDR_NXTHID, 714
MS_ADDR_NXTMMX, 714

MS_ADDR_NXTSERVO, 714
MS_ADDR_NXTSERVO_EM, 714
MS_ADDR_PFMATE, 715
MS_ADDR_PSPNX, 715
MS_ADDR_RTCLOCK, 715
MS_ADDR_RXMUX, 715
MS_CMD_ADPA_OFF, 715
MS_CMD_ADPA_ON, 715
MS_CMD_DEENERGIZED, 715
MS_CMD_ENERGIZED, 715
MiscConstants
 DEGREES_PER_RADIAN, 266
 FALSE, 266
 NA, 266
 PI, 266
 RADIANs_PER_DEGREE, 266
 TRUE, 266
Miscellaneous Comm module constants, 614
Miscellaneous NBC/NXC constants, 265
ModuleIDConstants
 ButtonModuleID, 264
 CommandModuleID, 264
 CommModuleID, 264
 DisplayModuleID, 264
 InputModuleID, 264
 IOCtrlModuleID, 264
 LoaderModuleID, 264
 LowSpeedModuleID, 264
 OutputModuleID, 264
 SoundModuleID, 265
 UIModuleID, 265
ModuleNameConstants
 ButtonModuleName, 262
 CommandModuleName, 262
 CommModuleName, 262
 DisplayModuleName, 262
 InputModuleName, 262
 IOCtrlModuleName, 262
 LoaderModuleName, 263
 LowSpeedModuleName, 263
 OutputModuleName, 263
 SoundModuleName, 263
 UIModuleName, 263
MS_1
 NBCCCommon.h, 940
MS_10
 TimeConstants, 489
 NBCCCommon.h, 940
 TimeConstants, 489
MS_100
 NBCCCommon.h, 940
 TimeConstants, 490
MS_150
 NBCCCommon.h, 941
 TimeConstants, 490
MS_2
 NBCCCommon.h, 941
 TimeConstants, 490
MS_20
 NBCCCommon.h, 941
 TimeConstants, 490
MS_200
 NBCCCommon.h, 941
 TimeConstants, 490
MS_250
 NBCCCommon.h, 941
 TimeConstants, 490
MS_3
 NBCCCommon.h, 941
 TimeConstants, 490
MS_30
 NBCCCommon.h, 941
 TimeConstants, 490
MS_300
 NBCCCommon.h, 941
 TimeConstants, 490
MS_350
 NBCCCommon.h, 941
 TimeConstants, 490
MS_4
 NBCCCommon.h, 941
 TimeConstants, 491
MS_40
 NBCCCommon.h, 942
 TimeConstants, 491
MS_400
 NBCCCommon.h, 942
 TimeConstants, 491
MS_450
 NBCCCommon.h, 942
 TimeConstants, 491

MS_5
 NBCCCommon.h, 942
 TimeConstants, 491
MS_50
 NBCCCommon.h, 942
 TimeConstants, 491
MS_500
 NBCCCommon.h, 942
 TimeConstants, 491
MS_6
 NBCCCommon.h, 942
 TimeConstants, 491
MS_60
 NBCCCommon.h, 942
 TimeConstants, 491
MS_600
 NBCCCommon.h, 942
 TimeConstants, 491
MS_7
 NBCCCommon.h, 942
 TimeConstants, 492
MS_70
 NBCCCommon.h, 943
 TimeConstants, 492
MS_700
 NBCCCommon.h, 943
 TimeConstants, 492
MS_8
 NBCCCommon.h, 943
 TimeConstants, 492
MS_80
 NBCCCommon.h, 943
 TimeConstants, 492
MS_800
 NBCCCommon.h, 943
 TimeConstants, 492
MS_9
 NBCCCommon.h, 943
 TimeConstants, 492
MS_90
 NBCCCommon.h, 943
 TimeConstants, 492
MS_900
 NBCCCommon.h, 943
 TimeConstants, 492
MS_ADDR_ACCLNX
 MindSensorsConstants, 713
 NBCCCommon.h, 943
MS_ADDR_CMPSNX
 MindSensorsConstants, 713
 NBCCCommon.h, 943
MS_ADDR_DISTNX
 MindSensorsConstants, 714
 NBCCCommon.h, 944
MS_ADDR_IVSENS
 MindSensorsConstants, 714
 NBCCCommon.h, 944
MS_ADDR_LINELDR
 MindSensorsConstants, 714
 NBCCCommon.h, 944
MS_ADDR_MTRMUX
 MindSensorsConstants, 714
 NBCCCommon.h, 944
MS_ADDR_NRLINK
 MindSensorsConstants, 714
 NBCCCommon.h, 944
MS_ADDR_NXTCAM
 MindSensorsConstants, 714
 NBCCCommon.h, 944
MS_ADDR_NXTHID
 MindSensorsConstants, 714
 NBCCCommon.h, 944
MS_ADDR_NXTMMX
 MindSensorsConstants, 714
 NBCCCommon.h, 944
MS_ADDR_NXTSERVO
 MindSensorsConstants, 714
 NBCCCommon.h, 944
MS_ADDR_NXTSERVO_EM
 MindSensorsConstants, 714
 NBCCCommon.h, 944
MS_ADDR_PFMATE
 MindSensorsConstants, 715
 NBCCCommon.h, 945
MS_ADDR_PSPNX
 MindSensorsConstants, 715
 NBCCCommon.h, 945
MS_ADDR_RTCLOCK
 MindSensorsConstants, 715
 NBCCCommon.h, 945
MS_ADDR_RXMUX
 MindSensorsConstants, 715

NBCCCommon.h, 945
MS_CMD_ADPA_OFF
 MindSensorsConstants, 715
 NBCCCommon.h, 945
MS_CMD_ADPA_ON
 MindSensorsConstants, 715
 NBCCCommon.h, 945
MS_CMD_DEENERGIZED
 MindSensorsConstants, 715
 NBCCCommon.h, 945
MS_CMD_ENERGIZED
 MindSensorsConstants, 715
 NBCCCommon.h, 945
MSACCLNx
 ACCL_CMD_RESET_CAL, 724
 ACCL_CMD_X_CAL, 724
 ACCL_CMD_X_CAL_END, 725
 ACCL_CMD_Y_CAL, 725
 ACCL_CMD_Y_CAL_END, 725
 ACCL_CMD_Z_CAL, 725
 ACCL_CMD_Z_CAL_END, 725
 ACCL_REG_SENS_LVL, 725
 ACCL_REG_X_ACCEL, 725
 ACCL_REG_X_OFFSET, 725
 ACCL_REG_X_RANGE, 725
 ACCL_REG_X_TILT, 725
 ACCL_REG_Y_ACCEL, 726
 ACCL_REG_Y_OFFSET, 726
 ACCL_REG_Y_RANGE, 726
 ACCL_REG_Y_TILT, 726
 ACCL_REG_Z_ACCEL, 726
 ACCL_REG_Z_OFFSET, 726
 ACCL_REG_Z_RANGE, 726
 ACCL_REG_Z_TILT, 726
MSACCLNxSLevel
 ACCL_SENSITIVITY_LEVEL_1,
 727
 ACCL_SENSITIVITY_LEVEL_2,
 727
 ACCL_SENSITIVITY_LEVEL_3,
 727
 ACCL_SENSITIVITY_LEVEL_4,
 727
MSADPAOff
 MindSensorsAPI, 170
MSADPAOn
 MindSensorsAPI, 171
 MindSensorsAPI, 171
MSDeenergize
 MindSensorsAPI, 171
MSDistNX
 DIST_CMD_CUSTOM, 716
 DIST_CMD_GP2D12, 716
 DIST_CMD_GP2D120, 716
 DIST_CMD_GP2YA02, 716
 DIST_CMD_GP2YA21, 716
 DIST_REG_DIST, 716
 DIST_REG_DIST1, 717
 DIST_REG_DIST_MAX, 717
 DIST_REG_DIST_MIN, 717
 DIST_REG_MODULE_TYPE, 717
 DIST_REG_NUM_POINTS, 717
 DIST_REG_VOLT, 717
 DIST_REG_VOLT1, 717
MSEnergize
 MindSensorsAPI, 171
MSIRTrain
 MindSensorsAPI, 172
MSNRLink
 NRLINK_CMD_2400, 722
 NRLINK_CMD_4800, 722
 NRLINK_CMD_FLUSH, 722
 NRLINK_CMD_IR_LONG, 722
 NRLINK_CMD_IR_SHORT, 723
 NRLINK_CMD_RUN_MACRO,
 723
 NRLINK_CMD_SET_PF, 723
 NRLINK_CMD_SET_RXC, 723
 NRLINK_CMD_SET_TRAIN, 723
 NRLINK_CMD_TX_RAW, 723
 NRLINK_REG_BYTES, 723
 NRLINK_REG_DATA, 723
 NRLINK_REG_EEPROM, 723
MSPFComboDirect
 MindSensorsAPI, 172
MSPFComboPWM
 MindSensorsAPI, 173
MSPFRawOutput
 MindSensorsAPI, 173
MSPFRepeat
 MindSensorsAPI, 174
MSPFSingleOutputCST
 MindSensorsAPI, 174

MSPFSingleOutputPWM
 MindSensorsAPI, 175

MSPFSinglePin
 MindSensorsAPI, 176

MSPFTrain
 MindSensorsAPI, 176

MSPSPNX
 PSP_CMD_ANALOG, 718
 PSP_CMD_DIGITAL, 718
 PSP_REG_BTNSET1, 718
 PSP_REG_BTNSET2, 718
 PSP_REG_XLEFT, 718
 PSP_REG_XRIGHT, 719
 PSP_REG_YLEFT, 719
 PSP_REG_YRIGHT, 719

MSPSPNxBtnSet1
 PSP_BTNSET1_DOWN, 719
 PSP_BTNSET1_L3, 719
 PSP_BTNSET1_LEFT, 720
 PSP_BTNSET1_R3, 720
 PSP_BTNSET1_RIGHT, 720
 PSP_BTNSET1_SELECT, 720
 PSP_BTNSET1_START, 720
 PSP_BTNSET1_UP, 720

MSPSPNxBtnSet2
 PSP_BTNSET2_CIRCLE, 721
 PSP_BTNSET2_CROSS, 721
 PSP_BTNSET2_L1, 721
 PSP_BTNSET2_L2, 721
 PSP_BTNSET2_R1, 721
 PSP_BTNSET2_R2, 721
 PSP_BTNSET2_SQUARE, 721
 PSP_BTNSET2_TRIANGLE, 721

MSRCXAbsVar
 MindSensorsAPI, 177

MSRCXAddToDatalog
 MindSensorsAPI, 177

MSRCXAndVar
 MindSensorsAPI, 177

MSRCXBatteryLevel
 MindSensorsAPI, 178

MSRCXBoot
 MindSensorsAPI, 178

MSRCXCalibrateEvent
 MindSensorsAPI, 178

MSRCXClearAllEvents

 MindSensorsAPI, 178

 MSRCXClearCounter
 MindSensorsAPI, 179

 MSRCXClearMsg
 MindSensorsAPI, 179

 MSRCXClearSensor
 MindSensorsAPI, 179

 MSRCXClearSound
 MindSensorsAPI, 179

 MSRCXClearTimer
 MindSensorsAPI, 179

 MSRCXCreateDatalog
 MindSensorsAPI, 180

 MSRCXDecCounter
 MindSensorsAPI, 180

 MSRCXDeleteSub
 MindSensorsAPI, 180

 MSRCXDeleteSubs
 MindSensorsAPI, 180

 MSRCXDeleteTask
 MindSensorsAPI, 180

 MSRCXDeleteTasks
 MindSensorsAPI, 181

 MSRCXDisableOutput
 MindSensorsAPI, 181

 MSRCXDivVar
 MindSensorsAPI, 181

 MSRCXEnableOutput
 MindSensorsAPI, 181

 MSRCXEvent
 MindSensorsAPI, 182

 MSRCXFloat
 MindSensorsAPI, 182

 MSRCXFwd
 MindSensorsAPI, 182

 MSRCXIIncCounter
 MindSensorsAPI, 182

 MSRCXIInvertOutput
 MindSensorsAPI, 183

 MSRCXMulVar
 MindSensorsAPI, 183

 MSRCXMuteSound
 MindSensorsAPI, 183

 MSRCXObvertOutput
 MindSensorsAPI, 183

 MSRCXOff

MindSensorsAPI, 184
MSRCXOn
 MindSensorsAPI, 184
MSRCXOnFor
 MindSensorsAPI, 184
MSRCXOnFwd
 MindSensorsAPI, 184
MSRCXOnRev
 MindSensorsAPI, 185
MSRCXOrVar
 MindSensorsAPI, 185
MSRCXPBTurnOff
 MindSensorsAPI, 185
MSRCXPing
 MindSensorsAPI, 185
MSRCXPlaySound
 MindSensorsAPI, 186
MSRCXPlayTone
 MindSensorsAPI, 186
MSRCXPlayToneVar
 MindSensorsAPI, 186
MSRCXPoll
 MindSensorsAPI, 186
MSRCXPollMemory
 MindSensorsAPI, 187
MSRCXRemote
 MindSensorsAPI, 187
MSRCXReset
 MindSensorsAPI, 187
MSRCXRev
 MindSensorsAPI, 187
MSRCXSelectDisplay
 MindSensorsAPI, 188
MSRCXSelectProgram
 MindSensorsAPI, 188
MSRCXSendSerial
 MindSensorsAPI, 188
MSRCXSet
 MindSensorsAPI, 188
MSRCXSetDirection
 MindSensorsAPI, 189
MSRCXSetEvent
 MindSensorsAPI, 189
MSRCXSetGlobalDirection
 MindSensorsAPI, 189
MSRCXSetGlobalOutput

 MindSensorsAPI, 190
MSRCXSetMaxPower
 MindSensorsAPI, 190
MSRCXSetMessage
 MindSensorsAPI, 190
MSRCXSetNRLinkPort
 MindSensorsAPI, 191
MSRCXSetOutput
 MindSensorsAPI, 191
MSRCXSetPower
 MindSensorsAPI, 191
MSRCXSetPriority
 MindSensorsAPI, 191
MSRCXSetSensorMode
 MindSensorsAPI, 192
MSRCXSetSensorType
 MindSensorsAPI, 192
MSRCXSetSleepTime
 MindSensorsAPI, 192
MSRCXSetTxPower
 MindSensorsAPI, 192
MSRCXSetUserDisplay
 MindSensorsAPI, 193
MSRCXSetVar
 MindSensorsAPI, 193
MSRCXSetWatch
 MindSensorsAPI, 193
MSRCXSgnVar
 MindSensorsAPI, 194
MSRCXStartTask
 MindSensorsAPI, 194
MSRCXStopAllTasks
 MindSensorsAPI, 194
MSRCXStopTask
 MindSensorsAPI, 194
MSRCXSubVar
 MindSensorsAPI, 194
MSRCXSumVar
 MindSensorsAPI, 195
MSRCXToggle
 MindSensorsAPI, 195
MSRCXUnlock
 MindSensorsAPI, 195
MSRCXUnmuteSound
 MindSensorsAPI, 195
MSReadValue

MindSensorsAPI, 196
MSScoutCalibrateSensor
 MindSensorsAPI, 196
MSScoutMuteSound
 MindSensorsAPI, 196
MSScoutSelectSounds
 MindSensorsAPI, 196
MSScoutSendVLL
 MindSensorsAPI, 197
MSScoutSetCounterLimit
 MindSensorsAPI, 197
MSScoutSetEventFeedback
 MindSensorsAPI, 197
MSScoutSetLight
 MindSensorsAPI, 197
MSScoutSetScoutMode
 MindSensorsAPI, 198
MSScoutSetScoutRules
 MindSensorsAPI, 198
MSScoutSetSensorClickTime
 MindSensorsAPI, 198
MSScoutSetSensorHysteresis
 MindSensorsAPI, 199
MSScoutSetSensorLowerLimit
 MindSensorsAPI, 199
MSScoutSetSensorUpperLimit
 MindSensorsAPI, 199
MSScoutSetTimerLimit
 MindSensorsAPI, 199
MSScoutUnmuteSound
 MindSensorsAPI, 200

NA
 MiscConstants, 266
 NBCCommon.h, 945
NBC Input port constants, 543
NBC sensor mode constants, 547
NBC sensor type constants, 544
NBCAPIDocs.h, 790
NBCCommon.h, 791
 ACCL_CMD_RESET_CAL, 838
 ACCL_CMD_X_CAL, 838
 ACCL_CMD_X_CAL_END, 839
 ACCL_CMD_Y_CAL, 839
 ACCL_CMD_Y_CAL_END, 839
 ACCL_CMD_Z_CAL, 839
ACCL_CMD_Z_CAL_END, 839
ACCL_REG_SENS_LVL, 839
ACCL_REG_X_ACCEL, 839
ACCL_REG_X_OFFSET, 839
ACCL_REG_X_RANGE, 839
ACCL_REG_X_TILT, 839
ACCL_REG_Y_ACCEL, 840
ACCL_REG_Y_OFFSET, 840
ACCL_REG_Y_RANGE, 840
ACCL_REG_Y_TILT, 840
ACCL_REG_Z_ACCEL, 840
ACCL_REG_Z_OFFSET, 840
ACCL_REG_Z_RANGE, 840
ACCL_REG_Z_TILT, 840
ACCL_SENSITIVITY_LEVEL_1, 840
ACCL_SENSITIVITY_LEVEL_2, 840
ACCL_SENSITIVITY_LEVEL_3, 841
ACCL_SENSITIVITY_LEVEL_4, 841
ActualSpeedField, 841
BITMAP_1, 841
BITMAP_2, 841
BITMAP_3, 841
BITMAP_4, 841
BITMAPS, 841
BlockTachoCountField, 841
BREAKOUT_REQ, 842
BT_ARM_CMD_MODE, 842
BT_ARM_DATA_MODE, 842
BT_ARM_OFF, 842
BT_BRICK_PORT_OPEN, 842
BT_BRICK_VISIBILITY, 842
BT_CMD_BYTE, 842
BT_CMD_READY, 842
BT_CONNECTION_0_ENABLE, 843
BT_CONNECTION_1_ENABLE, 843
BT_CONNECTION_2_ENABLE, 843
BT_CONNECTION_3_ENABLE, 843

BT_DEFAULT_INQUIRY_MAX, 843
BT_DEFAULT_INQUIRY_- TIMEOUT_LO, 843
BT_DEVICE_AWAY, 843
BT_DEVICE_EMPTY, 843
BT_DEVICE_KNOWN, 843
BT_DEVICE_NAME, 843
BT_DEVICE_UNKNOWN, 844
BT_DISABLE, 844
BT_ENABLE, 844
BTN1, 844
BTN2, 844
BTN3, 844
BTN4, 844
BTNCENTER, 844
BTNEXTIT, 844
BTNLEFT, 844
BTNRIGHT, 845
BTNSTATE_LONG_PRESSED_- EV, 845
BTNSTATE_LONG_RELEASED_- EV, 845
BTNSTATE_NONE, 845
BTNSTATE_PRESSED_EV, 845
BTNSTATE_PRESSED_STATE, 845
BTNSTATE_SHORT_- RELEASED_EV, 845
ButtonModuleID, 845
ButtonModuleName, 845
ButtonOffsetLongPressCnt, 845
ButtonOffsetLongRelCnt, 846
ButtonOffsetPressedCnt, 846
ButtonOffsetRelCnt, 846
ButtonOffsetShortRelCnt, 846
ButtonOffsetState, 846
CHAR_BIT, 846
CHAR_MAX, 846
CHAR_MIN, 846
CLUMP_DONE, 846
CLUMP_SUSPEND, 846
ColorSensorRead, 847
COM_CHANNEL_FOUR_- ACTIVE, 847
COM_CHANNEL_NONE_- ACTIVE, 847
COM_CHANNEL_ONE_ACTIVE, 847
COM_CHANNEL_THREE_- ACTIVE, 847
COM_CHANNEL_TWO_ACTIVE, 847
CommandModuleID, 847
CommandModuleName, 847
CommandOffsetActivateFlag, 847
CommandOffsetAwake, 847
CommandOffsetDeactivateFlag, 848
CommandOffsetFileName, 848
CommandOffsetFormatString, 848
CommandOffsetMemoryPool, 848
CommandOffsetOffsetDS, 848
CommandOffsetOffsetDVA, 848
CommandOffsetPRCHandler, 848
CommandOffsetProgStatus, 848
CommandOffsetSyncTick, 848
CommandOffsetSyncTime, 848
CommandOffsetTick, 849
CommBTCheckStatus, 849
CommBTConnection, 849
CommBTOnOff, 849
CommBTRRead, 849
CommBTWrite, 849
CommExecuteFunction, 849
CommHSCheckStatus, 849
CommHSControl, 849
CommHSRead, 849
CommHSSWrite, 850
CommLSCheckStatus, 850
CommLSRead, 850
CommLSWrite, 850
CommLSWriteEx, 850
CommModuleID, 850
CommModuleName, 850
CommOffsetBrickDataBdAddr, 850
CommOffsetBrickDataBluecoreVersion, 850
CommOffsetBrickDataBtHwStatus, 850
CommOffsetBrickDataBtStateStatus, 851

CommOffsetBrickDataName, 851
CommOffsetBrickDataTimeOut-Value, 851
CommOffsetBtConnectTableBdAddr, 851
CommOffsetBtConnectTableClassOfDevice, 851
CommOffsetBtConnectTableHandleNr, 851
CommOffsetBtConnect-TableLinkQuality, 851
CommOffsetBtConnectTableName, 851
CommOffsetBtConnectTablePin-Code, 851
CommOffsetBtConnectTa-bleStreamStatus, 851
CommOffsetBtDataMode, 852
CommOffsetBtDeviceCnt, 852
CommOffsetBtDeviceNameCnt, 852
CommOffsetBtDeviceTableBdAddr, 852
CommOffsetBtDeviceTableClassOfDevice, 852
CommOffsetBtDeviceTableDe-viceStatus, 852
CommOffsetBtDeviceTableName, 852
CommOffsetBtInBufBuf, 852
CommOffsetBtInBufInPtr, 852
CommOffsetBtInBufOutPtr, 852
CommOffsetBtOutBufBuf, 853
CommOffsetBtOutBufInPtr, 853
CommOffsetBtOutBufOutPtr, 853
CommOffsetHsAddress, 853
CommOffsetHsDataMode, 853
CommOffsetHsFlags, 853
CommOffsetHsInBufBuf, 853
CommOffsetHsInBufInPtr, 853
CommOffsetHsInBufOutPtr, 853
CommOffsetHsMode, 853
CommOffsetHsOutBufBuf, 854
CommOffsetHsOutBufInPtr, 854
CommOffsetHsOutBufOutPtr, 854
CommOffsetHsSpeed, 854
CommOffsetHsState, 854
CommOffsetPFunc, 854
CommOffsetPFuncTwo, 854
CommOffsetUsbInBufBuf, 854
CommOffsetUsbInBufInPtr, 854
CommOffsetUsbInBufOutPtr, 854
CommOffsetUsbOutBufBuf, 855
CommOffsetUsbOutBufInPtr, 855
CommOffsetUsbOutBufOutPtr, 855
CommOffsetUsbPollBufBuf, 855
CommOffsetUsbPollBufInPtr, 855
CommOffsetUsbPollBufOutPtr, 855
CommOffsetUsbState, 855
ComputeCalibValue, 855
CONN_BT0, 855
CONN_BT1, 855
CONN_BT2, 856
CONN_BT3, 856
CONN_HS4, 856
CONN_HS_1, 856
CONN_HS_2, 856
CONN_HS_3, 856
CONN_HS_4, 856
CONN_HS_5, 856
CONN_HS_6, 856
CONN_HS_7, 856
CONN_HS_8, 857
CONN_HS_ALL, 857
CT_ADDR_RFID, 857
CT_REG_DATA, 857
CT_REG_MODE, 857
CT_REG_STATUS, 857
DAC_MODE_DCOUT, 857
DAC_MODE_PWMVOLTAGE, 857
DAC_MODE_SAWNEGWAVE, 857
DAC_MODE_SAWPOSWAVE, 857
DAC_MODE_SINEWAVE, 858
DAC_MODE_SQUAREWAVE, 858
DAC_MODE_TRIANGLEWAVE, 858
DATA_MODE_GPS, 858
DATA_MODE_MASK, 858
DATA_MODE_NXT, 858
DATA_MODE_RAW, 858
DATA_MODE_UPDATE, 858

DatalogGetTimes, 858
DatalogWrite, 858
DEGREES_PER_RADIAN, 859
DGPS_REG_DISTANCE, 859
DGPS_REG_HEADING, 859
DGPS_REG_LASTANGLE, 859
DGPS_REG_LATITUDE, 859
DGPS_REG_LONGITUDE, 859
DGPS_REG_SETLATITUDE, 859
DGPS_REG_SETLONGITUDE, 859
DGPS_REG_STATUS, 859
DGPS_REG_TIME, 859
DGPS_REG_VELOCITY, 860
DGPS_REG_WAYANGLE, 860
DI_ADDR_ACCL, 860
DI_ADDR_DGPS, 860
DI_ADDR_GYRO, 860
DIACCL_CTRL1_FILT_BW125, 860
DIACCL_CTRL1_INT2TOINT1, 860
DIACCL_CTRL1_LEVELPULSE, 860
DIACCL_CTRL1_NO_XDETECT, 860
DIACCL_CTRL1_NO_YDETECT, 860
DIACCL_CTRL1_NO_ZDETECT, 861
DIACCL_CTRL1_PULSELEVEL, 861
DIACCL_CTRL1_PULSEPULSE, 861
DIACCL_CTRL1_THRESH_INT, 861
DIACCL_CTRL2_DETPOL_-, NEGAND, 861
DIACCL_CTRL2_DRIVE_-, STRONG, 861
DIACCL_CTRL2_LVLPOL_-, NEGAND, 861
DIACCL_INTERRUPT_LATCH_-, CLEAR1, 861
DIACCL_INTERRUPT_LATCH_-, CLEAR2, 861
DIACCL_MODE_GLVL2, 862
DIACCL_MODE_GLVL4, 862
DIACCL_MODE_GLVL8, 862
DIACCL_MODE_LVLDETECT, 862
DIACCL_MODE_MEASURE, 862
DIACCL_MODE_PLSDETECT, 862
DIACCL_MODE_STANDBY, 862
DIACCL_REG_CTRL1, 862
DIACCL_REG_CTRL2, 862
DIACCL_REG_DETECTSRC, 862
DIACCL_REG_I2CADDR, 863
DIACCL_REG_INTLATCH, 863
DIACCL_REG_LATENCYTM, 863
DIACCL_REG_LVLDETTHR, 863
DIACCL_REG_MODECTRL, 863
DIACCL_REG_OUTTEMP, 863
DIACCL_REG_PLSDETTHR, 863
DIACCL_REG_PLSDURVAL, 863
DIACCL_REG_STATUS, 863
DIACCL_REG_TIMEWINDOW, 863
DIACCL_REG_USERINFO, 864
DIACCL_REG_WHOAMI, 864
DIACCL_REG_X8, 864
DIACCL_REG_XHIGH, 864
DIACCL_REG_XHIGHDRIFT, 864
DIACCL_REG_XLOW, 864
DIACCL_REG_XLOWDRIFT, 864
DIACCL_REG_Y8, 864
DIACCL_REG_YHIGH, 864
DIACCL_REG_YHIGHDRIFT, 864
DIACCL_REG_YLOW, 865
DIACCL_REG_YLOWDRIFT, 865
DIACCL_REG_Z8, 865
DIACCL_REG_ZHIGH, 865
DIACCL_REG_ZHIGHDRIFT, 865
DIACCL_REG_ZLOW, 865
DIACCL_REG_ZLOWDRIFT, 865
DIACCL_STATUS_DATAOVER, 865
DIACCL_STATUS_DATAREADY, 865
DIACCL_STATUS_PARITYERR, 865

DIGI_PIN0, 866
DIGI_PIN1, 866
DIGI_PIN2, 866
DIGI_PIN3, 866
DIGI_PIN4, 866
DIGI_PIN5, 866
DIGI_PIN6, 866
DIGI_PIN7, 866
DIGYRO_CTRL1_-
 BANDWIDTH_1, 866
DIGYRO_CTRL1_-
 BANDWIDTH_2, 866
DIGYRO_CTRL1_-
 BANDWIDTH_3, 867
DIGYRO_CTRL1_-
 BANDWIDTH_4, 867
DIGYRO_CTRL1_DATARATE_-
 100, 867
DIGYRO_CTRL1_DATARATE_-
 200, 867
DIGYRO_CTRL1_DATARATE_-
 400, 867
DIGYRO_CTRL1_DATARATE_-
 800, 867
DIGYRO_CTRL1_NORMAL, 867
DIGYRO_CTRL1_POWERDOWN,
 867
DIGYRO_CTRL1_XENABLE, 867
DIGYRO_CTRL1_YENABLE, 867
DIGYRO_CTRL1_ZENABLE, 868
DIGYRO_CTRL2_CUTOFF_-
 FREQ_001, 868
DIGYRO_CTRL2_CUTOFF_-
 FREQ_002, 868
DIGYRO_CTRL2_CUTOFF_-
 FREQ_005, 868
DIGYRO_CTRL2_CUTOFF_-
 FREQ_01, 868
DIGYRO_CTRL2_CUTOFF_-
 FREQ_02, 868
DIGYRO_CTRL2_CUTOFF_-
 FREQ_05, 868
DIGYRO_CTRL2_CUTOFF_-
 FREQ_1, 868
DIGYRO_CTRL2_CUTOFF_-
 FREQ_2, 868
DIGYRO_CTRL2_CUTOFF_-
 FREQ_4, 868
DIGYRO_CTRL2_CUTOFF_-
 FREQ_8, 869
DIGYRO_CTRL2_HPMODE_-
 AUToint, 869
DIGYRO_CTRL2_HPMODE_-
 NORMAL, 869
DIGYRO_CTRL2_HPMODE_-
 REFSIG, 869
DIGYRO_CTRL2_HPMODE_-
 RESET, 869
DIGYRO_CTRL3_INT1_BOOT,
 869
DIGYRO_CTRL3_INT1_ENABLE,
 869
DIGYRO_CTRL3_INT1_-
 LOWACTIVE, 869
DIGYRO_CTRL3_INT2_-
 DATAREADY, 869
DIGYRO_CTRL3_INT2_EMPTY,
 869
DIGYRO_CTRL3_INT2_-
 OVERRUN, 870
DIGYRO_CTRL3_INT2_-
 WATERMARK, 870
DIGYRO_CTRL3_OPENDRAIN,
 870
DIGYRO_CTRL4_BIGENDIAN,
 870
DIGYRO_CTRL4_BLOCKDATA,
 870
DIGYRO_CTRL4_SCALE_2000,
 870
DIGYRO_CTRL4_SCALE_250,
 870
DIGYRO_CTRL4_SCALE_500,
 870
DIGYRO_CTRL5_FIFOENABLE,
 870
DIGYRO_CTRL5_HPENABLE,
 870
DIGYRO_CTRL5_INT1_SEL_1,
 871
DIGYRO_CTRL5_INT1_SEL_2,
 871

DIGYRO_CTRL5_INT1_SEL_3,
871
DIGYRO_CTRL5_OUT_SEL_1,
871
DIGYRO_CTRL5_OUT_SEL_2,
871
DIGYRO_CTRL5_OUT_SEL_3,
871
DIGYRO_CTRL5_REBOOTMEM,
871
DIGYRO_FIFOCTRL_BYPASS,
871
DIGYRO_FIFOCTRL_-
BYPASS2STREAM, 871
DIGYRO_FIFOCTRL_FIFO, 871
DIGYRO_FIFOCTRL_STREAM,
872
DIGYRO_FIFOCTRL_-
STREAM2FIFO, 872
DIGYRO_FIFOCTRL_-
WATERMARK_MASK,
872
DIGYRO_REG_CTRL1, 872
DIGYRO_REG_CTRL1AUTO, 872
DIGYRO_REG_CTRL2, 872
DIGYRO_REG_CTRL3, 872
DIGYRO_REG_CTRL4, 872
DIGYRO_REG_CTRL5, 872
DIGYRO_REG_FIFOCTRL, 872
DIGYRO_REG_FIFOSRC, 873
DIGYRO_REG_INT1_CFG, 873
DIGYRO_REG_INT1_DUR, 873
DIGYRO_REG_INT1_SRC, 873
DIGYRO_REG_INT1_XHI, 873
DIGYRO_REG_INT1_XLO, 873
DIGYRO_REG_INT1_YHI, 873
DIGYRO_REG_INT1_YLO, 873
DIGYRO_REG_INT1_ZHI, 873
DIGYRO_REG_INT1_ZLO, 873
DIGYRO_REG_OUTTEMP, 874
DIGYRO_REG_REFERENCE, 874
DIGYRO_REG_STATUS, 874
DIGYRO_REG_TEMPAUTO, 874
DIGYRO_REG_WHOAMI, 874
DIGYRO_REG_XHIGH, 874
DIGYRO_REG_XLOW, 874
DIGYRO_REG_XLOWBURST,
874
DIGYRO_REG_YHIGH, 874
DIGYRO_REG_YLOW, 874
DIGYRO_REG_YLOWBURST,
875
DIGYRO_REG_ZHIGH, 875
DIGYRO_REG_ZLOW, 875
DIGYRO_REG_ZLOWBURST,
875
DIGYRO_STATUS_XDATA, 875
DIGYRO_STATUS_XOVER, 875
DIGYRO_STATUS_XYZDATA,
875
DIGYRO_STATUS_XZOVER,
875
DIGYRO_STATUS_YDATA, 875
DIGYRO_STATUS_YOVER, 875
DIGYRO_STATUS_ZDATA, 876
DIGYRO_STATUS_ZOVER, 876
DISPLAY_BUSY, 876
DISPLAY_CHAR, 876
DISPLAY_CONTRAST_-
DEFAULT, 876
DISPLAY_CONTRAST_MAX, 876
DISPLAY_ERASE_ALL, 876
DISPLAY_ERASE_LINE, 876
DISPLAY_FILL_REGION, 876
DISPLAY_FRAME, 876
DISPLAY_HEIGHT, 877
DISPLAY_HORIZONTAL_LINE,
877
DISPLAY_MENUICONS_X_DIFF,
877
DISPLAY_MENUICONS_X_-
OFFS, 877
DISPLAY_MENUICONS_Y, 877
DISPLAY_ON, 877
DISPLAY_PIXEL, 877
DISPLAY_POPUP, 877
DISPLAY_REFRESH, 877
DISPLAY_REFRESH_DISABLED,
877
DISPLAY_VERTICAL_LINE, 878
DISPLAY_WIDTH, 878
DisplayExecuteFunction, 878

DisplayModuleID, 878
DisplayModuleName, 878
DisplayOffsetContrast, 878
DisplayOffsetDisplay, 878
DisplayOffsetEraseMask, 878
DisplayOffsetFlags, 878
DisplayOffsetNormal, 878
DisplayOffsetPBitmaps, 879
DisplayOffsetPFont, 879
DisplayOffsetPFunc, 879
DisplayOffsetPMenuIcons, 879
DisplayOffsetPMenuText, 879
DisplayOffsetPopup, 879
DisplayOffsetPScreens, 879
DisplayOffsetPStatusIcons, 879
DisplayOffsetPStatusText, 879
DisplayOffsetPStepIcons, 879
DisplayOffsetPTextLines, 880
DisplayOffsetStatusIcons, 880
DisplayOffsetStepIcons, 880
DisplayOffsetTextLinesCenterFlags,
 880
DisplayOffsetUpdateMask, 880
DIST_CMD_CUSTOM, 880
DIST_CMD_GP2D12, 880
DIST_CMD_GP2D120, 880
DIST_CMD_GP2YA02, 880
DIST_CMD_GP2YA21, 880
DIST_REG_DIST, 881
DIST_REG_DIST1, 881
DIST_REG_DIST_MAX, 881
DIST_REG_DIST_MIN, 881
DIST_REG_MODULE_TYPE, 881
DIST_REG_NUM_POINTS, 881
DIST_REG_VOLT, 881
DIST_REG_VOLT1, 881
DRAW_OPT_CLEAR, 881
DRAW_OPT_CLEAR_EXCEPT_-
 STATUS_SCREEN, 881
DRAW_OPT_CLEAR_PIXELS,
 882
DRAW_OPT_CLEAR_SCREEN_-
 MODES, 882
DRAW_OPT_CLEAR_WHOLE_-
 SCREEN, 882
DRAW_OPT_FILL_SHAPE, 882
DRAW_OPT_FONT_DIR_B2TL,
 882
DRAW_OPT_FONT_DIR_B2TR,
 882
DRAW_OPT_FONT_DIR_L2RB,
 882
DRAW_OPT_FONT_DIR_L2RT,
 882
DRAW_OPT_FONT_DIR_R2LB,
 882
DRAW_OPT_FONT_DIR_R2LT,
 882
DRAW_OPT_FONT_DIR_T2BL,
 883
DRAW_OPT_FONT_DIR_T2BR,
 883
DRAW_OPT_FONT_-
 DIRECTIONS, 883
DRAW_OPT_FONT_WRAP, 883
DRAW_OPT_INVERT, 883
DRAW_OPT_LOGICAL_AND,
 883
DRAW_OPT_LOGICAL_COPY,
 883
DRAW_OPT_LOGICAL_-
 OPERATIONS, 883
DRAW_OPT_LOGICAL_OR, 883
DRAW_OPT_LOGICAL_XOR,
 883
DRAW_OPT_NORMAL, 884
DRAW_OPT_POLYGON_-
 POLYLINE, 884
DrawCircle, 884
DrawEllipse, 884
DrawFont, 884
DrawGraphic, 884
DrawGraphicArray, 884
DrawLine, 884
DrawPoint, 884
DrawPolygon, 884
DrawRect, 885
DrawText, 885
EMETER_REG_AIN, 885
EMETER_REG_AOUT, 885
EMETER_REG_JOULES, 885
EMETER_REG_VIN, 885

EMETER_REG_VOUT, 885
EMETER_REG_WIN, 885
EMETER_REG_WOUT, 885
EOF, 885
ERR_ARG, 886
ERR_BAD_POOL_SIZE, 886
ERR_BAD_PTR, 886
ERR_CLUMP_COUNT, 886
ERR_COMM_BUFFER_FULL,
 886
ERR_COMM_BUS_ERR, 886
ERR_COMM_CHAN_INVALID,
 886
ERR_COMM_CHAN_NOT_-
 READY, 886
ERR_DEFAULT_OFFSETS, 886
ERR_FILE, 886
ERR_INSANE_OFFSET, 887
ERR_INSTR, 887
ERR_INVALID_FIELD, 887
ERR_INVALID_PORT, 887
ERR_INVALID_QUEUE, 887
ERR_INVALID_SIZE, 887
ERR_LOADER_ERR, 887
ERR_MEM, 887
ERR_MEMMGR_FAIL, 887
ERR_NO_ACTIVE_CLUMP, 887
ERR_NO_CODE, 888
ERR_NO_PROG, 888
ERR_NON_FATAL, 888
ERR_RC_BAD_PACKET, 888
ERR_RC_FAILED, 888
ERR_RC_ILLEGAL_VAL, 888
ERR_RC_UNKNOWN_CMD, 888
ERR_SPOTCHECK_FAIL, 888
ERR_VER, 888
FALSE, 888
FileClose, 889
FileDelete, 889
FileFindFirst, 889
FileFindNext, 889
FileOpenAppend, 889
FileOpenRead, 889
FileOpenReadLinear, 889
FileOpenWrite, 889
FileOpenWriteLinear, 889
FileOpenWriteNonLinear, 889
FileRead, 890
FileRename, 890
FileResize, 890
FileResolveHandle, 890
FileSeek, 890
FileTell, 890
FileWrite, 890
FRAME_SELECT, 890
FREQUENCY_MAX, 890
FREQUENCY_MIN, 890
GetStartTick, 891
GL_CAMERA_DEPTH, 891
GL_CIRCLE, 891
GL_CIRCLE_SIZE, 891
GL_CULL_BACK, 891
GL_CULL_FRONT, 891
GL_CULL_MODE, 891
GL_CULL_NONE, 891
GL_LINE, 891
GL_POINT, 891
GL_POLYGON, 892
GL_ROTATE_X, 892
GL_ROTATE_Y, 892
GL_ROTATE_Z, 892
GL_SCALE_X, 892
GL_SCALE_Y, 892
GL_SCALE_Z, 892
GL_TRANSLATE_X, 892
GL_TRANSLATE_Y, 892
GL_TRANSLATE_Z, 892
GL_ZOOM_FACTOR, 893
HS_ADDRESS_1, 893
HS_ADDRESS_2, 893
HS_ADDRESS_3, 893
HS_ADDRESS_4, 893
HS_ADDRESS_5, 893
HS_ADDRESS_6, 893
HS_ADDRESS_7, 893
HS_ADDRESS_8, 893
HS_ADDRESS_ALL, 893
HS_BAUD_115200, 894
HS_BAUD_1200, 894
HS_BAUD_14400, 894
HS_BAUD_19200, 894
HS_BAUD_230400, 894

HS_BAUD_2400, 894
HS_BAUD_28800, 894
HS_BAUD_3600, 894
HS_BAUD_38400, 894
HS_BAUD_460800, 894
HS_BAUD_4800, 895
HS_BAUD_57600, 895
HS_BAUD_7200, 895
HS_BAUD_76800, 895
HS_BAUD_921600, 895
HS_BAUD_9600, 895
HS_BAUD_DEFAULT, 895
HS_BYTES_REMAINING, 895
HS_CMD_READY, 895
HS_CTRL_EXIT, 895
HS_CTRL_INIT, 896
HS_CTRL_UART, 896
HS_DEFAULT, 896
HS_DISABLE, 896
HS_ENABLE, 896
HS_INIT_RECEIVER, 896
HS_INITIALISE, 896
HS_MODE_10_STOP, 896
HS_MODE_15_STOP, 896
HS_MODE_20_STOP, 896
HS_MODE_5_DATA, 897
HS_MODE_6_DATA, 897
HS_MODE_7_DATA, 897
HS_MODE_7E1, 897
HS_MODE_8_DATA, 897
HS_MODE_8N1, 897
HS_MODE_DEFAULT, 897
HS_MODE_E_PARITY, 897
HS_MODE_M_PARITY, 897
HS_MODE_MASK, 898
HS_MODE_N_PARITY, 898
HS_MODE_O_PARITY, 898
HS_MODE_S_PARITY, 898
HS_MODE_UART_RS232, 898
HS_MODE_UART_RS485, 898
HS_SEND_DATA, 898
HS_UART_MASK, 898
HS_UPDATE, 898
HT_ADDR_ACCEL, 898
HT_ADDR_ANGLE, 899
HT_ADDR_BAROMETRIC, 899
HT_ADDR_COLOR, 899
HT_ADDR_COLOR2, 899
HT_ADDR_COMPASS, 899
HT_ADDR_IRLINK, 899
HT_ADDR_IRRECEIVER, 899
HT_ADDR_IRSEEKER, 899
HT_ADDR_IRSEEKER2, 899
HT_ADDR_PROTOBOARD, 899
HT_ADDR_SUPERPRO, 900
HT_CH1_A, 900
HT_CH1_B, 900
HT_CH2_A, 900
HT_CH2_B, 900
HT_CH3_A, 900
HT_CH3_B, 900
HT_CH4_A, 900
HT_CH4_B, 900
HT_CMD_COLOR2_50HZ, 900
HT_CMD_COLOR2_60HZ, 901
HT_CMD_COLOR2_ACTIVE, 901
HT_CMD_COLOR2_BLCAL, 901
HT_CMD_COLOR2_FAR, 901
HT_CMD_COLOR2_LED_HI, 901
HT_CMD_COLOR2_LED_LOW,
 901
HT_CMD_COLOR2_NEAR, 901
HT_CMD_COLOR2_PASSIVE,
 901
HT_CMD_COLOR2_RAW, 901
HT_CMD_COLOR2_WBCAL, 901
HTANGLE_MODE_CALIBRATE,
 902
HTANGLE_MODE_NORMAL,
 902
HTANGLE_MODE_RESET, 902
HTANGLE_REG_ACDIR, 902
HTANGLE_REG_DC01, 902
HTANGLE_REG_DC02, 902
HTANGLE_REG_DC03, 902
HTANGLE_REG_DC04, 902
HTANGLE_REG_DC05, 902
HTANGLE_REG_DCAVG, 902
HTANGLE_REG_DCDIR, 903
HTANGLE_REG_MODE, 903
HTBAR_REG_CALIBRATION,
 903

HTBAR_REG_COMMAND, 903
HTBAR_REG_PRESSURE, 903
HTBAR_REG_TEMPERATURE,
 903
HTIR2_MODE_1200, 903
HTIR2_MODE_600, 903
HTIR2_REG_AC01, 903
HTIR2_REG_AC02, 903
HTIR2_REG_AC03, 904
HTIR2_REG_AC04, 904
HTIR2_REG_AC05, 904
HTIR2_REG_ACDIR, 904
HTIR2_REG_DC01, 904
HTIR2_REG_DC02, 904
HTIR2_REG_DC03, 904
HTIR2_REG_DC04, 904
HTIR2_REG_DC05, 904
HTIR2_REG_DCAVG, 904
HTIR2_REG_DCDIR, 905
HTIR2_REG_MODE, 905
HTPROTO_A0, 905
HTPROTO_A1, 905
HTPROTO_A2, 905
HTPROTO_A3, 905
HTPROTO_A4, 905
HTPROTO_REG_A0, 905
HTPROTO_REG_A1, 905
HTPROTO_REG_A2, 905
HTPROTO_REG_A3, 906
HTPROTO_REG_A4, 906
HTPROTO_REG_DCTRL, 906
HTPROTO_REG_DIN, 906
HTPROTO_REG_DOUT, 906
HTPROTO_REG_SRATE, 906
HTSPRO_A0, 906
HTSPRO_A1, 906
HTSPRO_A2, 906
HTSPRO_A3, 906
HTSPRO_DAC0, 907
HTSPRO_DAC1, 907
HTSPRO_REG_A0, 907
HTSPRO_REG_A1, 907
HTSPRO_REG_A2, 907
HTSPRO_REG_A3, 907
HTSPRO_REG_CTRL, 907
HTSPRO_REG_DAC0_FREQ, 907
HTSPRO_REG_DAC0_MODE, 907
HTSPRO_REG_DAC0_VOLTAGE,
 907
HTSPRO_REG_DAC1_FREQ, 908
HTSPRO_REG_DAC1_MODE, 908
HTSPRO_REG_DAC1_VOLTAGE,
 908
HTSPRO_REG_DCTRL, 908
HTSPRO_REG_DIN, 908
HTSPRO_REG_DLADDRESS, 908
HTSPRO_REG_DLCKSUM, 908
HTSPRO_REG_DLCONTROL, 908
HTSPRO_REG_DLLDATA, 908
HTSPRO_REG_DOUT, 908
HTSPRO_REG_LED, 909
HTSPRO_REG_MEMORY_20, 909
HTSPRO_REG_MEMORY_21, 909
HTSPRO_REG_MEMORY_22, 909
HTSPRO_REG_MEMORY_23, 909
HTSPRO_REG_MEMORY_24, 909
HTSPRO_REG_MEMORY_25, 909
HTSPRO_REG_MEMORY_26, 909
HTSPRO_REG_MEMORY_27, 909
HTSPRO_REG_MEMORY_28, 909
HTSPRO_REG_MEMORY_29, 910
HTSPRO_REG_MEMORY_2A,
 910
HTSPRO_REG_MEMORY_2B,
 910
HTSPRO_REG_MEMORY_2C,
 910
HTSPRO_REG_MEMORY_2D,
 910
HTSPRO_REG_MEMORY_2E,
 910
HTSPRO_REG_MEMORY_2F, 910
HTSPRO_REG_MEMORY_30, 910
HTSPRO_REG_MEMORY_31, 910
HTSPRO_REG_MEMORY_32, 910
HTSPRO_REG_MEMORY_33, 911
HTSPRO_REG_MEMORY_34, 911
HTSPRO_REG_MEMORY_35, 911
HTSPRO_REG_MEMORY_36, 911
HTSPRO_REG_MEMORY_37, 911
HTSPRO_REG_MEMORY_38, 911
HTSPRO_REG_MEMORY_39, 911

HTSPRO_REG_MEMORY_3A,
911
HTSPRO_REG_MEMORY_3B,
911
HTSPRO_REG_MEMORY_3C,
911
HTSPRO_REG_MEMORY_3D,
912
HTSPRO_REG_MEMORY_3E,
912
HTSPRO_REG_MEMORY_3F, 912
HTSPRO_REG_STROBE, 912
I2C_ADDR_DEFAULT, 912
I2C_OPTION_FAST, 912
I2C_OPTION_NORESTART, 912
I2C_OPTION_STANDARD, 912
I2C_REG_CMD, 912
I2C_REG_DEVICE_ID, 912
I2C_REG_VENDOR_ID, 913
I2C_REG_VERSION, 913
IN_1, 913
IN_2, 913
IN_3, 913
IN_4, 913
IN_MODE_ANGLESTEP, 913
IN_MODE_BOOLEAN, 913
IN_MODE_CELSIUS, 913
IN_MODE_FAHRENHEIT, 913
IN_MODE_MODEMASK, 914
IN_MODE_PCTFULLSCALE, 914
IN_MODE_PERIODCOUNTER,
914
IN_MODE_RAW, 914
IN_MODE_SLOPEMASK, 914
IN_MODE_TRANSITIONCNT,
914
IN_TYPE_ANGLE, 914
IN_TYPE_COLORBLUE, 914
IN_TYPE_COLOREXIT, 914
IN_TYPE_COLORFULL, 914
IN_TYPE_COLORGREEN, 915
IN_TYPE_COLORNONE, 915
IN_TYPE_COLORRED, 915
IN_TYPE_CUSTOM, 915
IN_TYPE_HISPEED, 915
IN_TYPE_LIGHT_ACTIVE, 915
IN_TYPE_LIGHT_INACTIVE, 915
IN_TYPE_LOWSPEED, 915
IN_TYPE_LOWSPEED_9V, 915
IN_TYPE_NO_SENSOR, 915
IN_TYPE_REFLECTION, 916
IN_TYPE_SOUND_DB, 916
IN_TYPE_SOUND_DBA, 916
IN_TYPE_SWITCH, 916
IN_TYPE_TEMPERATURE, 916
INPUT_BLACKCOLOR, 916
INPUT_BLANK, 916
INPUT_BLUE, 916
INPUT_BLUECOLOR, 916
INPUT_CAL_POINT_0, 916
INPUT_CAL_POINT_1, 917
INPUT_CAL_POINT_2, 917
INPUT_CUSTOM9V, 917
INPUT_CUSTOMACTIVE, 917
INPUT_CUSTOMINACTIVE, 917
INPUT_DIGI0, 917
INPUT_DIGI1, 917
INPUT_GREEN, 917
INPUT_GREENCOLOR, 917
INPUT_INVALID_DATA, 917
INPUT_NO_OF_COLORS, 918
INPUT_NO_OF_POINTS, 918
INPUT_PINCMD_CLEAR, 918
INPUT_PINCMD_DIR, 918
INPUT_PINCMD_MASK, 918
INPUT_PINCMD_READ, 918
INPUT_PINCMD_SET, 918
INPUT_PINCMD_WAIT, 918
INPUT_PINDIR_INPUT, 918
INPUT_PINDIR_OUTPUT, 919
INPUT_RED, 919
INPUT_REDCOLOR, 919
INPUT_RESETCAL, 919
INPUT_RUNNINGCAL, 919
INPUT_SENSORCAL, 919
INPUT_SENSOROFF, 919
INPUT_STARTCAL, 919
INPUT_WHITECOLOR, 919
INPUT_YELLOWCOLOR, 919
InputModeField, 920
InputModuleID, 920
InputModuleName, 920

InputOffsetADRaw, 920
InputOffsetColorADRaw, 920
InputOffsetColorBoolean, 920
InputOffsetColorCalibration, 920
InputOffsetColorCalibrationState,
 920
InputOffsetColorCalLimits, 920
InputOffsetColorSensorRaw, 920
InputOffsetColorSensorValue, 921
InputOffsetCustomActiveStatus, 921
InputOffsetCustomPctFullScale, 921
InputOffsetCustomZeroOffset, 921
InputOffsetDigiPinsDir, 921
InputOffsetDigiPinsIn, 921
InputOffsetDigiPinsOut, 921
InputOffsetInvalidData, 921
InputOffsetSensorBoolean, 921
InputOffsetSensorMode, 921
InputOffsetSensorRaw, 922
InputOffsetSensorType, 922
InputOffsetSensorValue, 922
InputPinFunction, 922
INT_MAX, 922
INT_MIN, 922
INTF_BT OFF, 922
INTF_BT ON, 922
INTF_CONNECT, 922
INTF_CONNECTBYNAME, 922
INTF_CONNECTREQ, 923
INTF_DISCONNECT, 923
INTF_DISCONNECTALL, 923
INTF_EXTREAD, 923
INTF_FACTORYRESET, 923
INTF_OPENSTREAM, 923
INTF_PINREQ, 923
INTF_REMOVEDevice, 923
INTF_SEARCH, 923
INTF_SENDDATA, 923
INTF_SENDFILE, 924
INTF_SETBTNAME, 924
INTF_SETCMDMODE, 924
INTF_STOPSEARCH, 924
INTF_VISIBILITY, 924
InvalidDataField, 924
IOCTRL_BOOT, 924
IOCTRL_POWERDOWN, 924
IOCtlModuleID, 924
IOCtlModuleName, 924
IOCtlOffsetPowerOn, 925
IOMapRead, 925
IOMapReadByID, 925
IOMapWrite, 925
IOMapWriteByID, 925
KeepAlive, 925
LCD_LINE1, 925
LCD_LINE2, 925
LCD_LINE3, 925
LCD_LINE4, 926
LCD_LINE5, 926
LCD_LINE6, 926
LCD_LINE7, 926
LCD_LINE8, 926
LDR_APPENDNOTPOSSIBLE,
 926
LDR_BTBUSY, 926
LDR_BTCONNECTFAIL, 926
LDR_BT TIMEOUT, 926
LDR_CMD_BOOTCMD, 926
LDR_CMD_BTFACTORYRESET,
 927
LDR_CMD_BTGETADR, 927
LDR_CMD_CLOSE, 927
LDR_CMD_-
 CLOSEMODHANDLE,
 927
LDR_CMD_CROPDATAFILE, 927
LDR_CMD_DELETE, 927
LDR_CMD_-
 DELETEUSERFLASH, 927
LDR_CMD_DEVICEINFO, 927
LDR_CMD_FINDFIRST, 927
LDR_CMD_-
 FINDFIRSTMODULE, 927
LDR_CMD_FINDNEXT, 928
LDR_CMD_-
 FINDNEXTMODULE, 928
LDR_CMD_IOMAPREAD, 928
LDR_CMD_IOMAPWRITE, 928
LDR_CMD_OPENAPPENDDATA,
 928
LDR_CMD_OPENREAD, 928

LDR_CMD_OPENREADLINEAR,
 928
LDR_CMD_OPENWRITE, 928
LDR_CMD_OPENWRITEDATA,
 928
LDR_CMD_-
 OPENWITELINEAR, 928
LDR_CMD_POLLCMD, 929
LDR_CMD_POLLCMDLEN, 929
LDR_CMD_READ, 929
LDR_CMD_RENAMEFILE, 929
LDR_CMD_RESIZEDATAFILE,
 929
LDR_CMD_-
 SEEKFROMCURRENT,
 929
LDR_CMD_SEEKFROMEND, 929
LDR_CMD_SEEKFROMSTART,
 929
LDR_CMD_SETBRICKNAME,
 929
LDR_CMD VERSIONS, 929
LDR_CMD_WRITE, 930
LDR_ENDOFFILE, 930
LDR_EOFEXPECTED, 930
LDR_FILEEXISTS, 930
LDR_FILEISBUSY, 930
LDR_FILEISFULL, 930
LDR_FILENOFOUND, 930
LDR_FILETX_CLOSEERROR,
 930
LDR_FILETX_DSTEXISTS, 930
LDR_FILETX_SRCMISSING, 930
LDR_FILETX_STREAMERROR,
 931
LDR_FILETX_TIMEOUT, 931
LDR_-
 HANDLEALREADYCLOSED,
 931
LDR_ILLEGALFILENAME, 931
LDR_ILLEGALHANDLE, 931
LDR_INPROGRESS, 931
LDR_INVALIDSEEK, 931
LDR_MODULENOTFOUND, 931
LDR_NOLINEARSPACE, 931
LDR_NOMOREFILES, 931
LDR_NOMOREHANDLES, 932
LDR_NOSPACE, 932
LDR_NOTLINEARFILE, 932
LDR_NOWRITEBUFFERS, 932
LDR_OUTOFCBOUNDARY, 932
LDR_REQPIN, 932
LDR_SUCCESS, 932
LDR_UNDEFINEDERROR, 932
LED_BLUE, 932
LED_NONE, 932
LED_RED, 933
LEGO_ADDR_EMETER, 933
LEGO_ADDR_TEMP, 933
LEGO_ADDR_US, 933
ListFiles, 933
LoaderExecuteFunction, 933
LoaderModuleID, 933
LoaderModuleName, 933
LoaderOffsetFreeUserFlash, 933
LoaderOffsetPFunc, 933
LONG_MAX, 934
LONG_MIN, 934
LOWSPEED_CH_NOT_READY,
 934
LOWSPEED_COMMUNICATING,
 934
LOWSPEED_DATA_RECEIVED,
 934
LOWSPEED_DONE, 934
LOWSPEED_ERROR, 934
LOWSPEED_IDLE, 934
LOWSPEED_INIT, 934
LOWSPEED_LOAD_BUFFER, 934
LOWSPEED_NO_ERROR, 935
LOWSPEED RECEIVING, 935
LOWSPEED_RX_ERROR, 935
LOWSPEED_TRANSMITTING,
 935
LOWSPEED_TX_ERROR, 935
LowSpeedModuleID, 935
LowSpeedModuleName, 935
LowSpeedOffsetChannelState, 935
LowSpeedOffsetErrorType, 935
LowSpeedOffsetInBufBuf, 935
LowSpeedOffsetInBufBytesToRx,
 936

LowSpeedOffsetInBufInPtr, 936
LowSpeedOffsetInBufOutPtr, 936
LowSpeedOffsetMode, 936
LowSpeedOffsetNoRestartOnRead,
 936
LowSpeedOffsetOutBufBuf, 936
LowSpeedOffsetOutBufBytesToRx,
 936
LowSpeedOffsetOutBufInPtr, 936
LowSpeedOffsetOutBufOutPtr, 936
LowSpeedOffsetSpeed, 936
LowSpeedOffsetState, 937
LR_COULD_NOT_SAVE, 937
LR_ENTRY_REMOVED, 937
LR_STORE_IS_FULL, 937
LR_SUCCESS, 937
LR_UNKNOWN_ADDR, 937
LSREAD_NO_RESTART_1, 937
LSREAD_NO_RESTART_2, 937
LSREAD_NO_RESTART_3, 937
LSREAD_NO_RESTART_4, 937
LSREAD_NO_RESTART_MASK,
 938
LSREAD_RESTART_ALL, 938
LSREAD_RESTART_NONE, 938
MAILBOX1, 938
MAILBOX10, 938
MAILBOX2, 938
MAILBOX3, 938
MAILBOX4, 938
MAILBOX5, 938
MAILBOX6, 938
MAILBOX7, 939
MAILBOX8, 939
MAILBOX9, 939
MAX_BT_MSG_SIZE, 939
MaxAccelerationField, 939
MaxSpeedField, 939
MemoryManager, 939
MENUICON_CENTER, 939
MENUICON_LEFT, 939
MENUICON_RIGHT, 940
MENUICONS, 940
MENUTEXT, 940
MessageRead, 940
MessageWrite, 940
MI_ADDR_XG1300L, 940
MIN_1, 940
MS_1, 940
MS_10, 940
MS_100, 940
MS_150, 941
MS_2, 941
MS_20, 941
MS_200, 941
MS_250, 941
MS_3, 941
MS_30, 941
MS_300, 941
MS_350, 941
MS_4, 941
MS_40, 942
MS_400, 942
MS_450, 942
MS_5, 942
MS_50, 942
MS_500, 942
MS_6, 942
MS_60, 942
MS_600, 942
MS_7, 942
MS_70, 943
MS_700, 943
MS_8, 943
MS_80, 943
MS_800, 943
MS_9, 943
MS_90, 943
MS_900, 943
MS_ADDR_ACCLNX, 943
MS_ADDR_CMPSNX, 943
MS_ADDR_DISTNX, 944
MS_ADDR_IVSENS, 944
MS_ADDR_LINEldr, 944
MS_ADDR_MTRMUX, 944
MS_ADDR_NRLINK, 944
MS_ADDR_NXTCAM, 944
MS_ADDR_NXTHID, 944
MS_ADDR_NXTMMX, 944
MS_ADDR_NXTSERVO, 944
MS_ADDR_NXTSERVO_EM, 944
MS_ADDR_PFMATE, 945

MS_ADDR_PSPNX, 945
MS_ADDR_RTCLOCK, 945
MS_ADDR_RXMUX, 945
MS_CMD_ADPA_OFF, 945
MS_CMD_ADPA_ON, 945
MS_CMD_DEENERGIZED, 945
MS_CMD_ENERGIZED, 945
NA, 945
NO_ERR, 945
NO_OF_BTNS, 946
NormalizedValueField, 946
NRLINK_CMD_2400, 946
NRLINK_CMD_4800, 946
NRLINK_CMD_FLUSH, 946
NRLINK_CMD_IR_LONG, 946
NRLINK_CMD_IR_SHORT, 946
NRLINK_CMD_RUN_MACRO,
 946
NRLINK_CMD_SET_PF, 946
NRLINK_CMD_SET_RCX, 946
NRLINK_CMD_SET_TRAIN, 947
NRLINK_CMD_TX_RAW, 947
NRLINK_REG_BYTES, 947
NRLINK_REG_DATA, 947
NRLINK_REG_EEPROM, 947
NULL, 947
NXTHID_CMD_ASCII, 947
NXTHID_CMD_DIRECT, 947
NXTHID_CMD_TRANSMIT, 947
NXTHID_MOD_LEFT_ALT, 947
NXTHID_MOD_LEFT_CTRL, 948
NXTHID_MOD_LEFT_GUI, 948
NXTHID_MOD_LEFT_SHIFT,
 948
NXTHID_MOD_NONE, 948
NXTHID_MOD_RIGHT_ALT, 948
NXTHID_MOD_RIGHT_CTRL,
 948
NXTHID_MOD_RIGHT_GUI, 948
NXTHID_MOD_RIGHT_SHIFT,
 948
NXTHID_REG_CMD, 948
NXTHID_REG_DATA, 948
NXTHID_REG_MODIFIER, 949
NXTLL_CMD_BLACK, 949
NXTLL_CMD_EUROPEAN, 949
NXTLL_CMD_INVERT, 949
NXTLL_CMD_POWERDOWN,
 949
NXTLL_CMD_POWERUP, 949
NXTLL_CMD_RESET, 949
NXTLL_CMD_SNAPSHOT, 949
NXTLL_CMD_UNIVERSAL, 949
NXTLL_CMD_USA, 949
NXTLL_CMD_WHITE, 950
NXTLL_REG_AVERAGE, 950
NXTLL_REG_BLACKDATA, 950
NXTLL_REG_BLACKLIMITS,
 950
NXTLL_REG_CALIBRATED, 950
NXTLL_REG_CMD, 950
NXTLL_REG_KD_FACTOR, 950
NXTLL_REG_KD_VALUE, 950
NXTLL_REG_KI_FACTOR, 950
NXTLL_REG_KI_VALUE, 950
NXTLL_REG_KP_FACTOR, 951
NXTLL_REG_KP_VALUE, 951
NXTLL_REG_RAWVOLTAGE,
 951
NXTLL_REG_RESULT, 951
NXTLL_REG_SETPOINT, 951
NXTLL_REG_STEERING, 951
NXTLL_REG_WHITEDATA, 951
NXTLL_REG_WHITELIMITS,
 951
NXTPM_CMD_RESET, 951
NXTPM_REG_CAPACITY, 951
NXTPM_REG_CMD, 952
NXTPM_REG_CURRENT, 952
NXTPM_REG_ERRORCOUNT,
 952
NXTPM_REG_GAIN, 952
NXTPM_REG_MAXCURRENT,
 952
NXTPM_REG_MAXVOLTAGE,
 952
NXTPM_REG_MINCURRENT,
 952
NXTPM_REG_MINVOLTAGE,
 952
NXTPM_REG_POWER, 952
NXTPM_REG_TIME, 952

NXTPM_REG_TOTALPOWER,
953
NXTPM_REG_USERGAIN, 953
NXTPM_REG_VOLTAGE, 953
NXTSE_ZONE_FRONT, 953
NXTSE_ZONE_LEFT, 953
NXTSE_ZONE_NONE, 953
NXTSE_ZONE_RIGHT, 953
NXTSERVO_CMD_EDIT1, 953
NXTSERVO_CMD_EDIT2, 953
NXTSERVO_CMD_GOTO, 953
NXTSERVO_CMD_HALT, 954
NXTSERVO_CMD_INIT, 954
NXTSERVO_CMD_PAUSE, 954
NXTSERVO_CMD_RESET, 954
NXTSERVO_CMD_RESUME, 954
NXTSERVO_EM_CMD_QUIT,
954
NXTSERVO_EM_REG_CMD, 954
NXTSERVO_EM_REG_-
EEPROM_END, 954
NXTSERVO_EM_REG_-
EEPROM_START, 954
NXTSERVO_POS_CENTER, 955
NXTSERVO_POS_MAX, 955
NXTSERVO_POS_MIN, 955
NXTSERVO_QPOS_CENTER, 955
NXTSERVO_QPOS_MAX, 955
NXTSERVO_QPOS_MIN, 955
NXTSERVO_REG_CMD, 955
NXTSERVO_REG_S1_POS, 955
NXTSERVO_REG_S1_QPOS, 955
NXTSERVO_REG_S1_SPEED,
955
NXTSERVO_REG_S2_POS, 956
NXTSERVO_REG_S2_QPOS, 956
NXTSERVO_REG_S2_SPEED,
956
NXTSERVO_REG_S3_POS, 956
NXTSERVO_REG_S3_QPOS, 956
NXTSERVO_REG_S3_SPEED,
956
NXTSERVO_REG_S4_POS, 956
NXTSERVO_REG_S4_QPOS, 956
NXTSERVO_REG_S4_SPEED,
956
NXTSERVO_REG_S5_POS, 956
NXTSERVO_REG_S5_QPOS, 957
NXTSERVO_REG_S5_SPEED,
957
NXTSERVO_REG_S6_POS, 957
NXTSERVO_REG_S6_QPOS, 957
NXTSERVO_REG_S6_SPEED,
957
NXTSERVO_REG_S7_POS, 957
NXTSERVO_REG_S7_QPOS, 957
NXTSERVO_REG_S7_SPEED,
957
NXTSERVO_REG_S8_POS, 957
NXTSERVO_REG_S8_QPOS, 957
NXTSERVO_REG_S8_SPEED,
958
NXTSERVO_REG_VOLTAGE, 958
NXTSERVO_SERVO_1, 958
NXTSERVO_SERVO_2, 958
NXTSERVO_SERVO_3, 958
NXTSERVO_SERVO_4, 958
NXTSERVO_SERVO_5, 958
NXTSERVO_SERVO_6, 958
NXTSERVO_SERVO_7, 958
NXTSERVO_SERVO_8, 958
OPARR_MAX, 959
OPARR_MEAN, 959
OPARR_MIN, 959
OPARR_SORT, 959
OPARR_STD, 959
OPARR_SUM, 959
OPARR_SUMSQR, 959
OUT_A, 959
OUT_AB, 959
OUT_ABC, 959
OUT_AC, 960
OUT_B, 960
OUT_BC, 960
OUT_C, 960
OUT_MODE_BRAKE, 960
OUT_MODE_COAST, 960
OUT_MODE_MOTORON, 960
OUT_MODE_REGMETHOD, 960
OUT_MODE_REGULATED, 960
OUT_OPTION_HOLDATLIMIT,
960

OUT_OPTION_-
 RAMPDOWNTOLIMIT,
 961
OUT_REGMODE_IDLE, 961
OUT_REGMODE_POS, 961
OUT_REGMODE_SPEED, 961
OUT_REGMODE_SYNC, 961
OUT_REGOPTION_NO_-
 SATURATION, 961
OUT_RUNSTATE_HOLD, 961
OUT_RUNSTATE_IDLE, 961
OUT_RUNSTATE_RAMPDOWN,
 961
OUT_RUNSTATE_RAMPUP, 962
OUT_RUNSTATE_RUNNING, 962
OutputModeField, 962
OutputModuleID, 962
OutputModuleName, 962
OutputOffsetActualSpeed, 962
OutputOffsetBlockTachoCount, 962
OutputOffsetFlags, 962
OutputOffsetMaxAccel, 963
OutputOffsetMaxSpeed, 963
OutputOffsetMode, 963
OutputOffsetMotorRPM, 963
OutputOffsetOptions, 963
OutputOffsetOverloaded, 963
OutputOffsetRegDParameter, 963
OutputOffsetRegIParameter, 963
OutputOffsetRegMode, 963
OutputOffsetRegPParameter, 964
OutputOffsetRegulationOptions, 964
OutputOffsetRegulationTime, 964
OutputOffsetRotationCount, 964
OutputOffsetRunState, 964
OutputOffsetSpeed, 964
OutputOffsetSyncTurnParameter,
 964
OutputOffsetTachoCount, 964
OutputOffsetTachoLimit, 964
OutputOptionsField, 965
OverloadField, 965
PF_CHANNEL_1, 965
PF_CHANNEL_2, 965
PF_CHANNEL_3, 965
PF_CHANNEL_4, 965
PF_CMD_BRAKE, 965
PF_CMD_FLOAT, 966
PF_CMD_FWD, 966
PF_CMD_REV, 966
PF_CMD_STOP, 966
PF_CST_CLEAR1_CLEAR2, 966
PF_CST_CLEAR1_SET2, 966
PF_CST_DECREMENT_PWM,
 966
PF_CST_FULL_FWD, 966
PF_CST_FULL_REV, 966
PF_CST_INCREMENT_PWM, 966
PF_CST_SET1_CLEAR2, 967
PF_CST_SET1_SET2, 967
PF_CST_TOGGLE_DIR, 967
PF_FUNC_CLEAR, 967
PF_FUNC_NOCHANGE, 967
PF_FUNC_SET, 967
PF_FUNC_TOGGLE, 967
PF_MODE_COMBO_DIRECT,
 967
PF_MODE_COMBO_PWM, 967
PF_MODE_SINGLE_OUTPUT_-
 CST, 967
PF_MODE_SINGLE_OUTPUT_-
 PWM, 968
PF_MODE_SINGLE_PIN_CONT,
 968
PF_MODE_SINGLE_PIN_TIME,
 968
PF_MODE_TRAIN, 968
PF_OUT_A, 968
PF_OUT_B, 968
PF_PIN_C1, 968
PF_PIN_C2, 968
PF_PWM_BRAKE, 968
PF_PWM_FLOAT, 968
PF_PWM_FWD1, 969
PF_PWM_FWD2, 969
PF_PWM_FWD3, 969
PF_PWM_FWD4, 969
PF_PWM_FWD5, 969
PF_PWM_FWD6, 969
PF_PWM_FWD7, 969
PF_PWM_REV1, 969
PF_PWM_REV2, 969

PF_PWM_REV3, 969
PF_PWM_REV4, 970
PF_PWM_REV5, 970
PF_PWM_REV6, 970
PF_PWM_REV7, 970
PFMATE_CHANNEL_1, 970
PFMATE_CHANNEL_2, 970
PFMATE_CHANNEL_3, 970
PFMATE_CHANNEL_4, 970
PFMATE_CMD_GO, 970
PFMATE_CMD_RAW, 970
PFMATE_MOTORS_A, 971
PFMATE_MOTORS_B, 971
PFMATE_MOTORS_BOTH, 971
PFMATE_REG_A_CMD, 971
PFMATE_REG_A_SPEED, 971
PFMATE_REG_B_CMD, 971
PFMATE_REG_B_SPEED, 971
PFMATE_REG_CHANNEL, 971
PFMATE_REG_CMD, 971
PFMATE_REG_MOTORS, 972
PI, 972
PID_0, 972
PID_1, 972
PID_2, 972
PID_3, 972
PID_4, 972
PID_5, 972
PID_6, 972
PID_7, 972
POOL_MAX_SIZE, 973
PowerField, 973
PROG_ABORT, 973
PROG_ERROR, 973
PROG_IDLE, 973
PROG_OK, 973
PROG_RESET, 973
PROG_RUNNING, 973
PSP_BTNSET1_DOWN, 974
PSP_BTNSET1_L3, 974
PSP_BTNSET1_LEFT, 974
PSP_BTNSET1_R3, 974
PSP_BTNSET1_RIGHT, 974
PSP_BTNSET1_SELECT, 974
PSP_BTNSET1_START, 974
PSP_BTNSET1_UP, 974
PSP_BTNSET2_CIRCLE, 974
PSP_BTNSET2_CROSS, 974
PSP_BTNSET2_L1, 975
PSP_BTNSET2_L2, 975
PSP_BTNSET2_R1, 975
PSP_BTNSET2_R2, 975
PSP_BTNSET2_SQUARE, 975
PSP_BTNSET2_TRIANGLE, 975
PSP_CMD_ANALOG, 975
PSP_CMD_DIGITAL, 975
PSP_REG_BTNSET1, 975
PSP_REG_BTNSET2, 975
PSP_REG_XLEFT, 976
PSP_REG_XRIGHT, 976
PSP_REG_YLEFT, 976
PSP_REG_YRIGHT, 976
RADIAN_PER_DEGREE, 976
RAND_MAX, 976
RandomEx, 976
RandomNumber, 976
RawValueField, 976
RC_PROP_BTNOFF, 976
RC_PROP_DEBUGGING, 977
RC_PROP_SLEEP_TIMEOUT, 977
RC_PROP_SOUND_LEVEL, 977
RCX_AbsVarOp, 977
RCX_AndVarOp, 977
RCX_AutoOffOp, 977
RCX_BatteryLevelOp, 977
RCX_BatteryLevelSrc, 977
RCX_BootModeOp, 977
RCX_CalibrateEventOp, 977
RCX_ClearAllEventsOp, 978
RCX_ClearCounterOp, 978
RCX_ClearMsgOp, 978
RCX_ClearSensorOp, 978
RCX_ClearSoundOp, 978
RCX_ClearTimerOp, 978
RCX_ClickCounterSrc, 978
RCX_ConstantSrc, 978
RCX_CounterSrc, 978
RCX_DatalogOp, 978
RCX_DatalogRawDirectSrc, 979
RCX_DatalogRawIndirectSrc, 979
RCX_DatalogSrcDirectSrc, 979
RCX_DatalogSrcIndirectSrc, 979

RCX_DatalogValueDirectSrc, 979
RCX_DatalogValueIndirectSrc, 979
RCX_DecCounterOp, 979
RCX_DeleteSubOp, 979
RCX_DeleteSubsOp, 979
RCX_DeleteTaskOp, 979
RCX_DeleteTasksOp, 980
RCX_DirectEventOp, 980
RCX_DisplayOp, 980
RCX_DivVarOp, 980
RCX_DurationSrc, 980
RCX_EventStateSrc, 980
RCX_FirmwareVersionSrc, 980
RCX_GlobalMotorStatusSrc, 980
RCX_GOutputDirOp, 980
RCX_GOutputModeOp, 980
RCX_GOutputPowerOp, 981
RCX_HysteresisSrc, 981
RCX_IncCounterOp, 981
RCX_IndirectVarSrc, 981
RCX_InputBooleanSrc, 981
RCX_InputModeOp, 981
RCX_InputModeSrc, 981
RCX_InputRawSrc, 981
RCX_InputTypeOp, 981
RCX_InputTypeSrc, 981
RCX_InputValueSrc, 982
RCX_IRModeOp, 982
RCX_LightOp, 982
RCX_LowerThresholdSrc, 982
RCX_LSBlinkTimeOp, 982
RCX_LSCalibrateOp, 982
RCX_LSHysteresisOp, 982
RCX_LSLowerThreshOp, 982
RCX_LSupperThreshOp, 982
RCX_MessageOp, 982
RCX_MessageSrc, 983
RCX_MulVarOp, 983
RCX_MuteSoundOp, 983
RCX_OnOffFloatOp, 983
RCX_OrVarOp, 983
RCX_OUT_A, 983
RCX_OUT_AB, 983
RCX_OUT_ABC, 983
RCX_OUT_AC, 983
RCX_OUT_B, 983
RCX_OUT_BC, 984
RCX_OUT_C, 984
RCX_OUT_FLOAT, 984
RCX_OUT_FULL, 984
RCX_OUT_FWD, 984
RCX_OUT_HALF, 984
RCX_OUT_LOW, 984
RCX_OUT_OFF, 984
RCX_OUT_ON, 984
RCX_OUT_REV, 984
RCX_OUT_TOGGLE, 985
RCX_OutputDirOp, 985
RCX_OutputPowerOp, 985
RCX_OutputStatusSrc, 985
RCX_PBTurnOffOp, 985
RCX_PingOp, 985
RCX_PlaySoundOp, 985
RCX_PlayToneOp, 985
RCX_PlayToneVarOp, 985
RCX_PollMemoryOp, 985
RCX_PollOp, 986
RCX_ProgramSlotSrc, 986
RCX_RandomSrc, 986
RCX_RemoteKeysReleased, 986
RCX_RemoteOp, 986
RCX_RemoteOutABackward, 986
RCX_RemoteOutAForward, 986
RCX_RemoteOutBBackward, 986
RCX_RemoteOutBForward, 986
RCX_RemoteOutCBackward, 986
RCX_RemoteOutCForward, 987
RCX_RemotePBMessage1, 987
RCX_RemotePBMessage2, 987
RCX_RemotePBMessage3, 987
RCX_RemotePlayASound, 987
RCX_RemoteSelProgram1, 987
RCX_RemoteSelProgram2, 987
RCX_RemoteSelProgram3, 987
RCX_RemoteSelProgram4, 987
RCX_RemoteSelProgram5, 987
RCX_RemoteStopOutOff, 988
RCX_ScoutCounterLimitSrc, 988
RCX_ScoutEventFBSrc, 988
RCX_ScoutLightParamsSrc, 988
RCX_ScoutOp, 988
RCX_ScoutRulesOp, 988

RCX_ScoutRulesSrc, 988
RCX_ScoutTimerLimitSrc, 988
RCX_SelectProgramOp, 988
RCX_SendUARTDataOp, 988
RCX_SetCounterOp, 989
RCX_SetDatalogOp, 989
RCX_SetEventOp, 989
RCX_SetFeedbackOp, 989
RCX_SetPriorityOp, 989
RCX_SetSourceValueOp, 989
RCX_SetTimerLimitOp, 989
RCX_SetVarOp, 989
RCX_SetWatchOp, 989
RCX_SgnVarOp, 989
RCX_SoundOp, 990
RCX_StartTaskOp, 990
RCX_StopAllTasksOp, 990
RCX_StopTaskOp, 990
RCX_SubVarOp, 990
RCX_SumVarOp, 990
RCX_TaskEventsSrc, 990
RCX_TenMSTimerSrc, 990
RCX_TimerSrc, 990
RCX_UARTSetupSrc, 990
RCX_UnlockFirmOp, 991
RCX_UnlockOp, 991
RCX_UnmuteSoundOp, 991
RCX_UploadDatalogOp, 991
RCX_UpperThresholdSrc, 991
RCX_VariableSrc, 991
RCX_ViewSourceValOp, 991
RCX_VLLOp, 991
RCX_WatchSrc, 991
ReadButton, 991
ReadLastResponse, 992
ReadSemData, 992
RegDValueField, 992
RegIValueField, 992
RegModeField, 992
RegPValueField, 993
RESET_ALL, 993
RESET_BLOCK_COUNT, 993
RESET_BLOCKANDTACHO, 993
RESET_COUNT, 993
RESET_NONE, 993
RESET_ROTATION_COUNT, 993
RFID_MODE_CONTINUOUS, 993
RFID_MODE_SINGLE, 994
RFID_MODE_STOP, 994
RICArg, 994
RICImgPoint, 994
RICImgRect, 994
RICMapArg, 995
RICMapElement, 995
RICMapFunction, 995
RICOpcircle, 995
RICOpcopybits, 996
RICOpdescription, 996
RICOpellipse, 996
RICOpline, 997
RICOpnumbox, 997
RICOppixel, 997
RICOppolygon, 998
RICOprect, 998
RICOpsprite, 998
RICOpvarmap, 999
RICPolygonPoints, 999
RICSpriteData, 999
ROTATE_QUEUE, 999
RotationCountField, 1000
RunStateField, 1000
SAMPLERATE_DEFAULT, 1000
SAMPLERATE_MAX, 1000
SAMPLERATE_MIN, 1000
ScaledValueField, 1000
SCHAR_MAX, 1001
SCHAR_MIN, 1001
SCOUT_FXR_ALARM, 1001
SCOUT_FXR_BUG, 1001
SCOUT_FXR_NONE, 1001
SCOUT_FXR_RANDOM, 1001
SCOUT_FXR_SCIENCE, 1001
SCOUT_LIGHT_OFF, 1001
SCOUT_LIGHT_ON, 1001
SCOUT_LR_AVOID, 1001
SCOUT_LR_IGNORE, 1002
SCOUT_LR_OFF_WHEN, 1002
SCOUT_LR_SEEK_DARK, 1002
SCOUT_LR_SEEK_LIGHT, 1002
SCOUT_LR_WAIT_FOR, 1002
SCOUT_MODE_POWER, 1002

SCOUT_MODE_STANDALONE,
1002
SCOUT_MR_CIRCLE_LEFT, 1002
SCOUT_MR_CIRCLE_RIGHT,
1002
SCOUT_MR_FORWARD, 1002
SCOUT_MR_LOOP_A, 1003
SCOUT_MR_LOOP_AB, 1003
SCOUT_MR_LOOP_B, 1003
SCOUT_MR_NO_MOTION, 1003
SCOUT_MR_ZIGZAG, 1003
SCOUT SNDSET_ALARM, 1003
SCOUT SNDSET_BASIC, 1003
SCOUT SNDSET_BUG, 1003
SCOUT SNDSET_NONE, 1003
SCOUT SNDSET_RANDOM,
1003
SCOUT SNDSET_SCIENCE, 1004
SCOUT_SOUND_1_BLINK, 1004
SCOUT_SOUND_2_BLINK, 1004
SCOUT_SOUND_COUNTER1,
1004
SCOUT_SOUND_COUNTER2,
1004
SCOUT_SOUND_ENTER -
BRIGHT, 1004
SCOUT_SOUND_ENTER_DARK,
1004
SCOUT_SOUND_ENTER -
NORMAL, 1004
SCOUT_SOUND_ENTERSA, 1004
SCOUT_SOUND_KEYERROR,
1004
SCOUT_SOUND_MAIL -
RECEIVED, 1005
SCOUT_SOUND_NONE, 1005
SCOUT_SOUND_REMOTE, 1005
SCOUT_SOUND_SPECIAL1, 1005
SCOUT_SOUND_SPECIAL2, 1005
SCOUT_SOUND_SPECIAL3, 1005
SCOUT_SOUND_TIMER1, 1005
SCOUT_SOUND_TIMER2, 1005
SCOUT_SOUND_TIMER3, 1005
SCOUT_SOUND_TOUCH1 -
PRES, 1005
SCOUT_SOUND_TOUCH1_REL,
1006
SCOUT_SOUND_TOUCH2 -
PRES, 1006
SCOUT_SOUND_TOUCH2_REL,
1006
SCOUT_TGS_LONG, 1006
SCOUT_TGS_MEDIUM, 1006
SCOUT_TGS_SHORT, 1006
SCOUT_TR_AVOID, 1006
SCOUT_TR_IGNORE, 1006
SCOUT_TR_OFF_WHEN, 1006
SCOUT_TR_REVERSE, 1006
SCOUT_TR_WAIT_FOR, 1007
SCREEN_BACKGROUND, 1007
SCREEN_LARGE, 1007
SCREEN_MODE_CLEAR, 1007
SCREEN_MODE_RESTORE, 1007
SCREEN_SMALL, 1007
SCREENS, 1007
SEC_1, 1007
SEC_10, 1008
SEC_15, 1008
SEC_2, 1008
SEC_20, 1008
SEC_3, 1008
SEC_30, 1008
SEC_4, 1008
SEC_5, 1008
SEC_6, 1008
SEC_7, 1008
SEC_8, 1009
SEC_9, 1009
SetScreenMode, 1009
SetSleepTimeoutVal, 1009
SHRT_MAX, 1009
SHRT_MIN, 1009
SIZE_OF_BDADDR, 1009
SIZE_OF_BRICK_NAME, 1009
SIZE_OF_BT_CONNECT -
TABLE, 1009
SIZE_OF_BT_DEVICE_TABLE,
1009
SIZE_OF_BT_NAME, 1010
SIZE_OF_BT_PINCODE, 1010
SIZE_OF_BTBUF, 1010

SIZE_OF_CLASS_OF_DEVICE,
1010
SIZE_OF_HSBUF, 1010
SIZE_OF_USBBUF, 1010
SIZE_OF_USBDATA, 1010
SOUND_CLICK, 1010
SOUND_DOUBLE_BEEP, 1010
SOUND_DOWN, 1010
SOUND_FAST_UP, 1011
SOUND_FLAGS_IDLE, 1011
SOUND_FLAGS_RUNNING, 1011
SOUND_FLAGS_UPDATE, 1011
SOUND_LOW_BEEP, 1011
SOUND_MODE_LOOP, 1011
SOUND_MODE_ONCE, 1011
SOUND_MODE_TONE, 1011
SOUND_STATE_FILE, 1011
SOUND_STATE_IDLE, 1011
SOUND_STATE_STOP, 1012
SOUND_STATE_TONE, 1012
SOUND_UP, 1012
SoundGetState, 1012
SoundModuleID, 1012
SoundModuleName, 1012
SoundOffsetDuration, 1012
SoundOffsetFlags, 1012
SoundOffsetFreq, 1012
SoundOffsetMode, 1012
SoundOffsetSampleRate, 1013
SoundOffsetSoundFilename, 1013
SoundOffsetState, 1013
SoundOffsetVolume, 1013
SoundPlayFile, 1013
SoundPlayTone, 1013
SoundSetState, 1013
SPECIALS, 1013
STAT_COMM_PENDING, 1013
STAT_MSG_EMPTY_MAILBOX,
1013
STATUSICON_BATTERY, 1014
STATUSICON_BLUETOOTH,
1014
STATUSICON_USB, 1014
STATUSICON_VM, 1014
STATUSICONS, 1014
STATUSTEXT, 1014
STEPICON_1, 1014
STEPICON_2, 1014
STEPICON_3, 1014
STEPICON_4, 1014
STEPICON_5, 1015
STEPICONS, 1015
STEPLINE, 1015
STOP_REQ, 1015
STROBE_READ, 1015
STROBE_S0, 1015
STROBE_S1, 1015
STROBE_S2, 1015
STROBE_S3, 1015
STROBE_WRITE, 1015
TachoCountField, 1016
TachoLimitField, 1016
TEMP_FQ_1, 1016
TEMP_FQ_2, 1016
TEMP_FQ_4, 1016
TEMP_FQ_6, 1016
TEMP_OS_ONESHOT, 1016
TEMP_POL_HIGH, 1017
TEMP_POL_LOW, 1017
TEMP_REG_CONFIG, 1017
TEMP_REG_TEMP, 1017
TEMP_REG_THIGH, 1017
TEMP_REG_TLOW, 1017
TEMP_RES_10BIT, 1017
TEMP_RES_11BIT, 1017
TEMP_RES_12BIT, 1017
TEMP_RES_9BIT, 1018
TEMP_SD_CONTINUOUS, 1018
TEMP_SD_SHUTDOWN, 1018
TEMP_TM_COMPARATOR, 1018
TEMP_TM_INTERRUPT, 1018
TEXTLINE_1, 1018
TEXTLINE_2, 1018
TEXTLINE_3, 1018
TEXTLINE_4, 1018
TEXTLINE_5, 1018
TEXTLINE_6, 1019
TEXTLINE_7, 1019
TEXTLINE_8, 1019
TEXTLINES, 1019
TIMES_UP, 1019
TONE_A3, 1019

TONE_A4, 1019
TONE_A5, 1019
TONE_A6, 1019
TONE_A7, 1019
TONE_AS3, 1020
TONE_AS4, 1020
TONE_AS5, 1020
TONE_AS6, 1020
TONE_AS7, 1020
TONE_B3, 1020
TONE_B4, 1020
TONE_B5, 1020
TONE_B6, 1020
TONE_B7, 1020
TONE_C4, 1021
TONE_C5, 1021
TONE_C6, 1021
TONE_C7, 1021
TONE_CS4, 1021
TONE_CS5, 1021
TONE_CS6, 1021
TONE_CS7, 1021
TONE_D4, 1021
TONE_D5, 1021
TONE_D6, 1022
TONE_D7, 1022
TONE_DS4, 1022
TONE_DS5, 1022
TONE_DS6, 1022
TONE_DS7, 1022
TONE_E4, 1022
TONE_E5, 1022
TONE_E6, 1022
TONE_E7, 1022
TONE_F4, 1023
TONE_F5, 1023
TONE_F6, 1023
TONE_F7, 1023
TONE_FS4, 1023
TONE_FS5, 1023
TONE_FS6, 1023
TONE_FS7, 1023
TONE_G4, 1023
TONE_G5, 1023
TONE_G6, 1024
TONE_G7, 1024
TONE_GS4, 1024
TONE_GS5, 1024
TONE_GS6, 1024
TONE_GS7, 1024
TOPLINE, 1024
TRAIN_CHANNEL_1, 1024
TRAIN_CHANNEL_2, 1024
TRAIN_CHANNEL_3, 1024
TRAIN_CHANNEL_ALL, 1025
TRAIN_FUNC_DECR_SPEED, 1025
TRAIN_FUNC_INCR_SPEED, 1025
TRAIN_FUNC_STOP, 1025
TRAIN_FUNC_TOGGLE_LIGHT, 1025
TRUE, 1025
TurnRatioField, 1025
TypeField, 1026
UCHAR_MAX, 1026
UF_PENDING_UPDATES, 1026
UF_UPDATE_MODE, 1026
UF_UPDATE_PID_VALUES, 1026
UF_UPDATE_RESET_BLOCK_COUNT, 1026
UF_UPDATE_RESET_COUNT, 1026
UF_UPDATE_RESET_ROTATION_COUNT, 1026
UF_UPDATE_SPEED, 1026
UF_UPDATE_TACHO_LIMIT, 1027
UI_BT_CONNECT_REQUEST, 1027
UI_BT_ERROR_ATTENTION, 1027
UI_BT_PIN_REQUEST, 1027
UI_BT_STATE_CONNECTED, 1027
UI_BT_STATE_OFF, 1027
UI_BT_STATE_VISIBLE, 1027
UI_BUTTON_ENTER, 1027
UI_BUTTON_EXIT, 1027
UI_BUTTON_LEFT, 1027
UI_BUTTON_NONE, 1028
UI_BUTTON_RIGHT, 1028

UI_FLAGS_BUSY, 1028
UI_FLAGS_DISABLE_EXIT, 1028
UI_FLAGS_DISABLE_LEFT_-
RIGHT_ENTER, 1028
UI_FLAGS_ENABLE_STATUS_-
UPDATE, 1028
UI_FLAGS_EXECUTE_LMS_-
FILE, 1028
UI_FLAGS_REDRAW_STATUS,
1028
UI_FLAGS_RESET_SLEEP_-
TIMER, 1028
UI_FLAGS_UPDATE, 1028
UI_STATE_BT_ERROR, 1029
UI_STATE_CONNECT_-
REQUEST, 1029
UI_STATE_DRAW_MENU, 1029
UI_STATE_ENTER_PRESSED,
1029
UI_STATE_EXECUTE_FILE, 1029
UI_STATE_EXECUTING_FILE,
1029
UI_STATE_EXIT_PRESSED, 1029
UI_STATE_INIT_DISPLAY, 1029
UI_STATE_INIT_INTRO, 1029
UI_STATE_INIT_LOW_-
BATTERY, 1029
UI_STATE_INIT_MENU, 1030
UI_STATE_INIT_WAIT, 1030
UI_STATE_LEFT_PRESSED, 1030
UI_STATE_LOW_BATTERY, 1030
UI_STATE_NEXT_MENU, 1030
UI_STATE_RIGHT_PRESSED,
1030
UI_STATE_TEST_BUTTONS,
1030
UI_VM_IDLE, 1030
UI_VM_RESET1, 1030
UI_VM_RESET2, 1031
UI_VM_RUN_FREE, 1031
UI_VM_RUN_PAUSE, 1031
UI_VM_RUN_SINGLE, 1031
UIModuleID, 1031
UIModuleName, 1031
UINT_MAX, 1031
UIOffsetAbortFlag, 1031
UIOffsetBatteryState, 1031
UIOffsetBatteryVoltage, 1032
UIOffsetBluetoothState, 1032
UIOffsetButton, 1032
UIOffsetError, 1032
UIOffsetFlags, 1032
UIOffsetForceOff, 1032
UIOffsetLMSfilename, 1032
UIOffsetOBPPointer, 1032
UIOffsetPMenu, 1032
UIOffsetRechargeable, 1032
UIOffsetRunState, 1033
UIOffsetSleepTimeout, 1033
UIOffsetSleepTimer, 1033
UIOffsetState, 1033
UIOffsetUsbState, 1033
UIOffsetVolume, 1033
ULONG_MAX, 1033
UpdateCalibCacheInfo, 1033
UpdateFlagsField, 1033
US_CMD_CONTINUOUS, 1034
US_CMD_EVENTCAPTURE,
1034
US_CMD_OFF, 1034
US_CMD_SINGLESHT, 1034
US_CMD_WARMRESET, 1034
US_REG_ACTUAL_ZERO, 1034
US_REG_CM_INTERVAL, 1034
US_REG_FACTORY_ACTUAL_-
ZERO, 1034
US_REG_FACTORY_SCALE_-
DIVISOR, 1034
US_REG_FACTORY_SCALE_-
FACTOR, 1035
US_REG_MEASUREMENT_-
UNITS, 1035
US_REG_SCALE_DIVISOR, 1035
US_REG_SCALE_FACTOR, 1035
USB_CMD_READY, 1035
USB_PROTOCOL_OVERHEAD,
1035
USHRT_MAX, 1035
WriteSemData, 1035
XG1300L_REG_2G, 1035
XG1300L_REG_4G, 1035
XG1300L_REG_8G, 1036

XG1300L_REG_ANGLE, 1036
XG1300L_REG_RESET, 1036
XG1300L_REG_TURNRATE, 1036
XG1300L_REG_XAXIS, 1036
XG1300L_REG_YAXIS, 1036
XG1300L_REG_ZAXIS, 1036
XG1300L_SCALE_2G, 1036
XG1300L_SCALE_4G, 1036
XG1300L_SCALE_8G, 1037
NBCInputPortConstants
 IN_1, 544
 IN_2, 544
 IN_3, 544
 IN_4, 544
NBCSensorModeConstants
 IN_MODE_ANGLESTEP, 548
 IN_MODE_BOOLEAN, 548
 IN_MODE_CELSIUS, 548
 IN_MODE_FAHRENHEIT, 548
 IN_MODE_MODEMASK, 548
 IN_MODE_PCTFULLSCALE, 548
 IN_MODE_PERIODCOUNTER,
 548
 IN_MODE_RAW, 549
 IN_MODE_SLOPEMASK, 549
 IN_MODE_TRANSITIONCNT,
 549
NBCSensorTypeConstants
 IN_TYPE_ANGLE, 545
 IN_TYPE_COLORBLUE, 545
 IN_TYPE_COLOREXIT, 545
 IN_TYPE_COLORFULL, 546
 IN_TYPE_COLORGREEN, 546
 IN_TYPE_COLORNONE, 546
 IN_TYPE_COLORRED, 546
 IN_TYPE_CUSTOM, 546
 IN_TYPE_HISPEED, 546
 IN_TYPE_LIGHT_ACTIVE, 546
 IN_TYPE_LIGHT_INACTIVE, 546
 IN_TYPE_LOWSPEED, 546
 IN_TYPE_LOWSPEED_9V, 546
 IN_TYPE_NO_SENSOR, 547
 IN_TYPE_REFLECTION, 547
 IN_TYPE_SOUND_DB, 547
 IN_TYPE_SOUND_DBA, 547
 IN_TYPE_SWITCH, 547
 IN_TYPE_TEMPERATURE, 547
NEQ
 cmpconst, 74
NO_ERR
 CommandModuleConstants, 82
 NBCCCommon.h, 945
NO_OF_BTNS
 ButtonNameConstants, 530
 NBCCCommon.h, 946
NormalizedValueField
 InputFieldConstants, 550
 NBCCCommon.h, 946
NRLLink2400
 MindSensorsAPI, 200
NRLLink4800
 MindSensorsAPI, 200
NRLINK_CMD_2400
 MSNRLLink, 722
 NBCCCommon.h, 946
NRLINK_CMD_4800
 MSNRLLink, 722
 NBCCCommon.h, 946
NRLINK_CMD_FLUSH
 MSNRLLink, 722
 NBCCCommon.h, 946
NRLINK_CMD_IR_LONG
 MSNRLLink, 722
 NBCCCommon.h, 946
NRLINK_CMD_IR_SHORT
 MSNRLLink, 723
 NBCCCommon.h, 946
NRLINK_CMD_RUN_MACRO
 MSNRLLink, 723
 NBCCCommon.h, 946
NRLINK_CMD_SET_PF
 MSNRLLink, 723
 NBCCCommon.h, 946
NRLINK_CMD_SET_RCX
 MSNRLLink, 723
 NBCCCommon.h, 946
NRLINK_CMD_SET_TRAIN
 MSNRLLink, 723
 NBCCCommon.h, 947
NRLINK_CMD_TX_RAW
 MSNRLLink, 723
 NBCCCommon.h, 947

NRLINK_REG_BYTES
 MSNRLink, 723
 NBCCCommon.h, 947

NRLINK_REG_DATA
 MSNRLink, 723
 NBCCCommon.h, 947

NRLINK_REG EEPROM
 MSNRLink, 723
 NBCCCommon.h, 947

NRLinkFlush
 MindSensorsAPI, 201

NRLinkIRLong
 MindSensorsAPI, 201

NRLinkIRShort
 MindSensorsAPI, 201

NRLinkSetPF
 MindSensorsAPI, 202

NRLinkSetRCX
 MindSensorsAPI, 202

NRLinkSetTrain
 MindSensorsAPI, 202

NRLinkTxRaw
 MindSensorsAPI, 203

NULL
 LoaderModuleConstants, 507
 NBCCCommon.h, 947

NumOut
 DisplayModuleFunctions, 343

NumOutEx
 DisplayModuleFunctions, 343

NXT firmware module IDs, 263

NXT firmware module names, 261

NXT Firmware Modules, 74

NXTHID_CMD_ASCII
 NBCCCommon.h, 947
 NXTHIDCommands, 742

NXTHID_CMD_DIRECT
 NBCCCommon.h, 947
 NXTHIDCommands, 742

NXTHID_CMD_TRANSMIT
 NBCCCommon.h, 947
 NXTHIDCommands, 742

NXTHID_MOD_LEFT_ALT
 NBCCCommon.h, 947
 NXTHIDModifiers, 741

NXTHID_MOD_LEFT_CTRL
 NBCCCommon.h, 948
 NXTHIDModifiers, 741

NXTHID_MOD_LEFT_GUI
 NBCCCommon.h, 948
 NXTHIDModifiers, 741

NXTHID_MOD_LEFT_SHIFT
 NBCCCommon.h, 948
 NXTHIDModifiers, 741

NXTHID_MOD_NONE
 NBCCCommon.h, 948
 NXTHIDModifiers, 741

NXTHID_MOD_RIGHT_ALT
 NBCCCommon.h, 948
 NXTHIDModifiers, 741

NXTHID_MOD_RIGHT_CTRL
 NBCCCommon.h, 948
 NXTHIDModifiers, 742

NXTHID_MOD_RIGHT_GUI
 NBCCCommon.h, 948
 NXTHIDModifiers, 742

NXTHID_MOD_RIGHT_SHIFT
 NBCCCommon.h, 948
 NXTHIDModifiers, 742

NXTHID_REG_CMD
 NBCCCommon.h, 948
 NXTHIDRegisters, 740

NXTHID_REG_DATA
 NBCCCommon.h, 948
 NXTHIDRegisters, 740

NXTHID_REG_MODIFIER
 NBCCCommon.h, 949
 NXTHIDRegisters, 740

NXTHIDAsciiMode
 MindSensorsAPI, 203

NXTHIDCommands
 NXTHID_CMD_ASCII, 742
 NXTHID_CMD_DIRECT, 742
 NXTHID_CMD_TRANSMIT, 742

NXTHIDDirectMode
 MindSensorsAPI, 203

NXTHIDLoadCharacter
 MindSensorsAPI, 204

NXTHIDModifiers
 NXTHID_MOD_LEFT_ALT, 741
 NXTHID_MOD_LEFT_CTRL, 741
 NXTHID_MOD_LEFT_GUI, 741

NXTHID_MOD_LEFT_SHIFT,
 741
NXTHID_MOD_NONE, **741**
NXTHID_MOD_RIGHT_ALT, **741**
NXTHID_MOD_RIGHT_CTRL,
 742
NXTHID_MOD_RIGHT_GUI, **742**
NXTHID_MOD_RIGHT_SHIFT,
 742

NXTHIDRegisters
 NXTHID_REG_CMD, **740**
 NXTHID_REG_DATA, **740**
 NXTHID_REG_MODIFIER, **740**

NXTHIDTransmit
 MindSensorsAPI, **204**

NXTLimits
 CHAR_BIT, **785**
 CHAR_MAX, **785**
 CHAR_MIN, **785**
 INT_MAX, **785**
 INT_MIN, **785**
 LONG_MAX, **785**
 LONG_MIN, **785**
 RAND_MAX, **785**
 SCHAR_MAX, **786**
 SCHAR_MIN, **786**
 SHRT_MAX, **786**
 SHRT_MIN, **786**
 UCHAR_MAX, **786**
 UINT_MAX, **786**
 ULONG_MAX, **786**
 USHRT_MAX, **786**

NXTLineLeaderCalibrateBlack
 MindSensorsAPI, **204**

NXTLineLeaderCalibrateWhite
 MindSensorsAPI, **205**

NXTLineLeaderCommands
 NXTLL_CMD_BLACK, **750**
 NXTLL_CMD_EUROPEAN, **750**
 NXTLL_CMD_INVERT, **750**
 NXTLL_CMD_POWERDOWN,
 750
 NXTLL_CMD_POWERUP, **750**
 NXTLL_CMD_RESET, **750**
 NXTLL_CMD_SNAPSHOT, **751**
 NXTLL_CMD_UNIVERSAL, **751**

NXTLL_CMD_USA, **751**
NXTLL_CMD_WHITE, **751**

NXTLineLeaderInvert
 MindSensorsAPI, **205**

NXTLineLeaderPowerDown
 MindSensorsAPI, **206**

NXTLineLeaderPowerUp
 MindSensorsAPI, **206**

NXTLineLeaderRegisters
 NXTLL_REG_AVERAGE, **748**
 NXTLL_REG_BLACKDATA, **748**
 NXTLL_REG_BLACKLIMITS,
 748
 NXTLL_REG_CALIBRATED, **748**
 NXTLL_REG_CMD, **748**
 NXTLL_REG_KD_FACTOR, **748**
 NXTLL_REG_KD_VALUE, **748**
 NXTLL_REG_KI_FACTOR, **748**
 NXTLL_REG_KI_VALUE, **748**
 NXTLL_REG_KP_FACTOR, **749**
 NXTLL_REG_KP_VALUE, **749**
 NXTLL_REG_RAWVOLTAGE,
 749
 NXTLL_REG_RESULT, **749**
 NXTLL_REG_SETPOINT, **749**
 NXTLL_REG_STEERING, **749**
 NXTLL_REG_WHITEDATA, **749**
 NXTLL_REG_WHITELIMITS,
 749

NXTLineLeaderReset
 MindSensorsAPI, **206**

NXTLineLeaderSnapshot
 MindSensorsAPI, **207**

NXTLL_CMD_BLACK
 NBCCommon.h, **949**

NXTLineLeaderCommands, **750**

NXTLL_CMD_EUROPEAN
 NBCCommon.h, **949**

NXTLineLeaderCommands, **750**

NXTLL_CMD_INVERT
 NBCCommon.h, **949**

NXTLineLeaderCommands, **750**

NXTLL_CMD_POWERDOWN
 NBCCommon.h, **949**

NXTLineLeaderCommands, **750**

NXTLL_CMD_POWERUP

NBCCCommon.h, 949
NXTLineLeaderCommands, 750
NXTLL_CMD_RESET
NBCCCommon.h, 949
NXTLineLeaderCommands, 750
NXTLL_CMD_SNAPSHOT
NBCCCommon.h, 949
NXTLineLeaderCommands, 751
NXTLL_CMD_UNIVERSAL
NBCCCommon.h, 949
NXTLineLeaderCommands, 751
NXTLL_CMD_USA
NBCCCommon.h, 949
NXTLineLeaderCommands, 751
NXTLL_CMD_WHITE
NBCCCommon.h, 950
NXTLineLeaderCommands, 751
NXTLL_REG_AVERAGE
NBCCCommon.h, 950
NXTLineLeaderRegisters, 748
NXTLL_REG_BLACKDATA
NBCCCommon.h, 950
NXTLineLeaderRegisters, 748
NXTLL_REG_BLACKLIMITS
NBCCCommon.h, 950
NXTLineLeaderRegisters, 748
NXTLL_REG_CALIBRATED
NBCCCommon.h, 950
NXTLineLeaderRegisters, 748
NXTLL_REG_CMD
NBCCCommon.h, 950
NXTLineLeaderRegisters, 748
NXTLL_REG_KD_FACTOR
NBCCCommon.h, 950
NXTLineLeaderRegisters, 748
NXTLL_REG_KD_VALUE
NBCCCommon.h, 950
NXTLineLeaderRegisters, 748
NXTLL_REG_KI_FACTOR
NBCCCommon.h, 950
NXTLineLeaderRegisters, 748
NXTLL_REG_KI_VALUE
NBCCCommon.h, 950
NXTLineLeaderRegisters, 748
NXTLL_REG_KP_FACTOR
NBCCCommon.h, 951
NXTLineLeaderRegisters, 749
NXTLL_REG_KP_VALUE
NBCCCommon.h, 951
NXTLineLeaderRegisters, 749
NXTLL_REG_RAWVOLTAGE
NBCCCommon.h, 951
NXTLineLeaderRegisters, 749
NXTLL_REG_RESULT
NBCCCommon.h, 951
NXTLineLeaderRegisters, 749
NXTLL_REG_SETPOINT
NBCCCommon.h, 951
NXTLineLeaderRegisters, 749
NXTLL_REG_STEERING
NBCCCommon.h, 951
NXTLineLeaderRegisters, 749
NXTLL_REG_WHITEDATA
NBCCCommon.h, 951
NXTLineLeaderRegisters, 749
NXTLL_REG_WHITELIMITS
NBCCCommon.h, 951
NXTLineLeaderRegisters, 749
NXTPM_CMD_RESET
NBCCCommon.h, 951
NXTPowerMeterCommands, 746
NXTPM_REG_CAPACITY
NBCCCommon.h, 951
NXTPowerMeterRegisters, 744
NXTPM_REG_CMD
NBCCCommon.h, 952
NXTPowerMeterRegisters, 744
NXTPM_REG_CURRENT
NBCCCommon.h, 952
NXTPowerMeterRegisters, 744
NXTPM_REG_ERRORCOUNT
NBCCCommon.h, 952
NXTPowerMeterRegisters, 744
NXTPM_REG_GAIN
NBCCCommon.h, 952
NXTPowerMeterRegisters, 744
NXTPM_REG_MAXCURRENT
NBCCCommon.h, 952
NXTPowerMeterRegisters, 744
NXTPM_REG_MAXVOLTAGE
NBCCCommon.h, 952
NXTPowerMeterRegisters, 744

NXTPM_REG_MINCURRENT
 NBCCCommon.h, 952
 NXTPowerMeterRegisters, 744

NXTPM_REG_MINVOLTAGE
 NBCCCommon.h, 952
 NXTPowerMeterRegisters, 744

NXTPM_REG_POWER
 NBCCCommon.h, 952
 NXTPowerMeterRegisters, 745

NXTPM_REG_TIME
 NBCCCommon.h, 952
 NXTPowerMeterRegisters, 745

NXTPM_REG_TOTALPOWER
 NBCCCommon.h, 953
 NXTPowerMeterRegisters, 745

NXTPM_REG_USERGAIN
 NBCCCommon.h, 953
 NXTPowerMeterRegisters, 745

NXTPM_REG_VOLTAGE
 NBCCCommon.h, 953
 NXTPowerMeterRegisters, 745

NXTPowerMeterCommands
 NXTPM_CMD_RESET, 746

NXTPowerMeterRegisters
 NXTPM_REG_CAPACITY, 744
 NXTPM_REG_CMD, 744
 NXTPM_REG_CURRENT, 744
 NXTPM_REG_ERRORCOUNT,
 744
 NXTPM_REG_GAIN, 744
 NXTPM_REG_MAXCURRENT,
 744
 NXTPM_REG_MAXVOLTAGE,
 744
 NXTPM_REG_MINCURRENT,
 744
 NXTPM_REG_MINVOLTAGE,
 744
 NXTPM_REG_POWER, 745
 NXTPM_REG_TIME, 745
 NXTPM_REG_TOTALPOWER,
 745
 NXTPM_REG_USERGAIN, 745
 NXTPM_REG_VOLTAGE, 745

NXTPowerMeterResetCounters
 MindSensorsAPI, 207

NXTSE_ZONE_FRONT
 NBCCCommon.h, 953
 NXTSumoEyesConstants, 746

NXTSE_ZONE_LEFT
 NBCCCommon.h, 953
 NXTSumoEyesConstants, 746

NXTSE_ZONE_NONE
 NBCCCommon.h, 953
 NXTSumoEyesConstants, 746

NXTSE_ZONE_RIGHT
 NBCCCommon.h, 953
 NXTSumoEyesConstants, 746

NXTSERVO_CMD_EDIT1
 NBCCCommon.h, 953
 NXTServoCommands, 738

NXTSERVO_CMD_EDIT2
 NBCCCommon.h, 953
 NXTServoCommands, 738

NXTSERVO_CMD_GOTO
 NBCCCommon.h, 953
 NXTServoCommands, 738

NXTSERVO_CMD_HALT
 NBCCCommon.h, 954
 NXTServoCommands, 739

NXTSERVO_CMD_INIT
 NBCCCommon.h, 954
 NXTServoCommands, 739

NXTSERVO_CMD_PAUSE
 NBCCCommon.h, 954
 NXTServoCommands, 739

NXTSERVO_CMD_RESET
 NBCCCommon.h, 954
 NXTServoCommands, 739

NXTSERVO_CMD_RESUME
 NBCCCommon.h, 954
 NXTServoCommands, 739

NXTSERVO_EM_CMD_QUIT
 NBCCCommon.h, 954
 NXTServoCommands, 739

NXTSERVO_EM_REG_CMD
 NBCCCommon.h, 954
 NXTServoRegisters, 732

NXTSERVO_EM_REG_EEPROM_-
 END
 NBCCCommon.h, 954
 NXTServoRegisters, 732

NXTSERVO_EM_REG_EEPROM_-
 START
 NBCCommon.h, 954
 NXTServoRegisters, 732

NXTSERVO_POS_CENTER
 NBCCommon.h, 955
 NXTServoPos, 735

NXTSERVO_POS_MAX
 NBCCommon.h, 955
 NXTServoPos, 735

NXTSERVO_POS_MIN
 NBCCommon.h, 955
 NXTServoPos, 736

NXTSERVO_QPOS_CENTER
 NBCCommon.h, 955
 NXTServoQPos, 736

NXTSERVO_QPOS_MAX
 NBCCommon.h, 955
 NXTServoQPos, 736

NXTSERVO_QPOS_MIN
 NBCCommon.h, 955
 NXTServoQPos, 736

NXTSERVO_REG_CMD
 NBCCommon.h, 955
 NXTServoRegisters, 732

NXTSERVO_REG_S1_POS
 NBCCommon.h, 955
 NXTServoRegisters, 732

NXTSERVO_REG_S1_QPOS
 NBCCommon.h, 955
 NXTServoRegisters, 733

NXTSERVO_REG_S1_SPEED
 NBCCommon.h, 955
 NXTServoRegisters, 733

NXTSERVO_REG_S2_POS
 NBCCommon.h, 956
 NXTServoRegisters, 733

NXTSERVO_REG_S2_QPOS
 NBCCommon.h, 956
 NXTServoRegisters, 733

NXTSERVO_REG_S2_SPEED
 NBCCommon.h, 956
 NXTServoRegisters, 733

NXTSERVO_REG_S3_POS
 NBCCommon.h, 956
 NXTServoRegisters, 733

NXTSERVO_REG_S3_QPOS
 NBCCommon.h, 956
 NXTServoRegisters, 733

NXTSERVO_REG_S3_SPEED
 NBCCommon.h, 956
 NXTServoRegisters, 733

NXTSERVO_REG_S4_POS
 NBCCommon.h, 956
 NXTServoRegisters, 733

NXTSERVO_REG_S4_QPOS
 NBCCommon.h, 956
 NXTServoRegisters, 733

NXTSERVO_REG_S4_SPEED
 NBCCommon.h, 956
 NXTServoRegisters, 734

NXTSERVO_REG_S5_POS
 NBCCommon.h, 956
 NXTServoRegisters, 734

NXTSERVO_REG_S5_QPOS
 NBCCommon.h, 957
 NXTServoRegisters, 734

NXTSERVO_REG_S5_SPEED
 NBCCommon.h, 957
 NXTServoRegisters, 734

NXTSERVO_REG_S6_POS
 NBCCommon.h, 957
 NXTServoRegisters, 734

NXTSERVO_REG_S6_QPOS
 NBCCommon.h, 957
 NXTServoRegisters, 734

NXTSERVO_REG_S6_SPEED
 NBCCommon.h, 957
 NXTServoRegisters, 734

NXTSERVO_REG_S7_POS
 NBCCommon.h, 957
 NXTServoRegisters, 734

NXTSERVO_REG_S7_QPOS
 NBCCommon.h, 957
 NXTServoRegisters, 734

NXTSERVO_REG_S7_SPEED
 NBCCommon.h, 957
 NXTServoRegisters, 734

NXTSERVO_REG_S8_POS
 NBCCommon.h, 957
 NXTServoRegisters, 735

NXTSERVO_REG_S8_QPOS

NBCCCommon.h, [957](#)
NXTServoRegisters, [735](#)

NXTSERVO_REG_S8_SPEED
NBCCCommon.h, [958](#)
NXTServoRegisters, [735](#)

NXTSERVO_REG_VOLTAGE
NBCCCommon.h, [958](#)
NXTServoRegisters, [735](#)

NXTSERVO_SERVO_1
NBCCCommon.h, [958](#)
NXTServoNumbers, [737](#)

NXTSERVO_SERVO_2
NBCCCommon.h, [958](#)
NXTServoNumbers, [737](#)

NXTSERVO_SERVO_3
NBCCCommon.h, [958](#)
NXTServoNumbers, [737](#)

NXTSERVO_SERVO_4
NBCCCommon.h, [958](#)
NXTServoNumbers, [737](#)

NXTSERVO_SERVO_5
NBCCCommon.h, [958](#)
NXTServoNumbers, [737](#)

NXTSERVO_SERVO_6
NBCCCommon.h, [958](#)
NXTServoNumbers, [737](#)

NXTSERVO_SERVO_7
NBCCCommon.h, [958](#)
NXTServoNumbers, [737](#)

NXTSERVO_SERVO_8
NBCCCommon.h, [958](#)
NXTServoNumbers, [738](#)

NXTServoCommands
NXTSERVO_CMD_EDIT1, [738](#)
NXTSERVO_CMD_EDIT2, [738](#)
NXTSERVO_CMD_GOTO, [738](#)
NXTSERVO_CMD_HALT, [739](#)
NXTSERVO_CMD_INIT, [739](#)
NXTSERVO_CMD_PAUSE, [739](#)
NXTSERVO_CMD_RESET, [739](#)
NXTSERVO_CMD_RESUME, [739](#)
NXTSERVO_EM_CMD_QUIT, [739](#)

NXTServoEditMacro
MindSensorsAPI, [208](#)

NXTServoGotoMacroAddress

MindSensorsAPI, [208](#)

NXTServoHaltMacro
MindSensorsAPI, [208](#)

NXTServoInit
MindSensorsAPI, [209](#)

NXTServoNumbers
NXTSERVO_SERVO_1, [737](#)
NXTSERVO_SERVO_2, [737](#)
NXTSERVO_SERVO_3, [737](#)
NXTSERVO_SERVO_4, [737](#)
NXTSERVO_SERVO_5, [737](#)
NXTSERVO_SERVO_6, [737](#)
NXTSERVO_SERVO_7, [737](#)
NXTSERVO_SERVO_8, [738](#)

NXTServoPauseMacro
MindSensorsAPI, [209](#)

NXTServoPos
NXTSERVO_POS_CENTER, [735](#)
NXTSERVO_POS_MAX, [735](#)
NXTSERVO_POS_MIN, [736](#)

NXTServoQPos
NXTSERVO_QPOS_CENTER, [736](#)
NXTSERVO_QPOS_MAX, [736](#)
NXTSERVO_QPOS_MIN, [736](#)

NXTServoQuitEdit
MindSensorsAPI, [210](#)

NXTServoRegisters
NXTSERVO_EM_REG_CMD, [732](#)
NXTSERVO_EM_REG_-EEPROM_END, [732](#)
NXTSERVO_EM_REG_-EEPROM_START, [732](#)
NXTSERVO_REG_CMD, [732](#)
NXTSERVO_REG_S1_POS, [732](#)
NXTSERVO_REG_S1_QPOS, [733](#)
NXTSERVO_REG_S1_SPEED, [733](#)
NXTSERVO_REG_S2_POS, [733](#)
NXTSERVO_REG_S2_QPOS, [733](#)
NXTSERVO_REG_S2_SPEED, [733](#)
NXTSERVO_REG_S3_POS, [733](#)
NXTSERVO_REG_S3_QPOS, [733](#)
NXTSERVO_REG_S3_SPEED, [733](#)
NXTSERVO_REG_S4_POS, [733](#)

NXTSERVO_REG_S4_QPOS, [733](#)
NXTSERVO_REG_S4_SPEED,
 [734](#)
NXTSERVO_REG_S5_POS, [734](#)
NXTSERVO_REG_S5_QPOS, [734](#)
NXTSERVO_REG_S5_SPEED,
 [734](#)
NXTSERVO_REG_S6_POS, [734](#)
NXTSERVO_REG_S6_QPOS, [734](#)
NXTSERVO_REG_S6_SPEED,
 [734](#)
NXTSERVO_REG_S7_POS, [734](#)
NXTSERVO_REG_S7_QPOS, [734](#)
NXTSERVO_REG_S7_SPEED,
 [734](#)
NXTSERVO_REG_S8_POS, [735](#)
NXTSERVO_REG_S8_QPOS, [735](#)
NXTSERVO_REG_S8_SPEED,
 [735](#)
NXTSERVO_REG_VOLTAGE, [735](#)

NXTServoReset
 MindSensorsAPI, [210](#)

NXTServoResumeMacro
 MindSensorsAPI, [210](#)

NXTSumoEyesConstants
 NXTSE_ZONE_FRONT, [746](#)
 NXTSE_ZONE_LEFT, [746](#)
 NXTSE_ZONE_NONE, [746](#)
 NXTSE_ZONE_RIGHT, [746](#)

Off
 OutputModuleFunctions, [281](#)

OffEx
 OutputModuleFunctions, [281](#)

OnFwd
 OutputModuleFunctions, [282](#)

OnFwdEx
 OutputModuleFunctions, [282](#)

OnFwdExPID
 OutputModuleFunctions, [282](#)

OnFwdReg
 OutputModuleFunctions, [283](#)

OnFwdRegEx
 OutputModuleFunctions, [283](#)

OnFwdRegExPID
 OutputModuleFunctions, [284](#)

OnFwdRegPID
 OutputModuleFunctions, [284](#)

OnFwdSync
 OutputModuleFunctions, [285](#)

OnFwdSyncEx
 OutputModuleFunctions, [285](#)

OnFwdSyncExPID
 OutputModuleFunctions, [286](#)

OnFwdSyncPID
 OutputModuleFunctions, [286](#)

OnRev
 OutputModuleFunctions, [287](#)

OnRevEx
 OutputModuleFunctions, [287](#)

OnRevExPID
 OutputModuleFunctions, [288](#)

OnRevReg
 OutputModuleFunctions, [288](#)

OnRevRegEx
 OutputModuleFunctions, [289](#)

OnRevRegExPID
 OutputModuleFunctions, [289](#)

OnRevRegPID
 OutputModuleFunctions, [290](#)

OnRevSync
 OutputModuleFunctions, [290](#)

OnRevSyncEx
 OutputModuleFunctions, [291](#)

OnRevSyncExPID
 OutputModuleFunctions, [291](#)

OnRevSyncPID
 OutputModuleFunctions, [292](#)

OPARR_MAX
 ArrayOpConstants, [476](#)
 NBCCommon.h, [959](#)

OPARR_MEAN
 ArrayOpConstants, [476](#)
 NBCCommon.h, [959](#)

OPARR_MIN
 ArrayOpConstants, [477](#)
 NBCCommon.h, [959](#)

OPARR_SORT
 ArrayOpConstants, [477](#)
 NBCCommon.h, [959](#)

OPARR_STD
 ArrayOpConstants, [477](#)

NBCCCommon.h, 959
OPARR_SUM
 ArrayOpConstants, 477
 NBCCCommon.h, 959
OPARR_SUMSQR
 ArrayOpConstants, 477
 NBCCCommon.h, 959
OpenFileAppend
 LoaderModuleFunctions, 467
OpenFileRead
 LoaderModuleFunctions, 467
OpenFileReadLinear
 LoaderModuleFunctions, 468
OUT_A
 NBCCCommon.h, 959
 OutputPortConstants, 559
OUT_AB
 NBCCCommon.h, 959
 OutputPortConstants, 559
OUT_ABC
 NBCCCommon.h, 959
 OutputPortConstants, 559
OUT_AC
 NBCCCommon.h, 960
 OutputPortConstants, 560
OUT_B
 NBCCCommon.h, 960
 OutputPortConstants, 560
OUT_BC
 NBCCCommon.h, 960
 OutputPortConstants, 560
OUT_C
 NBCCCommon.h, 960
 OutputPortConstants, 560
OUT_MODE BRAKE
 NBCCCommon.h, 960
 OutModeConstants, 565
OUT_MODE COAST
 NBCCCommon.h, 960
 OutModeConstants, 565
OUT_MODE MOTORON
 NBCCCommon.h, 960
 OutModeConstants, 565
OUT_MODE REGMETHOD
 NBCCCommon.h, 960
 OutModeConstants, 565
OUT_MODE_REGULATED
 NBCCCommon.h, 960
 OutModeConstants, 565
OUT_OPTION_HOLDATLIMIT
 NBCCCommon.h, 960
 OutOptionConstants, 566
OUT_OPTION_RAMPDOWNTOLIMIT
 NBCCCommon.h, 961
 OutOptionConstants, 566
OUT_REGMODE_IDLE
 NBCCCommon.h, 961
 OutRegModeConstants, 568
OUT_REGMODE_POS
 NBCCCommon.h, 961
 OutRegModeConstants, 568
OUT_REGMODE_SPEED
 NBCCCommon.h, 961
 OutRegModeConstants, 568
OUT_REGMODE_SYNC
 NBCCCommon.h, 961
 OutRegModeConstants, 568
OUT_REGOPTION_NO_-
 SATURATION
 NBCCCommon.h, 961
 OutRegOptionConstants, 566
OUT_RUNSTATE_HOLD
 NBCCCommon.h, 961
 OutRunStateConstants, 567
OUT_RUNSTATE_IDLE
 NBCCCommon.h, 961
 OutRunStateConstants, 567
OUT_RUNSTATE_RAMPDOWN
 NBCCCommon.h, 961
 OutRunStateConstants, 567
OUT_RUNSTATE_RAMPUP
 NBCCCommon.h, 962
 OutRunStateConstants, 567
OUT_RUNSTATE_RUNNING
 NBCCCommon.h, 962
 OutRunStateConstants, 567
OutModeConstants
 OUT_MODE BRAKE, 565
 OUT_MODE COAST, 565
 OUT_MODE MOTORON, 565
 OUT_MODE REGMETHOD, 565
 OUT_MODE_REGULATED, 565

OutOptionConstants
 OUT_OPTION_HOLDATLIMIT,
 [566](#)
 OUT_OPTION_-
 RAMPDOWNTOLIMIT,
 [566](#)
Output field constants, [569](#)
Output module, [79](#)
Output module constants, [79](#)
Output module functions, [276](#)
Output module IOMAP offsets, [575](#)
Output port constants, [559](#)
Output port mode constants, [564](#)
Output port option constants, [565](#)
Output port regulation mode constants,
 [568](#)
Output port run state constants, [566](#)
Output port update flag constants, [562](#)
Output regulation option constants, [566](#)
OutputFieldConstants
 ActualSpeedField, [570](#)
 BlockTachoCountField, [570](#)
 MaxAccelerationField, [571](#)
 MaxSpeedField, [571](#)
 OutputModeField, [571](#)
 OutputOptionsField, [571](#)
 OverloadField, [571](#)
 PowerField, [572](#)
 RegDValueField, [572](#)
 RegIValueField, [572](#)
 RegModeField, [572](#)
 RegPValueField, [573](#)
 RotationCountField, [573](#)
 RunStateField, [573](#)
 TachoCountField, [574](#)
 TachoLimitField, [574](#)
 TurnRatioField, [574](#)
 UpdateFlagsField, [575](#)
OutputIOMAP
 OutputOffsetActualSpeed, [576](#)
 OutputOffsetBlockTachoCount, [576](#)
 OutputOffsetFlags, [576](#)
 OutputOffsetMaxAccel, [576](#)
 OutputOffsetMaxSpeed, [576](#)
 OutputOffsetMode, [576](#)
 OutputOffsetMotorRPM, [576](#)
OutputOffsetOptions, [576](#)
OutputOffsetOverloaded, [576](#)
OutputOffsetRegDParameter, [577](#)
OutputOffsetRegIParameter, [577](#)
OutputOffsetRegMode, [577](#)
OutputOffsetRegPParameter, [577](#)
OutputOffsetRegulationOptions, [577](#)
OutputOffsetRegulationTime, [577](#)
OutputOffsetRotationCount, [577](#)
OutputOffsetRunState, [577](#)
OutputOffsetSpeed, [577](#)
OutputOffsetSyncTurnParameter,
 [578](#)
OutputOffsetTachoCount, [578](#)
OutputOffsetTachoLimit, [578](#)
OutputModeField
 NBCCommon.h, [962](#)
 OutputFieldConstants, [571](#)
OutputModuleFunctions
 Coast, [279](#)
 CoastEx, [280](#)
 Float, [280](#)
 GetOutPwnFreq, [280](#)
 GetOutRegulationOptions, [280](#)
 GetOutRegulationTime, [281](#)
 Off, [281](#)
 OffEx, [281](#)
 OnFwd, [282](#)
 OnFwdEx, [282](#)
 OnFwdExPID, [282](#)
 OnFwdReg, [283](#)
 OnFwdRegEx, [283](#)
 OnFwdRegExPID, [284](#)
 OnFwdRegPID, [284](#)
 OnFwdSync, [285](#)
 OnFwdSyncEx, [285](#)
 OnFwdSyncExPID, [286](#)
 OnFwdSyncPID, [286](#)
 OnRev, [287](#)
 OnRevEx, [287](#)
 OnRevExPID, [288](#)
 OnRevReg, [288](#)
 OnRevRegEx, [289](#)
 OnRevRegExPID, [289](#)
 OnRevRegPID, [290](#)
 OnRevSync, [290](#)

OnRevSyncEx, 291
OnRevSyncExPID, 291
OnRevSyncPID, 292
ResetAllTachoCounts, 292
ResetBlockTachoCount, 293
ResetRotationCount, 293
ResetTachoCount, 293
RotateMotor, 294
RotateMotorEx, 294
RotateMotorExPID, 294
RotateMotorPID, 295
SetOutPwnFreq, 296
SetOutRegulationOptions, 296
SetOutRegulationTime, 296
OutputModuleID
 ModuleIDConstants, 264
 NBCCommon.h, 962
OutputModuleName
 ModuleNameConstants, 263
 NBCCommon.h, 962
OutputOffsetActualSpeed
 NBCCommon.h, 962
 OutputIOMAP, 576
OutputOffsetBlockTachoCount
 NBCCommon.h, 962
 OutputIOMAP, 576
OutputOffsetFlags
 NBCCommon.h, 962
 OutputIOMAP, 576
OutputOffsetMaxAccel
 NBCCommon.h, 963
 OutputIOMAP, 576
OutputOffsetMaxSpeed
 NBCCommon.h, 963
 OutputIOMAP, 576
OutputOffsetMode
 NBCCommon.h, 963
 OutputIOMAP, 576
OutputOffsetMotorRPM
 NBCCommon.h, 963
 OutputIOMAP, 576
OutputOffsetOptions
 NBCCommon.h, 963
 OutputIOMAP, 576
OutputOffsetOverloaded
 NBCCommon.h, 963
 OutputIOMAP, 576
 OutputIOMAP, 576
 OutputOffsetRegDParameter
 NBCCommon.h, 963
 OutputIOMAP, 577
 OutputOffsetRegIParameter
 NBCCommon.h, 963
 OutputIOMAP, 577
 OutputOffsetRegMode
 NBCCommon.h, 963
 OutputIOMAP, 577
 OutputOffsetRegPParameter
 NBCCommon.h, 964
 OutputIOMAP, 577
 OutputOffsetRegulationOptions
 NBCCommon.h, 964
 OutputIOMAP, 577
 OutputOffsetRegulationTime
 NBCCommon.h, 964
 OutputIOMAP, 577
 OutputOffsetRotationCount
 NBCCommon.h, 964
 OutputIOMAP, 577
 OutputOffsetRunState
 NBCCommon.h, 964
 OutputIOMAP, 577
 OutputOffsetSpeed
 NBCCommon.h, 964
 OutputIOMAP, 577
 OutputOffsetSyncTurnParameter
 NBCCommon.h, 964
 OutputIOMAP, 578
 OutputOffsetTachoCount
 NBCCommon.h, 964
 OutputIOMAP, 578
 OutputOffsetTachoLimit
 NBCCommon.h, 964
 OutputIOMAP, 578
 OutputOptionsField
 NBCCommon.h, 965
 OutputFieldConstants, 571
 OutputPortConstants
 OUT_A, 559
 OUT_AB, 559
 OUT_ABC, 559
 OUT_AC, 560
 OUT_B, 560

OUT_BC, [560](#)
OUT_C, [560](#)
OutRegModeConstants
 OUT_REGMODE_IDLE, [568](#)
 OUT_REGMODE_POS, [568](#)
 OUT_REGMODE_SPEED, [568](#)
 OUT_REGMODE_SYNC, [568](#)
OutRegOptionConstants
 OUT_REGOPTION_NO_-_SATURATION, [566](#)
OutRunStateConstants
 OUT_RUNSTATE_HOLD, [567](#)
 OUT_RUNSTATE_IDLE, [567](#)
 OUT_RUNSTATE_RAMPDOWN, [567](#)
 OUT_RUNSTATE_RAMPUP, [567](#)
 OUT_RUNSTATE_RUNNING, [567](#)
OutUFConstants
 UF_PENDING_UPDATES, [562](#)
 UF_UPDATE_MODE, [562](#)
 UF_UPDATE_PID_VALUES, [562](#)
 UF_UPDATE_RESET_BLOCK_-_COUNT, [562](#)
 UF_UPDATE_RESET_COUNT, [562](#)
 UF_UPDATE_RESET_-_ROTATION_COUNT, [563](#)
 UF_UPDATE_SPEED, [563](#)
 UF_UPDATE_TACHO_LIMIT, [563](#)
OverloadField
 NBCCCommon.h, [965](#)
 OutputFieldConstants, [571](#)
PF/IR Train function constants, [683](#)
PF_CHANNEL_1
 NBCCCommon.h, [965](#)
 PFChannelConstants, [681](#)
PF_CHANNEL_2
 NBCCCommon.h, [965](#)
 PFChannelConstants, [681](#)
PF_CHANNEL_3
 NBCCCommon.h, [965](#)
 PFChannelConstants, [681](#)
PF_CHANNEL_4
 NBCCCommon.h, [965](#)
 PFChannelConstants, [681](#)
PF_CMD_BRAKE
 NBCCCommon.h, [965](#)
 PFCmdConstants, [680](#)
PF_CMD_FLOAT
 NBCCCommon.h, [966](#)
 PFCmdConstants, [680](#)
PF_CMD_FWD
 NBCCCommon.h, [966](#)
 PFCmdConstants, [680](#)
PF_CMD_REV
 NBCCCommon.h, [966](#)
 PFCmdConstants, [680](#)
PF_CMD_STOP
 NBCCCommon.h, [966](#)
 PFCmdConstants, [680](#)
PF_CST_CLEAR1_CLEAR2
 NBCCCommon.h, [966](#)
 PFCSTOptions, [687](#)
PF_CST_CLEAR1_SET2
 NBCCCommon.h, [966](#)
 PFCSTOptions, [687](#)
PF_CST_DECREMENT_PWM
 NBCCCommon.h, [966](#)
 PFCSTOptions, [687](#)
PF_CST_FULL_FWD
 NBCCCommon.h, [966](#)
 PFCSTOptions, [687](#)
PF_CST_FULL_REV
 NBCCCommon.h, [966](#)
 PFCSTOptions, [687](#)
PF_CST_INCREMENT_PWM
 NBCCCommon.h, [966](#)
 PFCSTOptions, [687](#)
PF_CST_SET1_CLEAR2
 NBCCCommon.h, [967](#)
 PFCSTOptions, [687](#)
PF_CST_SET1_SET2
 NBCCCommon.h, [967](#)
 PFCSTOptions, [687](#)
PF_CST_TOGGLE_DIR
 NBCCCommon.h, [967](#)
 PFCSTOptions, [688](#)
PF_FUNC_CLEAR
 NBCCCommon.h, [967](#)
 PFPinFuncs, [686](#)
PF_FUNC_NOCHANGE

NBCCCommon.h, 967
PFPinFuncs, 686

PF_FUNC_SET
 NBCCCommon.h, 967
 PFPinFuncs, 686

PF_FUNC_TOGGLE
 NBCCCommon.h, 967
 PFPinFuncs, 686

PF_MODE_COMBO_DIRECT
 NBCCCommon.h, 967
 PFModeConstants, 682

PF_MODE_COMBO_PWM
 NBCCCommon.h, 967
 PFModeConstants, 682

PF_MODE_SINGLE_OUTPUT_CST
 NBCCCommon.h, 967
 PFModeConstants, 682

PF_MODE_SINGLE_OUTPUT_PWM
 NBCCCommon.h, 968
 PFModeConstants, 682

PF_MODE_SINGLE_PIN_CONT
 NBCCCommon.h, 968
 PFModeConstants, 682

PF_MODE_SINGLE_PIN_TIME
 NBCCCommon.h, 968
 PFModeConstants, 682

PF_MODE_TRAIN
 NBCCCommon.h, 968
 PFModeConstants, 682

PF_OUT_A
 NBCCCommon.h, 968
 PFOOutputs, 685

PF_OUT_B
 NBCCCommon.h, 968
 PFOOutputs, 685

PF_PIN_C1
 NBCCCommon.h, 968
 PFPinConstants, 685

PF_PIN_C2
 NBCCCommon.h, 968
 PFPinConstants, 685

PF_PWM BRAKE
 NBCCCommon.h, 968
 PFPWMOptions, 688

PF_PWM_FLOAT
 NBCCCommon.h, 968

PFPWMOptions, 688

PF_PWM_FWD1
 NBCCCommon.h, 969
 PFPWMOptions, 689

PF_PWM_FWD2
 NBCCCommon.h, 969
 PFPWMOptions, 689

PF_PWM_FWD3
 NBCCCommon.h, 969
 PFPWMOptions, 689

PF_PWM_FWD4
 NBCCCommon.h, 969
 PFPWMOptions, 689

PF_PWM_FWD5
 NBCCCommon.h, 969
 PFPWMOptions, 689

PF_PWM_FWD6
 NBCCCommon.h, 969
 PFPWMOptions, 689

PF_PWM_FWD7
 NBCCCommon.h, 969
 PFPWMOptions, 689

PF_PWM_REV1
 NBCCCommon.h, 969
 PFPWMOptions, 689

PF_PWM_REV2
 NBCCCommon.h, 969
 PFPWMOptions, 689

PF_PWM_REV3
 NBCCCommon.h, 969
 PFPWMOptions, 689

PF_PWM_REV4
 NBCCCommon.h, 970
 PFPWMOptions, 690

PF_PWM_REV5
 NBCCCommon.h, 970
 PFPWMOptions, 690

PF_PWM_REV6
 NBCCCommon.h, 970
 PFPWMOptions, 690

PF_PWM_REV7
 NBCCCommon.h, 970
 PFPWMOptions, 690

PFChannelConstants
 PF_CHANNEL_1, 681
 PF_CHANNEL_2, 681

PF_CHANNEL_3, 681
PF_CHANNEL_4, 681
PFCmdConstants
 PF_CMD_BRAKE, 680
 PF_CMD_FLOAT, 680
 PF_CMD_FWD, 680
 PF_CMD_REV, 680
 PF_CMD_STOP, 680
PFCSTOptions
 PF_CST_CLEAR1_CLEAR2, 687
 PF_CST_CLEAR1_SET2, 687
 PF_CST_DECREMENT_PWM,
 687
 PF_CST_FULL_FWD, 687
 PF_CST_FULL_REV, 687
 PF_CST_INCREMENT_PWM, 687
 PF_CST_SET1_CLEAR2, 687
 PF_CST_SET1_SET2, 687
 PF_CST_TOGGLE_DIR, 688
PFMate channel constants, 730
PFMate motor constants, 729
PFMATE_CHANNEL_1
 NBCCommon.h, 970
 PFMateChannelConstants, 730
PFMATE_CHANNEL_2
 NBCCommon.h, 970
 PFMateChannelConstants, 730
PFMATE_CHANNEL_3
 NBCCommon.h, 970
 PFMateChannelConstants, 730
PFMATE_CHANNEL_4
 NBCCommon.h, 970
 PFMateChannelConstants, 730
PFMATE_CMD_GO
 NBCCommon.h, 970
 PFMateConstants, 728
PFMATE_CMD_RAW
 NBCCommon.h, 970
 PFMateConstants, 728
PFMATE_MOTORS_A
 NBCCommon.h, 971
 PFMateMotorConstants, 729
PFMATE_MOTORS_B
 NBCCommon.h, 971
 PFMateMotorConstants, 729
PFMATE_MOTORS_BOTH
 NBCCommon.h, 971
 PFMateMotorConstants, 729
NBCCommon.h, 971
PFMateMotorConstants, 729
PFMATE_REG_A_CMD
 NBCCommon.h, 971
 PFMateConstants, 728
PFMATE_REG_A_SPEED
 NBCCommon.h, 971
 PFMateConstants, 728
PFMATE_REG_B_CMD
 NBCCommon.h, 971
 PFMateConstants, 728
PFMATE_REG_B_SPEED
 NBCCommon.h, 971
 PFMateConstants, 728
PFMATE_REG_CHANNEL
 NBCCommon.h, 971
 PFMateConstants, 729
PFMATE_REG_CMD
 NBCCommon.h, 971
 PFMateConstants, 729
PFMATE_REG_MOTORS
 NBCCommon.h, 972
 PFMateConstants, 729
PFMateChannelConstants
 PFMATE_CHANNEL_1, 730
 PFMATE_CHANNEL_2, 730
 PFMATE_CHANNEL_3, 730
 PFMATE_CHANNEL_4, 730
PFMateConstants
 PFMATE_CMD_GO, 728
 PFMATE_CMD_RAW, 728
 PFMATE_REG_A_CMD, 728
 PFMATE_REG_A_SPEED, 728
 PFMATE_REG_B_CMD, 728
 PFMATE_REG_B_SPEED, 728
 PFMATE_REG_CHANNEL, 729
 PFMATE_REG_CMD, 729
 PFMATE_REG_MOTORS, 729
PFMateMotorConstants
 PFMATE_MOTORS_A, 729
 PFMATE_MOTORS_B, 729
 PFMATE_MOTORS_BOTH, 729
PFMateSend
 MindSensorsAPI, 211
PFMateSendRaw
 MindSensorsAPI, 211

PFModeConstants
 PF_MODE_COMBO_DIRECT,
 682
 PF_MODE_COMBO_PWM, 682
 PF_MODE_SINGLE_OUTPUT_-
 CST, 682
 PF_MODE_SINGLE_OUTPUT_-
 PWM, 682
 PF_MODE_SINGLE_PIN_CONT,
 682
 PF_MODE_SINGLE_PIN_TIME,
 682
 PF_MODE_TRAIN, 682
PFOoutputs
 PF_OUT_A, 685
 PF_OUT_B, 685
PFPinConstants
 PF_PIN_C1, 685
 PF_PIN_C2, 685
PFPinFuncs
 PF_FUNC_CLEAR, 686
 PF_FUNC_NOCHANGE, 686
 PF_FUNC_SET, 686
 PF_FUNC_TOGGLE, 686
PFPWMOptions
 PF_PWM BRAKE, 688
 PF_PWM_FLOAT, 688
 PF_PWM_FWD1, 689
 PF_PWM_FWD2, 689
 PF_PWM_FWD3, 689
 PF_PWM_FWD4, 689
 PF_PWM_FWD5, 689
 PF_PWM_FWD6, 689
 PF_PWM_FWD7, 689
 PF_PWM_REV1, 689
 PF_PWM_REV2, 689
 PF_PWM_REV3, 689
 PF_PWM_REV4, 690
 PF_PWM_REV5, 690
 PF_PWM_REV6, 690
 PF_PWM_REV7, 690
PI
 MiscConstants, 266
 NBCCCommon.h, 972
PID constants, 560
PID_0
 NBCCCommon.h, 972
 PIDConstants, 561
PID_1
 NBCCCommon.h, 972
 PIDConstants, 561
PID_2
 NBCCCommon.h, 972
 PIDConstants, 561
PID_3
 NBCCCommon.h, 972
 PIDConstants, 561
PID_4
 NBCCCommon.h, 972
 PIDConstants, 561
PID_5
 NBCCCommon.h, 972
 PIDConstants, 561
PID_6
 NBCCCommon.h, 972
 PIDConstants, 561
PID_7
 NBCCCommon.h, 972
 PIDConstants, 561
PIDConstants
 PID_0, 561
 PID_1, 561
 PID_2, 561
 PID_3, 561
 PID_4, 561
 PID_5, 561
 PID_6, 561
 PID_7, 561
PlayFile
 SoundModuleFunctions, 353
PlayFileEx
 SoundModuleFunctions, 353
PlayTone
 SoundModuleFunctions, 353
PlayToneEx
 SoundModuleFunctions, 354
PointOut
 DisplayModuleFunctions, 343
PointOutEx
 DisplayModuleFunctions, 344
PolyOut
 DisplayModuleFunctions, 344

PolyOutEx
 DisplayModuleFunctions, 345

POOL_MAX_SIZE
 CommandModuleConstants, 82
 NBCCCommon.h, 973

Power Function channel constants, 681

Power Function command constants, 680

Power Function CST options constants, 686

Power Function mode constants, 681

Power Function output constants, 684

Power Function pin constants, 685

Power Function PWM option constants, 688

Power Function single pin function constants, 685

PowerDown
 IOCtrlModuleFunctions, 461

PowerField
 NBCCCommon.h, 973
 OutputFieldConstants, 572

PowerOn constants, 505

PROG_ABORT
 CommandProgStatus, 502
 NBCCCommon.h, 973

PROG_ERROR
 CommandProgStatus, 502
 NBCCCommon.h, 973

PROG_IDLE
 CommandProgStatus, 502
 NBCCCommon.h, 973

PROG_OK
 CommandProgStatus, 503
 NBCCCommon.h, 973

PROG_RESET
 CommandProgStatus, 503
 NBCCCommon.h, 973

PROG_RUNNING
 CommandProgStatus, 503
 NBCCCommon.h, 973

Program status constants, 502

Property constants, 475

PSP_BTNSET1_DOWN
 MSPSPNXBtnSet1, 719
 NBCCCommon.h, 974

PSP_BTNSET1_L3

PSP_BTNSET1_LEFT
 MSPSPNXBtnSet1, 720
 NBCCCommon.h, 974

PSP_BTNSET1_R3
 MSPSPNXBtnSet1, 720
 NBCCCommon.h, 974

PSP_BTNSET1_RIGHT
 MSPSPNXBtnSet1, 720
 NBCCCommon.h, 974

PSP_BTNSET1_SELECT
 MSPSPNXBtnSet1, 720
 NBCCCommon.h, 974

PSP_BTNSET1_START
 MSPSPNXBtnSet1, 720
 NBCCCommon.h, 974

PSP_BTNSET1_UP
 MSPSPNXBtnSet1, 720
 NBCCCommon.h, 974

PSP_BTNSET2_CIRCLE
 MSPSPNXBtnSet2, 721
 NBCCCommon.h, 974

PSP_BTNSET2_CROSS
 MSPSPNXBtnSet2, 721
 NBCCCommon.h, 974

PSP_BTNSET2_L1
 MSPSPNXBtnSet2, 721
 NBCCCommon.h, 975

PSP_BTNSET2_L2
 MSPSPNXBtnSet2, 721
 NBCCCommon.h, 975

PSP_BTNSET2_R1
 MSPSPNXBtnSet2, 721
 NBCCCommon.h, 975

PSP_BTNSET2_R2
 MSPSPNXBtnSet2, 721
 NBCCCommon.h, 975

PSP_BTNSET2_SQUARE
 MSPSPNXBtnSet2, 721
 NBCCCommon.h, 975

PSP_BTNSET2_TRIANGLE
 MSPSPNXBtnSet2, 721
 NBCCCommon.h, 975

PSP_CMD_ANALOG
 MSPSPNX, 718

NBCCCommon.h, 975
PSP_CMD_DIGITAL
 MSPSPNX, 718
 NBCCCommon.h, 975
PSP_REG_BTNSET1
 MSPSPNX, 718
 NBCCCommon.h, 975
PSP_REG_BTNSET2
 MSPSPNX, 718
 NBCCCommon.h, 975
PSP_REG_XLEFT
 MSPSPNX, 718
 NBCCCommon.h, 976
PSP_REG_XRIGHT
 MSPSPNX, 719
 NBCCCommon.h, 976
PSP_REG_YLEFT
 MSPSPNX, 719
 NBCCCommon.h, 976
PSP_REG_YRIGHT
 MSPSPNX, 719
 NBCCCommon.h, 976
PSPNxAnalog
 MindSensorsAPI, 212
PSPNxDigital
 MindSensorsAPI, 212

RADIANS_PER_DEGREE
 MiscConstants, 266
 NBCCCommon.h, 976
RAND_MAX
 NBCCCommon.h, 976
 NXTLimits, 785
Random
 cstdlibAPI, 474
RandomEx
 NBCCCommon.h, 976
 SysCallConstants, 485
RandomNumber
 NBCCCommon.h, 976
 SysCallConstants, 485
RawValueField
 InputFieldConstants, 550
 NBCCCommon.h, 976
RC_PROP_BTTONOFF
 NBCCCommon.h, 976
RCPropertyConstants, 476
RC_PROP_DEBUGGING
 NBCCCommon.h, 977
 RCPropertyConstants, 476
RC_PROP_SLEEP_TIMEOUT
 NBCCCommon.h, 977
 RCPropertyConstants, 476
RC_PROP_SOUND_LEVEL
 NBCCCommon.h, 977
 RCPropertyConstants, 476
RCPropertyConstants
 RC_PROP_BTTONOFF, 476
 RC_PROP_DEBUGGING, 476
 RC_PROP_SLEEP_TIMEOUT, 476
 RC_PROP_SOUND_LEVEL, 476
RCX and Scout opcode constants, 669
RCX and Scout sound constants, 651
RCX and Scout source constants, 664
RCX constants, 645
RCX IR remote constants, 649
RCX output constants, 646
RCX output direction constants, 648
RCX output mode constants, 647
RCX output power constants, 648
RCX_AbsVarOp
 NBCCCommon.h, 977
 RCXOpcodeConstants, 671
RCX_AndVarOp
 NBCCCommon.h, 977
 RCXOpcodeConstants, 671
RCX_AutoOffOp
 NBCCCommon.h, 977
 RCXOpcodeConstants, 671
RCX_BatteryLevelOp
 NBCCCommon.h, 977
 RCXOpcodeConstants, 671
RCX_BatteryLevelSrc
 NBCCCommon.h, 977
 RCXSourceConstants, 665
RCX_BootModeOp
 NBCCCommon.h, 977
 RCXOpcodeConstants, 671
RCX_CalibrateEventOp
 NBCCCommon.h, 977
 RCXOpcodeConstants, 671
RCX_ClearAllEventsOp

NBCCCommon.h, 978
RCXOpcodeConstants, 671
RCX_ClearCounterOp
 NBCCCommon.h, 978
 RCXOpcodeConstants, 672
RCX_ClearMsgOp
 NBCCCommon.h, 978
 RCXOpcodeConstants, 672
RCX_ClearSensorOp
 NBCCCommon.h, 978
 RCXOpcodeConstants, 672
RCX_ClearSoundOp
 NBCCCommon.h, 978
 RCXOpcodeConstants, 672
RCX_ClearTimerOp
 NBCCCommon.h, 978
 RCXOpcodeConstants, 672
RCX_ClickCounterSrc
 NBCCCommon.h, 978
 RCXSourceConstants, 665
RCX_ConstantSrc
 NBCCCommon.h, 978
 RCXSourceConstants, 665
RCX_CounterSrc
 NBCCCommon.h, 978
 RCXSourceConstants, 665
RCX_DatalogOp
 NBCCCommon.h, 978
 RCXOpcodeConstants, 672
RCX_DatalogRawDirectSrc
 NBCCCommon.h, 979
 RCXSourceConstants, 665
RCX_DatalogRawIndirectSrc
 NBCCCommon.h, 979
 RCXSourceConstants, 665
RCX_DatalogSrcDirectSrc
 NBCCCommon.h, 979
 RCXSourceConstants, 666
RCX_DatalogSrcIndirectSrc
 NBCCCommon.h, 979
 RCXSourceConstants, 666
RCX_DatalogValueDirectSrc
 NBCCCommon.h, 979
 RCXSourceConstants, 666
RCX_DatalogValueIndirectSrc
 NBCCCommon.h, 979
 RCXSourceConstants, 666
RCXSourceConstants, 666
RCX_DecCounterOp
 NBCCCommon.h, 979
 RCXOpcodeConstants, 672
RCX_DeleteSubOp
 NBCCCommon.h, 979
 RCXOpcodeConstants, 672
RCX_DeleteSubsOp
 NBCCCommon.h, 979
 RCXOpcodeConstants, 672
RCX_DeleteTaskOp
 NBCCCommon.h, 979
 RCXOpcodeConstants, 672
RCX_DeleteTasksOp
 NBCCCommon.h, 980
 RCXOpcodeConstants, 673
RCX_DirectEventOp
 NBCCCommon.h, 980
 RCXOpcodeConstants, 673
RCX_DisplayOp
 NBCCCommon.h, 980
 RCXOpcodeConstants, 673
RCX_DivVarOp
 NBCCCommon.h, 980
 RCXOpcodeConstants, 673
RCX_DurationSrc
 NBCCCommon.h, 980
 RCXSourceConstants, 666
RCX_EventStateSrc
 NBCCCommon.h, 980
 RCXSourceConstants, 666
RCX_FirmwareVersionSrc
 NBCCCommon.h, 980
 RCXSourceConstants, 666
RCX_GlobalMotorStatusSrc
 NBCCCommon.h, 980
 RCXSourceConstants, 666
RCX_GOutputDirOp
 NBCCCommon.h, 980
 RCXOpcodeConstants, 673
RCX_GOutputModeOp
 NBCCCommon.h, 980
 RCXOpcodeConstants, 673
RCX_GOutputPowerOp
 NBCCCommon.h, 981
 RCXOpcodeConstants, 673

RCX_HysteresisSrc
 NBCCCommon.h, 981
 RCXSourceConstants, 666

RCX_IncCounterOp
 NBCCCommon.h, 981
 RCXOpcodeConstants, 673

RCX_IndirectVarSrc
 NBCCCommon.h, 981
 RCXSourceConstants, 666

RCX_InputBooleanSrc
 NBCCCommon.h, 981
 RCXSourceConstants, 667

RCX_InputModeOp
 NBCCCommon.h, 981
 RCXOpcodeConstants, 673

RCX_InputModeSrc
 NBCCCommon.h, 981
 RCXSourceConstants, 667

RCX_InputRawSrc
 NBCCCommon.h, 981
 RCXSourceConstants, 667

RCX_InputTypeOp
 NBCCCommon.h, 981
 RCXOpcodeConstants, 673

RCX_InputTypeSrc
 NBCCCommon.h, 981
 RCXSourceConstants, 667

RCX_InputValueSrc
 NBCCCommon.h, 982
 RCXSourceConstants, 667

RCX_IRModeOp
 NBCCCommon.h, 982
 RCXOpcodeConstants, 674

RCX_LightOp
 NBCCCommon.h, 982
 RCXOpcodeConstants, 674

RCX_LowerThresholdSrc
 NBCCCommon.h, 982
 RCXSourceConstants, 667

RCX_LSBlinkTimeOp
 NBCCCommon.h, 982
 RCXOpcodeConstants, 674

RCX_LSCalibrateOp
 NBCCCommon.h, 982
 RCXOpcodeConstants, 674

RCX_LSHysteresisOp

NBCCCommon.h, 982
RCXOpcodeConstants, 674

RCX_LSLowerThreshOp
 NBCCCommon.h, 982
 RCXOpcodeConstants, 674

RCX_LSSupperThreshOp
 NBCCCommon.h, 982
 RCXOpcodeConstants, 674

RCX_MessageOp
 NBCCCommon.h, 982
 RCXOpcodeConstants, 674

RCX_MessageSrc
 NBCCCommon.h, 983
 RCXSourceConstants, 667

RCX_MulVarOp
 NBCCCommon.h, 983
 RCXOpcodeConstants, 674

RCX_MuteSoundOp
 NBCCCommon.h, 983
 RCXOpcodeConstants, 674

RCX_OnOffFloatOp
 NBCCCommon.h, 983
 RCXOpcodeConstants, 675

RCX_OrVarOp
 NBCCCommon.h, 983
 RCXOpcodeConstants, 675

RCX_OUT_A
 NBCCCommon.h, 983
 RCXOutputConstants, 646

RCX_OUT_AB
 NBCCCommon.h, 983
 RCXOutputConstants, 646

RCX_OUT_ABC
 NBCCCommon.h, 983
 RCXOutputConstants, 646

RCX_OUT_AC
 NBCCCommon.h, 983
 RCXOutputConstants, 646

RCX_OUT_B
 NBCCCommon.h, 983
 RCXOutputConstants, 646

RCX_OUT_BC
 NBCCCommon.h, 984
 RCXOutputConstants, 647

RCX_OUT_C
 NBCCCommon.h, 984

RCXOutputConstants, 647
RCX_OUT_FLOAT
 NBCCCommon.h, 984
 RCXOutputMode, 647
RCX_OUT_FULL
 NBCCCommon.h, 984
 RCXOutputPower, 649
RCX_OUT_FWD
 NBCCCommon.h, 984
 RCXOutputDirection, 648
RCX_OUT_HALF
 NBCCCommon.h, 984
 RCXOutputPower, 649
RCX_OUT_LOW
 NBCCCommon.h, 984
 RCXOutputPower, 649
RCX_OUT_OFF
 NBCCCommon.h, 984
 RCXOutputMode, 647
RCX_OUT_ON
 NBCCCommon.h, 984
 RCXOutputMode, 647
RCX_OUT_REV
 NBCCCommon.h, 984
 RCXOutputDirection, 648
RCX_OUT_TOGGLE
 NBCCCommon.h, 985
 RCXOutputDirection, 648
RCX_OutputDirOp
 NBCCCommon.h, 985
 RCXOpcodeConstants, 675
RCX_OutputPowerOp
 NBCCCommon.h, 985
 RCXOpcodeConstants, 675
RCX_OutputStatusSrc
 NBCCCommon.h, 985
 RCXSourceConstants, 667
RCX_PBTurnOffOp
 NBCCCommon.h, 985
 RCXOpcodeConstants, 675
RCX_PingOp
 NBCCCommon.h, 985
 RCXOpcodeConstants, 675
RCX_PlaySoundOp
 NBCCCommon.h, 985
 RCXOpcodeConstants, 675
RCX_PlayToneOp
 NBCCCommon.h, 985
 RCXOpcodeConstants, 675
RCX_PlayToneVarOp
 NBCCCommon.h, 985
 RCXOpcodeConstants, 675
RCX_PollMemoryOp
 NBCCCommon.h, 985
 RCXOpcodeConstants, 675
RCX_PollOp
 NBCCCommon.h, 986
 RCXOpcodeConstants, 676
RCX_ProgramSlotSrc
 NBCCCommon.h, 986
 RCXSourceConstants, 667
RCX_RandomSrc
 NBCCCommon.h, 986
 RCXSourceConstants, 667
RCX_RemoteKeysReleased
 NBCCCommon.h, 986
 RCXRemoteConstants, 650
RCX_RemoteOp
 NBCCCommon.h, 986
 RCXOpcodeConstants, 676
RCX_RemoteOutABackward
 NBCCCommon.h, 986
 RCXRemoteConstants, 650
RCX_RemoteOutAForward
 NBCCCommon.h, 986
 RCXRemoteConstants, 650
RCX_RemoteOutBBackward
 NBCCCommon.h, 986
 RCXRemoteConstants, 650
RCX_RemoteOutBForward
 NBCCCommon.h, 986
 RCXRemoteConstants, 650
RCX_RemoteOutCBackward
 NBCCCommon.h, 986
 RCXRemoteConstants, 650
RCX_RemoteOutCForward
 NBCCCommon.h, 987
 RCXRemoteConstants, 650
RCX_RemotePBMessage1
 NBCCCommon.h, 987
 RCXRemoteConstants, 650
RCX_RemotePBMessage2

NBCCCommon.h, [987](#)
RCXRemoteConstants, [650](#)

RCX_RemotePBMessage3
 NBCCCommon.h, [987](#)
 RCXRemoteConstants, [651](#)

RCX_RemotePlayASound
 NBCCCommon.h, [987](#)
 RCXRemoteConstants, [651](#)

RCX_RemoteSelProgram1
 NBCCCommon.h, [987](#)
 RCXRemoteConstants, [651](#)

RCX_RemoteSelProgram2
 NBCCCommon.h, [987](#)
 RCXRemoteConstants, [651](#)

RCX_RemoteSelProgram3
 NBCCCommon.h, [987](#)
 RCXRemoteConstants, [651](#)

RCX_RemoteSelProgram4
 NBCCCommon.h, [987](#)
 RCXRemoteConstants, [651](#)

RCX_RemoteSelProgram5
 NBCCCommon.h, [987](#)
 RCXRemoteConstants, [651](#)

RCX_RemoteStopOutOff
 NBCCCommon.h, [988](#)
 RCXRemoteConstants, [651](#)

RCX_ScoutCounterLimitSrc
 NBCCCommon.h, [988](#)
 RCXSourceConstants, [668](#)

RCX_ScoutEventFBSrc
 NBCCCommon.h, [988](#)
 RCXSourceConstants, [668](#)

RCX_ScoutLightParamsSrc
 NBCCCommon.h, [988](#)
 RCXSourceConstants, [668](#)

RCX_ScoutOp
 NBCCCommon.h, [988](#)
 RCXOpcodeConstants, [676](#)

RCX_ScoutRulesOp
 NBCCCommon.h, [988](#)
 RCXOpcodeConstants, [676](#)

RCX_ScoutRulesSrc
 NBCCCommon.h, [988](#)
 RCXSourceConstants, [668](#)

RCX_ScoutTimerLimitSrc
 NBCCCommon.h, [988](#)

RCXSourceConstants, [668](#)

RCXSourceConstants, [668](#)

RCXSelectProgramOp
 NBCCCommon.h, [988](#)

RCXOpcodeConstants, [676](#)

RCX_SendUARTDataOp
 NBCCCommon.h, [988](#)
 RCXOpcodeConstants, [676](#)

RCX_SetCounterOp
 NBCCCommon.h, [989](#)
 RCXOpcodeConstants, [676](#)

RCX_SetDatalogOp
 NBCCCommon.h, [989](#)
 RCXOpcodeConstants, [676](#)

RCX_SetEventOp
 NBCCCommon.h, [989](#)
 RCXOpcodeConstants, [676](#)

RCX_SetFeedbackOp
 NBCCCommon.h, [989](#)
 RCXOpcodeConstants, [676](#)

RCX_SetPriorityOp
 NBCCCommon.h, [989](#)
 RCXOpcodeConstants, [677](#)

RCX_SetSourceValueOp
 NBCCCommon.h, [989](#)
 RCXOpcodeConstants, [677](#)

RCX_SetTimerLimitOp
 NBCCCommon.h, [989](#)
 RCXOpcodeConstants, [677](#)

RCX_SetVarOp
 NBCCCommon.h, [989](#)
 RCXOpcodeConstants, [677](#)

RCX_SetWatchOp
 NBCCCommon.h, [989](#)
 RCXOpcodeConstants, [677](#)

RCX_SgnVarOp
 NBCCCommon.h, [989](#)
 RCXOpcodeConstants, [677](#)

RCX_SoundOp
 NBCCCommon.h, [990](#)
 RCXOpcodeConstants, [677](#)

RCX_StartTaskOp
 NBCCCommon.h, [990](#)
 RCXOpcodeConstants, [677](#)

RCX_StopAllTasksOp
 NBCCCommon.h, [990](#)
 RCXOpcodeConstants, [677](#)

RCX_StopTaskOp
 NBCCCommon.h, 990
 RCXOpcodeConstants, 677

RCX_SubVarOp
 NBCCCommon.h, 990
 RCXOpcodeConstants, 678

RCX_SumVarOp
 NBCCCommon.h, 990
 RCXOpcodeConstants, 678

RCX_TaskEventsSrc
 NBCCCommon.h, 990
 RCXSourceConstants, 668

RCX_TenMSTimerSrc
 NBCCCommon.h, 990
 RCXSourceConstants, 668

RCX_TimerSrc
 NBCCCommon.h, 990
 RCXSourceConstants, 668

RCX_UARTSetupSrc
 NBCCCommon.h, 990
 RCXSourceConstants, 668

RCX_UnlockFirmOp
 NBCCCommon.h, 991
 RCXOpcodeConstants, 678

RCX_UnlockOp
 NBCCCommon.h, 991
 RCXOpcodeConstants, 678

RCX_UnmuteSoundOp
 NBCCCommon.h, 991
 RCXOpcodeConstants, 678

RCX_UploadDatalogOp
 NBCCCommon.h, 991
 RCXOpcodeConstants, 678

RCX_UpperThresholdSrc
 NBCCCommon.h, 991
 RCXSourceConstants, 668

RCX_VariableSrc
 NBCCCommon.h, 991
 RCXSourceConstants, 669

RCX_ViewSourceValOp
 NBCCCommon.h, 991
 RCXOpcodeConstants, 678

RCX_VLLOp
 NBCCCommon.h, 991
 RCXOpcodeConstants, 678

RCX_WatchSrc

NBCCCommon.h, 991
RCXSourceConstants, 669

RCXOpcodeConstants
 RCX_AbsVarOp, 671
 RCX_AndVarOp, 671
 RCX_AutoOffOp, 671
 RCX_BatteryLevelOp, 671
 RCX_BootModeOp, 671
 RCX_CalibrateEventOp, 671
 RCX_ClearAllEventsOp, 671
 RCX_ClearCounterOp, 672
 RCX_ClearMsgOp, 672
 RCX_ClearSensorOp, 672
 RCX_ClearSoundOp, 672
 RCX_ClearTimerOp, 672
 RCX_DatalogOp, 672
 RCX_DecCounterOp, 672
 RCX_DeleteSubOp, 672
 RCX_DeleteSubsOp, 672
 RCX_DeleteTaskOp, 672
 RCX_DeleteTasksOp, 673
 RCX_DirectEventOp, 673
 RCX_DisplayOp, 673
 RCX_DivVarOp, 673
 RCX_GOutputDirOp, 673
 RCX_GOutputModeOp, 673
 RCX_GOutputPowerOp, 673
 RCX_IncCounterOp, 673
 RCX_InputModeOp, 673
 RCX_InputTypeOp, 673
 RCX_IRModeOp, 674
 RCX_LightOp, 674
 RCX_LSBlinkTimeOp, 674
 RCX_LSCalibrateOp, 674
 RCX_LSHysteresisOp, 674
 RCX_LSLowerThreshOp, 674
 RCX_LSSupperThreshOp, 674
 RCX_MessageOp, 674
 RCX_MulVarOp, 674
 RCX_MuteSoundOp, 674
 RCX_OnOffFloatOp, 675
 RCX_OrVarOp, 675
 RCX_OutputDirOp, 675
 RCX_OutputPowerOp, 675
 RCX_PBTurnOffOp, 675
 RCX_PingOp, 675

RCX_PlaySoundOp, 675
RCX_PlayToneOp, 675
RCX_PlayToneVarOp, 675
RCX_PollMemoryOp, 675
RCX_PollOp, 676
RCX_RemoteOp, 676
RCX_ScoutOp, 676
RCX_ScoutRulesOp, 676
RCX_SelectProgramOp, 676
RCX_SendUARTDataOp, 676
RCX_SetCounterOp, 676
RCX_SetDatalogOp, 676
RCX_SetEventOp, 676
RCX_SetFeedbackOp, 676
RCX_SetPriorityOp, 677
RCX_SetSourceValueOp, 677
RCX_SetTimerLimitOp, 677
RCX_SetVarOp, 677
RCX_SetWatchOp, 677
RCX_SgnVarOp, 677
RCX_SoundOp, 677
RCX_StartTaskOp, 677
RCX_StopAllTasksOp, 677
RCX_StopTaskOp, 677
RCX_SubVarOp, 678
RCX_SumVarOp, 678
RCX_UnlockFirmOp, 678
RCX_UnlockOp, 678
RCX_UnmuteSoundOp, 678
RCX_UploadDatalogOp, 678
RCX_ViewSourceValOp, 678
RCX_VLLOp, 678

RCXOutputConstants
 RCX_OUT_A, 646
 RCX_OUT_AB, 646
 RCX_OUT_ABC, 646
 RCX_OUT_AC, 646
 RCX_OUT_B, 646
 RCX_OUT_BC, 647
 RCX_OUT_C, 647

RCXOutputDirection
 RCX_OUT_FWD, 648
 RCX_OUT_REV, 648
 RCX_OUT_TOGGLE, 648

RCXOutputMode
 RCX_OUT_FLOAT, 647

RCX_OUT_OFF, 647
RCX_OUT_ON, 647

RCXOutputPower
 RCX_OUT_FULL, 649
 RCX_OUT_HALF, 649
 RCX_OUT_LOW, 649

RCXRemoteConstants
 RCX_RemoteKeysReleased, 650
 RCX_RemoteOutABackward, 650
 RCX_RemoteOutAForward, 650
 RCX_RemoteOutBBackward, 650
 RCX_RemoteOutBForward, 650
 RCX_RemoteOutCBackward, 650
 RCX_RemoteOutCForward, 650
 RCX_RemotePBMessag1, 650
 RCX_RemotePBMessag2, 650
 RCX_RemotePBMessag3, 651
 RCX_RemotePlayASound, 651
 RCX_RemoteSelProgram1, 651
 RCX_RemoteSelProgram2, 651
 RCX_RemoteSelProgram3, 651
 RCX_RemoteSelProgram4, 651
 RCX_RemoteSelProgram5, 651
 RCX_RemoteStopOutOff, 651

RCXSoundConstants
 SOUND_CLICK, 652
 SOUND_DOUBLE_BEEP, 652
 SOUND_DOWN, 652
 SOUND_FAST_UP, 652
 SOUND_LOW_BEEP, 652
 SOUND_UP, 652

RCXSourceConstants
 RCX_BatteryLevelSrc, 665
 RCX_ClickCounterSrc, 665
 RCX_ConstantSrc, 665
 RCX_CounterSrc, 665
 RCX_DatalogRawDirectSrc, 665
 RCX_DatalogRawIndirectSrc, 665
 RCX_DatalogSrcDirectSrc, 666
 RCX_DatalogSrcIndirectSrc, 666
 RCX_DatalogValueDirectSrc, 666
 RCX_DatalogValueIndirectSrc, 666
 RCX_DurationSrc, 666
 RCX_EventStateSrc, 666
 RCX_FirmwareVersionSrc, 666
 RCX_GlobalMotorStatusSrc, 666

RCX_HysteresisSrc, 666
RCX_IndirectVarSrc, 666
RCX_InputBooleanSrc, 667
RCX_InputModeSrc, 667
RCX_InputRawSrc, 667
RCX_InputTypeSrc, 667
RCX_InputValueSrc, 667
RCX_LowerThresholdSrc, 667
RCX_MessageSrc, 667
RCX_OutputStatusSrc, 667
RCX_ProgramSlotSrc, 667
RCX_RandomSrc, 667
RCX_ScoutCounterLimitSrc, 668
RCX_ScoutEventFBSrc, 668
RCX_ScoutLightParamsSrc, 668
RCX_ScoutRulesSrc, 668
RCX_ScoutTimerLimitSrc, 668
RCX_TaskEventsSrc, 668
RCX_TenMSTimerSrc, 668
RCX_TimerSrc, 668
RCX_UARTSetupSrc, 668
RCX_UpperThresholdSrc, 668
RCX_VariableSrc, 669
RCX_WatchSrc, 669

Read
 LoaderModuleFunctions, 468

ReadACCLNxSensitivity
 MindSensorsAPI, 213

ReadACCLNxXOffset
 MindSensorsAPI, 213

ReadACCLNxXRange
 MindSensorsAPI, 213

ReadACCLNxYOffset
 MindSensorsAPI, 214

ReadACCLNxYRange
 MindSensorsAPI, 214

ReadACCLNxZOffset
 MindSensorsAPI, 215

ReadACCLNxZRange
 MindSensorsAPI, 215

ReadButton
 NBCCommon.h, 991
 SysCallConstants, 485

ReadButtonEx
 ButtonModuleFunctions, 380

ReadBytes

 LoaderModuleFunctions, 469

 MindSensorsAPI, 215

 MindSensorsAPI, 216

 MindSensorsAPI, 216

 MindSensorsAPI, 217

 MindSensorsAPI, 217

 LowSpeedModuleFunctions, 319

 LowSpeedModuleFunctions, 319

 LowSpeedModuleFunctions, 320

 LowSpeedModuleFunctions, 320

 LowSpeedModuleFunctions, 321

 LowSpeedModuleFunctions, 321

 LowSpeedModuleFunctions, 321

 NBCCommon.h, 992
 SysCallConstants, 485

 LoaderModuleFunctions, 469

 MindSensorsAPI, 218

 MindSensorsAPI, 218

 MindSensorsAPI, 218

 MindSensorsAPI, 219

 MindSensorsAPI, 219

 MindSensorsAPI, 220

 MindSensorsAPI, 220

ReadNXTPowerMeterErrorCount
 MindSensorsAPI, 220

ReadNXTPowerMeterMaxCurrent
 MindSensorsAPI, 221

ReadNXTPowerMeterMaxVoltage
 MindSensorsAPI, 221

ReadNXTPowerMeterMinCurrent
 MindSensorsAPI, 222

ReadNXTPowerMeterMinVoltage
 MindSensorsAPI, 222

ReadNXTPowerMeterPresentCurrent
 MindSensorsAPI, 222

ReadNXTPowerMeterPresentPower
 MindSensorsAPI, 223

ReadNXTPowerMeterPresentVoltage
 MindSensorsAPI, 223

ReadNXTPowerMeterTotalPowerConsumed
 MindSensorsAPI, 224

ReadNXTServoBatteryVoltage
 MindSensorsAPI, 224

ReadNXTServoPosition
 MindSensorsAPI, 224

ReadNXTServoSpeed
 MindSensorsAPI, 225

ReadSemData
 NBCCommon.h, 992
 SysCallConstants, 485

ReadSensor
 InputModuleFunctions, 305

ReadSensorColorEx
 InputModuleFunctions, 305

ReadSensorColorRaw
 InputModuleFunctions, 306

ReadSensorDIAccl
 DexterIndustriesAPI, 242

ReadSensorDIAccl8
 DexterIndustriesAPI, 242

ReadSensorDIAccl8Raw
 DexterIndustriesAPI, 243

ReadSensorDIAcclDrift
 DexterIndustriesAPI, 243

ReadSensorDIAcclRaw
 DexterIndustriesAPI, 244

ReadSensorDIAcclStatus
 DexterIndustriesAPI, 244

ReadSensorDIGPSDistanceToWaypoint
 DexterIndustriesAPI, 244

ReadSensorDIGPSHeading
 DexterIndustriesAPI, 245

ReadSensorDIGPSHeadingToWaypoint
 DexterIndustriesAPI, 245

ReadSensorDIGPSLatitude
 DexterIndustriesAPI, 245

ReadSensorDIGPSLongitude
 DexterIndustriesAPI, 245

ReadSensorDIGPSRelativeHeading
 DexterIndustriesAPI, 246

ReadSensorDIGPSStatus
 DexterIndustriesAPI, 246

ReadSensorDIGPSTime
 DexterIndustriesAPI, 246

ReadSensorDIGPSVelocity
 DexterIndustriesAPI, 247

ReadSensorDIGyro
 DexterIndustriesAPI, 247

ReadSensorDIGyroRaw
 DexterIndustriesAPI, 247

ReadSensorDIGyroStatus
 DexterIndustriesAPI, 248

ReadSensorDIGyroTemperature
 DexterIndustriesAPI, 248

ReadSensorEMeter
 LowSpeedModuleFunctions, 321

ReadSensorHTAccel
 HiTechnicAPI, 123

ReadSensorHTAngle
 HiTechnicAPI, 123

ReadSensorHTBarometric
 HiTechnicAPI, 124

ReadSensorHTColor
 HiTechnicAPI, 124

ReadSensorHTColor2Active
 HiTechnicAPI, 124

ReadSensorHTColorNum
 HiTechnicAPI, 125

ReadSensorHTCompass
 HiTechnicAPI, 125

ReadSensorHTEOPD
 HiTechnicAPI, 126

ReadSensorHTGyro
 HiTechnicAPI, 126

ReadSensorHTIRReceiver

HiTechnicAPI, 126
ReadSensorHTIRReceiverEx
 HiTechnicAPI, 127
ReadSensorHTIRSeeker
 HiTechnicAPI, 127
ReadSensorHTIRSeeker2AC
 HiTechnicAPI, 127
ReadSensorHTIRSeeker2Addr
 HiTechnicAPI, 128
ReadSensorHTIRSeeker2DC
 HiTechnicAPI, 128
ReadSensorHTIRSeekerDir
 HiTechnicAPI, 129
ReadSensorHTMagnet
 HiTechnicAPI, 129
ReadSensorHTNormalizedColor
 HiTechnicAPI, 130
ReadSensorHTNormalizedColor2Active
 HiTechnicAPI, 130
ReadSensorHTProtoAllAnalog
 HiTechnicAPI, 130
ReadSensorHTProtoAnalog
 HiTechnicAPI, 131
ReadSensorHTProtoDigital
 HiTechnicAPI, 131
ReadSensorHTProtoDigitalControl
 HiTechnicAPI, 132
ReadSensorHTRawColor
 HiTechnicAPI, 132
ReadSensorHTRawColor2
 HiTechnicAPI, 133
ReadSensorHTSuperProAllAnalog
 HiTechnicAPI, 133
ReadSensorHTSuperProAnalog
 HiTechnicAPI, 134
ReadSensorHTSuperProAnalogOut
 HiTechnicAPI, 134
ReadSensorHTSuperProDigital
 HiTechnicAPI, 135
ReadSensorHTSuperProDigitalControl
 HiTechnicAPI, 135
ReadSensorHTSuperProLED
 HiTechnicAPI, 135
ReadSensorHTSuperProProgramControl
 HiTechnicAPI, 136
ReadSensorHTSuperProStrobe
 HiTechnicAPI, 136
ReadSensorHTTouchMultiplexer
 HiTechnicAPI, 136
ReadSensorMIXG1300L
 MicroinfinityAPI, 252
ReadSensorMIXG1300LScale
 MicroinfinityAPI, 252
ReadSensorMSAccel
 MindSensorsAPI, 225
ReadSensorMSCompass
 MindSensorsAPI, 226
ReadSensorMSDROD
 MindSensorsAPI, 226
ReadSensorMSPlayStation
 MindSensorsAPI, 226
ReadSensorMSPressure
 MindSensorsAPI, 227
ReadSensorMSPressureRaw
 MindSensorsAPI, 227
ReadSensorMSRTClock
 MindSensorsAPI, 228
ReadSensorMSTilt
 MindSensorsAPI, 228
ReadSensorNXTSumoEyes
 MindSensorsAPI, 229
ReadSensorTemperature
 LowSpeedModuleFunctions, 322
ReadSensorUS
 LowSpeedModuleFunctions, 322
ReadSensorUSEx
 LowSpeedModuleFunctions, 323
RebootInFirmwareMode
 IOCtrlModuleFunctions, 461
ReceiveMessage
 CommModuleFunctions, 415
ReceiveRemoteBool
 CommModuleFunctions, 416
ReceiveRemoteMessageEx
 CommModuleFunctions, 416
ReceiveRemoteNumber
 CommModuleFunctions, 417
ReceiveRemoteString
 CommModuleFunctions, 417
RectOut
 DisplayModuleFunctions, 345
RectOutEx

DisplayModuleFunctions, 345
RegDValueField
 NBCCCommon.h, 992
 OutputFieldConstants, 572
RegIValueField
 NBCCCommon.h, 992
 OutputFieldConstants, 572
RegModeField
 NBCCCommon.h, 992
 OutputFieldConstants, 572
RegPValueField
 NBCCCommon.h, 993
 OutputFieldConstants, 573
Remote connection constants, 619
Remote control (direct commands) errors, 501
RemoteBluetoothFactoryReset
 CommModuleSCFunctions, 450
RemoteCloseFile
 CommModuleSCFunctions, 450
RemoteConnectionIdle
 CommModuleFunctions, 418
RemoteConnectionWrite
 CommModuleFunctions, 418
RemoteDatalogRead
 CommModuleDCFunctions, 436
RemoteDatalogSetTimes
 CommModuleDCFunctions, 436
RemoteDeleteFile
 CommModuleSCFunctions, 450
RemoteDeleteUserFlash
 CommModuleSCFunctions, 451
RemoteFindFirstFile
 CommModuleSCFunctions, 451
RemoteFindNextFile
 CommModuleSCFunctions, 452
RemoteGetBatteryLevel
 CommModuleDCFunctions, 436
RemoteGetBluetoothAddress
 CommModuleSCFunctions, 452
RemoteGetConnectionCount
 CommModuleDCFunctions, 437
RemoteGetConnectionName
 CommModuleDCFunctions, 437
RemoteGetContactCount
 CommModuleDCFunctions, 438
RemoteGetContactName
 CommModuleDCFunctions, 438
RemoteGetCurrentProgramName
 CommModuleDCFunctions, 438
RemoteGetDeviceInfo
 CommModuleSCFunctions, 453
RemoteGetFirmwareVersion
 CommModuleSCFunctions, 453
RemoteGetInputValues
 CommModuleDCFunctions, 439
RemoteGetOutputState
 CommModuleDCFunctions, 439
RemoteGetProperty
 CommModuleDCFunctions, 439
RemoteIOMapRead
 CommModuleSCFunctions, 454
RemoteIOMapWriteBytes
 CommModuleSCFunctions, 454
RemoteIOMapWriteValue
 CommModuleSCFunctions, 455
RemoteKeepAlive
 CommModuleDCFunctions, 440
RemoteLowspeedGetStatus
 CommModuleDCFunctions, 440
RemoteLowspeedRead
 CommModuleDCFunctions, 441
RemoteLowspeedWrite
 CommModuleDCFunctions, 441
RemoteMessageRead
 CommModuleDCFunctions, 441
RemoteMessageWrite
 CommModuleDCFunctions, 442
RemoteOpenAppendData
 CommModuleSCFunctions, 455
RemoteOpenRead
 CommModuleSCFunctions, 456
RemoteOpenWrite
 CommModuleSCFunctions, 456
RemoteOpenWriteData
 CommModuleSCFunctions, 457
RemoteOpenWriteLinear
 CommModuleSCFunctions, 457
RemotePlaySoundFile
 CommModuleDCFunctions, 442
RemotePlayTone
 CommModuleDCFunctions, 443

RemotePollCommand
 CommModuleSCFunctions, 458

RemotePollCommandLength
 CommModuleSCFunctions, 458

RemoteRead
 CommModuleSCFunctions, 459

RemoteRenameFile
 CommModuleSCFunctions, 459

RemoteResetMotorPosition
 CommModuleDCFunctions, 443

RemoteResetScaledValue
 CommModuleDCFunctions, 443

RemoteResetTachoCount
 CommModuleDCFunctions, 444

RemoteSetBrickName
 CommModuleSCFunctions, 460

RemoteSetInputMode
 CommModuleDCFunctions, 444

RemoteSetOutputState
 CommModuleDCFunctions, 445

Remote SetProperty
 CommModuleDCFunctions, 445

RemoteStartProgram
 CommModuleDCFunctions, 446

RemoteStopProgram
 CommModuleDCFunctions, 446

RemoteStopSound
 CommModuleDCFunctions, 447

RemoteWrite
 CommModuleSCFunctions, 460

RenameFile
 LoaderModuleFunctions, 470

RESET_ALL
 NBCCommon.h, 993
 TachoResetConstants, 564

RESET_BLOCK_COUNT
 NBCCommon.h, 993
 TachoResetConstants, 564

RESET_BLOCKANDTACHO
 NBCCommon.h, 993
 TachoResetConstants, 564

RESET_COUNT
 NBCCommon.h, 993
 TachoResetConstants, 564

RESET_NONE
 NBCCommon.h, 993

TachoResetConstants, 564

RESET_ROTATION_COUNT
 NBCCommon.h, 993
 TachoResetConstants, 564

ResetAllTachoCounts
 OutputModuleFunctions, 292

ResetBlockTachoCount
 OutputModuleFunctions, 293

ResetHTBarometricCalibration
 HiTechnicAPI, 137

ResetMIXG1300L
 MicroinfinityAPI, 252

ResetRotationCount
 OutputModuleFunctions, 293

ResetSensor
 InputModuleFunctions, 306

ResetSensorHTAngle
 HiTechnicAPI, 137

ResetSleepTimer
 CommandModuleFunctions, 369

ResetTachoCount
 OutputModuleFunctions, 293

ResizeFile
 LoaderModuleFunctions, 470

ResolveHandle
 LoaderModuleFunctions, 470

RFID_MODE_CONTINUOUS
 CTRFIDModeConstants, 753
 NBCCommon.h, 993

RFID_MODE_SINGLE
 CTRFIDModeConstants, 753
 NBCCommon.h, 994

RFID_MODE_STOP
 CTRFIDModeConstants, 753
 NBCCommon.h, 994

RFIDInit
 CodatexAPI, 237

RFIDMode
 CodatexAPI, 237

RFIDRead
 CodatexAPI, 238

RFIDReadContinuous
 CodatexAPI, 238

RFIDReadSingle
 CodatexAPI, 238

RFIDStatus

CodatexAPI, 239
RFIDStop
 CodatexAPI, 239
RIC Macro Wrappers, 253
RICArg
 NBCCCommon.h, 994
 RICMacros, 256
RICImgPoint
 NBCCCommon.h, 994
 RICMacros, 256
RICImgRect
 NBCCCommon.h, 994
 RICMacros, 256
RICMacros
 RICArg, 256
 RICImgPoint, 256
 RICImgRect, 256
 RICMapArg, 256
 RICMapElement, 257
 RICMapFunction, 257
 RICOOpCircle, 257
 RICOOpCopyBits, 258
 RICOOpDescription, 258
 RICOOpEllipse, 258
 RICOOpLine, 259
 RICOOpNumBox, 259
 RICOOpPixel, 259
 RICOOpPolygon, 259
 RICOOpRect, 260
 RICOOpSprite, 260
 RICOOpVarMap, 261
 RICPolygonPoints, 261
 RICSpriteData, 261
RICMapArg
 NBCCCommon.h, 995
 RICMacros, 256
RICMapElement
 NBCCCommon.h, 995
 RICMacros, 257
RICMapFunction
 NBCCCommon.h, 995
 RICMacros, 257
RICOOpCircle
 NBCCCommon.h, 995
 RICMacros, 257
RICOOpCopyBits

NBCCCommon.h, 996
RICMacros, 258
RICOOpDescription
 NBCCCommon.h, 996
 RICMacros, 258
RICOOpEllipse
 NBCCCommon.h, 996
 RICMacros, 258
RICOOpLine
 NBCCCommon.h, 997
 RICMacros, 259
RICOOpNumBox
 NBCCCommon.h, 997
 RICMacros, 259
RICOOpPixel
 NBCCCommon.h, 997
 RICMacros, 259
RICOOpPolygon
 NBCCCommon.h, 998
 RICMacros, 259
RICOOpRect
 NBCCCommon.h, 998
 RICMacros, 260
RICOOpSprite
 NBCCCommon.h, 998
 RICMacros, 260
RICOOpVarMap
 NBCCCommon.h, 999
 RICMacros, 261
RICPolygonPoints
 NBCCCommon.h, 999
 RICMacros, 261
RICSpriteData
 NBCCCommon.h, 999
 RICMacros, 261
ROTATE_QUEUE
 CommandVMState, 496
 NBCCCommon.h, 999
RotateMotor
 OutputModuleFunctions, 294
RotateMotorEx
 OutputModuleFunctions, 294
RotateMotorExPID
 OutputModuleFunctions, 294
RotateMotorPID
 OutputModuleFunctions, 295

RotationCountField
 NBCCCommon.h, 1000
 OutputFieldConstants, 573

RS485Control
 CommModuleFunctions, 418

RS485Disable
 CommModuleFunctions, 419

RS485Enable
 CommModuleFunctions, 419

RS485Initialize
 CommModuleFunctions, 420

RS485Read
 CommModuleFunctions, 420

RS485ReadEx
 CommModuleFunctions, 420

RS485Status
 CommModuleFunctions, 421

RS485Uart
 CommModuleFunctions, 421

RS485Write
 CommModuleFunctions, 422

RunNRLinkMacro
 MindSensorsAPI, 229

RunStateField
 NBCCCommon.h, 1000
 OutputFieldConstants, 573

SAMPLERATE_DEFAULT
 NBCCCommon.h, 1000
 SoundMisc, 521

SAMPLERATE_MAX
 NBCCCommon.h, 1000
 SoundMisc, 522

SAMPLERATE_MIN
 NBCCCommon.h, 1000
 SoundMisc, 522

ScaledValueField
 InputFieldConstants, 550
 NBCCCommon.h, 1000

SCHAR_MAX
 NBCCCommon.h, 1001
 NXTLimits, 786

SCHAR_MIN
 NBCCCommon.h, 1001
 NXTLimits, 786

Scout constants, 653

Scout light constants, 653

Scout light rule constants, 661

Scout mode constants, 658

Scout motion rule constants, 659

Scout sound constants, 654

Scout sound set constants, 657

Scout special effect constants, 663

Scout touch rule constants, 660

Scout transmit rule constants, 662

SCOUT_FXR_ALARM
 NBCCCommon.h, 1001
 ScoutSpecialEffectConstants, 663

SCOUT_FXR_BUG
 NBCCCommon.h, 1001
 ScoutSpecialEffectConstants, 663

SCOUT_FXR_NONE
 NBCCCommon.h, 1001
 ScoutSpecialEffectConstants, 663

SCOUT_FXR_RANDOM
 NBCCCommon.h, 1001
 ScoutSpecialEffectConstants, 664

SCOUT_FXR_SCIENCE
 NBCCCommon.h, 1001
 ScoutSpecialEffectConstants, 664

SCOUT_LIGHT_OFF
 NBCCCommon.h, 1001
 ScoutLightConstants, 654

SCOUT_LIGHT_ON
 NBCCCommon.h, 1001
 ScoutLightConstants, 654

SCOUT_LR_AVOID
 NBCCCommon.h, 1001
 ScoutLightRuleConstants, 662

SCOUT_LR_IGNORE
 NBCCCommon.h, 1002
 ScoutLightRuleConstants, 662

SCOUT_LR_OFF_WHEN
 NBCCCommon.h, 1002
 ScoutLightRuleConstants, 662

SCOUT_LR_SEEK_DARK
 NBCCCommon.h, 1002
 ScoutLightRuleConstants, 662

SCOUT_LR_SEEK_LIGHT
 NBCCCommon.h, 1002
 ScoutLightRuleConstants, 662

SCOUT_LR_WAIT_FOR

NBCCCommon.h, 1002
ScoutLightRuleConstants, 662

SCOUT_MODE_POWER
NBCCCommon.h, 1002
ScoutModeConstants, 659

SCOUT_MODE_STANDALONE
NBCCCommon.h, 1002
ScoutModeConstants, 659

SCOUT_MR_CIRCLE_LEFT
NBCCCommon.h, 1002
ScoutMotionRuleConstants, 659

SCOUT_MR_CIRCLE_RIGHT
NBCCCommon.h, 1002
ScoutMotionRuleConstants, 659

SCOUT_MR_FORWARD
NBCCCommon.h, 1002
ScoutMotionRuleConstants, 659

SCOUT_MR_LOOP_A
NBCCCommon.h, 1003
ScoutMotionRuleConstants, 660

SCOUT_MR_LOOP_AB
NBCCCommon.h, 1003
ScoutMotionRuleConstants, 660

SCOUT_MR_LOOP_B
NBCCCommon.h, 1003
ScoutMotionRuleConstants, 660

SCOUT_MR_NO_MOTION
NBCCCommon.h, 1003
ScoutMotionRuleConstants, 660

SCOUT_MR_ZIGZAG
NBCCCommon.h, 1003
ScoutMotionRuleConstants, 660

SCOUT SNDSET_ALARM
NBCCCommon.h, 1003
ScoutSndSetConstants, 658

SCOUT SNDSET_BASIC
NBCCCommon.h, 1003
ScoutSndSetConstants, 658

SCOUT SNDSET_BUG
NBCCCommon.h, 1003
ScoutSndSetConstants, 658

SCOUT SNDSET_NONE
NBCCCommon.h, 1003
ScoutSndSetConstants, 658

SCOUT SNDSET_RANDOM
NBCCCommon.h, 1003

ScoutSndSetConstants, 658

SCOUT SNDSET_SCIENCE
NBCCCommon.h, 1004
ScoutSndSetConstants, 658

SCOUT_SOUND_1_BLINK
NBCCCommon.h, 1004
ScoutSoundConstants, 655

SCOUT_SOUND_2_BLINK
NBCCCommon.h, 1004
ScoutSoundConstants, 655

SCOUT_SOUND_COUNTER1
NBCCCommon.h, 1004
ScoutSoundConstants, 655

SCOUT_SOUND_COUNTER2
NBCCCommon.h, 1004
ScoutSoundConstants, 655

SCOUT_SOUND_ENTER_BRIGHT
NBCCCommon.h, 1004
ScoutSoundConstants, 655

SCOUT_SOUND_ENTER_DARK
NBCCCommon.h, 1004
ScoutSoundConstants, 655

SCOUT_SOUND_ENTER_NORMAL
NBCCCommon.h, 1004
ScoutSoundConstants, 655

SCOUT_SOUND_ENTERSA
NBCCCommon.h, 1004
ScoutSoundConstants, 655

SCOUT_SOUND_KEYERROR
NBCCCommon.h, 1004
ScoutSoundConstants, 656

SCOUT_SOUND_MAIL_RECEIVED
NBCCCommon.h, 1005
ScoutSoundConstants, 656

SCOUT_SOUND_NONE
NBCCCommon.h, 1005
ScoutSoundConstants, 656

SCOUT_SOUND_REMOTE
NBCCCommon.h, 1005
ScoutSoundConstants, 656

SCOUT_SOUND_SPECIAL1
NBCCCommon.h, 1005
ScoutSoundConstants, 656

SCOUT_SOUND_SPECIAL2
NBCCCommon.h, 1005
ScoutSoundConstants, 656

SCOUT_SOUND_SPECIAL3
 NBCCCommon.h, 1005
 ScoutSoundConstants, 656

SCOUT_SOUND_TIMER1
 NBCCCommon.h, 1005
 ScoutSoundConstants, 656

SCOUT_SOUND_TIMER2
 NBCCCommon.h, 1005
 ScoutSoundConstants, 656

SCOUT_SOUND_TIMER3
 NBCCCommon.h, 1005
 ScoutSoundConstants, 656

SCOUT_SOUND_TOUCH1_PRES
 NBCCCommon.h, 1005
 ScoutSoundConstants, 657

SCOUT_SOUND_TOUCH1_REL
 NBCCCommon.h, 1006
 ScoutSoundConstants, 657

SCOUT_SOUND_TOUCH2_PRES
 NBCCCommon.h, 1006
 ScoutSoundConstants, 657

SCOUT_SOUND_TOUCH2_REL
 NBCCCommon.h, 1006
 ScoutSoundConstants, 657

SCOUT_TGS_LONG
 NBCCCommon.h, 1006
 ScoutTransmitRuleConstants, 663

SCOUT_TGS_MEDIUM
 NBCCCommon.h, 1006
 ScoutTransmitRuleConstants, 663

SCOUT_TGS_SHORT
 NBCCCommon.h, 1006
 ScoutTransmitRuleConstants, 663

SCOUT_TR_AVOID
 NBCCCommon.h, 1006
 ScoutTouchRuleConstants, 661

SCOUT_TR_IGNORE
 NBCCCommon.h, 1006
 ScoutTouchRuleConstants, 661

SCOUT_TR_OFF_WHEN
 NBCCCommon.h, 1006
 ScoutTouchRuleConstants, 661

SCOUT_TR_REVERSE
 NBCCCommon.h, 1006
 ScoutTouchRuleConstants, 661

SCOUT_TR_WAIT_FOR
 NBCCCommon.h, 1007
 ScoutTouchRuleConstants, 661

ScoutLightConstants
 SCOUT_LIGHT_OFF, 654
 SCOUT_LIGHT_ON, 654

ScoutLightRuleConstants
 SCOUT_LR_AVOID, 662
 SCOUT_LR_IGNORE, 662
 SCOUT_LR_OFF_WHEN, 662
 SCOUT_LR_SEEK_DARK, 662
 SCOUT_LR_SEEK_LIGHT, 662
 SCOUT_LR_WAIT_FOR, 662

ScoutModeConstants
 SCOUT_MODE_POWER, 659
 SCOUT_MODE_STANDALONE, 659

ScoutMotionRuleConstants
 SCOUT_MR_CIRCLE_LEFT, 659
 SCOUT_MR_CIRCLE_RIGHT, 659
 SCOUT_MR_FORWARD, 659
 SCOUT_MR_LOOP_A, 660
 SCOUT_MR_LOOP_AB, 660
 SCOUT_MR_LOOP_B, 660
 SCOUT_MR_NO_MOTION, 660
 SCOUT_MR_ZIGZAG, 660

ScoutSndSetConstants
 SCOUT SNDSET_ALARM, 658
 SCOUT SNDSET_BASIC, 658
 SCOUT SNDSET_BUG, 658
 SCOUT SNDSET_NONE, 658
 SCOUT SNDSET_RANDOM, 658
 SCOUT SNDSET_SCIENCE, 658

ScoutSoundConstants
 SCOUT_SOUND_1_BLINK, 655
 SCOUT_SOUND_2_BLINK, 655
 SCOUT_SOUND_COUNTER1, 655
 SCOUT_SOUND_COUNTER2, 655
 SCOUT_SOUND_ENTER_-BRIGHT, 655
 SCOUT_SOUND_ENTER_DARK, 655
 SCOUT_SOUND_ENTER_-NORMAL, 655

SCOUT_SOUND_ENTERSA, [655](#)
SCOUT_SOUND_KEYERROR, [656](#)
SCOUT_SOUND_MAIL_- RECEIVED, [656](#)
SCOUT_SOUND_NONE, [656](#)
SCOUT_SOUND_REMOTE, [656](#)
SCOUT_SOUND_SPECIAL1, [656](#)
SCOUT_SOUND_SPECIAL2, [656](#)
SCOUT_SOUND_SPECIAL3, [656](#)
SCOUT_SOUND_TIMER1, [656](#)
SCOUT_SOUND_TIMER2, [656](#)
SCOUT_SOUND_TIMER3, [656](#)
SCOUT_SOUND_TOUCH1_- PRES, [657](#)
SCOUT_SOUND_TOUCH1_REL, [657](#)
SCOUT_SOUND_TOUCH2_- PRES, [657](#)
SCOUT_SOUND_TOUCH2_REL, [657](#)

ScoutSpecialEffectConstants
 SCOUT_FXR_ALARM, [663](#)
 SCOUT_FXR_BUG, [663](#)
 SCOUT_FXR_NONE, [663](#)
 SCOUT_FXR_RANDOM, [664](#)
 SCOUT_FXR_SCIENCE, [664](#)

ScoutTouchRuleConstants
 SCOUT_TR_AVOID, [661](#)
 SCOUT_TR_IGNORE, [661](#)
 SCOUT_TR_OFF_WHEN, [661](#)
 SCOUT_TR_REVERSE, [661](#)
 SCOUT_TR_WAIT_FOR, [661](#)

ScoutTransmitRuleConstants
 SCOUT_TGS_LONG, [663](#)
 SCOUT_TGS_MEDIUM, [663](#)
 SCOUT_TGS_SHORT, [663](#)

SCREEN_BACKGROUND
 DisplayModuleConstants, [598](#)
 NBCCCommon.h, [1007](#)

SCREEN_LARGE
 DisplayModuleConstants, [598](#)
 NBCCCommon.h, [1007](#)

SCREEN_MODE_CLEAR
 DisplayModuleConstants, [598](#)
 NBCCCommon.h, [1007](#)

SCREEN_MODE_RESTORE
 DisplayModuleConstants, [599](#)
 NBCCCommon.h, [1007](#)

SCREEN_SMALL
 DisplayModuleConstants, [599](#)
 NBCCCommon.h, [1007](#)

SCREENS
 DisplayModuleConstants, [599](#)
 NBCCCommon.h, [1007](#)

SEC_1
 NBCCCommon.h, [1007](#)
 TimeConstants, [492](#)

SEC_10
 NBCCCommon.h, [1008](#)
 TimeConstants, [493](#)

SEC_15
 NBCCCommon.h, [1008](#)
 TimeConstants, [493](#)

SEC_2
 NBCCCommon.h, [1008](#)
 TimeConstants, [493](#)

SEC_20
 NBCCCommon.h, [1008](#)
 TimeConstants, [493](#)

SEC_3
 NBCCCommon.h, [1008](#)
 TimeConstants, [493](#)

SEC_30
 NBCCCommon.h, [1008](#)
 TimeConstants, [493](#)

SEC_4
 NBCCCommon.h, [1008](#)
 TimeConstants, [493](#)

SEC_5
 NBCCCommon.h, [1008](#)
 TimeConstants, [493](#)

SEC_6
 NBCCCommon.h, [1008](#)
 TimeConstants, [493](#)

SEC_7
 NBCCCommon.h, [1008](#)
 TimeConstants, [493](#)

SEC_8
 NBCCCommon.h, [1009](#)
 TimeConstants, [494](#)

SEC_9

NBCCCommon.h, 1009
TimeConstants, 494
SendMessage
 CommModuleFunctions, 422
SendRemoteBool
 CommModuleFunctions, 422
SendRemoteNumber
 CommModuleFunctions, 423
SendRemoteString
 CommModuleFunctions, 423
SendResponseBool
 CommModuleFunctions, 423
SendResponseNumber
 CommModuleFunctions, 424
SendResponseString
 CommModuleFunctions, 424
SendRS485Bool
 CommModuleFunctions, 424
SendRS485Number
 CommModuleFunctions, 425
SendRS485String
 CommModuleFunctions, 425
Sensor types and modes, 77
SetAbortFlag
 UiModuleFunctions, 388
SetACCLNxSensitivity
 MindSensorsAPI, 229
SetBatteryState
 UiModuleFunctions, 388
SetBluetoothState
 UiModuleFunctions, 389
SetBTDataMode
 CommModuleFunctions, 425
SetBTInputBuffer
 CommModuleFunctions, 426
SetBTInputBufferInPtr
 CommModuleFunctions, 426
SetBTInputBufferOutPtr
 CommModuleFunctions, 426
SetBTOutputBuffer
 CommModuleFunctions, 426
SetBTOutputBufferInPtr
 CommModuleFunctions, 427
SetBTOutputBufferOutPtr
 CommModuleFunctions, 427
SetButtonLongPressCount
 ButtonModuleFunctions, 381
SetButtonLongReleaseCount
 ButtonModuleFunctions, 381
SetButtonModuleValue
 CommandModuleFunctions, 369
SetButtonPressCount
 ButtonModuleFunctions, 381
SetButtonReleaseCount
 ButtonModuleFunctions, 381
SetButtonShortReleaseCount
 ButtonModuleFunctions, 382
SetButtonState
 ButtonModuleFunctions, 382
SetCommandFlags
 UiModuleFunctions, 389
SetCommandModuleBytes
 CommandModuleFunctions, 370
SetCommandModuleValue
 CommandModuleFunctions, 370
SetCommModuleBytes
 CommandModuleFunctions, 370
SetCommModuleValue
 CommandModuleFunctions, 371
SetDisplayContrast
 DisplayModuleFunctions, 346
SetDisplayDisplay
 DisplayModuleFunctions, 346
SetDisplayEraseMask
 DisplayModuleFunctions, 346
SetDisplayFlags
 DisplayModuleFunctions, 347
SetDisplayFont
 DisplayModuleFunctions, 347
SetDisplayModuleBytes
 CommandModuleFunctions, 371
SetDisplayModuleValue
 CommandModuleFunctions, 372
SetDisplayNormal
 DisplayModuleFunctions, 347
SetDisplayPopup
 DisplayModuleFunctions, 348
SetDisplayTextLinesCenterFlags
 DisplayModuleFunctions, 348
SetDisplayUpdateMask
 DisplayModuleFunctions, 348
SetHSAddress

CommModuleFunctions	427
SetHSDataMode	CommModuleFunctions, 427
SetHSFlags	CommModuleFunctions, 428
SetHSInputBuffer	CommModuleFunctions, 428
SetHSInputBufferInPtr	CommModuleFunctions, 428
SetHSInputBufferOutPtr	CommModuleFunctions, 428
SetHSMode	CommModuleFunctions, 429
SetHSOutputBuffer	CommModuleFunctions, 429
SetHSOutputBufferInPtr	CommModuleFunctions, 429
SetHSOutputBufferOutPtr	CommModuleFunctions, 430
SetHSSpeed	CommModuleFunctions, 430
SetHSState	CommModuleFunctions, 430
SetHTBarometricCalibration	HiTechnicAPI, 137
SetHTColor2Mode	HiTechnicAPI, 138
SetHTIRSeeker2Mode	HiTechnicAPI, 138
SetI2COptions	LowSpeedModuleFunctions, 323
SetInCustomActiveStatus	InputModuleFunctions, 306
SetInCustomPercentFullScale	InputModuleFunctions, 307
SetInCustomZeroOffset	InputModuleFunctions, 307
SetInDigiPinsDirection	InputModuleFunctions, 307
SetInDigiPinsOutputLevel	InputModuleFunctions, 308
SetInDigiPinsStatus	InputModuleFunctions, 308
SetInputModuleValue	CommandModuleFunctions, 372
SetInSensorBoolean	InputModuleFunctions, 308
SetIOCtlModuleValue	CommandModuleFunctions, 372
SetIOMapBytes	CommandModuleFunctions, 373
SetIOMapBytesByID	CommandModuleFunctions, 373
SetIOMapValue	CommandModuleFunctions, 374
SetIOMapValueByID	CommandModuleFunctions, 374
SetLoaderModuleValue	CommandModuleFunctions, 375
SetLowSpeedModuleBytes	CommandModuleFunctions, 375
SetLowSpeedModuleValue	CommandModuleFunctions, 375
SetNXTLineLeaderKdFactor	MindSensorsAPI, 230
SetNXTLineLeaderKdValue	MindSensorsAPI, 230
SetNXTLineLeaderKiFactor	MindSensorsAPI, 231
SetNXTLineLeaderKiValue	MindSensorsAPI, 231
SetNXTLineLeaderKpFactor	MindSensorsAPI, 231
SetNXTLineLeaderKpValue	MindSensorsAPI, 232
SetNXTLineLeaderSetpoint	MindSensorsAPI, 232
SetNXTServoPosition	MindSensorsAPI, 233
SetNXTServoQuickPosition	MindSensorsAPI, 233
SetNXTServoSpeed	MindSensorsAPI, 234
SetOnBrickProgramPointer	UiModuleFunctions, 389
SetOutputModuleValue	CommandModuleFunctions, 376
SetOutPwnFreq	OutputModuleFunctions, 296
SetOutRegulationOptions	OutputModuleFunctions, 296
SetOutRegulationTime	OutputModuleFunctions, 296

OutputModuleFunctions, 296
SetScreenMode
 NBCCommon.h, 1009
 SysCallConstants, 485
SetSensorColorBlue
 InputModuleFunctions, 308
SetSensorColorFull
 InputModuleFunctions, 309
SetSensorColorGreen
 InputModuleFunctions, 309
SetSensorColorNone
 InputModuleFunctions, 309
SetSensorColorRed
 InputModuleFunctions, 310
SetSensorDIAccl
 DexterIndustriesAPI, 248
SetSensorDIAcclDrift
 DexterIndustriesAPI, 249
SetSensorDIAcclEx
 DexterIndustriesAPI, 249
SetSensorDIGPSWaypoint
 DexterIndustriesAPI, 250
SetSensorDIGyro
 DexterIndustriesAPI, 250
SetSensorDIGyroEx
 DexterIndustriesAPI, 250
SetSensorEMeter
 InputModuleFunctions, 310
SetSensorHTEOPD
 HiTechnicAPI, 138
SetSensorHTGyro
 HiTechnicAPI, 139
SetSensorHTMagnet
 HiTechnicAPI, 139
SetSensorHTProtoDigital
 HiTechnicAPI, 139
SetSensorHTProtoDigitalControl
 HiTechnicAPI, 140
SetSensorHTSuperProAnalogOut
 HiTechnicAPI, 140
SetSensorHTSuperProDigital
 HiTechnicAPI, 140
SetSensorHTSuperProDigitalControl
 HiTechnicAPI, 141
SetSensorHTSuperProLED
 HiTechnicAPI, 141
SetSensorHTSuperProProgramControl
 HiTechnicAPI, 142
SetSensorHTSuperProStrobe
 HiTechnicAPI, 142
SetSensorLight
 InputModuleFunctions, 310
SetSensorLowspeed
 InputModuleFunctions, 311
SetSensorMIXG1300LScale
 MicroinfinityAPI, 253
SetSensorMode
 InputModuleFunctions, 311
SetSensorMSDRODActive
 MindSensorsAPI, 234
SetSensorMSDRODInactive
 MindSensorsAPI, 234
SetSensorMSPressure
 MindSensorsAPI, 235
SetSensorMSTouchMux
 MindSensorsAPI, 235
SetSensorNXTSumoEyesLong
 MindSensorsAPI, 235
SetSensorNXTSumoEyesShort
 MindSensorsAPI, 235
SetSensorSound
 InputModuleFunctions, 311
SetSensorTemperature
 InputModuleFunctions, 311
SetSensorTouch
 InputModuleFunctions, 312
SetSensorType
 InputModuleFunctions, 312
SetSensorUltrasonic
 InputModuleFunctions, 312
SetSleepTimeout
 UiModuleFunctions, 389
SetSleepTimeoutVal
 NBCCommon.h, 1009
 SysCallConstants, 486
SetSleepTimer
 UiModuleFunctions, 390
SetSoundDuration
 SoundModuleFunctions, 354
SetSoundFlags
 SoundModuleFunctions, 354
SetSoundFrequency

SoundModuleFunctions, 355
SetSoundMode
 SoundModuleFunctions, 355
SetSoundModuleState
 SoundModuleFunctions, 355
SetSoundModuleValue
 CommandModuleFunctions, 376
SetSoundSampleRate
 SoundModuleFunctions, 356
SetSoundState
 SoundModuleFunctions, 356
SetSoundVolume
 SoundModuleFunctions, 356
SetUIButton
 UiModuleFunctions, 390
SetUIModuleValue
 CommandModuleFunctions, 377
SetUIState
 UiModuleFunctions, 390
SetUSBInputBuffer
 CommModuleFunctions, 430
SetUSBInputBufferInPtr
 CommModuleFunctions, 431
SetUSBInputBufferOutPtr
 CommModuleFunctions, 431
SetUSBOutputBuffer
 CommModuleFunctions, 431
SetUSBOutputBufferInPtr
 CommModuleFunctions, 431
SetUSBOutputBufferOutPtr
 CommModuleFunctions, 432
SetUSBPollBuffer
 CommModuleFunctions, 432
SetUSBPollBufferInPtr
 CommModuleFunctions, 432
SetUSBPollBufferOutPtr
 CommModuleFunctions, 432
SetUSBState
 CommModuleFunctions, 432
SetUsbState
 UiModuleFunctions, 390
SetVMRunState
 UiModuleFunctions, 391
SetVolume
 UiModuleFunctions, 391
SHRT_MAX
NBCCCommon.h, 1009
NXTLimits, 786
SHRT_MIN
NBCCCommon.h, 1009
NXTLimits, 786
SignedRandom
cstdlibAPI, 474
SIZE_OF_BTADDR
CommMiscConstants, 615
NBCCCommon.h, 1009
SIZE_OF_BRICK_NAME
CommMiscConstants, 615
NBCCCommon.h, 1009
SIZE_OF_BT_CONNECT_TABLE
CommMiscConstants, 615
NBCCCommon.h, 1009
SIZE_OF_BT_DEVICE_TABLE
CommMiscConstants, 615
NBCCCommon.h, 1009
SIZE_OF_BT_NAME
CommMiscConstants, 615
NBCCCommon.h, 1010
SIZE_OF_BT_PINCODE
CommMiscConstants, 615
NBCCCommon.h, 1010
SIZE_OF_BTBUF
CommMiscConstants, 615
NBCCCommon.h, 1010
SIZE_OF_CLASS_OF_DEVICE
CommMiscConstants, 615
NBCCCommon.h, 1010
SIZE_OF_HSBUF
CommMiscConstants, 615
NBCCCommon.h, 1010
SIZE_OF_USBBUF
CommMiscConstants, 615
NBCCCommon.h, 1010
SIZE_OF_USBDATA
CommMiscConstants, 616
NBCCCommon.h, 1010
SizeOf
LoaderModuleFunctions, 471
Sound module, 85
Sound module constants, 516
Sound module functions, 349
Sound module IOMAP offsets, 520

Sound module miscellaneous constants, [521](#)
SOUND_CLICK
 NBCCCommon.h, [1010](#)
 RCXSoundConstants, [652](#)
SOUND_DOUBLE_BEEP
 NBCCCommon.h, [1010](#)
 RCXSoundConstants, [652](#)
SOUND_DOWN
 NBCCCommon.h, [1010](#)
 RCXSoundConstants, [652](#)
SOUND_FAST_UP
 NBCCCommon.h, [1011](#)
 RCXSoundConstants, [652](#)
SOUND_FLAGS_IDLE
 NBCCCommon.h, [1011](#)
 SoundFlagsConstants, [518](#)
SOUND_FLAGS_RUNNING
 NBCCCommon.h, [1011](#)
 SoundFlagsConstants, [518](#)
SOUND_FLAGS_UPDATE
 NBCCCommon.h, [1011](#)
 SoundFlagsConstants, [518](#)
SOUND_LOW_BEEP
 NBCCCommon.h, [1011](#)
 RCXSoundConstants, [652](#)
SOUND_MODE_LOOP
 NBCCCommon.h, [1011](#)
 SoundModeConstants, [519](#)
SOUND_MODE_ONCE
 NBCCCommon.h, [1011](#)
 SoundModeConstants, [519](#)
SOUND_MODE_TONE
 NBCCCommon.h, [1011](#)
 SoundModeConstants, [519](#)
SOUND_STATE_FILE
 NBCCCommon.h, [1011](#)
 SoundStateConstants, [518](#)
SOUND_STATE_IDLE
 NBCCCommon.h, [1011](#)
 SoundStateConstants, [518](#)
SOUND_STATE_STOP
 NBCCCommon.h, [1012](#)
 SoundStateConstants, [518](#)
SOUND_STATE_TONE
 NBCCCommon.h, [1012](#)
SoundStateConstants, [519](#)
SOUND_UP
 NBCCCommon.h, [1012](#)
 RCXSoundConstants, [652](#)
SoundFlags constants, [517](#)
SoundFlagsConstants
 SOUND_FLAGS_IDLE, [518](#)
 SOUND_FLAGS_RUNNING, [518](#)
 SOUND_FLAGS_UPDATE, [518](#)
SoundGetState
 NBCCCommon.h, [1012](#)
 SysCallConstants, [486](#)
SoundIOMAP
 SoundOffsetDuration, [520](#)
 SoundOffsetFlags, [520](#)
 SoundOffsetFreq, [520](#)
 SoundOffsetMode, [520](#)
 SoundOffsetSampleRate, [520](#)
 SoundOffsetSoundFilename, [520](#)
 SoundOffsetState, [521](#)
 SoundOffsetVolume, [521](#)
SoundMisc
 FREQUENCY_MAX, [521](#)
 FREQUENCY_MIN, [521](#)
 SAMPLERATE_DEFAULT, [521](#)
 SAMPLERATE_MAX, [522](#)
 SAMPLERATE_MIN, [522](#)
SoundMode constants, [519](#)
SoundModeConstants
 SOUND_MODE_LOOP, [519](#)
 SOUND_MODE_ONCE, [519](#)
 SOUND_MODE_TONE, [519](#)
SoundModuleFunctions
 GetSoundDuration, [351](#)
 GetSoundFrequency, [351](#)
 GetSoundMode, [352](#)
 GetSoundSampleRate, [352](#)
 GetSoundState, [352](#)
 GetSoundVolume, [352](#)
 PlayFile, [353](#)
 PlayFileEx, [353](#)
 PlayTone, [353](#)
 PlayToneEx, [354](#)
 SetSoundDuration, [354](#)
 SetSoundFlags, [354](#)
 SetSoundFrequency, [355](#)

SetSoundMode, [355](#)
SetSoundModuleState, [355](#)
SetSoundSampleRate, [356](#)
SetSoundState, [356](#)
SetSoundVolume, [356](#)
SoundModuleID
 ModuleIDConstants, [265](#)
 NBCCommon.h, [1012](#)
SoundModuleName
 ModuleNameConstants, [263](#)
 NBCCommon.h, [1012](#)
SoundOffsetDuration
 NBCCommon.h, [1012](#)
 SoundIOMAP, [520](#)
SoundOffsetFlags
 NBCCommon.h, [1012](#)
 SoundIOMAP, [520](#)
SoundOffsetFreq
 NBCCommon.h, [1012](#)
 SoundIOMAP, [520](#)
SoundOffsetMode
 NBCCommon.h, [1012](#)
 SoundIOMAP, [520](#)
SoundOffsetSampleRate
 NBCCommon.h, [1013](#)
 SoundIOMAP, [520](#)
SoundOffsetSoundFilename
 NBCCommon.h, [1013](#)
 SoundIOMAP, [520](#)
SoundOffsetState
 NBCCommon.h, [1013](#)
 SoundIOMAP, [521](#)
SoundOffsetVolume
 NBCCommon.h, [1013](#)
 SoundIOMAP, [521](#)
SoundPlayFile
 NBCCommon.h, [1013](#)
 SysCallConstants, [486](#)
SoundPlayTone
 NBCCommon.h, [1013](#)
 SysCallConstants, [486](#)
SoundSetState
 NBCCommon.h, [1013](#)
 SysCallConstants, [486](#)
SoundState constants, [518](#)
SoundStateConstants
 SOUND_STATE_FILE, [518](#)
 SOUND_STATE_IDLE, [518](#)
 SOUND_STATE_STOP, [518](#)
 SOUND_STATE_TONE, [519](#)
SPECIALS
 DisplayModuleConstants, [599](#)
 NBCCommon.h, [1013](#)
 Standard I2C constants, [587](#)
 Standard-C API functions, [267](#)
STAT_COMM_PENDING
 CommandModuleConstants, [82](#)
 NBCCommon.h, [1013](#)
STAT_MSG_EMPTY_MAILBOX
 CommandModuleConstants, [82](#)
 NBCCommon.h, [1013](#)
STATUSICON_BATTERY
 DisplayModuleConstants, [599](#)
 NBCCommon.h, [1014](#)
STATUSICON_BLUETOOTH
 DisplayModuleConstants, [599](#)
 NBCCommon.h, [1014](#)
STATUSICON_USB
 DisplayModuleConstants, [599](#)
 NBCCommon.h, [1014](#)
STATUSICON_VM
 DisplayModuleConstants, [599](#)
 NBCCommon.h, [1014](#)
STATUSICONS
 DisplayModuleConstants, [600](#)
 NBCCommon.h, [1014](#)
STATUSTEXT
 DisplayModuleConstants, [600](#)
 NBCCommon.h, [1014](#)
STEPICON_1
 DisplayModuleConstants, [600](#)
 NBCCommon.h, [1014](#)
STEPICON_2
 DisplayModuleConstants, [600](#)
 NBCCommon.h, [1014](#)
STEPICON_3
 DisplayModuleConstants, [600](#)
 NBCCommon.h, [1014](#)
STEPICON_4
 DisplayModuleConstants, [600](#)
 NBCCommon.h, [1014](#)
STEPICON_5

DisplayModuleConstants, 600
NBCCommon.h, 1015

STEPICONS
DisplayModuleConstants, 600
NBCCommon.h, 1015

STEPLINE
DisplayModuleConstants, 600
NBCCommon.h, 1015

STOP_REQ
CommandVMState, 496
NBCCommon.h, 1015

STROBE_READ
NBCCommon.h, 1015
StrobeCtrlConstants, 147

STROBE_S0
NBCCommon.h, 1015
StrobeCtrlConstants, 147

STROBE_S1
NBCCommon.h, 1015
StrobeCtrlConstants, 147

STROBE_S2
NBCCommon.h, 1015
StrobeCtrlConstants, 147

STROBE_S3
NBCCommon.h, 1015
StrobeCtrlConstants, 147

STROBE_WRITE
NBCCommon.h, 1015
StrobeCtrlConstants, 147

StrobeCtrlConstants
STROBE_READ, 147
STROBE_S0, 147
STROBE_S1, 147
STROBE_S2, 147
STROBE_S3, 147
STROBE_WRITE, 147

SuperPro analog output mode constants, 142

SuperPro digital pin constants, 145

SuperPro LED control constants, 144

SuperPro Strobe control constants, 146

SysCallConstants
ColorSensorRead, 479
CommBTCheckStatus, 479
CommBTConnection, 479
CommBTOnOff, 479

CommBTRead, 479
CommBTWrite, 480

CommExecuteFunction, 480

CommHSCheckStatus, 480

CommHSControl, 480

CommHSRead, 480

CommHSWrite, 480

CommLSCheckStatus, 480

CommLSRead, 480

CommLSWrite, 480

CommLSWriteEx, 480

ComputeCalibValue, 481

DatalogGetTimes, 481

DatalogWrite, 481

DisplayExecuteFunction, 481

DrawCircle, 481

DrawEllipse, 481

DrawFont, 481

DrawGraphic, 481

DrawGraphicArray, 481

DrawLine, 481

DrawPoint, 482

DrawPolygon, 482

DrawRect, 482

DrawText, 482

FileClose, 482

FileDelete, 482

FileFindFirst, 482

FileFindNext, 482

FileOpenAppend, 482

FileOpenRead, 482

FileOpenReadLinear, 483

FileOpenWrite, 483

FileOpenWriteLinear, 483

FileOpenWriteNonLinear, 483

FileRead, 483

FileRename, 483

FileResize, 483

FileResolveHandle, 483

FileSeek, 483

FileTell, 483

FileWrite, 484

GetStartTick, 484

InputPinFunction, 484

IOMapRead, 484

IOMapReadByID, 484

IOMapWrite, 484
IOMapWriteByID, 484
KeepAlive, 484
ListFiles, 484
LoaderExecuteFunction, 485
MemoryManager, 485
MessageRead, 485
MessageWrite, 485
RandomEx, 485
RandomNumber, 485
ReadButton, 485
ReadLastResponse, 485
ReadSemData, 485
SetScreenMode, 485
SetSleepTimeoutVal, 486
SoundGetState, 486
SoundPlayFile, 486
SoundPlayTone, 486
SoundSetState, 486
UpdateCalibCacheInfo, 486
WriteSemData, 486
System Call function constants, 477
System Command functions, 447

TachoCountField
 NBCCCommon.h, 1016
 OutputFieldConstants, 574
TachoLimitField
 NBCCCommon.h, 1016
 OutputFieldConstants, 574
Tachometer counter reset flags, 563
TachoResetConstants
 RESET_ALL, 564
 RESET_BLOCK_COUNT, 564
 RESET_BLOCKANDTACHO, 564
 RESET_COUNT, 564
 RESET_NONE, 564
 RESET_ROTATION_COUNT, 564
TEMP_FQ_1
 NBCCCommon.h, 1016
 TempI2CConstants, 591
TEMP_FQ_2
 NBCCCommon.h, 1016
 TempI2CConstants, 591
TEMP_FQ_4
 NBCCCommon.h, 1016

 TempI2CConstants, 591
TEMP_FQ_6
 NBCCCommon.h, 1016
 TempI2CConstants, 591
TEMP_OS_ONESHOT
 NBCCCommon.h, 1016
 TempI2CConstants, 591
TEMP_POL_HIGH
 NBCCCommon.h, 1017
 TempI2CConstants, 591
TEMP_POL_LOW
 NBCCCommon.h, 1017
 TempI2CConstants, 592
TEMP_REG_CONFIG
 NBCCCommon.h, 1017
 TempI2CConstants, 592
TEMP_REG_TEMP
 NBCCCommon.h, 1017
 TempI2CConstants, 592
TEMP_REG_THIGH
 NBCCCommon.h, 1017
 TempI2CConstants, 592
TEMP_REG_TLOW
 NBCCCommon.h, 1017
 TempI2CConstants, 592
TEMP_RES_10BIT
 NBCCCommon.h, 1017
 TempI2CConstants, 592
TEMP_RES_11BIT
 NBCCCommon.h, 1017
 TempI2CConstants, 592
TEMP_RES_12BIT
 NBCCCommon.h, 1017
 TempI2CConstants, 592
TEMP_RES_9BIT
 NBCCCommon.h, 1018
 TempI2CConstants, 592
TEMP_SD_CONTINUOUS
 NBCCCommon.h, 1018
 TempI2CConstants, 592
TEMP_SD_SHUTDOWN
 NBCCCommon.h, 1018
 TempI2CConstants, 593
TEMP_TM_COMPARATOR
 NBCCCommon.h, 1018
 TempI2CConstants, 593

TEMP_TM_INTERRUPT
 NBCCCommon.h, 1018
 TempI2CConstants, 593

TempI2CConstants
 TEMP_FQ_1, 591
 TEMP_FQ_2, 591
 TEMP_FQ_4, 591
 TEMP_FQ_6, 591
 TEMP_OS_ONESHOT, 591
 TEMP_POL_HIGH, 591
 TEMP_POL_LOW, 592
 TEMP_REG_CONFIG, 592
 TEMP_REG_TEMP, 592
 TEMP_REG_THIGH, 592
 TEMP_REG_TLOW, 592
 TEMP_RES_10BIT, 592
 TEMP_RES_11BIT, 592
 TEMP_RES_12BIT, 592
 TEMP_RES_9BIT, 592
 TEMP_SD_CONTINUOUS, 592
 TEMP_SD_SHUTDOWN, 593
 TEMP_TM_COMPARATOR, 593
 TEMP_TM_INTERRUPT, 593

Text line constants, 608

TEXTLINE_1
 DisplayTextLineConstants, 609
 NBCCCommon.h, 1018

TEXTLINE_2
 DisplayTextLineConstants, 609
 NBCCCommon.h, 1018

TEXTLINE_3
 DisplayTextLineConstants, 609
 NBCCCommon.h, 1018

TEXTLINE_4
 DisplayTextLineConstants, 609
 NBCCCommon.h, 1018

TEXTLINE_5
 DisplayTextLineConstants, 609
 NBCCCommon.h, 1018

TEXTLINE_6
 DisplayTextLineConstants, 609
 NBCCCommon.h, 1019

TEXTLINE_7
 DisplayTextLineConstants, 609
 NBCCCommon.h, 1019

TEXTLINE_8

DisplayTextLineConstants, 609
 NBCCCommon.h, 1019

TEXTLINES

TextOut
 DisplayModuleFunctions, 348

TextOutEx
 DisplayModuleFunctions, 349

Third-party NXT devices, 266

Time constants, 488

TimeConstants
 MIN_1, 489
 MS_1, 489
 MS_10, 489
 MS_100, 490
 MS_150, 490
 MS_2, 490
 MS_20, 490
 MS_200, 490
 MS_250, 490
 MS_3, 490
 MS_30, 490
 MS_300, 490
 MS_350, 490
 MS_4, 491
 MS_40, 491
 MS_400, 491
 MS_450, 491
 MS_5, 491
 MS_50, 491
 MS_500, 491
 MS_6, 491
 MS_60, 491
 MS_600, 491
 MS_7, 492
 MS_70, 492
 MS_700, 492
 MS_8, 492
 MS_80, 492
 MS_800, 492
 MS_9, 492
 MS_90, 492
 MS_900, 492
 SEC_1, 492
 SEC_10, 493

SEC_15, 493
SEC_2, 493
SEC_20, 493
SEC_3, 493
SEC_30, 493
SEC_4, 493
SEC_5, 493
SEC_6, 493
SEC_7, 493
SEC_8, 494
SEC_9, 494
TIMES_UP
 CommandVMState, 497
 NBCCCommon.h, 1019
Tone constants, 522
TONE_A3
 NBCCCommon.h, 1019
 ToneConstants, 523
TONE_A4
 NBCCCommon.h, 1019
 ToneConstants, 523
TONE_A5
 NBCCCommon.h, 1019
 ToneConstants, 524
TONE_A6
 NBCCCommon.h, 1019
 ToneConstants, 524
TONE_A7
 NBCCCommon.h, 1019
 ToneConstants, 524
TONE_AS3
 NBCCCommon.h, 1020
 ToneConstants, 524
TONE_AS4
 NBCCCommon.h, 1020
 ToneConstants, 524
TONE_AS5
 NBCCCommon.h, 1020
 ToneConstants, 524
TONE_AS6
 NBCCCommon.h, 1020
 ToneConstants, 524
TONE_AS7
 NBCCCommon.h, 1020
 ToneConstants, 524
TONE_B3
 NBCCCommon.h, 1020
 ToneConstants, 524
TONE_B4
 NBCCCommon.h, 1020
 ToneConstants, 524
TONE_B5
 NBCCCommon.h, 1020
 ToneConstants, 525
TONE_B6
 NBCCCommon.h, 1020
 ToneConstants, 525
TONE_B7
 NBCCCommon.h, 1020
 ToneConstants, 525
TONE_C4
 NBCCCommon.h, 1021
 ToneConstants, 525
TONE_C5
 NBCCCommon.h, 1021
 ToneConstants, 525
TONE_C6
 NBCCCommon.h, 1021
 ToneConstants, 525
TONE_C7
 NBCCCommon.h, 1021
 ToneConstants, 525
TONE_CS4
 NBCCCommon.h, 1021
 ToneConstants, 525
TONE_CS5
 NBCCCommon.h, 1021
 ToneConstants, 525
TONE_CS6
 NBCCCommon.h, 1021
 ToneConstants, 525
TONE_CS7
 NBCCCommon.h, 1021
 ToneConstants, 526
TONE_D4
 NBCCCommon.h, 1021
 ToneConstants, 526
TONE_D5
 NBCCCommon.h, 1021
 ToneConstants, 526
TONE_D6
 NBCCCommon.h, 1022

ToneConstants, 526
TONE_D7
 NBCCCommon.h, 1022
 ToneConstants, 526
TONE_DS4
 NBCCCommon.h, 1022
 ToneConstants, 526
TONE_DS5
 NBCCCommon.h, 1022
 ToneConstants, 526
TONE_DS6
 NBCCCommon.h, 1022
 ToneConstants, 526
TONE_DS7
 NBCCCommon.h, 1022
 ToneConstants, 526
TONE_E4
 NBCCCommon.h, 1022
 ToneConstants, 526
TONE_E5
 NBCCCommon.h, 1022
 ToneConstants, 527
TONE_E6
 NBCCCommon.h, 1022
 ToneConstants, 527
TONE_E7
 NBCCCommon.h, 1022
 ToneConstants, 527
TONE_F4
 NBCCCommon.h, 1023
 ToneConstants, 527
TONE_F5
 NBCCCommon.h, 1023
 ToneConstants, 527
TONE_F6
 NBCCCommon.h, 1023
 ToneConstants, 527
TONE_F7
 NBCCCommon.h, 1023
 ToneConstants, 527
TONE_FS4
 NBCCCommon.h, 1023
 ToneConstants, 527
TONE_FS5
 NBCCCommon.h, 1023
 ToneConstants, 527
TONE_FS6
 NBCCCommon.h, 1023
 ToneConstants, 527
TONE_FS7
 NBCCCommon.h, 1023
 ToneConstants, 528
TONE_G4
 NBCCCommon.h, 1023
 ToneConstants, 528
TONE_G5
 NBCCCommon.h, 1023
 ToneConstants, 528
TONE_G6
 NBCCCommon.h, 1024
 ToneConstants, 528
TONE_G7
 NBCCCommon.h, 1024
 ToneConstants, 528
TONE_GS4
 NBCCCommon.h, 1024
 ToneConstants, 528
TONE_GS5
 NBCCCommon.h, 1024
 ToneConstants, 528
TONE_GS6
 NBCCCommon.h, 1024
 ToneConstants, 528
TONE_GS7
 NBCCCommon.h, 1024
 ToneConstants, 528
ToneConstants
 TONE_A3, 523
 TONE_A4, 523
 TONE_A5, 524
 TONE_A6, 524
 TONE_A7, 524
 TONE_AS3, 524
 TONE_AS4, 524
 TONE_AS5, 524
 TONE_AS6, 524
 TONE_AS7, 524
 TONE_B3, 524
 TONE_B4, 524
 TONE_B5, 525
 TONE_B6, 525
 TONE_B7, 525

TONE_C4, [525](#)
TONE_C5, [525](#)
TONE_C6, [525](#)
TONE_C7, [525](#)
TONE_CS4, [525](#)
TONE_CS5, [525](#)
TONE_CS6, [525](#)
TONE_CS7, [526](#)
TONE_D4, [526](#)
TONE_D5, [526](#)
TONE_D6, [526](#)
TONE_D7, [526](#)
TONE_DS4, [526](#)
TONE_DS5, [526](#)
TONE_DS6, [526](#)
TONE_DS7, [526](#)
TONE_E4, [526](#)
TONE_E5, [527](#)
TONE_E6, [527](#)
TONE_E7, [527](#)
TONE_F4, [527](#)
TONE_F5, [527](#)
TONE_F6, [527](#)
TONE_F7, [527](#)
TONE_FS4, [527](#)
TONE_FS5, [527](#)
TONE_FS6, [527](#)
TONE_FS7, [528](#)
TONE_G4, [528](#)
TONE_G5, [528](#)
TONE_G6, [528](#)
TONE_G7, [528](#)
TONE_GS4, [528](#)
TONE_GS5, [528](#)
TONE_GS6, [528](#)
TONE_GS7, [528](#)

TOPLINE
 DisplayModuleConstants, [600](#)
 NBCCommon.h, [1024](#)

TRAIN_CHANNEL_1
 IRTrainChannels, [684](#)
 NBCCommon.h, [1024](#)

TRAIN_CHANNEL_2
 IRTrainChannels, [684](#)
 NBCCommon.h, [1024](#)

TRAIN_CHANNEL_3

IRTrainChannels, [684](#)
NBCCommon.h, [1024](#)

TRAIN_CHANNEL_ALL
 IRTrainChannels, [684](#)
 NBCCommon.h, [1025](#)

TRAIN_FUNC_DECN_SPEED
 IRTrainFuncs, [683](#)
 NBCCommon.h, [1025](#)

TRAIN_FUNC_INCR_SPEED
 IRTrainFuncs, [683](#)
 NBCCommon.h, [1025](#)

TRAIN_FUNC_STOP
 IRTrainFuncs, [683](#)
 NBCCommon.h, [1025](#)

TRAIN_FUNC_TOGGLE_LIGHT
 IRTrainFuncs, [683](#)
 NBCCommon.h, [1025](#)

TRUE
 MiscConstants, [266](#)
 NBCCommon.h, [1025](#)

TurnRatioField
 NBCCommon.h, [1025](#)
 OutputFieldConstants, [574](#)

TypeField
 InputFieldConstants, [550](#)
 NBCCommon.h, [1026](#)

UCHAR_MAX
 NBCCommon.h, [1026](#)
 NXTLimits, [786](#)

UF_PENDING_UPDATES
 NBCCommon.h, [1026](#)
 OutUFConstants, [562](#)

UF_UPDATE_MODE
 NBCCommon.h, [1026](#)
 OutUFConstants, [562](#)

UF_UPDATE_PID_VALUES
 NBCCommon.h, [1026](#)
 OutUFConstants, [562](#)

UF_UPDATE_RESET_BLOCK_-
 COUNT
 NBCCommon.h, [1026](#)
 OutUFConstants, [562](#)

UF_UPDATE_RESET_COUNT
 NBCCommon.h, [1026](#)
 OutUFConstants, [562](#)

UF_UPDATE_RESET_ROTATION_-
COUNT
 NBCCCommon.h, 1026
 UiFlagsConstants, 534

UF_UPDATE_SPEED
 NBCCCommon.h, 1026
 UiFlagsConstants, 534

UF_UPDATE_TACHO_LIMIT
 NBCCCommon.h, 1027
 UiFlagsConstants, 534

Ui module, 86

Ui module constants, 533

Ui module functions, 382

Ui module IOMAP offsets, 541

UI_BT_CONNECT_REQUEST
 NBCCCommon.h, 1027
 UiBluetoothStateConstants, 539

UI_BT_ERROR_ATTENTION
 NBCCCommon.h, 1027
 UiBluetoothStateConstants, 539

UI_BT_PIN_REQUEST
 NBCCCommon.h, 1027
 UiBluetoothStateConstants, 539

UI_BT_STATE_CONNECTED
 NBCCCommon.h, 1027
 UiBluetoothStateConstants, 539

UI_BT_STATE_OFF
 NBCCCommon.h, 1027
 UiBluetoothStateConstants, 539

UI_BT_STATE_VISIBLE
 NBCCCommon.h, 1027
 UiBluetoothStateConstants, 539

UI_BUTTON_ENTER
 NBCCCommon.h, 1027
 UiButtonConstants, 538

UI_BUTTON_EXIT
 NBCCCommon.h, 1027
 UiButtonConstants, 538

UI_BUTTON_LEFT
 NBCCCommon.h, 1027
 UiButtonConstants, 538

UI_BUTTON_NONE
 NBCCCommon.h, 1028
 UiButtonConstants, 538

UI_BUTTON_RIGHT
 NBCCCommon.h, 1028

UiButtonConstants, 538

UI_FLAGS_BUSY
 NBCCCommon.h, 1028
 UiFlagsConstants, 534

UI_FLAGS_DISABLE_EXIT
 NBCCCommon.h, 1028
 UiFlagsConstants, 534

UI_FLAGS_DISABLE_LEFT_RIGHT_-
ENTER
 NBCCCommon.h, 1028
 UiFlagsConstants, 534

UI_FLAGS_ENABLE_STATUS_-
UPDATE
 NBCCCommon.h, 1028
 UiFlagsConstants, 534

UI_FLAGS_EXECUTE_LMS_FILE
 NBCCCommon.h, 1028
 UiFlagsConstants, 535

UI_FLAGS_REDRAW_STATUS
 NBCCCommon.h, 1028
 UiFlagsConstants, 535

UI_FLAGS_RESET_SLEEP_TIMER
 NBCCCommon.h, 1028
 UiFlagsConstants, 535

UI_FLAGS_UPDATE
 NBCCCommon.h, 1028
 UiFlagsConstants, 535

UI_STATE_BT_ERROR
 NBCCCommon.h, 1029
 UiStateConstants, 536

UI_STATE_CONNECT_REQUEST
 NBCCCommon.h, 1029
 UiStateConstants, 536

UI_STATE_DRAW_MENU
 NBCCCommon.h, 1029
 UiStateConstants, 536

UI_STATE_ENTER_PRESSED
 NBCCCommon.h, 1029
 UiStateConstants, 536

UI_STATE_EXECUTE_FILE
 NBCCCommon.h, 1029
 UiStateConstants, 536

UI_STATE_EXECUTING_FILE
 NBCCCommon.h, 1029
 UiStateConstants, 536

UI_STATE_EXIT_PRESSED

NBCCCommon.h, 1029
UiStateConstants, 536

UI_STATE_INIT_DISPLAY
NBCCCommon.h, 1029
UiStateConstants, 536

UI_STATE_INIT_INTRO
NBCCCommon.h, 1029
UiStateConstants, 537

UI_STATE_INIT_LOW_BATTERY
NBCCCommon.h, 1029
UiStateConstants, 537

UI_STATE_INIT_MENU
NBCCCommon.h, 1030
UiStateConstants, 537

UI_STATE_INIT_WAIT
NBCCCommon.h, 1030
UiStateConstants, 537

UI_STATE_LEFT_PRESSED
NBCCCommon.h, 1030
UiStateConstants, 537

UI_STATE_LOW_BATTERY
NBCCCommon.h, 1030
UiStateConstants, 537

UI_STATE_NEXT_MENU
NBCCCommon.h, 1030
UiStateConstants, 537

UI_STATE_RIGHT_PRESSED
NBCCCommon.h, 1030
UiStateConstants, 537

UI_STATE_TEST_BUTTONS
NBCCCommon.h, 1030
UiStateConstants, 537

UI_VM_IDLE
NBCCCommon.h, 1030
UiVMRunStateConstants, 540

UI_VM_RESET1
NBCCCommon.h, 1030
UiVMRunStateConstants, 540

UI_VM_RESET2
NBCCCommon.h, 1031
UiVMRunStateConstants, 540

UI_VM_RUN_FREE
NBCCCommon.h, 1031
UiVMRunStateConstants, 540

UI_VM_RUN_PAUSE
NBCCCommon.h, 1031

UiVMRunStateConstants, 541

UI_VM_RUN_SINGLE
NBCCCommon.h, 1031
UiVMRunStateConstants, 541

UiBluetoothStateConstants
UI_BT_CONNECT_REQUEST,
539

UI_BT_ERROR_ATTENTION, 539

UI_BT_PIN_REQUEST, 539

UI_BT_STATE_CONNECTED, 539

UI_BT_STATE_OFF, 539

UI_BT_STATE_VISIBLE, 539

UIButton constants, 538

UiButtonConstants
UI_BUTTON_ENTER, 538

UI_BUTTON_EXIT, 538

UI_BUTTON_LEFT, 538

UI_BUTTON_NONE, 538

UI_BUTTON_RIGHT, 538

UiFlagsConstants
UI_FLAGS_BUSY, 534

UI_FLAGS_DISABLE_EXIT, 534

UI_FLAGS_DISABLE_LEFT_-
RIGHT_ENTER, 534

UI_FLAGS_ENABLE_STATUS_-
UPDATE, 534

UI_FLAGS_EXECUTE_LMS_-
FILE, 535

UI_FLAGS_REDRAW_STATUS,
535

UI_FLAGS_RESET_SLEEP_-
TIMER, 535

UI_FLAGS_UPDATE, 535

UiIOMAP
UIOffsetAbortFlag, 542

UIOffsetBatteryState, 542

UIOffsetBatteryVoltage, 542

UIOffsetBluetoothState, 542

UIOffsetButton, 542

UIOffsetError, 542

UIOffsetFlags, 542

UIOffsetForceOff, 542

UIOffsetLMSfilename, 542

UIOffsetOBPPointer, 542

UIOffsetPMenu, 543

UIOffsetRechargeable, 543

UIOffsetRunState, [543](#)
UIOffsetSleepTimeout, [543](#)
UIOffsetSleepTimer, [543](#)
UIOffsetState, [543](#)
UIOffsetUsbState, [543](#)
UIOffsetVolume, [543](#)
UiModuleFunctions
 ForceOff, [385](#)
 GetAbortFlag, [385](#)
 GetBatteryLevel, [385](#)
 GetBatteryState, [385](#)
 GetBluetoothState, [385](#)
 GetCommandFlags, [386](#)
 GetOnBrickProgramPointer, [386](#)
 GetRechargeableBattery, [386](#)
 GetSleepTimeout, [386](#)
 GetSleepTimer, [387](#)
 GetUIButton, [387](#)
 GetUIState, [387](#)
 GetUsbState, [387](#)
 GetVMRunState, [388](#)
 GetVolume, [388](#)
 SetAbortFlag, [388](#)
 SetBatteryState, [388](#)
 SetBluetoothState, [389](#)
 SetCommandFlags, [389](#)
 SetOnBrickProgramPointer, [389](#)
 SetSleepTimeout, [389](#)
 SetSleepTimer, [390](#)
 SetUIButton, [390](#)
 SetUIState, [390](#)
 SetUsbState, [390](#)
 SetVMRunState, [391](#)
 SetVolume, [391](#)
UiModuleID
 ModuleIDConstants, [265](#)
 NBCCommon.h, [1031](#)
UiModuleName
 ModuleNameConstants, [263](#)
 NBCCommon.h, [1031](#)
UINT_MAX
 NBCCommon.h, [1031](#)
 NXTLimits, [786](#)
UIOffsetAbortFlag
 NBCCommon.h, [1031](#)
 UiIOMAP, [542](#)
UIOffsetBatteryState
 NBCCommon.h, [1031](#)
 UiIOMAP, [542](#)
UIOffsetBatteryVoltage
 NBCCommon.h, [1032](#)
 UiIOMAP, [542](#)
UIOffsetBluetoothState
 NBCCommon.h, [1032](#)
 UiIOMAP, [542](#)
UIOffsetButton
 NBCCommon.h, [1032](#)
 UiIOMAP, [542](#)
UIOffsetError
 NBCCommon.h, [1032](#)
 UiIOMAP, [542](#)
UIOffsetFlags
 NBCCommon.h, [1032](#)
 UiIOMAP, [542](#)
UIOffsetForceOff
 NBCCommon.h, [1032](#)
 UiIOMAP, [542](#)
UIOffsetLMSfilename
 NBCCommon.h, [1032](#)
 UiIOMAP, [542](#)
UIOffsetOBPPointer
 NBCCommon.h, [1032](#)
 UiIOMAP, [542](#)
UIOffsetPMenu
 NBCCommon.h, [1032](#)
 UiIOMAP, [543](#)
UIOffsetRechargeable
 NBCCommon.h, [1032](#)
 UiIOMAP, [543](#)
UIOffsetRunState
 NBCCommon.h, [1033](#)
 UiIOMAP, [543](#)
UIOffsetSleepTimeout
 NBCCommon.h, [1033](#)
 UiIOMAP, [543](#)
UIOffsetSleepTimer
 NBCCommon.h, [1033](#)
 UiIOMAP, [543](#)
UIOffsetState
 NBCCommon.h, [1033](#)
 UiIOMAP, [543](#)
UIOffsetUsbState

NBCCCommon.h, 1033
UiIOMAP, 543

UIOffsetVolume
NBCCCommon.h, 1033
UiIOMAP, 543

UIState constants, 535

UiStateConstants
UI_STATE_BT_ERROR, 536
UI_STATE_CONNECT_-
REQUEST, 536
UI_STATE_DRAW_MENU, 536
UI_STATE_ENTER_PRESSED,
536
UI_STATE_EXECUTE_FILE, 536
UI_STATE_EXECUTING_FILE,
536
UI_STATE_EXIT_PRESSED, 536
UI_STATE_INIT_DISPLAY, 536
UI_STATE_INIT_INTRO, 537
UI_STATE_INIT_LOW_-
BATTERY, 537
UI_STATE_INIT_MENU, 537
UI_STATE_INIT_WAIT, 537
UI_STATE_LEFT_PRESSED, 537
UI_STATE_LOW_BATTERY, 537
UI_STATE_NEXT_MENU, 537
UI_STATE_RIGHT_PRESSED, 537
UI_STATE_TEST_BUTTONS, 537

UiVMRunStateConstants
UI_VM_IDLE, 540
UI_VM_RESET1, 540
UI_VM_RESET2, 540
UI_VM_RUN_FREE, 540
UI_VM_RUN_PAUSE, 541
UI_VM_RUN_SINGLE, 541

ULONG_MAX
NBCCCommon.h, 1033
NXTLimits, 786

Ultrasonic sensor constants, 588

UpdateCalibCacheInfo
NBCCCommon.h, 1033
SysCallConstants, 486

UpdateFlagsField
NBCCCommon.h, 1033
OutputFieldConstants, 575

US_CMD_CONTINUOUS

NBCCCommon.h, 1034
USI2CConstants, 589

US_CMD_EVENTCAPTURE
NBCCCommon.h, 1034
USI2CConstants, 589

US_CMD_OFF
NBCCCommon.h, 1034
USI2CConstants, 589

US_CMD_SINGLESHT
NBCCCommon.h, 1034
USI2CConstants, 589

US_CMD_WARMRESET
NBCCCommon.h, 1034
USI2CConstants, 589

US_REG_ACTUAL_ZERO
NBCCCommon.h, 1034
USI2CConstants, 589

US_REG_CM_INTERVAL
NBCCCommon.h, 1034
USI2CConstants, 589

US_REG_FACTORY_ACTUAL_ZERO
NBCCCommon.h, 1034
USI2CConstants, 590

US_REG_FACTORY_SCALE_-
DIVISOR
NBCCCommon.h, 1034
USI2CConstants, 590

US_REG_FACTORY_SCALE_FACTOR
NBCCCommon.h, 1035
USI2CConstants, 590

US_REG_MEASUREMENT_UNITS
NBCCCommon.h, 1035
USI2CConstants, 590

US_REG_SCALE_DIVISOR
NBCCCommon.h, 1035
USI2CConstants, 590

US_REG_SCALE_FACTOR
NBCCCommon.h, 1035
USI2CConstants, 590

USB_CMD_READY
CommStatusCodesConstants, 638
NBCCCommon.h, 1035

USB_PROTOCOL_OVERHEAD
CommMiscConstants, 616
NBCCCommon.h, 1035

UseRS485

CommModuleFunctions, 433
USHRT_MAX
 NBCCCommon.h, 1035
 NXTLimits, 786
USI2CConstants
 US_CMD_CONTINUOUS, 589
 US_CMD_EVENTCAPTURE, 589
 US_CMD_OFF, 589
 US_CMD_SINGLESHOT, 589
 US_CMD_WARMRESET, 589
 US_REG_ACTUAL_ZERO, 589
 US_REG_CM_INTERVAL, 589
 US_REG_FACTORY_ACTUAL_-
 ZERO, 590
 US_REG_FACTORY_SCALE_-
 DIVISOR, 590
 US_REG_FACTORY_SCALE_-
 FACTOR, 590
 US_REG_MEASUREMENT_-
 UNITS, 590
 US_REG_SCALE_DIVISOR, 590
 US_REG_SCALE_FACTOR, 590

VM run state constants, 540
VM state constants, 496

Wait
 CommandModuleFunctions, 377
Write
 LoaderModuleFunctions, 471
WriteBytes
 LoaderModuleFunctions, 471
WriteBytesEx
 LoaderModuleFunctions, 472
WriteI2CRegister
 LowSpeedModuleFunctions, 323
WriteLn
 LoaderModuleFunctions, 472
WriteLnString
 LoaderModuleFunctions, 473
WriteNRLinkBytes
 MindSensorsAPI, 236
WriteSemData
 NBCCCommon.h, 1035
 SysCallConstants, 486
WriteString

LoaderModuleFunctions, 473
XG1300L_REG_2G
 NBCCCommon.h, 1035
 XG1300LConstants, 782
XG1300L_REG_4G
 NBCCCommon.h, 1035
 XG1300LConstants, 782
XG1300L_REG_8G
 NBCCCommon.h, 1036
 XG1300LConstants, 782
XG1300L_REG_ANGLE
 NBCCCommon.h, 1036
 XG1300LConstants, 783
XG1300L_REG_RESET
 NBCCCommon.h, 1036
 XG1300LConstants, 783
XG1300L_REG_TURNRATE
 NBCCCommon.h, 1036
 XG1300LConstants, 783
XG1300L_REG_XAXIS
 NBCCCommon.h, 1036
 XG1300LConstants, 783
XG1300L_REG_YAXIS
 NBCCCommon.h, 1036
 XG1300LConstants, 783
XG1300L_REG_ZAXIS
 NBCCCommon.h, 1036
 XG1300LConstants, 783
XG1300L_SCALE_2G
 NBCCCommon.h, 1036
 XG1300LScaleConstants, 784
XG1300L_SCALE_4G
 NBCCCommon.h, 1036
 XG1300LScaleConstants, 784
XG1300L_SCALE_8G
 NBCCCommon.h, 1037
 XG1300LScaleConstants, 784
XG1300LConstants
 MI_ADDR_XG1300L, 782
 XG1300L_REG_2G, 782
 XG1300L_REG_4G, 782
 XG1300L_REG_8G, 782
 XG1300L_REG_ANGLE, 783
 XG1300L_REG_RESET, 783
 XG1300L_REG_TURNRATE, 783

XG1300L_REG_XAXIS, [783](#)
XG1300L_REG_YAXIS, [783](#)
XG1300L_REG_ZAXIS, [783](#)
XG1300LScaleConstants
 XG1300L_SCALE_2G, [784](#)
 XG1300L_SCALE_4G, [784](#)
 XG1300L_SCALE_8G, [784](#)