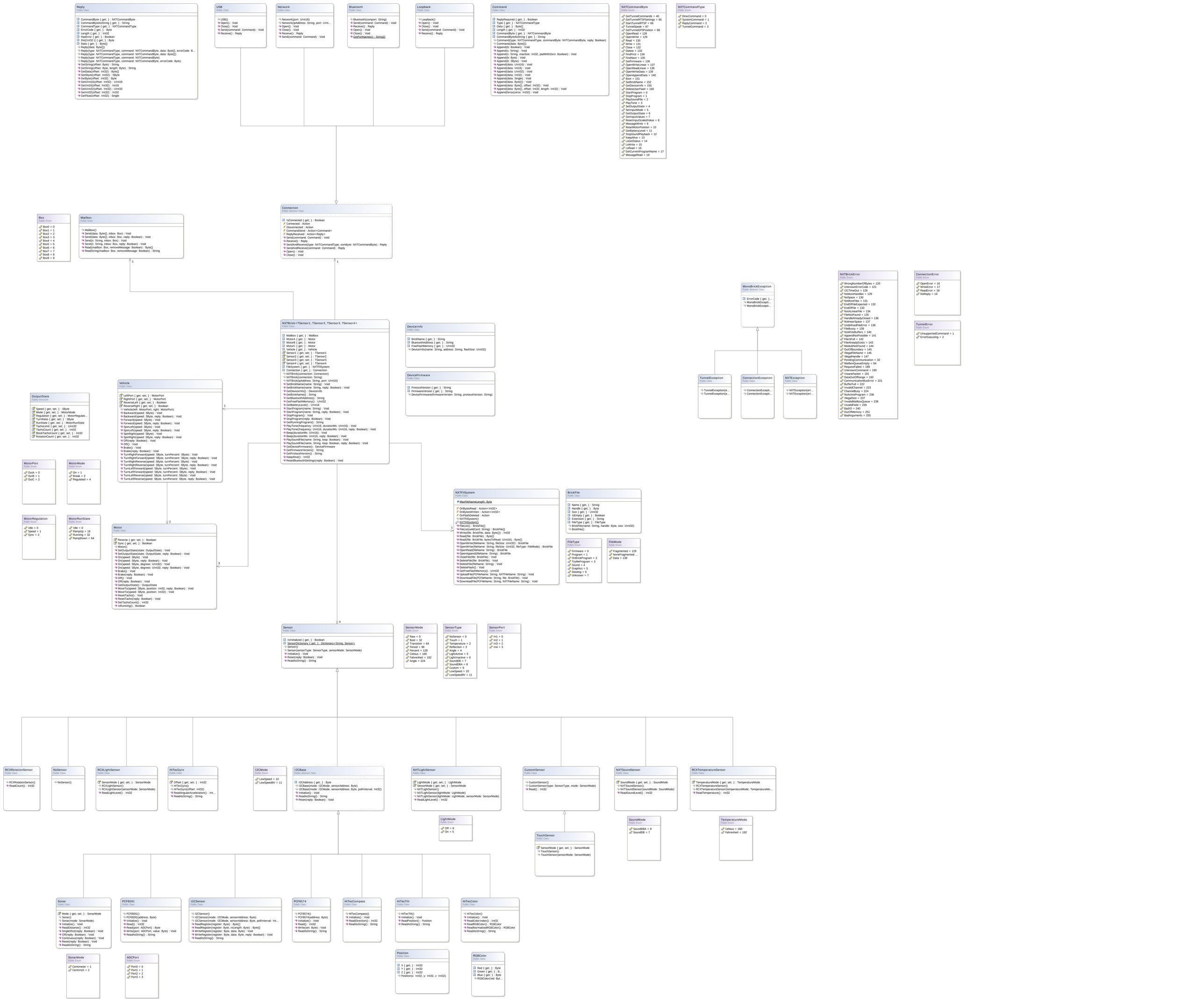
Reply
Public Class
CommandByte { get; } : NXTCommandByte
CommandByteAsString { get; } : String
CommandType { get; } : NXTCommandType
ErrorCode { get; } : Byte
Length { get; } : Int32
HasError { get; } : Boolean
this[Int32 i] { get; } : Byte
Data { get; } : Byte[]
Reply(data: Byte[])
Reply(type: NXTCommandType, command: NXTCommandByte, data: Byte[], errorCode: B...
Reply(type: NXTCommandType, command: NXTCommandByte, data: Byte[])
Reply(type: NXTCommandType, command: NXTCommandByte)
Reply(type: NXTCommandType, command: NXTCommandByte, errorCode: Byte)
GetString(offset: Byte) : String
GetString(offset: Byte, length: Byte) : String
GetData(offset: Int32) : Byte[]
GetSbyte(offset: Int32) : SByte
GetByte(offset: Int32) : Byte
GetInt16(offset: Int32) : Int16
GetInt16(offset: Int32) : Int16
GetUInt32(offset: Int32) : UInt32
GetInt32(offset: Int32) : Int32
GetFloat(offset: Int32) : Single
USB
Public Class
USB()
Open() : Void
Close() : Void
Send(command: Command) : Void
Receive() : Reply
Network
Public Class
Network(port: UInt16)
Network(ipAddress: String, port: UInt...
Open() : Void
Close() : Void
Receive() : Reply
Send(command: Command) : Void
Bluetooth
Public Class
Bluetooth(comport: String)
Send(command: Command) : Void
Receive() : Reply
Open() : Void
Close() : Void
GetPortNames() : String[]
Loopback
Public Class
Loopback()
Open() : Void
Close() : Void
Send(command: Command) : Void
Receive() : Reply
Command
Public Class
ReplyRequired { get; } : Boolean
Type { get; } : NXTCommandType
Data { get; } : Byte[]
Length { get; } : Int32
CommandByte { get; } : NXTCommandByte
CommandByteAsString { get; } : String
Command(type: NXTCommandType, commandByte: NXTCommandByte, reply: Boolean)
Command(data: Byte[])
Append(b: Boolean) : Void
Append(s: String) : Void
Append(s: String, maxSize: Int32, padWithZero: Boolean) : Void
Append(b: Byte) : Void
Append(b: SByte) : Void
Append(data: UInt16) : Void
Append(data: Int16) : Void
Append(data: UInt32) : Void
Append(data: Int32) : Void
Append(data: Single) : Void
Append(data: Byte[]) : Void
Append(data: Byte[], offset: Int32) : Void
Append(data: Byte[], offset: Int32, length: Int32) : Void
AppendZeros(zeros: Int32) : Void
NXTCommandByte
Public Enum
GetTunnelCommands = 40
GetTunnelRTSPSettings = 65
StartTunnelRTSP = 66
TunnelSpeak = 67
GetTunnelGPSPosition = 68
OpenRead = 128
OpenWrite = 129
Read = 130
Write = 131
Close = 132
Delete = 133
FindFirst = 134
FindNext = 135
GetFirmware = 136
OpenWriteLinear = 137
OpenReadLinear = 138
OpenWriteData = 139
OpenAppendData = 140
Boot = 151
SetBrickName = 152
GetDeviceInfo = 155
DeleteUserFlash = 160
StartProgram = 0
StopProgram = 1
PlaySoundFile = 2
PlayTone = 3
SetOutputState = 4
SetInputMode = 5
GetOutputState = 6
GetInputValues = 7
ResetInputScaledValue = 8
MessageWrite = 9
ResetMotorPosition = 10
GetBatteryLevel = 11
StopSoundPlayback = 12
KeepAlive = 13
LsGetStatus = 14
LsWrite = 15
LsRead = 16
GetCurrentProgramName = 17
MessageRead = 19
NXTCommandType
Public Enum
DirecCommand = 0
SystemCommand = 1
ReplyCommand = 2
TunnelCommand = 3
Connection
Public Abstract Class
IsConnected { get; } : Boolean
Connected : Action
Disconnected : Action
CommandSend : Action<Command>
ReplyReceived : Action<Reply>
Send(command: Command) : Void
Receive() : Reply
SendAndReceive(type: NXTCommandType, combyte: NXTCommandByte) : Reply
SendAndReceive(command: Command) : Reply
Open() : Void
Close() : Void
Box
Public Enum
Box0 = 0
Box1 = 1
Box2 = 2
Box3 = 3
Box4 = 4
Box5 = 5
Box6 = 6
Box7 = 7
Box8 = 8
Box9 = 9
Mailbox
Public Class
Mailbox()
Send(data: Byte[], inbox: Box) : Void
Send(data: Byte[], inbox: Box, reply: Boolean) : Void
Send(s: String, inbox: Box) : Void
Send(s: String, inbox: Box, reply: Boolean) : Void
Read(mailbox: Box, removeMessage: Boolean) : Byte[]
ReadString(mailbox: Box, removeMessage: Boolean) : String
1
1
NXTBrick<TSensor1, TSensor2, TSensor3, TSensor4>
Public Class
Mailbox { get; } : Mailbox
MotorA { get; } : Motor
MotorB { get; } : Motor
MotorC { get; } : Motor
Vehicle { get; } : Vehicle
Sensor1 { get; set; } : TSensor1
Sensor2 { get; set; } : TSensor2
Sensor3 { get; set; } : TSensor3
Sensor4 { get; set; } : TSensor4
FileSystem { get; } : NXTFileSystem
Connection { get; } : Connection
NXTBrick(connection: Connection)
NXTBrick(connection: String)
NXTBrick(ipAddress: String, port: UInt16)
SetBrickName(name: String) : Void
SetBrickName(name: String, reply: Boolean) : Void
GetDeviceInfo() : DeviceInfo
GetBrickName() : String
GetBluetoothAddress() : String
GetFreeFlashMemory() : UInt32
GetBatteryLevel() : UInt16
StartProgram(name: String) : Void
StartProgram(name: String, reply: Boolean) : Void
StopProgram() : Void
StopProgram(reply: Boolean) : Void
GetRunningProgram() : String
PlayTone(frequency: UInt16, durationMs: UInt16) : Void
PlayTone(frequency: UInt16, durationMs: UInt16, reply: Boolean) : Void
Beep(durationMs: UInt16) : Void
Beep(durationMs: UInt16, reply: Boolean) : Void
PlaySoundFile(name: String, loop: Boolean) : Void
PlaySoundFile(name: String, loop: Boolean, reply: Boolean) : Void
GetDeviceFirmware() : DeviceFirmware
GetFirmwareVersion() : String
GetProtocolVersion() : String
KeepAlive() : Int32
ResetBluetoothSettings(reply: Boolean) : Void
DeviceInfo
Public Class
BrickName { get; } : String
BluetoothAddress { get; } : String
FreeFlashMemory { get; } : UInt32
DeviceInfo(name: String, address: String, flashSize: UInt32)
DeviceFirmware
Public Class
ProtocolVersion { get; } : String
FirmwareVersion { get; } : String
DeviceFirmware(firmwareVersion: String, protocolVersion: String)
MonoBrickException
Public Abstract Class
ErrorCode { get; }...
MonoBrickExcepti...
MonoBrickExcepti...
TunnelException
Public Class
TunnelException(e...
TunnelException(e...
ConnectionException
Public Class
ConnectionExcepti...
ConnectionExcepti...
NXTException
Public Class
NXTException(err...
NXTException(err...
NXTBrickError
Public Enum
WrongNumberOfBytes = 120
UnknownErrorCode = 121
I2CTimeOut = 128
NoMoreHandles = 129
NoSpace = 130
NoMoreFiles = 131
EndOfFileExpected = 132
EndOfFile = 133
NotALinearFile = 134
FileNotFound = 135
HandleAlreadyClosed = 136
NoLinearSpace = 137
UndefinedFileError = 138
FileBusy = 139
NoWriteBuffers = 140
AppendNotPossible = 141
FileFull = 142
FileAlreadyExists = 143
ModuleNotFound = 144
OutOfBoundary = 145
IllegalFileName = 146
IllegalHandle = 147
PendingCommunication = 32
MailboxQueueEmpty = 64
RequestFailed = 189
UnknownCommand = 190
InsanePacket = 191
DataOutOfRange = 192
CommunicationBusError = 221
BufferFull = 222
InvalidChannel = 223
ChannelBusy = 224
NoActiveProgram = 236
IllegalSize = 237
InvalidMailboxQueue = 238
InvalidField = 239
BadIO = 240
OutOfMemory = 251
BadArguments = 255
ConnectionError
Public Enum
OpenError = 16
WriteError = 17
ReadError = 18
NoReply = 19
TunnelError
Public Enum
UnsupportedCommand = 1
ErrorExecuting = 2
OutputState
Public Structure
Speed { get; set; } : SByte
Mode { get; set; } : MotorMode
Regulation { get; set; } : MotorRegulati...
TurnRatio { get; set; } : SByte
RunState { get; set; } : MotorRunState
TachoLimit { get; set; } : UInt32
TachoCount { get; set; } : Int32
BlockTachoCount { get; set; } : Int32
RotationCount { get; set; } : Int32
Vehicle
Public Class
LeftPort { get; set; } : MotorPort
RightPort { get; set; } : MotorPort
ReverseLeft { get; set; } : Boolean
ReverseRight { get; set; } : Boolean
Vehicle(left: MotorPort, right: MotorPort)
Backward(speed: SByte) : Void
Backward(speed: SByte, reply: Boolean) : Void
Forward(speed: SByte) : Void
Forward(speed: SByte, reply: Boolean) : Void
SpinLeft(speed: SByte) : Void
SpinLeft(speed: SByte, reply: Boolean) : Void
SpinRight(speed: SByte) : Void
SpinRight(speed: SByte, reply: Boolean) : Void
Off(reply: Boolean) : Void
Off() : Void
Brake() : Void
Brake(reply: Boolean) : Void
TurnRightForward(speed: SByte, turnPercent: SByte) : Void
TurnRightForward(speed: SByte, turnPercent: SByte, reply: Boolean) : Void
TurnRightReverse(speed: SByte, turnPercent: SByte) : Void
TurnRightReverse(speed: SByte, turnPercent: SByte, reply: Boolean) : Void
TurnLeftForward(speed: SByte, turnPercent: SByte) : Void
TurnLeftForward(speed: SByte, turnPercent: SByte, reply: Boolean) : Void
TurnLeftReverse(speed: SByte, turnPercent: SByte) : Void
TurnLeftReverse(speed: SByte, turnPercent: SByte, reply: Boolean) : Void
MotorPort
Public Enum
OutA = 0
OutB = 1
OutC = 2
MotorMode
Public Enum
On = 1
Break = 2
Regulated = 4
MotorRegulation
Public Enum
Idle = 0
Speed = 1
Sync = 2
MotorRunState
Public Enum
Idle = 0
RampUp = 16
Running = 32
RampDown = 64
Motor
Public Class
Reverse { get; set; } : Boolean
Sync { get; set; } : Boolean
Motor()
SetOutputState(state: OutputState) : Void
SetOutputState(state: OutputState, reply: Boolean) : Void
On(speed: SByte) : Void
On(speed: SByte, reply: Boolean) : Void
On(speed: SByte, degrees: UInt32) : Void
On(speed: SByte, degrees: UInt32, reply: Boolean) : Void
Brake() : Void
Brake(reply: Boolean) : Void
Off() : Void
Off(reply: Boolean) : Void
GetOutputState() : OutputState
MoveTo(speed: SByte, position: Int32, reply: Boolean) : Void
MoveTo(speed: SByte, position: Int32) : Void
ResetTacho() : Void
ResetTacho(reply: Boolean) : Void
GetTachoCount() : Int32
IsRunning() : Boolean
NXTFileSystem
Public Class
MaxFileNameLength: Byte
OnBytesRead : Action<Int32>
OnBytesWritten : Action<Int32>
OnFlashDeleted : Action
NXTFileSystem()
NXTFileSystem()
FileList() : BrickFile[]
FileList(wildCard: String) : BrickFile[]
Write(file: BrickFile, data: Byte[]) : Int32
Read(file: BrickFile) : Byte[]
Read(file: BrickFile, bytesToRead: UInt16) : Byte[]
OpenWrite(fileName: String, fileSize: UInt32) : BrickFile
OpenWrite(fileName: String, fileSize: UInt32, fileType: FileMode) : BrickFile
OpenRead(fileName: String) : BrickFile
OpenAppend(fileName: String) : BrickFile
CloseFile(file: BrickFile) : Void
DeleteFile(file: BrickFile) : Void
DeleteFile(fileName: String) : Void
DeleteFlash() : Void
GetFreeFlashMemory() : UInt32
UploadFile(PCFileName: String, NXTFileName: String) : Void
DownloadFile(PCFileName: String, file: BrickFile) : Void
DownloadFile(PCFileName: String, NXTFileName: String) : Void
BrickFile
Public Class
Name { get; } : String
Handle { get; } : Byte
Size { get; } : UInt32
IsEmpty { get; } : Boolean
Extension { get; } : String
FileType { get; } : FileType
BrickFile(name: String, handle: Byte, size: UInt32)
BrickFile()
FileType
Public Enum
Firmware = 0
Program = 1
OnBrickProgram = 2
TryMeProgram = 3
Sound = 4
Graphics = 5
Datalog = 6
Unknown = 7
FileMode
Public Enum
Fragmented = 129
NoneFragmented ...
Data = 139
Sensor
Public Class
IsInitialized { get; } : Boolean
SensorDictionary { get; } : Dictionary<String, Sensor>
Sensor()
Sensor(sensorType: SensorType, sensorMode: SensorMode)
Initialize() : Void
Reset(reply: Boolean) : Void
ReadAsString() : String
SensorMode
Public Enum
Raw = 0
Boot = 32
Transition = 64
Period = 96
Percent = 128
Celsius = 160
Fahrenheit = 192
Angle = 224
SensorType
Public Enum
NoSensor = 0
Touch = 1
Temperature = 2
Reflection = 3
Angle = 4
LightActive = 5
LightInactive = 6
SoundDB = 7
SoundDBA = 8
Custom = 9
LowSpeed = 10
LowSpeed9V = 11
SensorPort
Public Enum
In1 = 0
In2 = 1
In3 = 2
In4 = 3
RCXRotationSensor
Public Class
RCXRotationSensor()
ReadCount() : Int32
NoSensor
Public Class
NoSensor()
RCXLightSensor
Public Class
SensorMode { get; set; } : SensorMode
RCXLightSensor()
RCXLightSensor(sensorMode: SensorMode)
ReadLightLevel() : Int32
HiTecGyro
Public Class
Offset { get; set; } : Int32
HiTecGyro()
HiTecGyro(offset: Int32)
ReadAngularAcceleration() : Int...
ReadAsString() : String
I2CMode
Public Enum
LowSpeed = 10
LowSpeed9V = 11
I2CBase
Public Abstract Class
I2CAddress { get; } : Byte
I2CBase(mode: I2CMode, sensorAddress: Byte)
I2CBase(mode: I2CMode, sensorAddress: Byte, pollInterval: Int32)
Initialize() : Void
ReadAsString() : String
Reset(reply: Boolean) : Void
NXTLightSensor
Public Class
LightMode { get; set; } : LightMode
SensorMode { get; set; } : SensorMode
NXTLightSensor()
NXTLightSensor(lightMode: LightMode)
NXTLightSensor(lightMode: LightMode, sensorMode: SensorMode)
ReadLightLevel() : Int32
LightMode
Public Enum
Off = 6
On = 5
CustomSensor
Public Class
CustomSensor()
CustomSensor(type: SensorType, mode: SensorMode)
Read() : Int32
TouchSensor
Public Class
SensorMode { get; set; } : SensorMode
TouchSensor()
TouchSensor(sensorMode: SensorMode)
NXTSoundSensor
Public Class
SoundMode { get; set; } : SoundMode
NXTSoundSensor()
NXTSoundSensor(soundMode: SoundMode)
ReadSoundLevel() : Int32
SoundMode
Public Enum
SoundDBA = 8
SoundDB = 7
RCXTemperatureSensor
Public Class
TemperatureMode { get; set; } : TemperatureMode
RCXTemperatureSensor()
RCXTemperatureSensor(temperatureMode: TemperatureMo...
ReadTemperature() : Int32
TemperatureMode
Public Enum
Celsius = 160
Fahrenheit = 192
Sonar
Public Class
Mode { get; set; } : SonarMode
Sonar()
Sonar(mode: SonarMode)
Initialize() : Void
ReadDistance() : Int32
SingleShot(reply: Boolean) : Void
Off(reply: Boolean) : Void
Continuous(reply: Boolean) : Void
Reset(reply: Boolean) : Void
ReadAsString() : String
SonarMode
Public Enum
Centimeter = 1
CentInch = 2
PCF8591
Public Class
PCF8591()
PCF8591(address: Byte)
Initialize() : Void
Read() : Int32
Read(port: ADCPort) : Byte
Write(port: ADCPort, value: Byte) : Void
ReadAsString() : String
ADCPort
Public Enum
Port0 = 0
Port1 = 1
Port2 = 2
Port3 = 3
I2CSensor
Public Class
I2CSensor()
I2CSensor(mode: I2CMode, sensorAddress: Byte)
I2CSensor(mode: I2CMode, sensorAddress: Byte, pollInterval: Int...
ReadRegister(register: Byte) : Byte[]
ReadRegister(register: Byte, nLength: Byte) : Byte[]
WriteRegister(register: Byte, data: Byte) : Void
WriteRegister(register: Byte, data: Byte, reply: Boolean) : Void
ReadAsString() : String
PCF8574
Public Class
PCF8574()
PCF8574(address: Byte)
Initialize() : Void
Read() : Int32
Write(set: Byte) : Void
ReadAsString() : String
HiTecCompass
Public Class
HiTecCompass()
Initialize() : Void
ReadDirection() : Int32
ReadAsString() : String
HiTecTilt
Public Class
HiTecTilt()
Initialize() : Void
ReadPosition() : Position
ReadAsString() : String
Position
Public Class
X { get; } : Int32
Y { get; } : Int32
Z { get; } : Int32
Position(x: Int32, y: Int32, z: Int32)
HiTecColor
Public Class
HiTecColor()
Initialize() : Void
ReadColorIndex() : Int32
ReadRGBColor() : RGBColor
ReadNormalizedRGBColor() : RGBColor
ReadAsString() : String
RGBColor
Public Class
Red { get; } : Byte
Green { get; } : B...
Blue { get; } : Byte
RGBColor(red: Byt...