**Overview:** This document documents the CAN messages used in a NEOS system. The NEOS uses a specialized bit rate that is not supported by most CAN bus interfaces to allow audio to be streamed over the bus. The bitrate for the NEOS bus is 1024000Bits/Second and uses 11-bit CAN IDs with 8-byte messages. Some of CAN messages are documented using byte fields and others are with word fields. Word 0 🡪 Data Byte 0 (LSB) + Data Byte 1 (MSB).

A NEOS system is comprised of “nodes” connected via the CAN bus. There are 3 types of nodes: slaves, a master and a display. The slave nodes are the “buttons/LEDs” that a user interacts with. The master node contains the game logic and streams audio to the slave. The display node is the score board. Each node has an internal 1-byte address. The slave nodes use addresses 0-15. The master node address is 247 and the display node is at address 248. The display and master nodes have fixed addresses and the slaves are addressed during installation.

**Slave Oriented Messages**

**Message:** **LED\_COMMAND**

**Message ID:** 0x0010

**Description:**  This message is used to turn on the Button node LEDs. This command can be used to fade the node from a start color to a end color. The brightness parameters for each LED color are 6-bits and timing parameters are 10 bits. There is also a timeout parameter where the LEDs will turn off after a timeout value. All of the parameters are tightly packed into the 8-bytes of the CAN message. This requires one to used proper bit shifting to create a proper LED command. Note that while there are fields for blue color, the current hardware only implements red and green.

**Message Structure:**

**Byte 0 :** Red start brightness + 2 LSB’s of green start brightness shifted up 6 bits

**Byte 1 :** Green start brightness 4 MSB’s + 4 LSB’s of blue start brightness shifted up 4 bits  
**Byte 2 :** 2 MSB’s of the blue start brightness + red brightness target at end of fade shifted up 2 bits

**Byte 3 :** Green brightness target at end of the fade + 2 LSB’s of blue brightness target at end of the fade shifted up 6 bits

**Byte 4 :** 4 MSB’s of the blue brightness target at end of fade+ 4 LSB’s of the 10-bit LED timeout shifted up 4 bits

**Byte 5 :** 6 MSB’s of 10-bit LED timeout in 10ms steps, (use 0 for no timeout) + 2 LSB’s of the fade time

**Byte 6 :** 8 MSB’s of LED fade time in 10ms steps (0 for no fading)

**Byte 7 :** Node Number (a value of 0xff make all nodes accept command)

**Message:** **BUTTON\_PRESS**

**Message ID:** 0x00F0

**Description:**  This message is sent when a button on a slave node is pressed.

**Message Structure:**

**Byte 0** : Node where the button was hit

**Byte 1** : n/a  
**Byte 2** : n/a

**Byte 3** : n/a

**Byte 4** : n/a

**Byte 5** : n/a

**Byte 6** : n/a

**Byte 7** : n/a

**Message:** **RESET\_ALL\_SLAVE\_NODES**

**Message ID:** 0x00E0

**Description:**  This message resets all of the button slave nodes and the score display. It clears the CAN Tx/Rx buffers, turns off the audio and turns the LEDs off.

**Message Structure:**

**Byte 0** : n/a

**Byte 1** : n/a  
**Byte 2** : n/a

**Byte 3** : n/a

**Byte 4** : n/a

**Byte 5** : n/a

**Byte 6** : n/a

**Byte 7** : n/a

**Message:** **AUDIO\_OFF\_ALL\_NODES**

**Message ID:** 0x008E

**Description:**  This message turns off the audio on all nodes.

**Message Structure:**

**Byte 0** : n/a

**Byte 1** : n/a  
**Byte 2** : n/a

**Byte 3** : n/a

**Byte 4** : n/a

**Byte 5** : n/a

**Byte 6** : n/a

**Byte 7** : n/a

**Message:** **NOOP**

**Message ID:** 0x008C

**Description:**  No-operation. This command is used to fill can receive mailboxes to work around message ordering issues.

**Message Structure:**

**Byte 0** : n/a

**Byte 1** : n/a  
**Byte 2** : n/a

**Byte 3** : n/a

**Byte 4** : n/a

**Byte 5** : n/a

**Byte 6** : n/a

**Byte 7** : n/a

**Message:** **SET\_SLAVE\_TO\_ADDRESS**

**Message ID:** 0x00E3

**Description:**  Sets the slave to the address in byte 0 when its button is pressed.

**Message Structure:**

**Byte 0** : Node Address to Accept

**Byte 1** : n/a  
**Byte 2** : n/a

**Byte 3** : n/a

**Byte 4** : n/a

**Byte 5** : n/a

**Byte 6** : n/a

**Byte 7** : n/a

**Message:** **CANCEL\_SET\_SLAVE\_TO\_ADDRESS**

**Message ID:** 0x00E4

**Description:**  Cancels the command issued with a SET\_SLAVE\_TO\_ADDRESS command.

**Message Structure:**

**Byte 0** : n/a

**Byte 1** : n/a  
**Byte 2** : n/a

**Byte 3** : n/a

**Byte 4** : n/a

**Byte 5** : n/a

**Byte 6** : n/a

**Byte 7** : n/a

**Message:** **PLAY\_INTERNAL\_SOUND**

**Message ID:** 0x008B

**Description:**  Each of the slaves has a limited amount of internal FLASH memory to store “internal sounds”. This command can trigger the internal sounds. This function was added to overcome limitations in the audio streaming methods which only allow for 4 simultaneous sounds spread across all the nodes. The internal sounds allow for common sounds to be generated internally. As of SVN BUILD 42, the following internal sounds are available:

#define INTERNAL\_SOUND\_POSITIVE\_FEEDBACK 0

#define INTERNAL\_SOUND\_SELECTION 1

#define INTERNAL\_SOUND\_FF\_WATER\_HIT 2

#define INTERNAL\_SOUND\_FF\_WATER\_MISS 3

#define INTERNAL\_SOUND\_FF\_FIRE\_APPEAR 4

#define INTERNAL\_SOUND\_FF\_FIRE\_OUT 5

**Message Structure:**

**Byte 0**: Node number to play sound (0xFF commands all nodes to play internal sound)

**Byte 1** : Sound Number (See macros above)

**Byte 2** : Volume—0x00 is off and 0xFF is Full  
**Byte 3** : Number of Times to Repeat

**Byte 4** : Stream to switch to when complete (0 – 3)

**Byte 5** : Volume after playback is complete

**Byte 6** : Audio Status after playback complete (0 is audio off, 1 is audio on)

**Byte 7** : n/a

**Message:** **PLAY\_AUDIO\_STREAM**

**Message ID:** 0x0088

**Description:**  Commands a slave node to accept a particular audio stream for a programmable time period.

**Message Structure:**

**Byte 0 :** Node number. 0xFF commands all nodes

**Byte 1 :** Bits 0,1 🡪Stream to switch to

Bits 2,3 🡪 Stream after Timeout

Bit 4 🡪 Audio Status (0 = Off, 1 = On)

Bit 5 🡪 Audio Enable After Timeout (0 = Off, 1 = On)

**Byte 2 :** LSB of Timeout value in 1/100 sec steps (0xFFFF if no timeout)

**Byte 3 :** MSB of Timeout value in 1/100 sec steps

**Byte 4 :** Audio Volume

**Byte 5 :** Audio Volume After Timeout

**Byte 6 :** n/a

**Byte 7** **:** n/a

**Message:** **SYNC\_NODE\_TO\_STREAM**

**Message ID:** 0x0089

**Description:**  Synchronizes a node to its current audio stream. Node will set its internal mute, then look for a write to buffer address 0x0 followed by a write to 0x80. When the write to 0x80 is seen, they will reset their read indexes to 0x0 to read data exactly 0x80 behind what’s been written. This is used to re-sync a specific node to an already playing stream, such as when an active button needs to switch back to background music

**Message Structure:**

**Byte 0 :** Node number.

**Byte 1 :** n/a

**Byte 2 :** n/a

**Byte 3 :** n/a

**Byte 4 :** n/a

**Byte 5 :** n/a

**Byte 6 :** n/a

**Byte 7** **:** n/a

**Message:** **SYNC\_NODE\_TO\_STREAM**

**Message ID:** 0x008A

**Description:**  Same function as SYNC\_NODE\_TO\_STREAM except that all slave nodes are command to synchronize to the stream specified in byte 0.

**Message Structure:**

**Byte 0 :** Stream number for all the slave nodes to sync to.

**Byte 1 :** n/a

**Byte 2 :** n/a

**Byte 3 :** n/a

**Byte 4 :** n/a

**Byte 5 :** n/a

**Byte 6 :** n/a

**Byte 7** **:** n/a

**Message:** **NODE\_OPERATIONS**

**Message ID:** 0x02F2

**Description:**  This message is used for testing/debugging purposes. Currently applies to slave nodes (not the scoreboard or master). The first byte is always a command and the others are dependent on which command was sent

**Message Structure:**

**Byte 0:** Command:

**NODE\_PING** 0x01

**NODE\_PONG** 0x02

**NODE\_STATISCALLY\_ACCEPT\_UID** 0x03

**NODE\_UNCONDITIONALLY\_ACCEPT\_UID** 0x04

**NODE\_READ\_SUPPLY\_VOLTAGE** 0x05

**NODE\_SUPPLY\_VOLTAGE\_RESPONSE** 0x06

**NODE\_LOOP\_BACK** 0x07

**NODE\_LOOP\_BACK\_RESPONSE** 0x08

Command Specific details:

*NODE\_PING*: Used to test for nodes on the bus

**Byte 1:** Node ID of the ping.

**Bytes 2- 7**: n/a

Node will light its LEDs to YELLOW when it receives this command for a random time between 0.5 to 2 seconds and then switch to green. Node will respond with a NODE\_PONG message with this organization:

**Byte 0: NODE\_PONG**

**Byte 1:** Node ID of the ping.

**Bytes 2- 7**: pseudo-random bytes populated with stdlib rand()

*NODE\_STATISCALLY\_ACCEPT\_UID*: Accept UID if and only if an internal random number generator

is less than 0x8000

*NODE\_UNCONDITIONALLY\_ACCEPT\_UID*: Node that should accept the UID without condition

**Byte 2-5: UID to Accept (32-bit little endian ordered)**

**Bytes 6,7: n/a**

After this command is received, node will respond with this message:

**Byte 0:** NODE\_PONG\_CODE

**Byte 1: Node ID that received the command**

**Byte 2-5: UID of the node. If the UID had not changed with the statistical command, the old UID will be sent.**   
**Bytes 6,7**: n/a

*READ\_SUPPLY\_VOLTAGE*:

**Byte 1:** Node Address of the Command

**Bytes 2-7:** n/a

The node will respond with a **NODE\_SUPPLY\_VOLTAGE\_RESPONSE** commandmessage

**Byte 1:** Responding node address

**Bytes 2-3:** 16-Bit little endian Voltage reading. 310 Counts/Volt

**Bytes 4-7**: n/a

*NODE\_LOOP\_BACK:*

**Byte 1: Node Number to loop data back**

**Bytes 2-7:** **Loop Back Data:**

After receiving this command, the receiving node will respond with a NODE\_LOOP\_BACK\_RESPONSE\_MESSAGE:

**Byte 1:** Node number providing loop back data

**Bytes 2-7:** Loop back data sent with a NODE\_LOOP\_BACK\_COMMAND

**Scoreboard Oriented Messages**

**Message:** **SEND\_SCORE\_SEGMENT\_DATA**

**Message ID:** 0x0070

**Description:**  Commands the segments on the score display. The segment mapping is shown below.



**Message Structure:**

**Byte 0 :** Display Address (0xF8)

**Byte 1 :** Brightness 0x00-0x40

**Byte 2 :** Digit 0 (far right character of display) bit 0=**segment a,** bit 1= **segment b,** etc.

**Byte 3 :** Digit 1

**Byte 4 :** Digit 2

**Byte 5 :** Digit 3

**Byte 6 :** Digit 4

**Byte 7** **:** Digit 5

**Message:** **SCORE\_LED\_COMMAND**

**Message ID:** 0x0072

**Description:**  Turns on the score button LEDs. Each selector button on the score board has an integrated LED

**Message Structure:**

**Byte 0 :** Display Address (0xF8)

**Byte 1 :** 0

**Byte 2 :** LEDs 0-7

**Byte 3 :** LEDs 8-15

**Byte 4 :** LEDs 16-19

**Byte 5 :** n/a

**Byte 6 :** n/a

**Byte 7** **:** n/a

**Message:** **SCORE\_BUTTONS\_CHANGED**

**Message ID:** 0x00F3

**Description:**  This message is sent by a score display when a score board/selector button is pressed.

**Message Structure:**

**Byte 0 :** Display Address (0xF8)

**Byte 1 :** n/a

**Byte 2 :** Score Buttons 0-7

**Byte 3 :** Score Buttons 8-15

**Byte 4 :** Score Buttons 16-19

**Byte 5 :** n/a

**Byte 6 :** n/a

**Byte 7** **:** n/a

**Master Oriented Messages**

**Message:** **SET\_MASTER\_SYSTEM\_MODE**

**Message ID:** 0x02F3

**Description:**  Resets all the timers, Resets the GameState to INIT and the DiagnosticState to INIT. Note: This message is for debugging purposes only. Not for general consumption.

**Message Structure:**

**Byte 0:** Master System MODE:

GAME\_ACTIVE 0x00

USB\_FLASH\_UPDATE 0x01

SYSTEM\_IDLE 0x02

SYSTEM\_SOFTWARE\_RESET 0x03

SYSTEM\_DIAGNOSTICS 0x04

SYSTEM\_BOOT 0xFE

**Byte 1-7:** n/a

**Audio Stream Oriented Messages**

**Message:** **WRITE\_AUDIO\_STREAM**

**Message ID:** 0x0100 – 0x1FF

**Description:**  This message set is for the audio streaming functions of the NEOS. The NEOS master transmits 4 audio streams on the CAN bus. To play a sound, the master will command a slave to grab data from a particular stream for a particular amount of time (the sound length) and then switch to another stream(such as the background music stream) or turns its internal volume to 0x00 (don’t play any sounds). These 4 audio streams are continuously transmitting. It is the job of the master to continuously provide stream data at a sample rate that matches the slave. This timing must match for continuous audio without glitches. Each slave has an internal audio buffer that holds data to be sent to its D/A at an 8KHz rate. This buffer is dividing to a top and bottom section (ping-pong style). The slave plays data from one of the sections while it waits for the other to be filled. The master’s stream timing must be matched to the slave’s to achieve proper buffer synchronization. To achieve this timing, the master and slave microcontrollers are of the same type and use the same reference frequency (set by a crystal). The master also uses the same D/A interrupt routine as the slaves except that it does not output any real data to its D/A port. It only uses it for timing purposes.

The slaves have 256-byte audio buffers. This is segmented into an upper and lower half of 128 bytes each. Data is streamed at an 8KHz sample rate with each audio sample being in a signed 8-bit format (2’s complement). The messages for the audio streams use CAN ID’s 0x100 to 0x1FF (or 0x00 to 0xFF with a 0x100 offset). Since each CAN message is 8-bytes, think of the messages as virtual memory “writes” performed at 8-bytes at a time. A 256 (0xFF) ID span can map to a total memory space 2048 bytes (256 \* 8). Each audio stream gets its own “space” of 0x40 CAN message IDs:

**CAN IDs**

0x100 – 0x3ff🡪 Stream 0

0x140 – 0x7ff🡪 Stream 1

0x180 – 0xBff🡪 Stream 2

0x1C0 – 0x1ff🡪 Stream 3

Each stream is allocated a CAN ID space of 0x40 messages which can address a buffer of 512 bytes (0x40 \* 8). In the current firmware, the slave audio buffers are 256 bytes in length. The lower half of this buffer is mapped to CAN Id offsets 0x00 – 0xF and the upper half of the buffer is mapped to CAN ID offset 0x20 to 0x2F.

**CAN ID Buffer Mapping**

0x100-0x10F : Stream 0 Lower Buffer Half ------ 0x120-0x12F : Stream 0 Upper Buffer Half

0x140-0x10F : Stream 1 Lower Buffer Half ------ 0x160-0x16F : Stream 1 Upper Buffer Half

0x180-0x10F : Stream 2 Lower Buffer Half ------ 0x1A0-0x1AF : Stream 2 Upper Buffer Half

0x1C0-0x10F : Stream 3 Lower Buffer Half ------ 0x1D0-0x1DF : Stream 3 Upper Buffer Half

The master node is constantly writing to these lower and upper buffers in succession. Slaves can be synchronized to a stream via a **SYNC\_NODE\_TO\_STREAM.** The **SYNC\_NODE\_TO\_STREAM** message will command the slave node to synchronize its buffer operations to ensure that is correctly outputting one half of the audio buffer while the master is filling the other.