

Control of BSS Audio Soundweb London Series using AMX

(Soundweb London Architect v3.00)

This document is outlined as follows:

Introduction

Section 1 – Getting Started

Section 2 – User Function/Subroutine Definitions

Examples of Sending Commands

Section 3 – Receiving Commands/Feedback

Section 4 – Adding Additional Functionality

Section 5 – Programmer Function/Subroutine Definitions

Introduction:

This purpose of this document is to outline the control of a BSS Audio Soundweb London by an AMX Netlinx control system using IP/RS232 communications. The full protocol specification for the Soundweb London can be obtained by downloading the latest software from <http://www.soundweb-london.com/>. The 3rd Party Control Interface Kit [London DI kit .pdf] is located in the London Architect directory when installed. The Soundweb London has such a large range of controls that only the most common controls are included. If the control of a certain device/parameter is not included in section 1, it can be added by reading section 4 (Adding Additional Functionality) and using the London DI kit.pdf.

Section 1: Getting Started

One sample AMX program has been included for references; BSS Basic. This system has a touch panel, Nextlinx code, and a sample London Architect file. This program shows many, many examples of different commands, string parsing, true-feedback, and more. Please open these programs as you read through this guide for clarity.

Comm Port = 115200, N, 8, 1

IP = TCP/IP; Port 1023

“Friendly” name = A constant, either a CHAR or an INTEGER, that is declared and assigned a friendly name within AMX.

Ex: CHAR Blu_RoomMixer[] = {\$11,\$8B,\$03,\$00,\$01,\$0C}; //Mixer FRIENDLY Name

State-Variable = A constant within BSS Soundweb London that identifies every control. Every control in each Soundweb London processing object is assigned a unique SV (state-variable). The value of the SV designates the current state of that control.

HiQnet address = The 6 byte address of every device within Soundweb London Architect. The HiQnet address is obtained in the **“properties” window** (not the DI Tool bar) of the wanted device. It is made up of Node Address<2bytes>, VD<1byte>, ObjectID<3bytes>. The object ID is generated by the numerical order a device is placed within Soundweb London.

Ex. = Means example. This is the exact code in AMX syntax needed to execute a command.

Subscribe = When you subscribe to a State-Variable, the Soundweb London will send an unsolicited updates automatically whenever that state-variable is changed in order to keep the AMX system in sync with the London without requiring extra effort from the programmer to set up

‘polling’, or requiring the AMX processor to constantly check for updates. The first time the subscribe message is sent the Soundweb London will respond with its current state much like a ‘GET’ statement. The Soundweb London will keep sending updates until a ‘UNSUBSCRIBE’ command is sent.

Every string length is 17 bytes...kinda, sorta, with an ‘and’ and a ‘but’. Every command is 17 bytes before it is transmitted. However, if a character holds the value of a reserved char then it is replaced with an escape character \$1B and the reserved char has \$80 (128 decimal) added to it before transmitting. This code takes care of everything for the programmer but if the programmer desires to understand this protocol in greater depth then see London DI kit.pdf for full details.

Lastly, inside London Architect (LA) there is a Serial Toolbar which will show the EXACT string needed to control anything inside LA by simply clicking on the desired control. This is a very useful tool to supplement this code and is very beneficial for debugging purposes.

Section 2: User Function/Subroutine Definitions

‘SET_MIXER’ - controls state-variables listed in PARAM for any Mixer or Automixer device

SET_MIXER (char OBJECT[6], integer INPUT, integer OUTPUT, integer PARAM, integer VALUE)

- OBJECT = “Friendly” name; Copy and Paste HiQnet address from properties window in London Architect
- INPUT = input channel; 1 - # of inputs on device; a zero indicates output only control
- OUTPUT = output channel; 1,3; ‘1’ is Master out on Mono or Left output on a Stereo. A ‘3’ is Right output on stereo. A zero indicates input only control.
- PARAM = “Friendly” name; Type of action to execute (ie. MUTE, GROUP, SOLO)
- VALUE = ON or OFF; use a 1 for ON and a zero for OFF

Possible Parameter Values:

PARAM

| | |
|------------|--|
| MUTE | - Mute/Unmute an input/output (Value = 0 or 1) |
| SOLO | - Turn ON/OFF the SOLO function on input faders (Value = 0 or 1) |
| AUX | - Mute/Unmute Aux outputs (Value = 0 or 1) |
| GROUP | -Route/UnRoute Groups; Mute/UnMute Group master outputs (Value = 0 or 1) |
| OVERRIDE | -Turn ON/OFF input Overrides (Value = 0 or 1) |
| AUTO | -Turn ON/OFF input Auto (Value = 0 or 1) |
| PAN | -Use VALUE of 0-100 (percent) |
| OFF_GAIN | -Use VALUE of 0-100 (percent) |
| AUX_GAIN | -Use VALUE of 0-100 (percent) |
| GROUP_GAIN | -Use VALUE of 0-100 (percent) |

Ex: SET_MIXER(Blu_RoomMixer,2,0,MUTE,1) – Mutes input 2 on Blu_Room Mixer

Ex: SET_MIXER(Blu_RoomMixer,0,1,MUTE,0) – UnMutes master output on Blu_Room Mixer

Ex: SET_MIXER(Blu_RoomMixer,3,4,GROUP,1) – Assigns input 3 to Group 4 on Blu_Room Mixer

Ex: SET_MIXER(Blu_RoomMixer,0,1,GROUP,1) – Mutes Group 1 master output

Ex: SET_MIXER(Blu_RoomMixer,0,1,GROUP_GAIN,73) – Sets group 1 gain to 73% (0dB)

‘SET_ROOMCOMBINE’ - controls state-variables listed in PARAM for a Room Combine object

SET_ROOMCOMBINE (char OBJECT[6], integer INPUT, integer OUTPUT, integer PARAM, integer VALUE)

- OBJECT = “Friendly” name; Copy and Paste HiQnet address from properties window in London Architect

- INPUT = input channel; 1 - # of inputs on device; a '0' indicates output only control
- OUPUT = output channel; 1 - #of outputs on device; a '0' indicates input only control.
- PARAM = "Friendly" name; Type of action to execute (ie. SOURCE_GAIN, BGM_GAIN, SOURCE_MUTE, BGM_MUTE)
- VALUE = ON/OFF/MUTE/UNMUTE (0 or 1) for discrete Mute SV's. All GAIN's are 0-100 (percent)

**BGM = Background Music

**RC = Room Combine Object

Possible Parameter Values:

PARAM

| | |
|-------------|---|
| SOURCE_MUTE | - Mute/Unmute's the Source input (Value = 0 or 1) |
| BGM_MUTE | - Mute/Unmute's the BGM input (Value = 0 or 1) |
| MASTER_MUTE | - Mute/Unmute's the Master output (Value = 0 or 1) |
| BGM_SELECT | -Selects the BGM source; (Value = 0 - # of BGM inputs) |
| SOURCE_GAIN | -Sets the Source input fader to Value (Use VALUE of 0-100 (percent)) |
| BGM_GAIN | -Sets the BGM input fader to Value (Use VALUE of 0-100 (percent)) |
| MASTER_GAIN | -Sets the Master output fader to Value (Use VALUE of 0-100 (percent)) |
| PARTITION | -Open/Closes a partition |
| GROUP | -Sets/Gets the room's Group SV |

Ex: SET_ROOMCOMBINE(Hall_A,2,0,SOURCE_MUTE,1) – Mutes Room 2's Source input on Hall_A Room Combine object (RC)

Ex: SET_ROOMCOMBINE(Hall_A,2,0,SOURCE_MUTE,0) – Un-Mutes Room 2's Source input on Hall_A Room Combine Object

Ex: SET_ROOMCOMBINE(Hall_A,0,4,MASTER_MUTE,1) – Mutes Room 4's Master output on Hall_A Room Combine object (RC)

Ex: SET_ROOMCOMBINE(Hall_A,0,2,MASTER_GAIN,73) – Set's Room 2's Master output fader on Hall_A to 73% (0dB)

Ex: SET_ROOMCOMBINE(Hall_A,2,0,BGM_GAIN,60) – Set's Room 2's BGM input fader on Hall_A to 60%

Ex: SET_ROOMCOMBINE(Hall_A,1,0,BGM_SELECT,6) – Selects Room 1's BGM source on Hall_A Room Combine object (RC) to input 6.

Ex: SET_ROOMCOMBINE(HALL_A,3,0,PARTITION,1) – Open's Partition #3

Ex: SET_ROOMCOMBINE(HALL_A,3,0,PARTITION,0) – Close's Partition #3

NOTE: When rooms are combined all of their controls are linked. If a mute is pressed in room 1 then all other rooms that are combined with room 1 will mute as well. True feedback will be automatically reported for the linked rooms without the programmer needing to add any additional code.

'SET_VAL' – controls state-variables that are in DEVICE and listed by PARAM

SET_VAL(char OBJECT[6], integer DEVICE, integer INPUT, integer OUTPUT, integer PARAM, integer VALUE)

- OBJECT = "Friendly" name; Copy and Paste HiQ address from properties window in London Architect
- DEVICE = "Friendly" name; Used to tell functions what type of device is being controlled
- INPUT = input channel; 1 - # of inputs on device; a zero indicates output only control
- OUPUT = output channel; 1 -# of outputs on a device. A zero indicates input only control.
- PARAM = "Friendly" name; Type of action to execute (ie. MUTE, GROUP, SOLO)
- VALUE = Value of PARAM (ie. a 1or 0; action to execute mute, unmute, route, unroute,etc)

Possible Parameter Values:

| <u>DEVICE</u> | <u>PARAM</u> | <u>VALUE</u> |
|---------------|--------------|--------------------|
| ROUTER | MUTE | 1 or 0 (on or off) |

| | | |
|-----------------------|-----------|--|
| MM <matrix mixer> | ROUTE | 0 - # of inputs for Source_Selector |
| SOURCE_MATRIX | ATTACK | 0 = Off for Phantom Pwr, 100 = On for Phantom Pwr; (%) |
| SOURCE_SELECTOR | RELEASED | |
| GAIN (Single Channel) | REFERENCE | |
| N_GAIN | PHANTOM | |
| INPUT_CARD | | |
| OUTPUT_CARD | | |

Ex: SET_VAL(JukeBox, ROUTER,2,16,ROUTE,1) – Routes input 2 to output 16 on JukeBox Router
 Ex: SET_VAL(BallRoom, MM, 1,1,UNMUTE,0) – UnMutes crosspoint input 1 output 1 on a Matrix
 Ex: SET_VAL(BGM, SOURCE_SELECTOR, 0,0, ROUTE,2) – Sets the Source Selector to input 2
 Ex: SET_VAL(IO_CARD1,INPUT_CARD,1,0, PHANTOM,100) – Turns ON Phantom Pwr CardA (%)

‘SET_GAIN%’ – controls gain state-variables that are in *DEVICE* and assigns level given by *PERCENT*
 -see LONDON DI Kit.pdf for full use and explanation of gain values

SET_GAIN%(char OBJECT[6], integer DEVICE, integer INPUT, integer OUTPUT, integer PERCENT)
 - OBJECT = “Friendly” name; Copy and Paste HiQ address from properties window in London Architect
 - DEVICE = “Friendly” name; Used to tell functions what type of device is being controlled
 - INPUT = input channel; 1 - # of inputs on device; a zero indicates output only control
 - OUPUT = output channel; 1 -# of outputs on a device. A zero indicates input only control.
 - PERCENT = Level of gain represented by integer in decimal. Increments of 1% = 1/3dB

Possible Parameter Values:

| | |
|-------------------|--|
| <u>DEVICE</u> | <u>PERCENT</u> |
| MIXER | 0-100 <-70dB = 0; 0dB = 73; +10dB = 100> |
| MM <matrix mixer> | |
| AUTOMIXER | |
| GAIN | |
| N_GAIN | |
| INPUT_CARD | |
| OUTPUT_CARD | |

Ex: SET_GAIN%(Blu_Room,AUTOMIXER,2,0,0) – Sets gain to -70dB on input 2 of an Automixer
 Ex: SET_GAIN%(BallRoom, MM, 1,1,73) – Sets crosspoint gain input1 output 1 to 0dB
 Ex: SET_GAIN%(HouseVol, N_GAIN,3,0,100) – Sets input 3 gain to +10dB of N-Gain device

‘SET_GAIN’ – controls gain state-variables that are in *DEVICE* and assign’s level given by *VALUE*
 -see LONDON DI Kit.pdf for full use and explanation of gain values

SET_GAIN(char OBJECT[6], integer DEVICE, integer INPUT, integer OUTPUT, slong VALUE)
 - OBJECT = “Friendly” name; Copy and Paste HiQ address from properties window in London Architect
 - DEVICE = “Friendly” name; Used to tell functions what type of device is being controlled
 - INPUT = input channel; 1 - # of inputs on device; a zero indicates output only control
 - OUPUT = output channel; 1 -# of outputs on a device. A zero indicates input only control.
 - VALUE = Level of gain represented by SLONG in decimal

Possible Parameter Values:

| <u>DEVICE</u> | <u>VALUE</u> |
|-------------------|--|
| MIXER | -300,000 – 100,000 <0dB = 0; 10dB = 100,000> |
| MM <matrix mixer> | |
| AUTOMIXER | |
| GAIN | |
| N_GAIN | |
| INPUT_CARD | |
| OUTPUT_CARD | |

Ex: SET_GAIN(Blu_Room,AUTOMIXER,2,0,0) – Sets gain to 0dB on input 2 of an automixer
 Ex: SET_GAIN(BallRoom, MM, 1,1,-300000) – Sets crosspoint gain input1 output 1 to -70dB
 Ex: SET_GAIN(HouseVol, N_GAIN,0,1,100000) – Sets Master Out Gain to +10dB of N-Gain device

‘SET_PRESET’ – recalls device and parameter presets

SET_PRESET(integer PRESET_TYPE, integer PRESET_NUMBER)
 - PRESET_TYPE = “PARAMETER_PRESET” or “DEVICE_PRESET”
 - PRESET_NUMBER = # of preset; 1 – a lot

Ex: SET_PRESET(PARAMETER_PRESET, 1) – Recalls Parameter preset one
 Ex: SET_PRESET(DEVICE_PRESET, 2) – Recalls Device preset two

‘SUBSCRIBE%’ – When you subscribe to a State-Variable, the Soundweb London will send an update string whenever that state-variable is changed. The first time the subscribe message is sent the Soundweb London will respond with its current state much like a ‘GET’ statement. The Soundweb London will keep sending updates until a ‘UNSUBSCRIBE%’ is sent. **Typically used for Gain states, Meter states, etc.**

SUBSCRIBE%(char OBJECT[6], integer DEVICE, integer INPUT, integer OUTPUT, integer PARAM)
 - OBJECT = “Friendly” name; Copy and Paste HiQ address from properties window in London Architect
 - DEVICE = “Friendly” name; Used to tell functions what type of device is being controlled
 - INPUT = input channel; 1 - # of inputs on device; a zero indicates output only control
 - OUPUT = output channel; 1 -# of outputs on a device. A zero indicates input only control.
 - PARAM = “Friendly” name; action to execute

Possible Parameter Values:

| <u>DEVICE</u> | <u>PARAM</u> |
|-------------------|---|
| ROUTER | GAIN |
| MM <matrix mixer> | METER (will return state of meter 10 times per second by default) |
| SOURCE_MATRIX | |
| SOURCE_SELECTOR | |
| MIXER | |
| AUTOMIXER | |
| GAIN | |
| N_GAIN | |
| INPUT_CARD | |
| OUTPUT_CARD | |

Ex: SUBSCRIBE%(Blu_Room, Mixer,1,0,GAIN) – Will subscribe to input 1 fader
 Ex: SUBSCRIBE%(Blu_Room, Mixer,0,1,GAIN) – Will subscribe to master output fader

To stop receiving updates:

Ex: UN_SUBSCRIBE%(Blu_Room, MIXER,1,0,GAIN) – Will un_subscribe to input 1 fader

‘SUBSCRIBE’ – When you subscribe to a State-Variable, the Soundweb London will send an update string whenever that state-variable is changed. The first time the subscribe message is sent the SoundwebLondon will respond with its current state much like a ‘GET’ statement. The Soundweb London will keep sending updates until a ‘UNSUBSCRIBE’ is sent. **Typically used for discrete states such as Mute, UnMute, ON, OFF, Route, UnRoute.**

SUBSCRIBE(char OBJECT[6], integer DEVICE, integer INPUT, integer OUTPUT, integer PARAM)

- OBJECT = “Friendly” name; Copy and Paste HiQ address from properties window in London Architect
- DEVICE = “Friendly” name; Used to tell functions what type of device is being controlled
- INPUT = input channel; 1 - # of inputs on device; a zero indicates output only control
- OUPUT = output channel; 1 -# of outputs on a device. A zero indicates input only control.
- PARAM = “Friendly” name; action to execute

Possible Parameter Values:

| <u>DEVICE</u> | <u>PARAM</u> |
|-------------------|-------------------------------|
| ROUTER | MUTE or UNMUTE (same thing) |
| MM <matrix mixer> | ROUTE or UNROUTE (same thing) |
| SOURCE_MATRIX | GAIN |
| SOURCE_SELECTOR | METER |
| MIXER | GROUP |
| AUTOMIXER | AUX |
| GAIN | OVERRIDE |
| N_GAIN | SOLO |
| INPUT_CARD | AUTO |
| OUTPUT_CARD | |
| METER | |

Ex: SUBSCRIBE(Blu_Room, MIXER,1,0,MUTE) – Will subscribe to input 1 Mute/UnMute

Ex: SUBSCRIBE(Blu_Room, MIXER,0,1,MUTE) – Will subscribe to master output Mute/UnMute

Ex: SUBSCRIBE(JukeBox, ROUTER,3,1, ROUTE) – Will subscribe to crosspoint input 3 output 1

To stop receiving updates:

Ex: UN_SUBSCRIBE(Blu_Room, MIXER,1,0,MUTE) – Will un_subscribe to input 1 Mute/UnMute

‘SET_TELEPHONE’ - controls state-variables listed in PARAM for a Telephone Hybrid object

SET_TELEPHONE (char OBJECT[6], integer INPUT, integer OUTPUT, integer PARAM, integer VALUE)

- OBJECT = “Friendly” name; Copy and Paste HiQnet address from properties window in London Architect
- INPUT = input channel; Used only for PARAMs dealing with speed dial and telephone number. See PARAM below for more details. Otherwise this will be 0 as well.
- OUPUT = output channel; this will always be 0 for this object.
- PARAM = “Friendly” name; Type of action to execute (ie. RX_GAIN, TX_GAIN, RX_MUTE, TX_MUTE)
- VALUE = ON/OFF/MUTE/UNMUTE (0 or 1) for discrete Mute SV’s. All GAIN’s are 0-100 (percent). For AUTO_ANSWER the values are from 0-10.

Possible Parameter Values:

PARAM

| | |
|-------------------------|---|
| BUTTON_0 | - The 0 button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| BUTTON_1 | - The 1 button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| BUTTON_2 | - The 2 button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| BUTTON_3 | - The 3 button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| BUTTON_4 | - The 4 button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| BUTTON_5 | - The 5 button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| BUTTON_6 | - The 6 button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| BUTTON_7 | - The 7 button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| BUTTON_8 | - The 8 button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| BUTTON_9 | - The 9 button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| PAUSE | -The pause (.) button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| CLEAR | - The clear button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| INTERNATIONAL | - The international (+) button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| BACKSPACE | - The backspace button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| REDIAL | - The redial button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| FLASH | - The flash button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| ASTERISK | - The asterisk (*) button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| POUND | - The pound (#) button for the telephone (Value = 0) (NO FEEDBACK DON'T SUBSCRIBE) |
| TX_MUTE | - Mute/Unmute's the TX (Value = 0 or 1) |
| RX_MUTE | - Mute/Unmute's the RX (Value = 0 or 1) |
| TX_GAIN | -Sets the TX fader to Value (Use VALUE of 0-100 (percent)) |
| RX_GAIN | -Sets the RX fader to Value (Use VALUE of 0-100 (percent)) |
| DTMF_GAIN | -Sets the DTMF fader to Value (Use VALUE of 0-100 (percent)) |
| DIAL_TONE_GAIN | -Sets the Dial tone fader to Value (Use VALUE of 0-100 (percent)) |
| RING_GAIN | -Sets the Ring fader to Value (Use VALUE of 0-100 (percent)) |
| SPEED_DIAL_STORE_SELECT | -Pushes one of the Store speed dial buttons (VALUE = 0) (INPUT = 1- # of STORE buttons) (NO FEEDBACK DON'T SUBSCRIBE) |
| SPEED_DIAL_SELECT | -Pushes one of the speed dial buttons (VALUE = 0) (INPUT = 1- # of Speed Dial buttons) (NO FEEDBACK DON'T SUBSCRIBE) |
| AUTO_ANSWER | -Sets the auto answer for the telephone (VALUE = 0-10) |
| DIAL_HANGUP | -The Dial/Hang-up button for the telephone (VALUE = 0) (Gives feedback – 1 if dialing and 0 if hung up) |

PARAM'S THAT SHOULD ONLY BE SUBSCRIBED TO

| | |
|------------------|---|
| TELEPHONE_NUMBER | -The number you're calling or going to call. Must use four feedback slots to receive number and must use DISPLAY_NUMBER function to convert the four feedback positions to a string that is the whole number. |
|------------------|---|

INCOMING_CALL

-Whether or not you're getting a incoming call or not. (0 = no call, 1= incoming call)

Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,BUTTON_0,0) – push 0 button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,BUTTON_1,0) – push 1 button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,BUTTON_2,0) – push 2 button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,BUTTON_3,0) – push 3 button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,BUTTON_4,0) – push 4 button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,BUTTON_5,0) – push 5 button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,BUTTON_6,0) – push 6 button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,BUTTON_7,0) – push 7 button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,BUTTON_8,0) – push 8 button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,BUTTON_9,0) – push 9 button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,ASTERISK,0) – push asterisk (*) button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,POUND,0) – push pound (#) button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,PAUSE,0) – push pause (,) button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,INTERNATIONAL,0) – push international (+) button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,CLEAR,0) – push clear button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,BACKSPACE,0) – push backspace button on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,TX_MUTE,1) – Mute's the TX on TELEPHONE_B
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,TX_MUTE,0) – unMute's the TX on TELEPHONE_B.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,TX_GAIN,70) – Set's the TX fader on TELEPHONE_B to 70%.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,DTMF_GAIN,20) – Set's the DTMF fader on TELEPHONE_B to 20%.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,DIAL_HANGUP,0) – push Dial/hang-up button on TELEPHONE_B (This will either dial or hang-up).
Ex: 'SET_TELEPHONE'(TELEPHONE_B,0,0,AUTO_ANSWER,3) – Set's auto answer on TELEPHONE_B to 3 rings.
Ex: 'SET_TELEPHONE'(TELEPHONE_B,1,0,SPEED_DIAL_STORE_SELECT,0) – push speed dial store button #1 on TELEPHONE_B (stores the current number in speed dial slot #1).
Ex: 'SET_TELEPHONE'(TELEPHONE_B,3,0,SPEED_DIAL_SELECT,0) – push speed dial button #3 on TELEPHONE_B (dials the number found in speed dial slot #3).

NOTE: Here is an example of dealing with TELEPHONE_NUMBER and INCOMING_CALL

SUBSCRIBING:

TELEPHONE_NUMBER:

FOR(i = 1;i<=4;i++) // FB[57] - FB[60] is used for the main telephone number

```
{  
    CALL 'SUBSCRIBE'(IO_CARD2,TELEPHONE,i,0,TELEPHONE_NUMBER)//Main number  
}
```

INCOMING_CALL:

CALL 'SUBSCRIBE'(IO_CARD2,TELEPHONE,0,0,INCOMING_CALL)//FB[63]

EX: getting the number from the four slots in the FB array to a single string.

If the main number was subscribed to the slots 57-60 in the FB array this is how you would call the function:

```
LOCAL_VAR CHAR[32] Main_Number;  
Main_Number = DISPLAY_NUMBER(FB[57],FB[58],FB[59],FB[60]);
```

If you would like to send a number like 1-800-555-5555 to the box without having to push the buttons individually you can use the SET_NUMBER call:

```
call 'SET_NUMBER'(IO_CARD2,'18005555555');// will set the number for the hybrid card in slot  
2 to 1-800-555-5555.
```

SECTION 3 – Receiving Commands from the Soundweb London

The AMX module is set up to receive and parse all incoming messages and compare these strings to the FB[][] array for true-feedback purposes. If pseudo feedback is desired, simply comment out the DATA_EVENT for the device. If true feedback is desired, follow the next two steps:

Step 1. Decide which state-variables (ie mutes, gains) you want to receive true feedback. Subscribe to these variables in the DATA_EVENT:ONLINE event. **IMPORTANT:** Make sure the SUBSCRIBE commands lie between the STARTUP = 1 and STARTUP = 0 flags. These flags will allow the subscribe commands to be logged into the FB[][] array. If a SUBSCRIBE or SUBSCRIBE% command is used anywhere else in the program it will only be used as a 'GET' statement and will not be used for true-feedback. The numerical order that you placed the SUBSCRIBE or SUBSCRIBE% commands in the DATA_EVENT: ONLINE section will determine the FB[] dimension for live feedback. *See example program for implementation.*

Ex:

```
DATA_EVENT: ONLINE
```

```
STARTUP = 1  
SUBSCRIBE(Blu_Room, MIXER,1,0,MUTE) // = FB[1][]  
SUBSCRIBE(Blu_Room, MIXER,0,1,MUTE) // = FB[2][]  
SUBSCRIBE%(Blu_Room, MIXER,0,1,GAIN)// = FB[3][]  
STARTUP = 0
```

Step 2. In DEFINE_PROGRAM, assign the feedback button/bargraph to the FB array. The [][][12] element will hold the value of a Route, UnRoute, Mute, UnMute, On, OFF (ie this element will be a 1 or 0) . If a gain is subscribed to then the FB[][][9-12] elements hold the 4 byte data value.

```
DEFINE_PROGRAM
```

```
[dvTP,1] = FB[1][12]; //assigns button 1 to the mute state of input 1 on Blu_Room mixer.  
[dvTP,3] = FB[2][12]; //assigns button 3 to the mute state of the master output on Blu_Room mixer
```

```
(* TO CONTROL A FADER WHEN USING AN ACTIVE FADER IN NETLINX*)  
(*IF USING SUBSCRIBE% <PERCENT> IN STARTUP AND NETLINX BARGRAPH HAS RANGE 0-  
100 IN TP PROPERTIES*)
```

```
LEVEL_EVENT (dvTP,2)  
(  
    'SET_GAIN%'(Blu_Room, MIXER,1,0, LEVEL.VALUE)
```

)

(* TO UPDATE VOLUME BAR GRAPHS WHEN USING A NON ACTIVE BAR GRAPH *)
(*IF USING SUBSCRIBE % IN STARTUP AND NETLINX BARGARH HAS RANGE 0-100 IN TP PROPERTIES*)

VOLUME = MID_STRING(FB[3],10,1); //taking element [10] <RANGE 0-100>
SEND_LEVEL dvTP, 2, VOLUME;

(* TO UPDATE VOLUME BAR GRAPHS WHEN USING A NON ACTIVE BAR GRAPH*)
(*IF USING SUBSCRIBE IN STARTUP AND NETLINX BARGRAPH HAS RANGE 0-255*)
VOLUME = MID_STRING(FB[3],9,4); //taking elements [9][10][11][12] <RANGE -300,000-100,000>
intVOL = CHAR_SLONGVAL(VOLUME); //turning chars into integers
MYVOL = scaleRange(intVOL,0,100,0,255); //SCALE PERCENT 0-100 TO 0-255 BAR GRAPH SCALE
SEND_LEVEL dvTP, 2, MYVOL;

NOTE: The FB array is made up of 12 characters; Node<2bytes> VD<1byte> Object<3bytes>
SV<2bytes> Value<4bytes>

Section 4: Adding Additional Functionality

The BSS Soundweb London gives the ability to control nearly everything through a 3rd party controller. These controls range from the standard functions of muting, unmuting, gains, etc. to controlling the front panel contrast, sleep brightness, active brightness, etc. This AMX module provides the control for the most common functions. If a programmer wants to add additional functionality to this module, then the programmer must use the full programming specification (see Section 1- Startup) and add the SV to the module.

Let's add a Graphic EQ subroutine. First, we need to declare a new subroutine.

'SET_GRAPHIC'(char OBJECT[6], integer BAND, slong DATA)
- OBJECT = "Friendly" name; Copy and Paste HiQ address from properties window in London
- BAND = # of band to be adjusted; range <1 to 30> by default.
- DATA = signed long holding value in dB; range <-10 to 10>

Ex:

DEFINE_CONSTANT
CHAR GRAPHIC_EQ[] = {\$11,\$8B,\$03,\$00,\$01,\$0C}; //GRAPHIC_EQ FRIENDLY NAME

DEFINE_CALL 'SET_GRAPHIC'(char OBJECT[6], integer BAND, slong DATA)
{
LOCAL_VAR INTEGER S_V; //state variable
LOCAL_VAR INTEGER S_VLOW;
LOCAL_VAR INTEGER S_VHIGH;
LOCAL_VAR INTEGER DEVICE;
LOCAL_VAR INTEGER iDATA[4]; //will receive 4 element int array converted from slong
LOCAL_VAR CHAR cDATA[4]; //will hold 4 elements of data in char form
LOCAL_VAR CHAR MY_STRING[13];
LOCAL_VAR CHAR EVENT;

EVENT = \$88; //for all discrete or scalar audio values

S_V = (BAND -1)+32; //formula from p. 87 full programming spec
S_VLOW = LO_BYTE(S_V); //LOW BYTE
S_VHIGH = HI_BYTE(S_V); //HIGH BYTE

```

CALL 'SLONG_BYTES'((DATA*10000),iDATA); //Turn slong into integer array; by_ref
cDATA = "iDATA[1], iDATA[2], iDATA[3], iDATA[4]"; //convert integer array into char array.

MY_STRING = "EVENT,OBJECT,S_VHIGH,S_VLOW, cDATA";
CALL 'CHECKSUM'(MY_STRING)

}

```

To use this new subroutine:

Ex:

```

CALL 'SET_GRAPHIC'(GRAPHIC_EQ,1,5) //set Band 1 to +5dB
CALL 'SET_GRAPHIC'(GRAPHIC_EQ,2,-5) //set Band 2 to -5dB
CALL 'SET_GRAPHIC'(GRAPHIC_EQ,3,7) //set Band 3 to +7dB

```

Section 5: Programmer Subroutine/Function Definitions

This section gives the definitions of all the subroutine/function definitions that are used to construct and format commands. These processes are ONLY called from within a user subroutine/functionS and do not need to be called directly from the program. Any modification of these definitions will result in incorrect commands being sent from within the program.

‘CHECKSUM’ – *Takes formed string and calculates checksum. The routine checks for, and replaces, all special characters. Finally it adds pre-limiter \$2, delimiter \$3, and sends string out comm port or TCP/IP Port 1023.*

```

DEFINE_CALL 'CHECKSUM'(char MY_STRING[13]) //CALCULATE AND SEND STRING
- MY_STRING[13] = EVENT<1 byte> NODE<2 bytes> VD<1 byte> OBJECT<3 bytes>
  SV<2 bytes> DATA<4 bytes>

```

‘SLONG_BYTES’ – *Takes an slong variable and returns a four element integer array using GAIN_A and passing by reference. Used to convert 32bit slong values to 4 position integer array in order to send out comm port.*

```

DEFINE_CALL 'SLONG_BYTES'(slong intGAIN, integer GAIN_A[4])
- intGAIN = variable holding value of gain (or other devices using 32bit slong parameter)
- GAIN = four element integer array used to return converted 32bit slong.

```

‘LONG_BYTES’ – *Takes a long variable and returns a four element integer array using GAIN_A and passing by reference. Used to convert 32bit long values to 4 position integer array in order to send out comm port.*

```

DEFINE_CALL 'LONG_BYTES'(long intGAIN, integer GAIN_A[4])
- intGAIN = variable holding value of gain (or other devices using 32bit long parameter)
- GAIN = four element integer array used to return converted 32bit long.

```

‘PROCESS_FEEDBACK’ – *Takes the parsed string from the DATA_EVENT and compares it to values stored in the FB[][] array for true feedback purposes.*

```

DEFINE_CALL 'PROCESS_FEEDBACK'(char RECEIVED_STRING[13])
- RECEIVED_STRING = string made up of EVENT<1 byte> NODE<2 bytes> VD<1 byte>
  OBJECT<3 bytes> SV<2 bytes> DATA<4 bytes>

```

‘GET_SV’ – *Using the data passed to from other functions/calls it calculates and returns the SV.*

DEFINE_FUNCTION integer GET_SV(integer DEVICE, integer INPUT, integer OUTPUT, integer PARAM) //GET STATE VARIABLE

- DEVICE = “Friendly” name; Used to tell functions what type of device is being controlled
- INPUT = input channel; 1 - # of inputs on device; a zero indicates output only control
- OUPUT = output channel; 1 -# of outputs on a device. A zero indicates input only control.
- PARAM = “Friendly” name; action to execute

‘IS_SPECIAL’ – *TAKES A SINGLE CHARACTER AND CHECKS TO SEE IF IT IS A RESERVED OR “SPECIAL” CHARACTER. IF IT IS, IT RETURNS A 1 OR ‘TRUE’.*

DEFINE_FUNCTION integer IS_SPECIAL(char SPECIAL_CHAR)

- SPECIAL_CHAR = character that needs to be checked to see if it is a special character.

‘HI_BYTE’ – *Takes a 16 bit integer and returns the upper 8bits (1byte).*

DEFINE_FUNCTION integer HI_BYTE(integer IBYTE)

- IBYTE = 16 bit integer value

‘LO_BYTE’ – *Takes a 16 bit integer and returns the lower 8bits (1byte).*

DEFINE_FUNCTION integer LO_BYTE(integer IBYTE)

- IBYTE = 16 bit integer value

‘CHAR_LONGVAL’ – *Takes a four element character array and returns a long. Used to take the 4 bytes of the data field on an incoming string and convert it to a long in order to perform mathematical conversions.*

DEFINE_FUNCTION long CHAR_LONGVAL(char VALUE[4])

- VALUE – the four bytes of the command string holding a particular value such as a gain.

‘FOUR_BITS_TOSTRING’ – *takes lower 4 bits and converts them into a char that represents a number or symbol on a telephone. See function for the conversion table. Used by the DISPLAY_NUMBER function to display the phone number off a telephone hybrid card.*

DEFINE_FUNCTION CHAR[1] FOUR_BITS_TOSTRING(LONG convert)

- convert – the four bits that need to be converted into a char.

‘DISPLAY_NUMBER’ – *Takes the four feedback arrays that represent the phone number of the telephone hybrid card and returns a string representing that number. Example of how to use is shown above under SET_TELEPHONE section 2.*

DEFINE_FUNCTION CHAR[32] DISPLAY_NUMBER(CHAR first[12],char second[12],char third[12],char fourth[12])

- it doesn’t matter in what order you give the feedback arrays that represent the phone number into this function, because it will put them in order for you.
- first – one of the four feedback arrays that are part of the number
- second – one of the four feedback arrays that are part of the number

- third – one of the four feedback arrays that are part of the number
- fourth – one of the four feedback arrays that are part of the number

SE 6/15/06

SE: updated 1/28/08

JJ: updated 7/29/10