iMS4 Independent Channel control.

There are changes in the API/SDK to reflect the new features of the Rev-C synthesizer.
This includes independent power control of the four channels.

1:For rev-C , Amplitude Control should be set to ::INDEPENDENT

enum iMS::SignalPath::AmplitudeControl

Selects Amplitude Control source for each of the 4 RF Channel outputs.

The RF signal outputs from the Synthesiser feature channel bandwidth filtering and an RF mixer with a selectable control source. The mixer input can be routed to signal modulation.

Enumerator	
OFF	Turn RF outputs off.
EXTERNAL	Route RF mixer inputs to analogue modulation external signals.
WIPER_1	Connect RF mixer inputs to digital pot wiper 1.
WIPER_2	Connect RF mixer inputs to digital pot wiper 2.
INDEPENDENT	RF mixer inputs are controlled by digital pot. Use this on hardware that supports independent control per channel.

2: The Amplitude for <u>each</u> channel needs to be updated

For X/Y AO deflector applications:
Channel 1 value = Channel 2 value
Channel 3 value = Channel 4 value

```
bool iMS::SignalPath::UpdateRFAmplitude ( const AmplitudeControl  src,
                                          const Percent &         ampl,
                                          const RFChannel &       chan = RFChannel::all
                                        )
```

Scales the Digital Potentiometer mixer drive level up & down.

The 2 digital potentiometers on the Synthesiser can be selected to apply a DC drive level to the IF input of a wideband RF mixer in the output channel signal conditioning, thereby acting as an amplitude control voltage.

This function sets the drive level of the 2 digital potentiometers. The AmplitudeControl input determines which potentiometer is updated, if it is set to anything other than WIPER_1 or WIPER_2, the request is ignored and the function returns false.

Parameters

[in] **src** Which of the two digital potentiometers to update
[in] **ampl** the percentage of maximum amplitude scaling to update the potentiometer to
[in] **chan** On Synthesisers that support independent channel amplitude control, this can be used to select which channel to apply the change to

Returns

true if the amplitude update request was sent successfully

Since

1.0

3: LUT compensation files can be generated for each axis independently
For X/Y AO deflector applications:
Channel 1 LUT = Channel 2 LUT
Channel 3 LUT = Channel 4 LUT

enum iMS::SignalPath::CompensationScope

Chooses the applicable scope for a Compenation Table: globally or per channel.

A CompensationTable downloaded to a Synthesiser can be applied to FAP data in one of two methods. With Global scope, there is a single table applied universally to all channels. With Channel scope, multiple tables are downloaded to the Synthesiser, one for each RF Channel.

Note that the size of storage space on the Synthesiser remains constant in both scopes so the resolution of channel scope compared to global scope is the reciprocal of the number of RF channels.

Since

1.6

Enumerator	
GLOBAL	Single CompensationTable applied universally to all channels.
CHANNEL	Multiple CompensationTable's applied per channel.

An example code snippet to illustrate:

```
SignalPath sp(*myiMS);

// Import a channel-scoped LUT here
CompensationTableImporter cti("343nm100-140M-revC.LUT");
if (!cti.IsValid()) {
    std::cout << "LUT File is Not Valid!" << std::endl;
    while (!_kbhit());
    return 0;
}
else if (cti.IsGlobal()) {
    std::cout << "This example needs a Channel-Scoped LUT.  Yours is Global" << std::endl;
    while (!_kbhit());
    return 0;
}

// Retrieve channel LUTs from file and download them to iMS
CompensationTable channel_lut;
for (int i = RFChannel::min; i <= cti.Channels(); i++) {
    std::cout << "Downloading Channel " << i << " LUT" << std::endl;

    // Gets table for channel 'i' from file
    channel_lut = cti.RetrieveChannelLUT(RFChannel(i));

    // Download to iMS
    if (!WaitForCompensationDownload(*myiMS, RFChannel(i), channel_lut)) {
        std::cout << "Aborting.. <press any key>" << std::endl;
        while (!_kbhit());
        return 0;
    }
}

// Enable compensation
sp.EnableImagePathCompensation(SignalPath::Compensation::ACTIVE, SignalPath::Compensation::ACTIVE);

// Double check signal path is in channel scope
if (sp.QueryCompensationScope() == SignalPath::CompensationScope::CHANNEL) {
    std::cout << "Synthesiser set for CHANNEL Compensation Scope" << std::endl;
}
else {
    std::cout << "Synthesiser set for GLOBAL Compensation Scope" << std::endl;
}

// Set RF Amplitude to use internal level control (not external modulation)
sp.SwitchRFAmplitudeControlSource(SignalPath::AmplitudeControl::INDEPENDENT, RFChannel::all);

// Update per-channel RF Amplitude levels
sp.UpdateRFAmplitude(SignalPath::AmplitudeControl::INDEPENDENT, Percent(30), RFChannel(1));
sp.UpdateRFAmplitude(SignalPath::AmplitudeControl::INDEPENDENT, Percent(30), RFChannel(2));
sp.UpdateRFAmplitude(SignalPath::AmplitudeControl::INDEPENDENT, Percent(30), RFChannel(3));
sp.UpdateRFAmplitude(SignalPath::AmplitudeControl::INDEPENDENT, Percent(30), RFChannel(4));
```

4: The remaining RF power control function, UpdateDDSPowerLevel is the same for both rev-B and rev-C e.g. **`sp.UpdateDDSPowerLevel(80);`**

bool iMS::SignalPath::UpdateDDSPowerLevel (const Percent & power)

Scales the DDS device (Direct Digital Synthesis RF signal generator) power up & down.

The RF signal generator device on the Synthesiser converts frequency, amplitude and phase data into the 4 RF signals that drive the output of the Synthesiser. The device can be configured to (maximum power)

Parameters

[in] **power** the percentage of maximum power at which the DDS should drive RF signals into the output signal conditioning

Returns

true if the power update request was sent successfully

Since

1.0