
Controlling and Monitoring DSP Conductor Configurations

1. Introduction

DSP Conductor™ is a development environment that allows you to graphically define audio DSP algorithms, generate DSP code, and adjust the parameters of the running code.

Although DSP Conductor is intended for use by product engineers rather than end users, it is often desirable to allow end users to control a subset of DSP parameters. This control may be accomplished by a local microcontroller attached to the host port of the DSP or by a remote application communicating to the DSP via the IP-based simple network management protocol (SNMP). All parameters in a stand-alone DSP configuration may be adjusted by either of these methods. The DSP Conductor for CobraNet™ application itself uses SNMP to control and monitor configurations.

This application note describes how to identify and interface with the parameters you designate as end-user controllable.

2. Generating Deliverables

Once you have built, adjusted, and are satisfied with a DSP configuration you may use the *Generate deliverables...* item in the *Tools* menu to generate reference files for the configuration. Clicking *Generate deliverables...* will bring up a directory browser so that you may designate a destination directory for the files. The *Generate deliverables...* function will then create data files in this directory, including a header file with the same root name as the configuration and an *.h* extension. This is the *configuration header file*.

3. Interpreting the Configuration Header File

For each DSP parameter in the configuration, four header definitions are generated. All four are of the following format:

```
#define dspc_<primitive name>_<primitive ruid>_<parameter name>_<metric> <value>
```

<primitive name> - Name given to the primitive in the XML files that define the primitive. A single and unique name is given to each primitive in the device library. Refer to the application note AN277, “*Creating DSP Conductor Primitives*” for detailed descriptions of the XML files. The primitive name is typically a more technical description of the primitive than the name shown on the corresponding processing block in the GUI. Examples include *generator_sine_wave* and *mixer_NxM*.

`<primitive ruid>` - The primitive's relatively unique identifier (RUID) is a number that uniquely identifies an instance of a primitive within a configuration. The primitive RUID is required to differentiate multiple instances of the same primitive in a configuration.

`<parameter name>` - Name given to the parameter in the XML files that define the primitive. Primitives typically have numerous parameters, each uniquely named. Refer to the application note AN277, “*Creating DSP Conductor Primitives*” for detailed descriptions of the XML files. Example parameter names include *c1*, *hold_count*, and *send*. For multi-channel devices such as mixers, the parameter name may also be constructed to include a channel number in the form `<parameter name>_<channel number>`.

A different `<metric>` code is used for each of the four entries associated with a DSP parameter. The four metric codes, *wo*, *rw*, *fb* and *sg* are described below.

wo - Word Offset

Word offset of the parameter. The value is used to generate HMI addresses and SNMP OIDs. See [Section 6. Forming HMI Addresses](#) and [Section 5. Forming SNMP OIDs](#) sections below for a complete description of the use of word offsets.

rw - Read/Write

Access rights for the parameter: 0 - read-only, 1 - read/write. Access rights come into play when generating HMI addresses and SNMP OIDs as there are separate regions assigned to read-only and read/write parameters.

fb - Fraction Bits

Number of fractional bits in the parameter. The DSP targeted by DSP Conductor is a 32-bit, fixed-point processor. All DSP Conductor parameters are 32 bits. The value of this definition indicates the fixed position of the decimal point for the parameter: 0 - the parameter is an integer with full 32-bit range, 31 - the parameter is a fraction with its absolute value less than or equal to one.

sg - Sign

Indicates whether the parameter carries a signed or unsigned value: 0 - unsigned, 1 - signed.

4. Parameter Values

All DSP Conductor parameters are 32-bit values. While the sign (*sg*) and fraction bits (*fb*) attributes give some indication of how the DSP might interpret the values, in the end, it is the specifics of the DSP implementation that determine exactly how values are interpreted. The way values are interpreted can be discerned by examining the *Crunch Functions* section in the primitive's implementation XML file. For details on the implementation XML file refer to Cirrus application note AN277, “*Creating DSP Conductor Primitives*”.

5. Forming SNMP OIDs

All SNMP variables are referenced by the object identifier (OID). An OID is a dot-separated string of integers enumerating the path from the root of the SNMP management information base (MIB) to the variable. Firmware supporting DSP Conductor features a *dspExtensions* SNMP extension agent rooted at *iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).peakAudio(2680).cobraNet(1).dspExtensions(4)*.

DSP Conductor parameters under this extension are separated into two tables of 32-bit integers. The first table, rooted at *dspExtensions(4).control(2).controlRWTable(2)*, holds read-only parameters. The second table, rooted at *dspExtensions(4).control(2).controlROTable(4)* holds read/write parameters. Note that access to SNMP tables is achieved using 1-based indices. The word offsets specified in the configuration header file are 0 based.

To construct an OID for a read/write variable, append the parameter's word offset (wo) plus one to the OID for the *controlRWTable* values:

```
iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).peakAudio(2680).cobraNet(1).dspExtensions(4).control(2).controlRWTable(2).controlRWEntry(1).controlRWValue(2).
```

For example, the OID for a read/write parameter with word offset 5 is 1.3.6.1.4.1.2680.1.4.2.2.1.2.6

To construct an OID for a read-only variable, append the parameter's word offset (wo) plus one to the OID for the *controlROTable* values:

```
iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).peakAudio(2680).cobraNet(1).dspExtensions(4).control(2).controlROTable(4).controlROEntry(1).controlROValue(2).
```

For example, the OID for a read-only parameter with word offset 13 is 1.3.6.1.4.1.2680.1.4.2.4.1.2.14

5.1 1.0.0 Beta OIDs

The 1.0.0 beta version of DSP Conductor uses an early revision of the *dspExtensions* MIB with slightly different OIDs as follows:

Read/write base OID:

```
iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).peakAudio(2680).cobraNet(1).dspExtensions(4).controlRWTable(3).controlRWEntry(1).controlRWValue(2)
```

Read-only base OID:

```
iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).peakAudio(2680).cobraNet(1).dspExtensions(4).controlROTable(5).controlROEntry(1).controlROValue(2)
```

6. Forming HMI Addresses

Parameters accessed through HMI are located in two separate address blocks. Access to the parameters is controlled by access control rights.

Read/write parameters are based at HMI address 0x76000. To generate an HMI address for a specific read/write parameter, add 0x76000 to the word offset (wo) definition for the parameter.

Read-only parameters are based at HMI address 0x7A000. To generate an HMI address for a specific read-only parameter, add 0x7A000 to the word offset (wo) definition for the parameter.

Contacting Cirrus Logic Support

For all product questions and inquiries contact a Cirrus Logic Sales Representative.

To find the one nearest to you go to www.cirrus.com

IMPORTANT NOTICE

Cirrus Logic, Inc. and its subsidiaries ("Cirrus") believe that the information contained in this document is accurate and reliable. However, the information is subject to change without notice and is provided "AS IS" without warranty of any kind (express or implied). Customers are advised to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, indemnification, and limitation of liability. No responsibility is assumed by Cirrus for the use of this information, including use of this information as the basis for manufacture or sale of any items, or for infringement of patents or other rights of third parties. This document is the property of Cirrus and by furnishing this information, Cirrus grants no license, express or implied under any patents, mask work rights, copyrights, trademarks, trade secrets or other intellectual property rights. Cirrus owns the copyrights associated with the information contained herein and gives consent for copies to be made of the information only for use within your organization with respect to Cirrus integrated circuits or other products of Cirrus. This consent does not extend to other copying such as copying for general distribution, advertising or promotional purposes, or for creating any work for resale.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). CIRRUS PRODUCTS ARE NOT DESIGNED, AUTHORIZED OR WARRANTED FOR USE IN AIRCRAFT SYSTEMS, MILITARY APPLICATIONS, PRODUCTS SURGICALLY IMPLANTED INTO THE BODY, AUTOMOTIVE SAFETY OR SECURITY DEVICES, LIFE SUPPORT PRODUCTS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF CIRRUS PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK AND CIRRUS DISCLAIMS AND MAKES NO WARRANTY, EXPRESS, STATUTORY OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE, WITH REGARD TO ANY CIRRUS PRODUCT THAT IS USED IN SUCH A MANNER. IF THE CUSTOMER OR CUSTOMER'S CUSTOMER USES OR PERMITS THE USE OF CIRRUS PRODUCTS IN CRITICAL APPLICATIONS, CUSTOMER AGREES, BY SUCH USE, TO FULLY INDEMNIFY CIRRUS, ITS OFFICERS, DIRECTORS, EMPLOYEES, DISTRIBUTORS AND OTHER AGENTS FROM ANY AND ALL LIABILITY, INCLUDING ATTORNEYS' FEES AND COSTS, THAT MAY RESULT FROM OR ARISE IN CONNECTION WITH THESE USES.

Cirrus Logic, Cirrus, the Cirrus Logic logo designs, DSP Conductor, and CobraNet are trademarks of Cirrus Logic, Inc. All other brand and product names in this document may be trademarks or service marks of their respective owners.