Technical Notes on using Analog Devices' DSP components and development tools
Phone: (800) ANALOG-D, FAX: (781) 461-3010, EMAIL: dsp.support@analog.com, FTP: ftp.analog.com, WEB: www.analog.com/dsp

# A Quick Primer on ELF and DWARF

*Contributed by Ken Butler,*
*Assemblers and Simulator Products Manager*

## Introduction

The 4.1 VisualDSP® SHARC® and all newer VisualDSP tools for other ADI DSPs will generate ELF/DWARF-2 object files and executables. This application note will help explain what this means for users, especially users that have been using the 3.3 and 4.0 VisualDSP (SHARC) tools. It will also explain why the change has been made.

## ELF Object File Format

ELF (Executable and Linking Format) is an object file format originally developed by Unix System Laboratories. It was selected by the Tools Interface Standards committee as the basis for a portable object file format that works on a variety of operating systems. Refer to the standard "Executable and Linkable Format (ELF), Tools Interface Standards (TIS), Portable Formats Specification, Version 1.1."

In the Analog Devices, Inc. (ADI) tools all linkable objects (.doj) and all executable images (.dxe) are stored as ELF files.

ADI started using ELF in the 4.0 VisualDSP tools for SHARC. Presently, the only ADI software development tools that do not use ELF are the ADSP-218x tools that generate AEXE; this will change with the 7.0 release of the ADSP-218x tools.

## DWARF debug information format

DWARF is a format for the information generated by compilers, assemblers and linkers that are necessary for symbolic source-level debugging. It is a debugging information format that does not favor the design of any compiler or debugger. It was also designed to meet the debugging needs of different languages in a unified manner. ADI uses version 2 of the DWARF standard, sometimes referred to as DWARF-2. Refer to the standard, "DWARF Debugging Information Format, Tools Interface Standards (TIS), Portable Formats Specification, Version 1.1".

In ADI tools, the debugging information produced by the compiler and assembler is in DWARF-2 format. The VisualDSP debugger reads DWARF-2 debug information.

The software development tools for the ADSP-21160 (now in beta) and the ADSP-TS001 (now in beta) are complete ELF/DWARF-2 toolsets. The tools for the SHARC will be ELF/DWARF-2 starting with the 4.1 release. The ADSP-218x and ADSP-219x tools will be ELF/DWARF-2 starting with the 7.0 release. All toolsets for new processors will be ELF/DWARF-2.

## Previous formats

The 3.3 SHARC tools produced COFF object files and executables. The debugging information was in SDB. (A good reference for COFF and SDB is "Understanding and Using COFF" by Gintaras R. Gircys", O'reilly & Associates). The 4.0 VisualDSP SHARC tools used ELF object files but maintained the SDB format for debug information. The 4.1 SHARC tools will be an ELF/DWARF toolset. (The table below shows object and debug formats for ADI tools).

The ADSP-218x tools use AEXE as an object file format. The 7.0 release of theADSP- 218x tools will be an ELF/DWARF toolset.

**The following table shows object and debug formats for ADI tools:**

| **Toolset** | **3.3 SHARC** | **4.0 SHARC** | **4.1 SHARC** | **6.1 218x** | **7.0 218x** | **21160** | **TS001** |
|---|---|---|---|---|---|---|---|
| Object Format | COFF | **ELF** | **ELF** | AEXE | **ELF** | **ELF** | **ELF** |
| Debug Format | SDB | SDB | **DWARF** | NONE | **DWARF** | **DWARF** | **DWARF** |
| Object file extension | .obj | .doj | **.doj** | .obj | **.doj** | **.doj** | **.doj** |
| Executable extension | .exe | .dxe | **.dxe** | .exe | **.dxe** | **.dxe** | **.dxe** |

## Object File Implications

If you are starting a new program the object file format and debug information format shouldn't affect how you program at all. There are the general advantages of using industry standard file formats as described towards the end of this note.

If you have existing code targeted for a non-ADI part, the object file format shouldn't affect you either. You will need to recompile your C code, translate and assemble any assembly code, and then proceed.

If you are migrating from previous versions of ADI tools, you may notice a difference. The ELF/DWARF object format is richer in information than the previous object formats. ADI will provide object file converters (e.g., coff2elf that converts from SHARC 3.3 COFF format to ELF, or aexe2elf that converts from 218x AEXE format to ELF) that will convert existing object files and executables to the ELF/DWARF format.

ADI recommends using the latest tools to rebuild from source. But in many cases users will have libraries for which they may not have source code. Using the object file converters

When the SHARC 4.0 tools were released, users were able to convert COFF object files to ELF. In addition, the VisualDSP debugger in the 4.0 release was able to read and debug COFF object files directly (without conversion). Users who had 3.3 generated files that contained debugging information would find that the object file converter would preserve the debugging information. This behavior is unique to the 4.0 SHARC tools (and the 4.0.1 service pack). In general, most object file converters will not preserve debug information.

## A SHARC Example

If you had an object library of math routines that were in COFF format, you would be able to run the coff2elf conversion tool over the library to produce an object library of math routines that were in ELF format. You could then use the ELF/DWARF tools for the rest of your development. You would be able to use the math routines as before.

## Converting an Existing Object File with Debug Information

If you have an existing object file that already contains debug information you have two choices. If you have the source you can recompile to produce a file in the ELF/DWARF format. This file would have debugging information that will be more complete than the original file.

If you don't have source you can convert the file to ELF/DWARF and continue development. You would be able to link the file in and it will execute correctly. In the debugger you would only have symbol table information, you would not have source level debugging information -- the latter information is what is lost in the conversion from COFF to ELF/DWARF. (On the other hand, your ability to do source level debugging is already limited since you don't have source).

**Advantages of ELF Objects**

The ADI tools changed the object file format in order to better support our DSP architectures. On the SHARC, the COFF file required extensions to support the advanced linker support for overlay linking and multiprocessor linking. For the 218x tools, the object file support does not provide any means of carrying debugging information with the object file.

COFF can be extended (normally a vendor will refer to their version of ECOFF -- Extended COFF). However, every time you extend COFF or make a variation on the extension you often need to "retool" all of the existing tools that operate on the ECOFF. In contrast, ELF has been designed so it can be extended without affecting the existing tools that produce or consume the ELF object file format.

COFF also has some other artificial limitations including a limit on section (segment) name lengths, object file size (COFF is a 16-bit object file format); and ELF is portable across different hosts.

An ELF file produced on a Sun/Solaris machine can be simply transferred to a PC and read by tools on that machine without any conversion. In order to move this file in COFF, the user would have to run a utility, CSWAP, that would fix up the file to account for the different byte-ordering between the Sparc and the PC.

By selecting a generalized standard object file format ADI is able to use the same object file format for all of the toolsets. This commonality reduces the development time needed for building any piece of the ELF/DWARF toolchain. In addition, the tools are generally more robust as maintenance and enhancements will be shared by all tools.

**Advantages of DWARF Debug Information**

It's important to keep in mind that debug formats and object files are independent. The 4.0 SHARC tools did support the SDB format that was used by the 3.3 tools. (This is why it was possible to debug 3.3 objects using the 4.0 tools). However there are good reasons to change to a new debug format. The existing format would have to change to support debugging of overlays, and this was put into the DWARF format. There is room in DWARF for future expansion of debugging capability for features like advanced language support (e.g., C++) or debugging of optimized code.

Users will also notice more complete and robust debugging information through using DWARF-2. There were some aspects of C that weren't adequately represented in the previous format that are using DWARF-2.

For the ADSP-218x DSPs, adding a true debug format to the ADSP-218x object file format will greatly improve the capabilities of any 218x debugger. The AEXE format does not include any kind of debugging information in the object file. The debugger was able to reconstruct a lot of the information from processing sources files and compiler output, but this process was extremely error-prone, and was guaranteed to be incorrect if a source file had changed since a file was compiled.

## Conclusion

The use of ELF and DWARF-2 for Analog Devices DSP development tools results from user demands for more and better tool set features. Use of these formats results in a more robust development development environment today, and will help tool developers enhance the tool sets in upcoming releases.