1. Introduction

The Application Resource Kit™ (ARK) is a set of software building blocks that works behind the scenes to make it easier to develop fitting software. “Behind the scenes” means that it is not part of the user interface – audiologists and dispensers will not be aware of it.

ARK is intended to significantly shorten the software development time required to add new hearing aids to a fitting module or standalone fitting software. It simplifies or eliminates the repetitive tasks of programming different types of controller chips and measuring hearing aid performance curves. This allows more time for the fitting software development team to focus on innovative user interfaces and fitting methodologies.

ARK-based applications can be developed in virtually any language that supports 64-bit operating systems and Windows® 7 (Windows® 95, 98, ME, 2000, XP or Windows NT® 4.0). The set of freely available ARK components and sample applications with source code, called ARKbase and ARKsdk, is available now on the ARK website http://ark.onsemi.com/index.php. A detailed tutorial and reference guide for programmers is also available at the same location.

Architecture

Figure 1 shows the components that make up the ARK framework and the applications that might use it. ARK controller components allow an application to communicate with the DSP using a programming box such as the HiPro or DSP Programmer from ON Semiconductor. ARK product components contain the product-specific details such as the mapping between product and controller parameters, and the electrical and acoustic models that represent the product’s characteristics. The ARK Core Component defines the interfaces between the application and the DSP and product components, and contains functionality that is requires by every component.

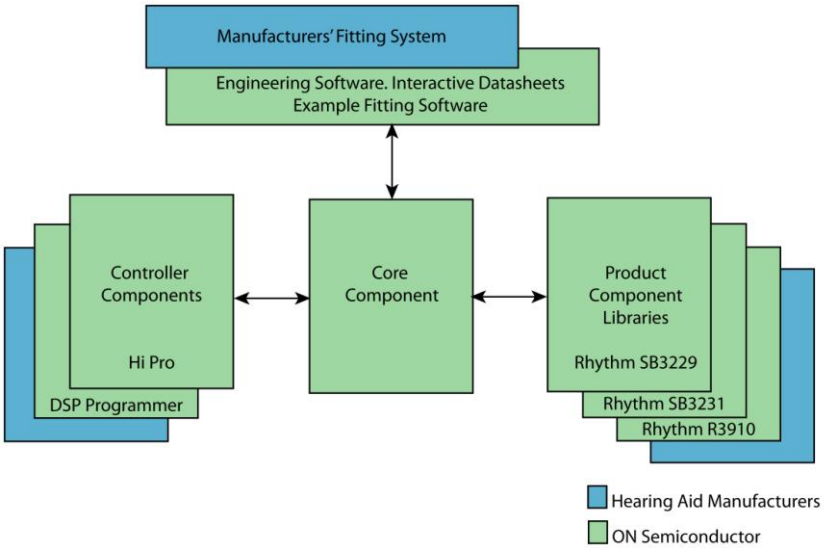

Figure 1. An overview of the ARK components and applications

The ARK framework uses Microsoft's Component Object Model (COM) technology, also known as ActiveX®. This allows ARK to tightly integrate with 32-bit development environments such as Microsoft Visual Basic®, Borland® Delphi and Microsoft Visual Studio®. COM also simplifies component packaging and distribution. Each component is a separate "plug-in" DLL that can be upgraded independently, maintain backwards compatibility. Controller and product components can also be installed and removed at any time – whenever an ARK-based application or fitting module starts up the ARK Core Component queries the system to determine which components have been installed.

NOAH®

NOAH is a commonly used software framework in the hearing industry. As shown in Figure 2, hearing aid manufacturers and independent software vendors develop three types of modules that plug into the NOAH framework: fitting modules, measurement modules and office management modules. NOAH also provides a shell from which the user launches these modules. As the diagram indicates, ARK can be used to develop NOAH-compatible fitting modules.

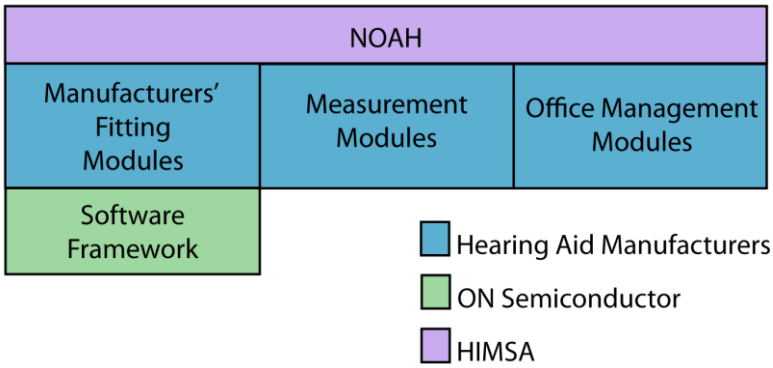

Figure 2. Software in the hearing industry

2. The ARK Core Component

Before discussing the ARK Core Component, some terms need to be defined. In ARK, a *client* is an application that uses ARK components. An ARK component is also known as a *server*. These terms are more commonly used when the *servers* and *clients* reside on separate machines, such as a web server and a web browser (client). They are also applicable when the *server* is a COM DLL and the *client* is the application that uses the DLL.

The Core Component has three major functions. First it defines the interfaces (APIs) by which all of the *servers* must abide. Think of an interface as a contract between the *client* and *server*, where the *server* promises to implement standard functionality. The actual implementation details of a *server* are hidden, minimizing the amount of "knowledge" the *client* has to have about the *server's* internal workings. This allows multiple *servers* to have dramatically different behaviour, yet be accessed with the same interface. This means that client code can be written once to be compatible with any ARK component. A design goal of the ARK framework is to keep the interfaces as simple as possible while offering the required flexibility and functionality.

The second function of the Core Component is to perform many of the routine tasks that the controller and product components would normally perform. There are certain types of objects and collections of objects that are heavily reused within ARK components. Component developers do not have to replicate them in their own code.

The third function of the Core Component is to determine which components are installed on a system and build lists of product and controller components that the *client* can use.

3. Controller Components

Controller components contain the knowledge necessary to communicate with hearing aid controllers through a programming box such as HiPro. Typically, there is one component per programming box, with each component supporting multiple controllers, but this is determined by the component developer. The only requirement is that a controller component implements the standard API that defines the set of operations: Open, Close, WhichChip, Init, Timing, Read, Write and Burn.

ARKbase includes a HiPro component that work with Foundation, Consolidator, Voyageur and Wolverine based pre-configured products. The Open and Close operations affect the programming box and are controller-independent. The WhichChip operation detects or sets which controller is connected and decides which code should be executed for the remaining operations (Init, Timing, Read, Write and Burn). All of this functionality is constrained in a single DLL. Support for additional programming boxes can be added through additional DLLs. Support for additional controllers can be added to the components provided by ON Semiconductor or in third-party ARK controller components.

On the client side, an application can query the core component for the list of available programming boxes and then displays the list in a list box so the user can select which box to use. The software then calls the controller operations through the standard API, regardless of which programming box was selected.

4. Product Components

In the same way that controller components have a common interface for every programming box and controller, product components have one interface for every product. “Products” can be hybrids that input and output electrical signals, or hearing aids complete with shells, tubing and transducers. The product API covers two main areas: parameter mapping and generating data for characteristic curves.

Parameter Mapping

Fitting software typically has a number of controls available for each product that allows the user to adjust the products parameters. The parameter mapping feature maps controller parameters (e.g R1, R2, R3) onto product parameters such as gain, threshold and compression

ratio. In some cases, the product parameters map directly onto single controller parameters, while in others, multiple controller parameters may be combined into one product parameter. Parameter mapping is essential as the product may have numerous controller parameters.

Characteristic Curves

A product component helps in plotting characteristic curves by calculating the output for given input levels, frequencies and product parameter settings. For supported products, ON Semiconductor provides compiled code that simulates the electrical part of the hearing aid system. As illustrated in Figure 3, the manufacturer can then combine the electrical model with input and output acoustic models to produce the response of the entire hearing aid.

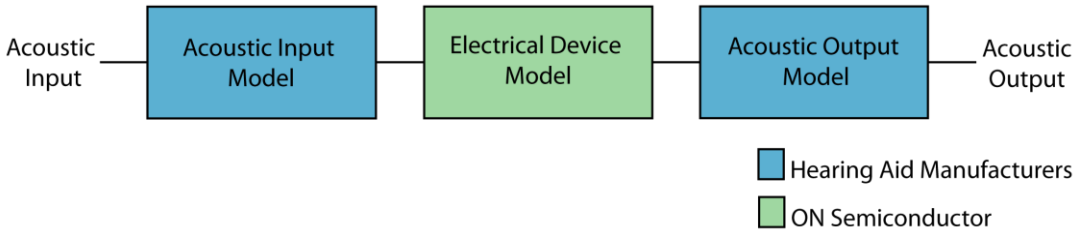

Figure 3. Combining acoustic and electrical models to produce characteristic curves

As a first approximation, the hearing aid manufacturer will be able to combine an electrical device model together with microphone and receiver models (which would correspond to the acoustic input model and acoustic output model, respectively). The manufacturer could then take several measurements of actual assembled hearing aids to refine the acoustic models. A key goal of ARK is to assist manufacturers in the rapid development of new products by not having to measure acoustic curves for every parameter combination.

Libraries and Future Development

Libraries allow manufacturers to combine multiple product components into a single DLL for easy distribution. Manufacturers may wish to combine products into libraries by product line, or combine their entire product family into a single library. For example, ARKbase includes Demo Libraries for pre-configured products such as Rhythm SB3229. These libraries contain many different products which include ones with electrical models (these can be used to electrically verify the performance of the device) and sample acoustic models.

These products also include samples of some possible device customizations that can be done, such as setting the sampling rate (16 kHz or 32 kHz), enabling generic biquads and setting up the functionality of the MMI (Man/Machine Interface).

ON Semiconductor has developed a library management tool, ARKonline that assists developers in combining compiled electrical models, transducer models, parameter maps and fitting support algorithms for multiple products into a library.

ON Semiconductor also offers a sample auto-fit algorithm that can be integrated into ARK product components.

5. Applications

To help component (server) developers and application (client) developers, ON Semiconductor has developed an application called Interactive Data Sheet (IDS) (see Figure 5) which includes the controller functionality and also displays the frequency and I/O response for a selected product at the current parameter settings. Source code is provided for both applications to give the developer a working example as a good starting point.

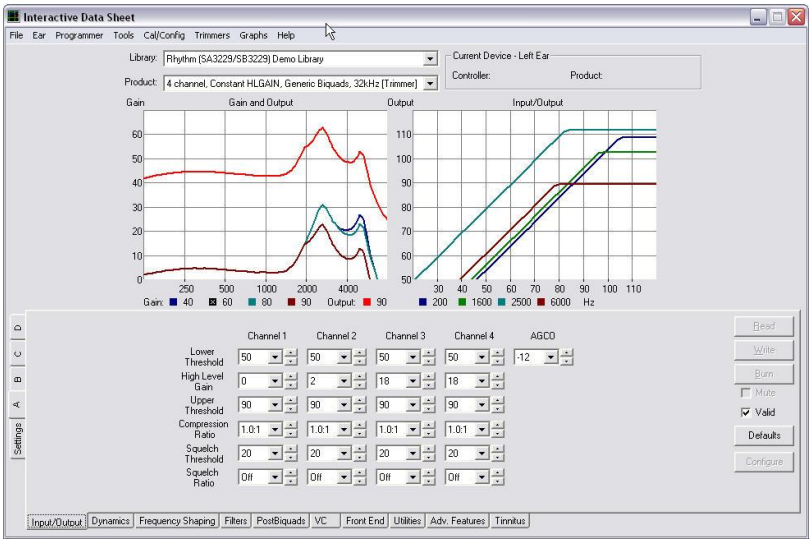

Figure 4. Interactive Data Sheet screensho*t*

Download ARK here.