



ASR1600/C Speech application development guide for Sega's DreamCast platform

1. Introduction

Although human speech varies from person to person and people's voices differ for males, females and children and different accents exist for languages, L&H provides a speaker independent speech recognition technology that requires no enrollment or training for different speakers. This means that the L&H ASR can be used to create applications that use continuous speech where the user is unknown.

The L&H ASR continuous speech recognition engines are language independent speech recognizers that will operate with all languages that come available from Lernout & Hauspie Speech Products. L&H provides with the recognition engines speaker-independent models for a specific language. They can be used by any *user* and for any *vocabulary* that is defined by your application.

The ASR1600 recognition engine is the latest member in L&H family of continuous speech recognition engines. The ASR1600 is the natural successor of the ASR1500.

Continuous speech recognition systems recognize phrases of continuously spoken words. On the other hand, isolated word recognition systems require small pauses between the words. The L&H recognition engines provide continuous speech recognition capabilities that allow the users to speak in a natural and fluent way.

Isolated word recognition, key word spotting and connected digit recognition are provided by L&H as special cases to continuous speech recognition and require no special interface.

State of the art, real-time, continuous speech recognition systems today require the speech recognition process to be guided by a *context*, e.g., some knowledge of the phrases that are expected. Contexts are application dependent and have to be built by the application designer. The L&H ASR SDK provides the application developer with *context development* facilities that allow the design of arbitrary contexts. Arbitrary contexts are built using the L&H *BNF grammar programming language* and a *grammar compiler*.

L&H contexts are *hierarchical*. This means that contexts can be used from within other contexts. This feature enables you to build up a library of continuous speech recognition contexts that can be used consistently. For example a context could be designed to describe "date" speech input, all higher level grammars that need to incorporate dates could use this.

Following special purpose contexts are provided:

- Isolated Word-Recognition: a special context allowing isolated word recognition.
- Key-Word-Spotting: a special context allowing spotting one keyword in a phrase.
- Connected Digits: a special context to recognize digit strings.

The L&H Lexicon Toolkit (LexTool) supports multiple *dictionaries* for each language that contain words and their phonetic transcriptions. The *standard dictionary* is implemented as an *expert system* that translates words into their phonetic transcriptions for the specific language. In this way, L&H provides you with a dictionary that contains **all** words of the language.

However, some words that originate from other languages, proper nouns, words that belong to a specialized field of interest, etc. can yield sub-optimal or even wrong results when using these expert systems. To correct for such cases, L&H provides *exception dictionaries* with the proper interface functions to handle specific exception words and their phonetic transcription.

In this version *user word capabilities* have been added for applications that requires the user to add words at runtime (without typing) which do not belong to a pre-defined word list.

2. Definitions of Concepts

Speech Recognition Engine

The speech recognition engine performs the actual speech recognition tasks. The engine can be operated through the platform independent API 1600/C¹.

All data required by the recognition engine can be passed as buffers and should be loaded into memory by the User Application. This data is provided with the engine or can be created with the Development Tools.

Language

A language is a concept embedded in the API 1600/C which contains all information and knowledge required for speech recognition purposes for a certain language. A language contains:

- language dependent data for the engine.
- link to speaker independent *user* model data.

Dictionaries

A dictionary is in essence an associated list of words and a list of possible pronunciations of that word. Dictionaries are used in the process of context building (grammar compilers, add word functionality) where you can specify which dictionary to use for a certain language.

The development tools gives access to an expert system to translate words into phonetic transcriptions, that acts as a dictionary for a language. Exception dictionaries make extensions to this expert dictionary since some words that originate from other languages, i.e., proper nouns, words that belong to a specialized field of interest, etc. can yield sub-optimal or wrong results when using expert systems. A dictionary contains an exception part and has access to the general expert system for the specific language. A standard dictionary is provided.

The API1600/C uses this dictionary-related functionality indirectly.

User

A user is a concept embedded in the ASRAPI² that contains all information and knowledge required for user management with speech recognition purposes. It contains all data related to a speaker (user) enrolled for the speech recognition engine and the methods how to manage this data.

Generic user data for each language that enables for speaker independent recognition is provided within the installation package for a language. Users can register for a certain language.

The user concept is a part of the ASR database system. The ASR database is present in the development environment but not on embedded systems. Therefore the API 1600/C has no knowledge of the concept *user*.

¹ The API 1600/C stands for the ASR1600 consumer API.

² The ASRAPI stands for the ASR engine for a Win32 environment. This is used by the L&H ASR development tools.

Grammar

Continuous speech recognition requires not only a vocabulary (list of words) but also a grammar that is a set of relations between the words in a phrase. In essence a grammar is the formal description of the phrases a recognizer can expect. A very simple grammar is an isolated word grammar, which states that an allowed phrase is a word belonging to a specific vocabulary.

Continuous speech grammars are application dependent and are written by the application developer in a specific grammar language and are compiled with a *grammar compiler* into *contexts*.

The L&H ASR1000, ASR1500 and ASR1600 Development Tools provide a BNF grammar compiler. This enables the compilation of grammars written in the L&H BNF grammar language³.

Syntax

Syntaxes are defined here as limited domain grammars for which grammar compilation can be omitted.

Contexts can be created making a reference to the syntax they are to use.

Syntaxes included are:

- Isolated word syntax (IWS): allowed phrase is a word belonging to a specific vocabulary.
- Key word spotting (KWS) syntax: allowed phrases are phrases that contain a word belonging to a specific vocabulary.
- Continuous digit (CD) syntax: allowed phrases are digit strings of arbitrary length.

Context

The Development tools provide the possibility to compile grammars into a representation that is suited for L&H recognition engines. These compiled grammars are called contexts. All the necessary data of languages, users, grammars, symbols, vocabulary, etc. needed for in the recognition engine is handled in the concept of a context.

Contexts are structured around symbols that describe a set of phrases. Each context has at least one exported symbol that can be activated on a recognition engine.

Contexts can be structured hierarchically. Each context can make references to symbols exported in other contexts. These symbols, which are used in a context but defined in other contexts, are the imported symbols of the context. Contexts that have imported symbols can not be activated on a recognizer until all imported symbols have been resolved. A context that imports a symbol can be linked with a context that exports the symbol, in order to resolve the imported symbol.

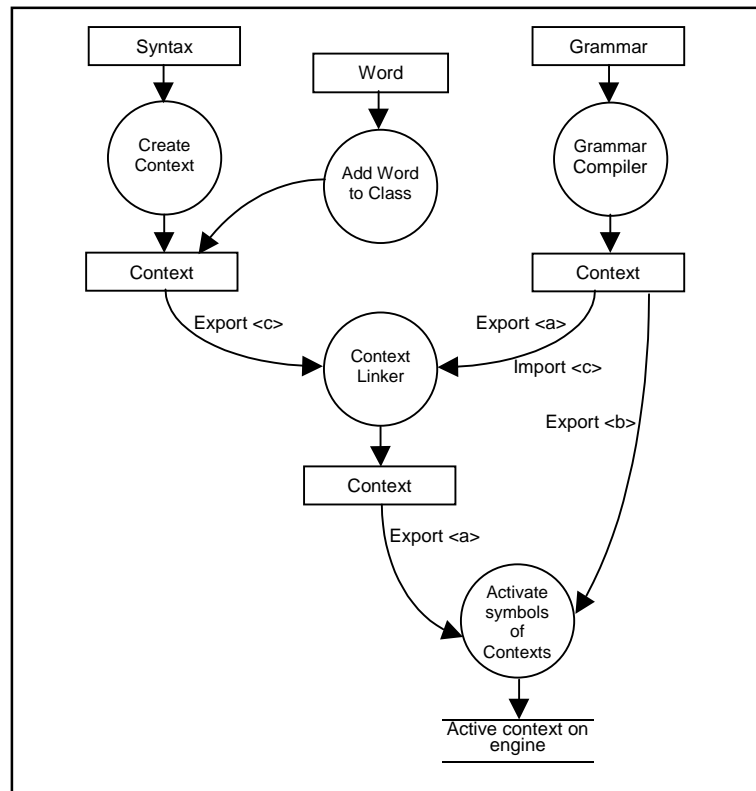
This feature allows the application developer to build a library of contexts that have exported symbols that are likely to be reused in different applications, like free-format numbers or date & time.

A list of symbols belonging to different contexts can be activated together on a recognition engine enabling for the recognition of the union of the phrases defined by all symbols. Activation of contexts takes care of the loading of language and user data to the recognizer. The current version of the ASR1600 requires that all symbols of all contexts are in the same language.

³ See the PC based Development Tools User Guide for the full specification.

Contexts can contain classes that are in essence lists of equivalent words in a certain grammatical context. New words can be added to classes, and words can be deleted from classes with the Lexicon ToolKit without recompilation of the grammar that generated the context. Phonetic transcription of words belonging to classes remains modifiable. This feature provides a dynamic context capability allowing for the design of contexts with empty classes where the words are added and deleted from context at runtime, under control of the application software.

The relation between grammars, syntaxes and word classes and the context development flow chart is represented graphically in the following figure:



The grammar and its vocabulary must be well designed to guarantee a high recognition performance. Following a few simple guidelines when selecting word-sets can greatly improve the recognition rate of an application:

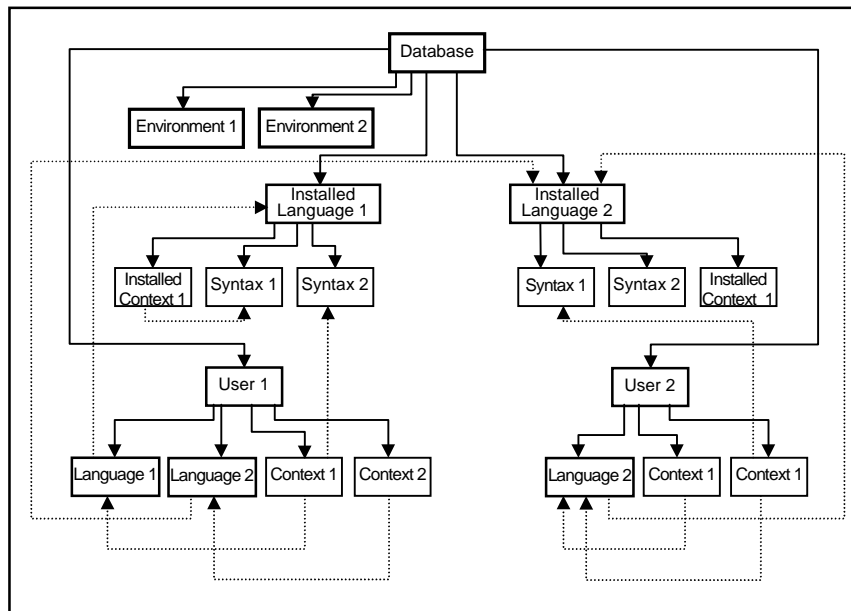
- Avoid the use of short words in word classes whenever possible, especially monosyllabic words.
- Avoid the use of similar sounding words in word classes whenever possible.
- Limit the context perplexity (the average number of words following a word calculated over all words in the context).
- Limit the maximal branching of the context (the maximum number of words following a word calculated over all words in the context).
- Limit the number of words in a word class. This may require splitting up a word class in a number of smaller word classes and modifying the grammar to avoid having all word classes active at the same moment. For instance, a long list of employee-names may be subdivided according to department, and the grammar modified to require a department name to be stated before the employee name.

ASR Database

The centralized repository for all information required by the speech recognition engine on the PC. The development tools are built on this engine. All *users*, *languages*, *grammars* and *vocabularies* that are to be used with a recognition engine and a specific setup are organized in a database.

Databases can be shared by many applications simultaneously. This allows for maintenance without interrupting running speech applications. New contexts can be compiled, new languages can be installed, new users created and multiple recognizers can be serviced.

The general structure of a database is shown in the following figure:



A database consists of the following items:

- Environments
- Installed languages
- Installed syntax's
- Installed contexts
- Users
- Languages
- Contexts

Environments

The speech recognition engine can run on different platforms (e.g., Engine running in Win16, Win32, RISC processor or on a DSP-board). The necessary information about the platform is put in the database and is called an Environment.

Installed Languages

On initial installation a number of languages are installed in the database. Included in an installed language are the speaker independent models, the standard exception dictionary for this language and the necessary DLL's for the functionality of adding new words to a context. This data is shared between all users in the database.

Installed Syntax's

For each installed language, there can be several syntaxes installed. An installed syntax can be used to create a context for a certain user. The user then can add words in the word classes (a list of words) of the syntax or delete them. One example is an isolated word syntax: the user adds the words to be recognized to the only word class in the syntax. Each syntax is linked to a specific language and cannot be used for other languages.

This data is shared between all users in the database.

Installed Contexts

For each installed language, there can also be a number of predefined contexts called installed contexts. A context is a combination of the language, the syntax and the words. The user can register such a context for own private use. To be able to register such a context, the user must have registered the language that is linked to this context. After registering he can add or delete words from the word classes in the context. This data is shared between all users in the database.

Users

To be able to use this database on a recognizer, you must create at least one user, register at least one language for this user and register an installed context or compile one grammar. Registering of a language means copying all the language related data that can be user dependent. The same is true when you register a context. For the context there are two cases: the context is making use of an installed syntax or the context is generated with the grammar compiler. For the first one, only the list of words the user has added is stored with the user. For the second one, not only the list of words but also the grammar itself is copied to the user. That's why if you ask for the syntax name of the second one, you get an empty string (meaning not linked to an installed syntax).

Languages

Each user can have several languages registered for private use. The user can only register or create contexts for the languages he has registered.



Contexts

Each user can register installed contexts, import contexts with the grammar compiler or create contexts with an installed syntax for own private use.

The L&H ASR Development tools

3. Application Context Development

The development cycle needed for speech enabled user interfaces is described below. Design of user interfaces is more an art than a craft.

Man-machine dialog design

This is the conceptual design stage where the application designer defines the speech enabled user interface that is required by his application. Important for the design are the possible speech prompts, the possible voice commands, the associated actions and the behavior of each command in the dialog at each state of the dialog.

Context development

Based on the conceptual design and the voice inputs that were defined, contexts that formally describe this expected voice input have to be developed.

Development tools for this purpose are provided with the L&H Development Tools. A phonetic expert system is provided to design vocabularies for isolated word recognition tasks and keyword spotting. Grammar development tools are provided to develop continuous speech contexts.

Performance testing

Speech recognition performance on the contexts that were developed can be tested in real-time or with prerecorded speech files with the evaluator program provided with the L&H Development Tools.

4. Creating context and word class buffers for DreamCast platform

To produce context and word class buffers that can be used on the embedded system following steps have to be completed:

1. Make a BNF file and import this in the ASRAPI database or create a context based on a syntax with the LexTool. For more information about how to produce a BNF file consult the ASR PC based development tools user guide.
2. You can test the context for performance on the PC platform.
3. Export the context in the LexTool using the asr1600 export format. As a result a context buffer and a word class buffer are written out.
4. The context buffer is useable on PC. The word class buffer has to be split up into a word and a class buffer. Use the CONVWCL tool for that purpose.

The buffers that are written out by the LexTool are not directly useable on the platform. The use of this conversion program is:

```
CONVWCL inputfile outputfile1(*) outputfile2(**)
```

(*) Outputfile1 corresponds to the classes output buffer. For the format of the classes buffer consult appendix A.

(**) Outputfile2 corresponds to the words output buffer. For the format of the word buffer consult appendix A.

5. On the DreamCast platform the class buffer can be passed to the API with the CasrImportClass function. The word buffer is not used directly used by the API. It is up to the application to interpret the buffer. A handy function is present in the ctxfuncs.c file, which can be used in most of the applications that want word string feedback. If you do not wish to use the word string information on the platform you can use the showctxinfo tool to extract the word id's from the words.

APPENDIX A

Classes buffer.

DWORD: Byte size of this buffer
DWORD: Number of classes (Nc)
char sz[]: Class name 0
char sz[]: Class name 1
...
char sz[]: Class name Nc-1

Words buffer.

DWORD: Byte size of this buffer
DWORD: Number of words (Nw)
char sz[]: Word name 0
char sz[]: Word name 1
...
char sz[]: Word name Nw-1