16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

# WATCH-DICTAPHONE FOR AUTOMATIC MEDICAL CODIFICATION

*S. Grassi [(1)], P. Heck [(2)], P. Stadelmann [(1)], P. Cotofrei [(3)], P. Dinissen [(2)], F. Meylan [(2)], J.-P. Mignot [(2)], J.-N. Pfeuti [(2)], F. Piccini [(2)], P. Geiser[(1,†)], G. Biundo[(1)], P. -A. Farine [(1)], J. Grupp [(2)] and K. Stoffel [(3)]*

(1) Institute of Microtechnology / Electronics and Signal Processing Laboratory, University of Neuchâtel
A.-L. Breguet 2, CH-2000, Neuchâtel, Switzerland, http://www-imt.unine.ch/esplab
phone: + (41) 32 718 34 22, fax: + (41) 32 718 34 02, email: sara.grassi@unine.ch, pierre-andre.farine@unine.ch

(†) Now with "Phonak Communications AG"

(2) ASULAB, Division of The Swatch Group R&D Ltd, Rue des Sors 3, CH-2074, Marin, Switzerland
phone: + (41) 32 755 56 66, fax: + (41) 32 755 59 66, email: pheck@asulab.ch, jgrupp@asulab.ch, http://www.asulab.ch

(3) Information Management Institute, University of Neuchâtel, http://www.unine.ch/imi

## ABSTRACT

*This paper describes the development of a voice data recording watch and the accompanying software and hardware needed to build an integrated solution for medical data capture and automatic codification. The developed system is used to capture on the stand-alone watch, either by voice recording or by form-filling, the relevant information on the healthcare service being provided. The captured information is stored locally and then downloaded to a PC, via a USB connection. In the PC, speech is translated first into text and then into medical codification, with minimum user-intervention. The resulting information is stored into local or centralized patient databases, to be sent later to 3rd-party SW for billing, reporting and statistics. The user controls the watch either via a touch-screen based interface or via a voice-driven interface for hands-free operation. Although this work is multidisciplinary, in this paper the emphasis is put on the description of the DSP implementation.*

## 1. INTRODUCTION

The TARMED medical codification [1] for healthcare tariff structure was introduced in the whole medical sector in Switzerland (hospital, private clinics, physicians and other healthcare providers) in January 2004, with the goals of transparent cost control and savings through a more efficient and reliable electronic data capture and data exchange among all actors of the healthcare system. However, TARMED complexity, with about 4'600 codes, described with the help of about 10'000 rules, induces a heavy work overload for the healthcare providers. The aim of the work described in this paper is to produce an integrated solution, ranging from data capture to automatic medical codification, to overcome the work overload generated by the introduction of TARMED, and to exploit its potential cost savings.

Although different SW and HW solutions exist currently in the market, none of them features a truly portable recording device, fulfilling the needs of the healthcare professionals for hands-free operation and mobile data capture. Thus, the core component of the developed system is the "watch-dictaphone", a user-friendly, voice-controlled portable voice recording device, having the form-factor of a wristwatch.

The work described in this paper was carried out through the contribution of many people with expertise in different fields such as medical practice and public healthcare, system and user interface conception, watch and hardware (HW) design, display and audio chain in watches, microcontroller software (SW), PC SW and network and database applications, as well as speech processing and DSP implementation. However, in this paper, the emphasis is put on the description of the DSP implementation.

The paper is organized as follows. The overall system is described in Section 2. The DSP implementation is presented in Section 3. Field-test results and a rough comparison with existing devices is given in Sections 4 and 5. Finally, conclusions and future work are given in Section 6.

## 2. OVERALL SYSTEM DESCRIPTION

The main components of the developed system are: 1) The watch-dictaphone. 2) The watch cradle for battery recharge and USB connection to the PC. 3) The PC software application.

The watch-dictaphone works either standalone or attached to the PC through the cradle. The standalone watch is used for capturing and storing locally the relevant information on the healthcare service being provided, either as voice recordings or as filled-forms. Later, when the watch is put on the cradle, data is downloaded to the PC, where the speech recordings are translated first into text and then into medical codification, with minimum user-intervention, mostly for verification and approval after each automatic translation stage. The captured and processed information is stored into local or centralized patient databases, to be sent later to 3rd-party SW for automatic billing, reporting and statistics, or to generate and print paper filled-forms with barcodes.

The system supports multiple institutions for the same user, e.g. a doctor that works at both a private practice and a hospital. The user controls the *stand-alone* watch with two conceptually similar user interfaces (UI):

- Touch-screen driven: touch sensitive watch crystal and three push-buttons for user input, associated with a three-line LCD display and the watch hands for user feedback. The watch crystal provides 13 touch sensitive zones, allowing the selection of voice recording commands such as play/pause, record, rewind, as well as the 0-9 digits and commands for browsing menus and choosing menu options.

- Hands-free voice driven: command and control (C&C) automatic speech recognition (ASR) for user input, LCD display and watch hands for user feedback.

When the watch is on the cradle (connected to the PC) both UIs are switched off, and the user controls the system through the Graphical User Interface (GUI) of the PC SW (§ 2.3). The PC controls the watch via a USB connection, with two possible operation modes. In the default "audio mode", the PC sees the watch as a standard USB audio card, and the watch is controlled by the audio drivers of the PC operating system. In the non-standard "data mode" the PC SW application issues "ftp style" commands to the watch via the USB port to perform tasks such as managing the files stored in the watch. The USB audio mode is mainly used for the training, taking into account the watch audio chain, of the PC speech recognition (Speech-to-Text) (§ 2.3).

### 2.1 The watch-dictaphone

The first prototype of the watch-dictaphone (see Figure 1) has the following characteristics:

- 47 mm diameter and 15.8 mm height.
- Rechargeable 150 mAh Li-Ion battery.
- Water resistant up to 30 m, to withstand the frequent hand-wash of the healthcare providers.
- Audio chain, with low-power AD/DA converters, amplifier, microphone and (piezoelectric) speaker.
- User authentication for secure data access.

The watch has three processors:

- Proprietary CoolRisc microcontroller, handling watch functions, LCD display, push-buttons, touch screen, and UIs, as well as communication with the DSP processor.
- A DSP processor, of the C55 Family from Texas Instrument [2] to manage audio acquisition, C&C ASR, speech compression (coding), and storage in a FLASH-NAND based proprietary file system allowing audio edition (segment insertion and deletion). The DSP processor also implements communication with the CoolRisc and the USB controller.
- A USB controller to manage the USB communication between the PC and the DSP.

### 2.2 Audio and data files in the watch-dictaphone

When the watch is on the cradle, several data files are downloaded from the PC and stored in the watch, including identification of the user, the institution and the watch, models for the barcode "paper-forms" to be filled, audio help files, trained models for the ASR, and the patient list (id number, name, date of birth, gender) selected by the user from its medical practice database.

When the watch is stand-alone, the user can record into audio files either the description of medical acts for codification or notes related to a patient, to an institution or to private purposes. Creation and modification timestamps are added to each audio file. When a patient is unknown to the watch, the user is asked to record an audio patient identification file. Additionally, to conform to a common hospital practice, medical acts can also be recorded by selecting and filling forms with "checkmarks" on the watch, as well as inputting parameters such as administered quantities.

### 2.3 The PC software

The PC software architecture was designed around a GUI from which the functional modules are called: the authentication module (for PC and watch users), the control speech module (based on


Figure 1 - The watch-dictaphone and its cradle

Scansoft Dragon NaturallySpeaking engine [3] and including User-Training, Vocabulary-Building, Acoustic-Optimizer and automatic Speech-to-Text functionalities), the text processing module (text parser, transcription analyser, synonyms lookup, macro-codes builder and Tarmed Explorer classifier – a code classification tool for TARMED, based on a WEB interface and using ontological information to dynamically structure the requests' results [4], [5]) and the file management module (to control the data transfer from/to the watch).

The downloaded, processed data is stored into patient databases, either local (in case of an office-based medical practice) or distributed (in the case of a bigger medical practice, such as a hospital). The PC software also contains a data management module allowing intelligent search of the filled-forms with barcodes, for visualization and printing.

The PC software features maintenance and configuration operations such as firmware upgrade of the DSP processor and the USB controller, training of the ASR in the watch, watch alarm and date setting, institution creation/deletion on the watch, DSP File system formatting and initialization as well as download of pre-recorded speech files to the watch.

## 3. DSP IMPLEMENTATION [2], [6], [7]

### 3.1 HW architecture

The HW block diagram of the DSP connections, within the complete system, is shown in Figure 2. Two external memories (not shown) are connected to the DSP: A 4-Mbit FLASH-NOR for storing program code and a 256-Mbit FLASH-NAND for storing data files.

The DSP has three serial ports (MCBSP0, MCBSP1, MCBSP2) which are respectively connected to the USB controller, the Audio Integrated Circuit (AIC) and the CoolRisc.

The CoolRisc sends commands to the DSP via a **4-line SPI bus** connected to MCBSP2. The **RDY** line is used to avoid overrun. As the DSP is the slave of the SPI bus, it cannot initiate a data transfer over the SPI, but it can use the line **RQST** to ask the CoolRisc to initiate a data transfer, which is then used by the DSP to send commands to the CR.

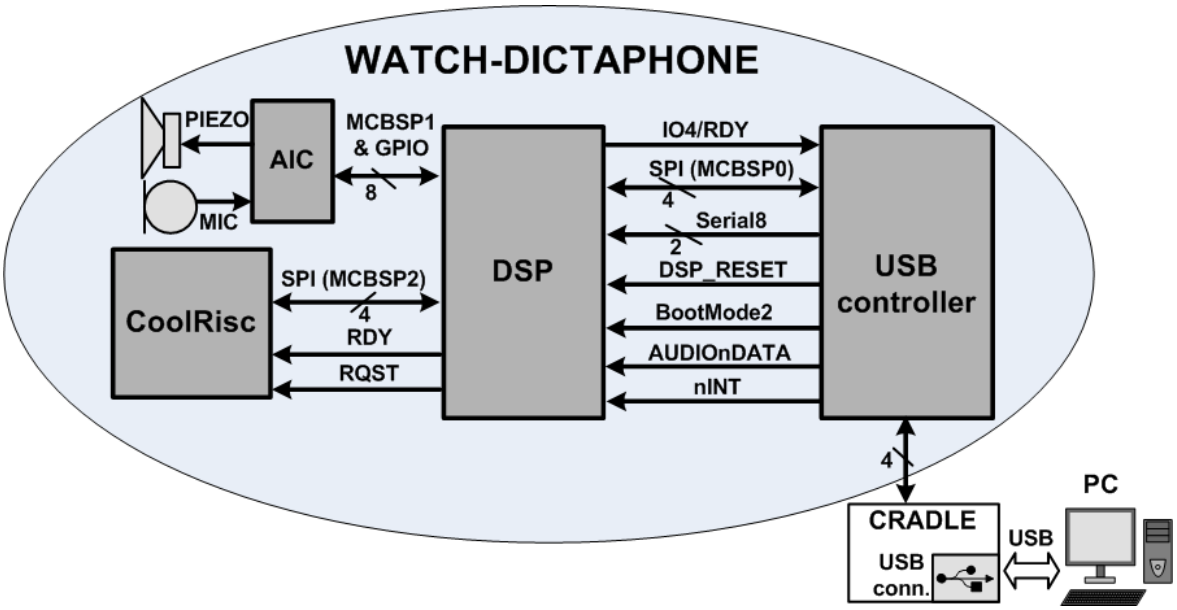


Figure 2 - : Watch-dictaphone HW block diagram

The USB controller configures the USB operation mode (data or audio, § 2) by first setting the line **AUDIOnDATA** and then sending an interrupt over **nINT** to the DSP, which reconfigures the interface. The DSP uses the line **IO4/RDY** to tell the USB controller that it is ready for transmission. The USB controller monitors this line before transmitting to avoid overrun. Packet communication in audio or data mode is carried out over the **4-line SPI bus** connected to MCBSP0. The USB controller can reset the DSP using the **DSP_RESET** line, and can control the DSP boot mode with the line **BootMode2**. Communication in boot mode over serial port is performed over the **2-line Serial8 bus.**

An Audio Integrated Circuit (AIC) is used for audio acquisition and playback. It contains AD/DA converters used in 16-bit mode, 16 kHz sampling frequency, as required by the speech codec (§ 3.3). The base input frequency is given by a 16 MHz quartz oscillator also used by the DSP. The DSP configures the AIC via an SPI bus, built over 4 generic DSP I/O. On one side (analog audio) the AIC codec is connected to the loudspeaker and the microphone. On the digital audio side the AIC is tied to the MCBSP1 of the DSP, through a 4-wire I2S bus. The DSP manages the audio I/O using two DMA channels serving MCBSP1 interruptions, with a double buffer scheme and buffer size of 320 words matching the frame size of the speech codec (§ 3.3).

### 3.2 SW architecture

The DSP SW architecture closely matches the HW architecture (§ 3.1) and the system user scenario (§ 2).

When the CoolRisc detects that the watch is on the cradle, it enables the USB in audio mode and disables the touch screen interface. The watch is then seen as a standard audio card by the PC. When the PC SW application is launched, it switches the USB mode from audio to data and issues “FTP-style” commands to the DSP via the USB connection, to perform tasks such as managing the files in the DSP file system, or writing a program in the DSP FLASH-NOR. At exit, the PC application sets the USB in audio mode again. Finally, when the CoolRisc detects that the watch is removed from the cradle, it disables the USB, and enables the touch screen interface. Note that the CoolRisc controls the USB enabling / disabling, whereas the PC controls the USB audio / data mode switching.

The DSP is controlled by either the CR, via an SPI bus connected to MCBSP2, or by the PC, through the USB controller, via another SPI bus connected to MCBSP0. In both cases, the DSP is the slave of the SPI bus. The MCBSP2 is configured to interrupt the DSP CPU when the CoolRisc initiates a data transfer (sends a command) to the DSP over the SPI bus. A DMA channel is used to transfer data from the MCBSP0 to memory and is configured to interrupt the DSP CPU when the USB controller has sent a packet to the DSP over the SPI bus.

In the main routine of the DSP software, the file system is mounted and the MCBSP0 and MCBSP2 interrupts are enabled. Then the DSP enters idle mode, waiting for a HW interrupt due to the CoolRisc initiating a data transfer to send a command to the DSP. When the DSP receives such an interrupt, it posts a SW interrupt to a routine that receives and executes the command. One particular command is for opening the USB channel for communication with the PC.

Figure 3 shows a diagram of the DSP software, containing the different SW modules, and the interaction with the user interfaces. The different SW modules are:

**CR-communication:** manages the CoolRisc-DSP communication protocol through SPI.

**ASR:** 3rd party Automatic Speech Recognition.

**CR-command:** serves the received CoolRisc commands.

**AIC:** driver for the Audio Integrated Circuit**.**

**Codec:** speech compression / decompression (§ 3.3).

**FM:** File Manager (§ 3.4).

**FS:** File System (§ 3.4).

**NAND:** FLASH-NAND HW Abstraction Layer (§ 3.4).

**FTP:** serves the "FTP-style" commands from the PC, received via the USB port through the USB controller.

**USB:** manages the communication protocol through SPI between the USB controller and the DSP.

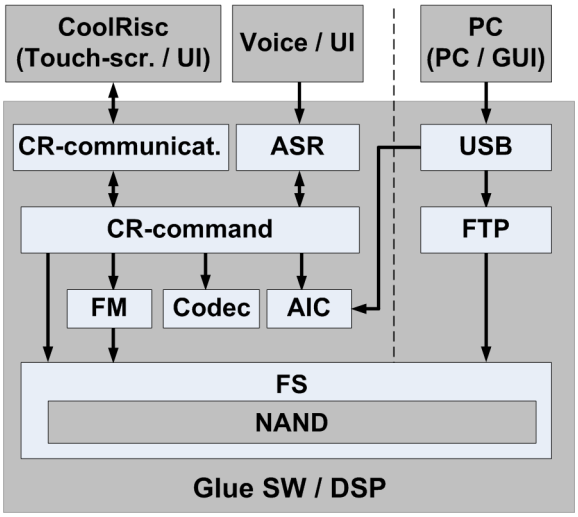


Figure 3 - DSP software diagram

### 3.3 Speech coding

The implemented speech codec (compression / decompression) is based on the ITU-T G722.1 digital wideband codec [8], which has a bandwidth of 50 Hz to 7 kHz, and a bit rate of 24 kbit/s (allowing up-to 3 hours of recorded audio). The digital input to the encoder is in 16-bit 2's complement format at a sample rate of 16 kHz. The algorithm is based on Modulated Lapped Transform (MLT). It operates on 20 ms frames (320 samples) of audio. The description of the coding algorithm is given in terms of bit-exact, fixed point mathematical operations. The C code delivered with the standard is based on a set of "basic operations", which are routines implementing fixed-point operations. These 16- and 32-bit basic operators resemble assembly language instructions that are commonly found in DSP, thus allowing C-code implementations that can be directly mapped to DSP assembly code.

The C55 is a 16-bit fixed point DSP. Its C-Compiler [9] provides a set of intrinsics to implement the fixed-point basic operations commonly found in the C-code for standard audio codecs. These intrinsics produce assembly language statements directly into the program. The C-code delivered with the standard was thus ported using these intrinsics.

### 3.4 File System and File Manager

We have developed a proprietary FLASH-NAND File System to fit into the limited resources available, in particular to avoid adding an external RAM. The file system manages all the low-level organization of the files in the Flash (e.g. block allocation), yet the exposed API is very similar to usual high-level file manipulations libraries such as *stdio* in the standard C library. It implements a cache mechanism to provide fast access to data and reduce the number of accesses to the Flash. Since the reliability of this kind of memories degrades over time, an error detection and correction scheme is used to validate all data read from the device. Moreover, all data programmed to the Flash are immediately re-read and compared to the source buffer to ensure that it was correctly memorized. The high-level API provides functions for:

- managing the Flash device (mount, unmount, format, get free space).
- managing files (open, close, delete, get file size).
- manipulating file data (read, write, seek, tell).
- managing directories (get directory listing, delete entire directory).

Lower-level internal routines handle all conversion between the high-level routines and the Flash structure, such as translating a read operation at a given offset in a file into a series of page loads. They also handle page allocations and management, garbage collection (i.e. recovery of Flash space no longer in use) and memory allocation. Finally, the FLASH-NAND HAL (HW Abstraction Layer) interfaces the Flash device and provides functions for reading and writing Flash pages.

The File Manager is built on top of the file system. It makes it possible to insert and remove data anywhere in a file. To this effect, it uses a collection of data files and a reconstruction file, but all details are handled internally and hidden from the application. The API defines a custom file type, and provides usual functions (open, close, read, write, seek, remove data, delete file …) as well as storage and retrieval of the creation and modification date and time.

The file organization and the file content were designed to support the required behavior of the watch in the different utilization cases, within the constraints imposed by the File System and the File Manager.

### 3.5 DSP development environment and tools

For DSP code development, we have used Code Composer Studio V3.1. To decouple the DSP development from the parallel development of the CoolRisc software, we have simulated the CR with an SPI communication box, connected to the DSP, and controlled by the PC via an USB port. A PC program was written for implementing the DSP-CR communication protocol and simulating sending "CoolRisc commands" to the DSP and retrieving the DSP answer to these commands.

Each of the "FTP-Style" PC-DSP commands is implemented in both the "DSP-side" (FTP server) and the "PC-Side" (FTP client, written in C-language using the GNU environment). Each command is then tested using a PC program written with MS Visual C 6.0. Finally, we have written a minimal GUI-based PC program for demonstrating and testing the use of the routines contained in the FTP client, and to support the development and testing of the other DSP software modules. This program was written using MS Visual C++ 6.0. The GUI was implemented using Gtk+ V2.0.

### 3.6 Power consumption

During watch-only operation of the device, the DSP processor is switched off and all the watch functions are handled by the low-power CoolRisc microcontroller (§ 2.1) with a minimal power consumption of a few micro-amperes. When operations such as audio recording or ASR are required, the DSP is switched on and clocked at the base frequency of 4 MHz. Clock frequency is dynamically scaled according to computational requirements of each operation mode: when more complex operations are performed, the DSP clock frequency is temporarily scaled-up to a value ranging from 48 to 84MHz, and later scaled-down to the base clock frequency. The value of the frequency increase and its duration depends on the computational power required.

The resulting power consumption according to the working mode is the following: 20.5 mA in recording, 38.8 mA in playback (including the load of the loudspeaker), 17.5 mA in voice command, and 0.09 mA in watch mode (including LCD).

The fully charged 150 mAh Li-Ion battery is typically sufficient for about 2 hours of recording (50% in hands free), plus 1 hour of

playback time, plus 2 hours of form filling (50% in hands free), plus 2 weeks in watch mode.

Ongoing work concerns further DSP software optimisation for power saving in order to increase the standby and working time of the watch in standalone operation, so as to be sufficient for two whole workweeks of the healthcare provider (currently we exceed, by almost one week, the initial target of fulfilling the needs of one workday).

## 4. FIELD TESTS

To date, thirty prototypes have been produced. For a first evaluation, the system was tested during 10 weeks by medical personnel (10 persons), in a pilot site consisting of different ambulatory services at a Swiss university hospital which uses forms to be filled with checkmarks for recording the services provided. Twenty different models of the forms to be filled were involved in the test. Typically, the device was used 1 to 4 hours per day for filling forms and recording reports.

The touch-interface was found easy to learn and use. On the other hand, the voice-interface was not frequently used, mostly as a complement to the tactile interface. The ASR training was an obstacle for many users, and in noisy environments the recognition was sometimes triggered by words pronounced by other people. More testing and development work should be done on this interface to make it more ergonomic and robust.

There was a high level of acceptance of the product, mostly due to the wristwatch form-factor, the users assimilating it immediately to a personal belonging. The quality of the sound recorded and the sound playback were judged excellent compared to existing voice recorders and PDAs the users were acquainted with.

## 5. STATE-OF-THE-ART COMPARISON

It is difficult to compare the developed device to existing products as, to the extent of our knowledge, the developed device has a unique set of features tailored to the application: very small volume and waterproof wristwatch form-factor; integrated microphone and speaker as well as USB connection and charging through case contact (no plugs); extended tactile interface; voice interface for hands-free operation; pre-organized capture of audio and non-audio information essential for the medical application and seamless integration into the complete medical codification system. Here we consider for a rough comparison: “voice recorder and media player wristwatches”, digital voice recorders (dictaphones), and tablet PC or PDA based Electronic Medical Recording (EMR) systems.

There exist few watches with audio recording/playing capabilities intended for leisure as well as professional digital voice recorders (such as the Olympus DS-50 or the WS-331M [10]). They are not waterproof, having plug connectors for external speakers and USB, and lack capabilities to capture pre-organized audio and non-audio medical data for further processing. On the other hand, their recording time is higher than in the watch-dictaphone, as this feature was dimensioned according to the needs of the application, to allow smaller size and power consumption and the availability of other necessary features. The larger volume of the dictaphones (a factor of 2-3 compared to the watch-dictaphone) allows also a larger display and a more comfortable tactile interface, at the cost of portability and hands-free use.

Tablet PC or PDA EMR systems have larger display and recording capabilities, but are not truly portable and hands-free.

## 6. CONCLUSIONS AND FUTURE WORK

We have presented the development of an integrated solution for data capture and TARMED medical codification for the healthcare domain, with emphasis on the description of the DSP implementation, where one of the main challenges is the implementation of sophisticated speech processing functions into a miniaturized and portable low-power device as well as the design of an ergonomic hands-free voice driven interface.

Future work includes reengineering the watches, taking into account the results of the field tests and reducing the manufacturing costs.

The developed system is easily extendable, with only SW modifications, for use in other medical practices, in other languages and with other codification systems such as Diagnosis-Related Groups (DRG) codes. Finally, it is possible to use the system for other applications requiring data capture and code classification, even outside the healthcare domain, such as customs classification of the products in transit.

A watch with speech-to-text has a much broader application field outside of any classification system, e.g. to dictate personal e-mails. This application however is much more price sensitive and the present development is considered as a precursor of such a general consumer product.

## 7. ACKNOWLEDGEMENTS

This work was partly supported by the Swiss Federal Office for Professional Education and Technology (OPET) through the Innovation Promotion Agency (CTI) under Grant CTI 6935.1 ENS-ES (“ATTra-Watch” project). We are most grateful to all the people who have contributed to this work: medical Dr. J.-C. Vergriete and medical Dr. P.-Y. Sandoz; R. Patthey; R. Calame at “Hôpital de la Béroche”; Dr. M. Ansorge at IMT; J.-J. Bart, A. Comtesse, Y. Ferri, J.-C. Martin, and R. Nicolet at ASULAB; and P.-A. Lautenschlager and D. Uccelli at Axiome Alpha S.A.

## REFERENCES

[1] http://www.tarmedsuisse.ch

[2] http://www.ti.com/

[3] http://www.nuance.com/naturallyspeaking/sdk/

[4] T. Kurz and K. Stoffel, "Web-Based Tools for Codification with Medical Ontologies in Switzerland", in *Proc. ECAI 2006*, Riva del Garda, Italy, 28 August 2006, pp. 642-646.

[5] http://taurus.unine.ch/tarmed

[6] S. M. Kuo and Bob H. Lee, *Real-Time Digital Signal Processing: Implementations, Application and Experiments with the TMS320C55X*. John Wiley & Sons, 2001.

[7] R. Oshana, *DSP Software Development Techniques for Embedded and Real-Time Systems*. Newnes, 2005.

[8] ITU-T Recommendation G.722.1. *Low-complexity coding at 24 and 32 kbit/s for hands-free operation in systems with low frame loss*. http://www.itu.int/rec/T-REC-G.722.1/en

[9] *TMS320C55x Optimizing C/C++ Compiler User's Guide*. http://focus.ti.com/lit/ug/spru281f/spru281f.pdf.

[10] http://www.olympus.co.uk/consumer/2581_2616.htm