Development of a Surround A/V decoder using Linux and embedded/Qt

Ardis Technologies BV

14 May 2003, Version 1.0

Table Of Contents

1 Introduction

This paper describes the development of an AV Controller with the involvement of two companies, the Scottish client and ArdisTech, a Dutch technical − ICT company. ArdisTech was responsible for the software development including integrating and interfacing to proprietary and licensed software modules and including integrating and interfacing to software files for electronic devices such as FPGA’s (Field Programmable Gate Array’s) and emulation and test software. In addition we did some of the electronic hardware design work. It was the client’s responsibility to do all hardware design and manufacturing and supply proper code and documentation of of all components to be integrated.

1.1 What is an AV Controller?

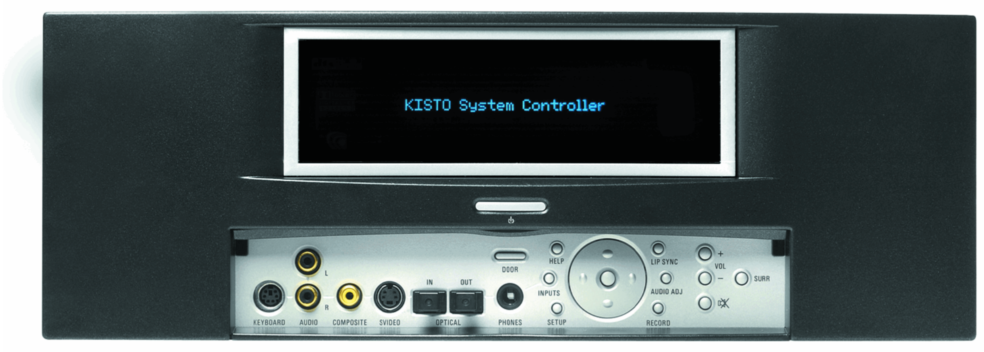


Figure 1.

An AV controller switches audio inputs with or without conversion and when required does surround decoding to drive a set of stereo, 5.1 or 7.1 power amplifier inputs. In addition, such product switches video from multiple inputs to a watch output for a television, monitor or projector. Audio and video outputs are also available for recording while watching and listening to the same or different sources for playback. Such product can be classified as belonging to the family of Home Theatre/Entertainment Products.

Source material can be from players using DVD−V, CD, SACD and DVD−A disks and many others such as computer, tape recorder, tuner, camcorder, satellite receiver, cable and VCR.

Figure 1 shows the front of the product.

1.2 Functionality

Although the product is currently available the functionality was not fixed at all when we started. Therefore the software was designed to be hardware independent. Specifications became more and more defined as we went along.

The product itself is best described as, a cross point switcher for analogue audio inputs with a 40x16 matrix, a switcher for digital audio inputs, a switcher for analogue composite, S−video and component video inputs and a cross switcher between analogue and digital audio with or without sample rate conversion and AD/DA conversion. Audio can be surround sound decoded and or post processed into various output formats. The available surround decoding algorithms can be changed during the lifetime of the product but obvious decoding algorithms such as Dolby Digital, Dolby Digital EX, DTS Matrix ES, DTS Discrete 6.1, DTS 96/24, Neo:6, Dolby Prologic II, Dolby Headphone, Lip sync and a proprietary algorithm called Limbik Party are available from the start. The AD and DA conversion runs at 96 KHz, 24 bit and when required audio signals are up sampled and down sampled using sample rate converters.

The front of the machine has a dot matrix VFD screen with a 256 x 64 pixel resolution, a number of buttons including a navigator, headphone socket, auxiliary audio in − and output and video inputs as shown in figure 1.

Part of the electronics of the product can be switched into low power mode without disabling the record path.

The navigator allows scrolling through the menu’s as shown in figure 2

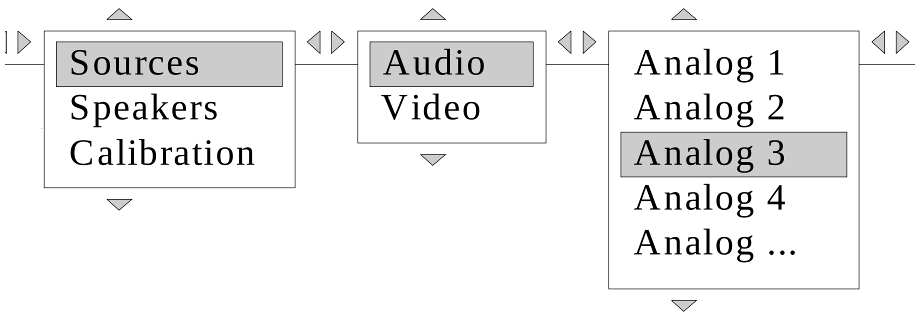
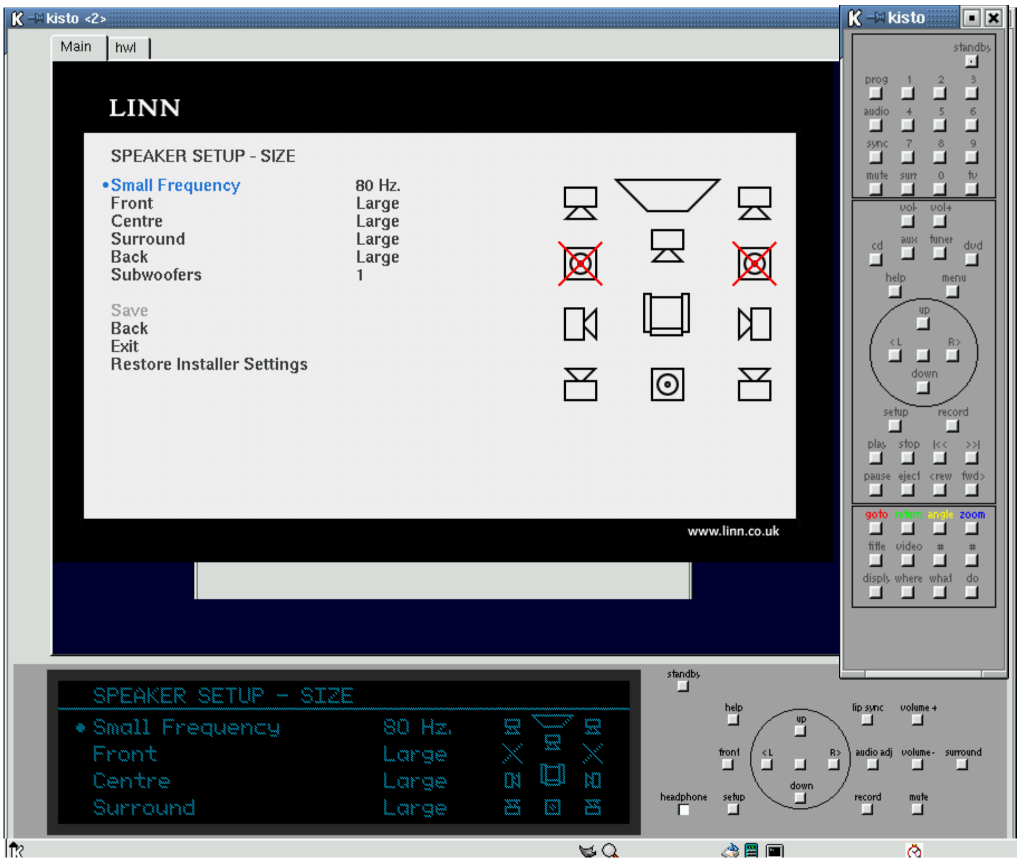


Figure 2.

An emulation example of the speaker calibration screen both on the VFD and monitor are shown in figures 3. Also an emulation of the handset is shown.



SPEAKER SETUP - SIZE

•Small Frequency	80 Hz.
Front	Large
Centre	Large
Surround	Large
Back	Large
Subwoofers	1

Save
Back
Exit
Restore Installer Settings

```
SPEAKER SETUP - SIZE
• Small Frequency    80 Hz.
  Front              Large
  Centre             Large
  Surround           Large
```

Figure 3.

Audio and video inputs can be randomly combined and be made available to represent for example DVD player 1. That way only DVD has to be selected to get access to the possibilities of a multifunctional DVD player. These settings can be saved in so called profiles. Five of these profiles can be kept in the product. These profiles can be up − and downloaded from/to the product via an RS232 port or ethernet. Surround modes offered depend on the incoming stream and speaker set−up. There are numerous in− and output connectors on the product. A number of these are duplicated on a different socket type. Duplicates are not mentioned here.

In and Outputs	Number
Stereo Analogue Audio Inputs	10
Stereo Digital Audio Inputs	6
Composite (Component) Video Inputs	12(4)
S−Video Inputs	6
Component (PrYPbHV) Video Inputs	1
RGBHV Video Input	1
Stereo Analogue Audio Outputs	Front, Side, Rear, Centre/LFE
Stereo Digital Audio Outputs	2
Stereo Tape out 1/Sub 1,2	1
Stereo Tape 2 out	1
Composite (Component) Video Outputs, Watch	3 (1)
Composite (Component) Video Outputs, Record	3 (1)
Component (PrYPbHV) Video Output	1
S−Video Output, Watch	1
S−Video Output, Record	1
Front Stereo Analogue Audio Input	1
Front Stereo Digital Audio Input	1
Front Composite Video Input	1
Front S−Video Input	1
Front Stereo Digital Audio Output	1
Stereo Headphones	1
RS232 ports	2

In and Outputs	Number
Proprietary Knekt ports	2
Proprietary RCU port	1
10 Mbit/sec Ethernet port	1
PS2 Keyboard port	1
Flasher Output	1
IR/RC5 Receiver	1
GPO’s	4

1.3 Additional Requirements

Because this product is the first in a line of products and since both functionality and specifications was expected to change during development, the software is designed for reusability and hardware independence.

In addition because this embedded product is a typical consumer product to be used in living rooms both soft− and hardware needed to fulfil typical requirements such as : no noise, fast power up and down sequence, high reliability and the absence of a hardrive.

1.4 Selection of the DSP for Surround Decoding

In the beginning the proper SHARC DSP from Analog Devices was not available. That particular DSP was selected because it promised the largest flexibility to select decoding options any time any place. If an end user wanted to upgrade with an additional decoding algorithm this would be possible "in the field".

1.5 Processor with CRT Controller

Possible choices for the processor were DragonBall VZ, Arm and PowerPC processors. The DragonBall VZ is based on the 68K core and limited to 33MHz. This could cause the On Screen graphics drawing to become too slow. The choice for PowerPC instead of ARM is mainly based on the fact that there is a PowerPC chip available with graphic controller and almost all the IO on the chip. This is the MPC823e chip. The output of the MPC823e is CCIR 4:2:2 format with 720 pixels per line. A video encoder must be used to generate PAL or NTSC video. An application note from Motorola describes how to insert this video into the program video stream for On Screen Display (OSD). This processor is also frequently used in set top boxes.

1.6 Other Intelligent Hardware Components

Other intelligent hardware modules apart from the DSP's for the software to interface to are a component from Universal Electronics (Valhalla), a client supplied H8S microcontroller module and a PS2 keyboard decoder.

1.7 Choice of the Operating System

At present 10% of product development in general is based on MMU (memory management unit) type open source environments. This is a huge success considering that embedded open source applications only started to surface between one and two years ago. Furthermore, studies show that this percentage will increase to 50 % in 2004. Open source means that you control every aspect of the development process. Using an MMU and open source means that time to market is much faster, changes can be made at the last minute and debugging of the application software suddenly becomes much easier. Cross platform development and emulations are easily set up.

The client until today used microcontrollers for most of their products. We suggested to use embedded Linux as operating system for this particular product and the client agreed. Many processors nowadays have a Linux port available which means that all the features of Linux can be used when needed. Because of its ability to work in a small footprint, its flexibility, the availability of the source code which let us fully control the firmware development process and its stability and reliability made Linux the perfect choice.

1.8 The Graphical Toolbox

In the case of a product such as this where graphical (colour) based display of information is a key issue, the additional use of a proper graphical toolbox helps to decrease development time substantially.

After checking out several possibilities embedded/Qt from Trolltech was selected as the graphical environment. Embedded/Qt is upward compatible with Qt. Qt is a cross platform graphical environment which at present can run under all important operating systems. Emulation of an application on different platforms therefore is easy. A graphical interface for example can be developed and emulated on any computer before it is installed on the target platform. Once target platform specifications are set, the development of both hardware and software can be done in parallel and independent from each other.

Since the product both had On Screen Graphics (OSG) and a VFD screen, an emulation of the graphical interface of the product would of course need to include both display types, the appropriate buttons, handset and emulations of all hardware components.

Embedded/Qt was successfully used as the graphical toolbox to make the display software for both the VFD display and the monitor. In Qt the character set is not limited to one font size or font style. Also non−western character sets can be used. However to display the same information in e.g. Japanese a VFD display needs to have a higher pixel resolution. Different character sets require different pixel resolutions:

Language	Horizontal resolution	Vertical resolution
Western	5	7
Thai	8	22
Korean	16	16
Japanese	14	14
Traditional Chinese	24	24

The second display of course is the (TV) monitor. See the figure 3.

1.9 Planning

Despite the evolving specifications of the product to be planned the project has been accepted for a fixed price with some time scale flexibility. The project phases 0 to 4 where defined as follows:

Phase 0: Defining product features and the preliminary product description as well as, the development approach and presenting rough cost estimates. Phase 1: Detailed design planning and specification of the product including responsibilities and delivery of an emulation environment showing how the Gui of the product could be. Phase 2: Deliverance of tested preproduction software prototypes. During Phase 3 certification issues have been taken care of, software bugs were solved and improvements made. Phase 4: Market launch.

2 Hardware and General Design

To have a quick start an evaluation board with the proper processor and Linux installed was bought. This evaluation board allowed us to develop and prepare the firmware in an early stage of the project. It also allowed us to use the electronic schematics supplied with the evaluation board to develop the first PowerPC prototype board.

Because of earlier mentioned requirements the decision to develop an emulation environment which included the software versions of all electronic counterparts to " dry test" hardware functionality turned out to be very helpful. This emulation environment was fully integrated in our development situation. Both approaches assured a quick start of software testing ones the first hardware prototypes became available.

In figure 4 a block diagram of the hardware is given. Blue shows video while green shows audio. Grey stands for computer type components. There is 64 Mbyte of SDRAM and 16 Mbyte of EEPROM available to the PowerPC in the product.

The main communication bus is I2C although communication with the H8S module, the Valhalla chip and PS2 keyboard controller uses RS232 and communication with the DSP’s is fully software based using SPI. I2C is a unidirectional bus so one has to assume that the commands are received although an I2C expander allows for interrupts to be used. All RS232 communication is interrupt driven. Communication with the DSP proceeds with writing a command to the SPI port and then on a regular interval poll if a Status register in the DSP has changed. Each action of a front button results in an interrupt from the I2C expander after which the data is red. The majority of the RS232 ports and the ethernet port are integrated in the PowerPC. The VFD display is directly connected to the PowerPC. All RC5 handling is done in the RC5 routing and H8S module using a proprietary software module of the client.

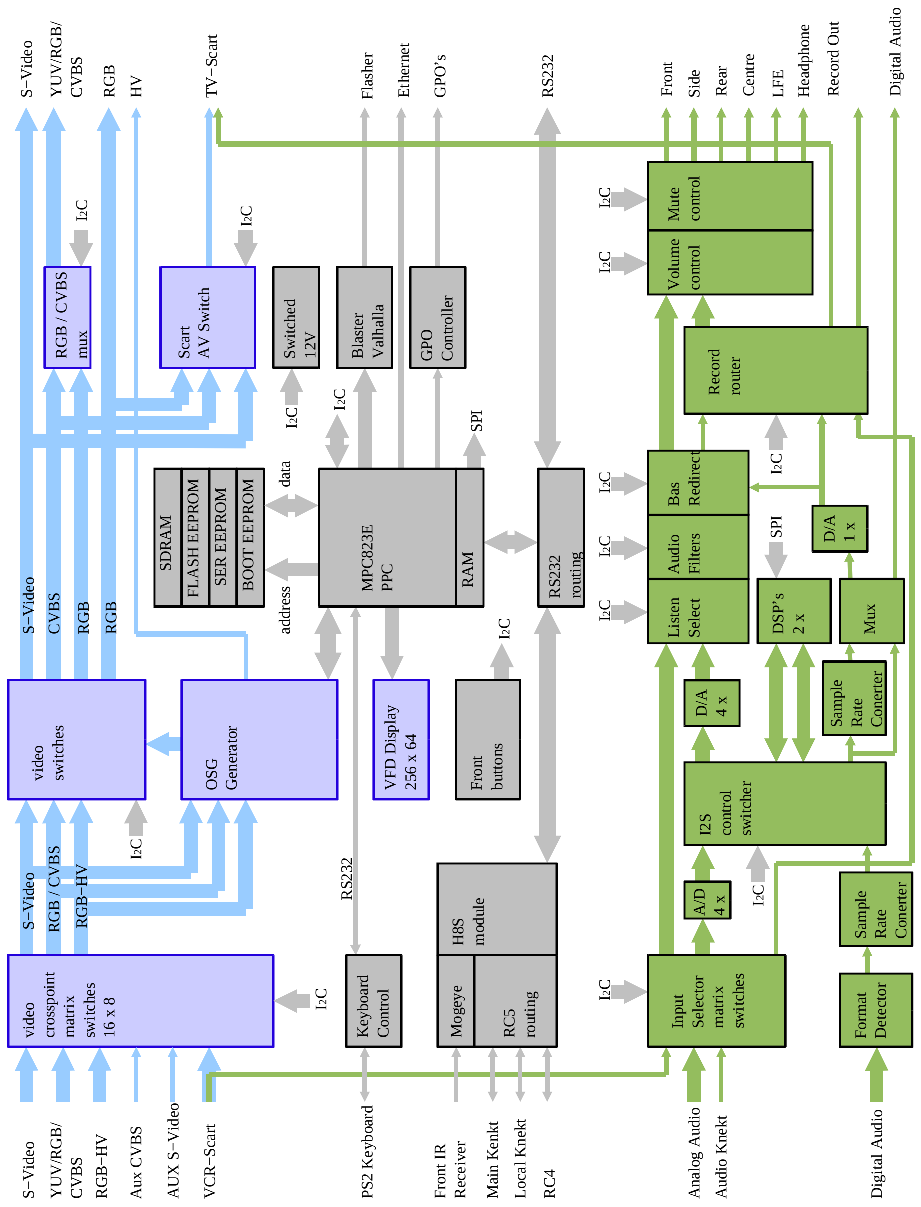

S-Video
YUV/RGB/ CVBS
RGB-HV
Aux CVBS
AUX S-Video
VCR-Scart
PS2 Keyboard
Front IR Receiver
Main Kenkt
Local Knekt
RC4
Analog Audio
Audio Knekt
Digital Audio
video crosspoint matrix switches 16 x 8
I2C
Keyboard Control
Mogeye
RC5 routing
H8S module
Input Selector matrix switches
A/D 4 x
Format Detector
Sample Rate Conerter
S-Video
RGB / CVBS
RGB-HV
video switches
OSG Generator
VFD Display 256 x 64
Front buttons
RS232
I2S control switcher
D/A 4 x
Sample Rate Conerter
S-Video
CVBS
RGB
RGB
address
data
SDRAM
FLASH EEPROM
SER EEPROM
BOOT EEPROM
MPC823E PPC
RAM
RS232 routing
SPI
Listen Select
Audio Filters
Bas Redirect
DSP's 2 x
Mux
D/A 1 x
SPI
Record router
RGB / CVBS mux
Scart AV Switch
Switched 12V
Blaster Valhalla
GPO Controller
Volume control
Mute control
S-Video
YUV/RGB/ CVBS
RGB
HV
TV-Scart
Flasher
Ethernet
GPO's
RS232
Front
Side
Rear
Centre
LFE
Headphone
Record Out
Digital Audio

Figure 4.

2.1 Analogue Audio Inputs

The many analogue audio inputs are routed through a 40 x 16 matrix. The signal can be kept " as is" and routed straight to the Listen Select or it can be digitised for routing through the DSP and then to the Listen Select via the DA converters. The signals are then fed to the Audio filters, Analogue Bass Redirect and through the Volume and Mute Control to the outputs. In the "as is" case when an analogue 5.1 signal is offered − e.g. SACD − the audio output is 5.1. When the signal is routed through the DSP the decoding, post processing algorithms in the DSP and the speaker set−up determine the multichannel output format.

Next to decoding algorithms the DSP can also do digital bass redirect, delay, lip sync and volume control.

2.2 Digital Audio Inputs

The digital inputs are routed to a format detector and then to a sample rate converter through a 16 x 8 I2S cross point matrix FPGA after which the signal is converted either directly to analogue or routed through the DSP’s and then converted or routed to the so called record path or TV−Scart socket via a sample rate converter, multiplexer and the record DA converter.

2.3 Composite (CVBS) Video Inputs

The composite video signals are routed through its 16 x 8 crosspoint Matrix Switcher to Video Switches. In Video Switches the signal is keyed on/off with the video signals generated by the Power PC for on screen graphics to the OSG Generator. Since inputs for CVBS can also be used for component (RGB or PrYPb) video, the video signal follows different routes in the Video Switches module. No conversion of the video signal is done.

2.4 Y/C (S−video) Video Inputs

The Y/C video signals are routed through its 16 x 8 crosspoint matrix switcher to Video Switches. In Video Switches the signal is keyed on/off with the OSG signal coming from the OSG Generator. No conversion of the video signal is done.

2.5 Component (RGB, PrYPb, RGBHV) Video Inputs

The data flow path is identical to that described for the composite signal. No conversion is done.

2.6 Analogue Audio Outputs

There are 8 analogue outputs which are labelled in stereo pairs: Front, Side, Rear and Centre/LFE. In addition there are two record outputs; Tape 1 and Tape 2. Tape 1 can also be used as extra subwoofer outputs. Also the audio signal is routed to the TV−Scart output.

2.7 Headphones

There are three headphone modes: stereo headphone for any analogue signal which is offered as an analogue signal, stereo headphone for an incoming PCM signal and Dolby Headphone which engages the Dolby Headphone postprocessor in the DSP. The latter assures a surround sound experience over a stereo headphone.

2.8 Digital Audio Outputs

The digital audio is coming from the Mux. The signal to the Mux either comes from the I2S switcher via the sample rate converter or it bypasses this component and comes straight from the I2S switcher to the Mux.

2.9 Composite Video Outputs

The RGB/CVBS Mux determines whether a composite or component video signal is available on the output socket.

2.10 Y/C (S−video) Video Outputs

The S−video output can only be connected to an S−video input. Both a watch output for monitor, and a record output for a VCR are available.

2.11 Component (RGB,PrYPb,RGBHV) Video Output

In the case of RGB or CVBS inputs the availability of which output format is used is determined by the RGB/CVBS Mux. The RGBHV signal has a separate BNC input and output.

2.12 Scart

Both a VCR−Scart (Input) and TV−Scart (Output) connector are available and the routing from and to the Scart connector has a large flexibility. Which video signal is available on the TV−Scart output connector is determined by the Scart AV Switcher.

2.13 GPO’s

The are four GPO outputs. For each of these the pulse length and polarity can be set. These GPO’s are used to control mechanical devices in a home theatre environment such as curtains.

2.14 Other In and Outputs

Other Input/Outputs are for the PS2 Keyboard, Front IR receiver, Knekt, RCU, Flasher, Ethernet and two RS232 ports for Projector control, control of other multimedia equipment and uploading and downloading data.

3 Software Architecture

In the block diagram (figure 5) a visualisation of the software architecture is shown.

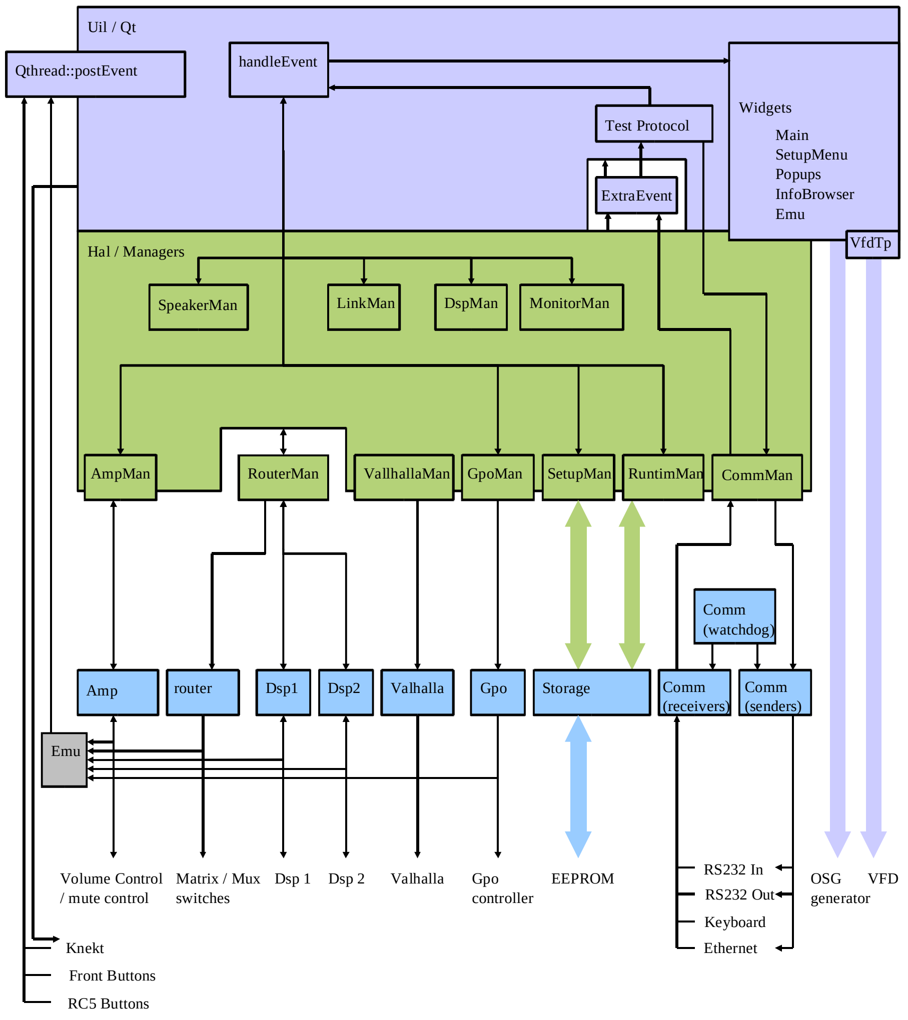


Figure 5.

The UI (with Qt) and all managers excluding ExtraEvent and Routerman are running in the main thread. ExtraEvent is an active/reactive object which convert event from the active/reactive system into Qt events. Routerman acts as a supervisor for all managers to assure that actions are done in the proper order. Most of the hardware implementation layer modules also have their own thread. Amp will do ramping, RS232 waits for data and so on.

Because many of the processes are asynchronous our (soft) realtime framework was used to implement these processes in a fast and stabile manner. This framework is part of the AOS Toolbox. It contains tools to handle memory management, object containers, input and output streams, event driven realtime features, file management and others.

Modules shown in figure 5 communicates with the underlying hardware via the following layered mechanism:

The **User Interface Layer** (UIL) comprises all the software that communicates between the user and the rest of the system. The UIL never directly accesses the hardware.

The **Hardware Abstraction Layer** (HAL) provides a virtual image of the available hardware using a number of classes. For example, a hardware switch matrix device is represented by an object of class `SwitchMatrix (figure 6)`, and only instances of this class know how to communicate with the corresponding hardware device. Most of the activity in the HAL is a result of UIL requests. The HAL can be divided in the hardware driver layer and the hardware emulation layer. The hardware emulation layer is active in parallel to the hardware layer.

The **Hardware Layer** is where the real−world hardware lives. The hardware devices communicate with the HAL layer only. Unless the system is compiled for hardware emulation, this layer has no associated software.A product such as this with matrixes, multiplexers, switches, routing and modules lent itself perfectly for an object based UML (Unified Modelling Language) approach and C++ as the language combined with the Gnu C++ compiler and linker, XEmacs as word processor and Gdb as debugger with CVS as the revision database system. A brief example of an OO approach for a switch matrix is given in figure 6.

3.1 UML Example

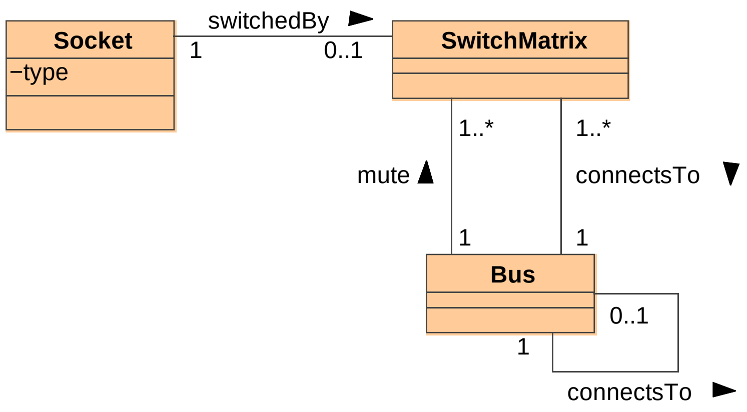


Figure 6.

The audio or video signal received from sockets (figure 6) is routed through a software controlled switch matrix device, represented by class `SwitchMatrix`. An instance of this class knows how to control the hardware device it represents via the earlier mentioned layered structure.A switch matrix is connected to a bus, for example a 16−channel analogue audio bus. It is the responsibility of a switch matrix to connect a socket’s incoming signal to a given channel of the connected−to bus. An instance of class `SwitchMatrix` can perform that task.

Several switch matrices can be connected to a single bus. To prevent routing conflicts, i.e. when more that one socket signal is connected to a given bus channel, class `Bus` is introduced. An instance of class `Bus` is able to detect routing conflicts on the hardware bus it represents and can instruct any associated `SwitchMatrix` instance to mute a given signal.

Using this type of approach all physical elements which we expected to turn up one way or another in the hardware where modelled using UML.

3.2 Handle Changing Specifications

The lack of exact hardware specifications forced us to make following decisions in an early stage of the project, to use an ASCII based netlist structure instead of a in software coded one and to create and use emulation versions.

Netlist Structure

What exactly is a netlist structure? To explain we first define an object called node where node represents for example an input, socket, matrix, dsp, AD or DA converter. Each node has its own logic which it uses to decide the output to select to follow a routing path. The information required to perform such a disconnect or connect sequence is laid down in a netlist structure.

To be able to quickly respond to hardware changes the netlist structure is made into an ASCII file instead of being directly coded in the source. The ASCII file is parsed upon power up and then the hardware representation is created. Even the Gui depends on this ASCII file and gets the names and other information of inputs and outputs on power up from the parser as well.

Emulation Versions

We have discussed the presence of emulation versions before. All emulation versions also include a graphical user interface and they represent switches, muxes, amps, DSP's and GPO's as shown in figure 7.

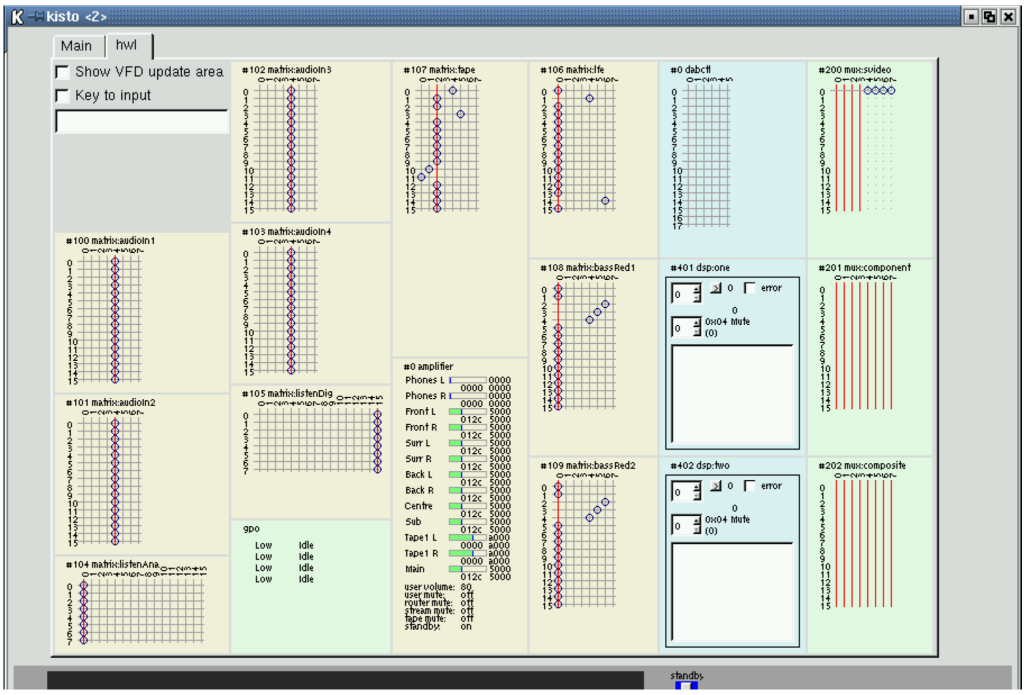


Figure 7.

3.3 State Machines

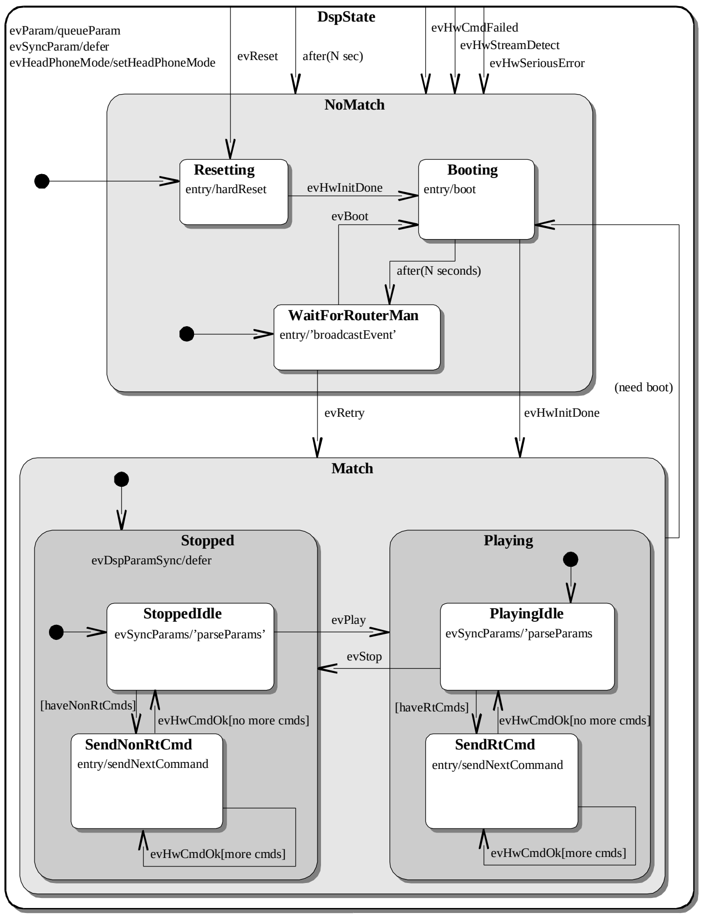


Figure 8.

As an example the state machine of the DSP is shown (figure 8). To be able to handle the complex and asynchronous state machines of the DSP's an extended version of the State design pattern is used. In the State design pattern all methods and attributes which are part of a certain state are realised in a class. State machines have been used for other parts of the software modules as well.

3.4 Font Selection

The product has so called On Screen Graphics (OSG). Because one watches video, dvd's and TV signals on a television, monitor or projector with quality from composite interlaced (low quality) to component progressive scan (higher quality) the OSG has to be readable on a simple TV in the living room at a distance of 3 meters or larger. This sets the actual lower limit of the font size to 14 points. Also a TV with 525/60 or 625/50 vertical lines interlaced and a colour resolution horizontal of 360 pixels plus the problems one has with a composite signal with colour edges differs considerably from what is seen on a high resolution computer monitor at a reading distance of 50 cm.

All these arguments had to be taken into account when designing the screen layouts for the product. Preferable the designer wants to use vectored fonts instead of having to deal with bit map ones. However there are large quality differences between the available anti−aliasing engines. Furthermore anti−aliasing involves a considerable amount of processor (graphics or normal) time. In many embedded application such as this one the processor selected is no match for a 2 GHz one.

3.5 Selection of the HTML Browser

Since both hardware and software of the product may change during its lifetime, the client wanted to have an on line help/info manual. The easiest, fastest and most flexible way to accommodate this was to use a HTML browser. After trying several possibilities we decided to modify and use the HTML browser which comes with embedded/Qt.

3.6 Testing on Different Locations

The hardware of the product was developed in the clients R/D department in Scotland while the software development was done at ArdisTech in the Netherlands. Once the hardware started to become available it was important for the electronic engineers which were responsible for the hardware development to have easy access to a proper test environments with proper test programs. In addition upon request of each of these engineers test program versions should be immediately available for them as if sitting next to them.

For that purpose the system shown in figure 9 was build.

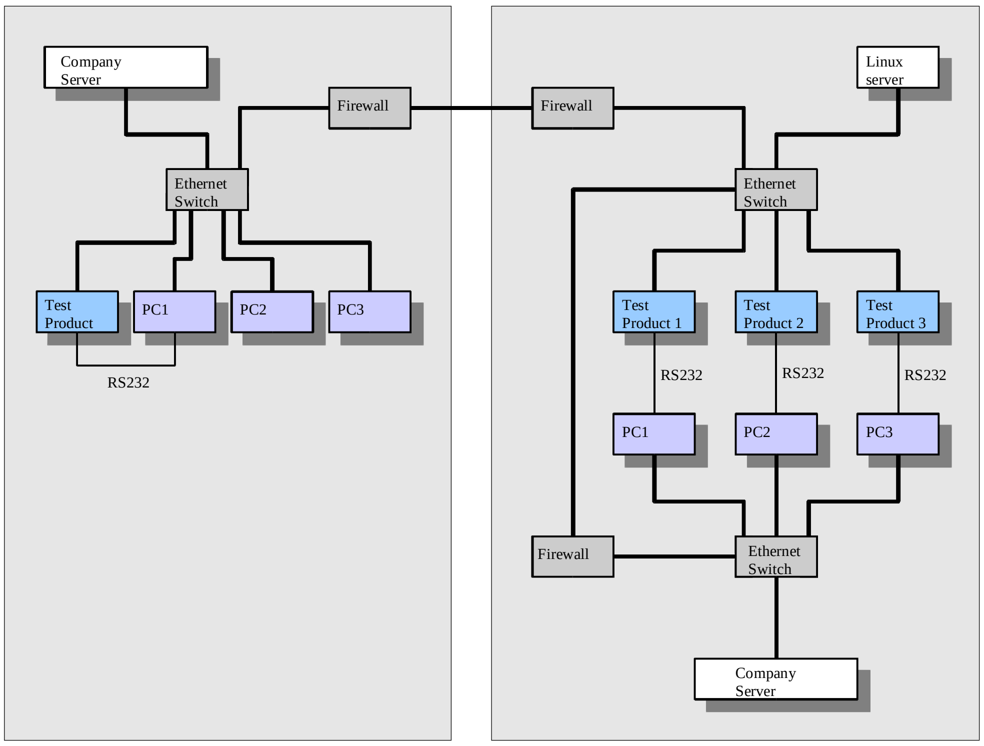


Figure 9.

Upon request test programs and new software versions were placed on the Linux server. Each engineer could then upload the program required on to hers/his test platform and communicate using Telnet on their own PC. This system has been proven to be very flexible and once all software including the file for the programmable electronic (FPGA's) was integrated test products could have all their software changed instantly.