Alef Null International
Signal Processing Division

User’s Manual

For DSP CARD 4

8 April, 1995

Kaj Wiik, OH6EH
email: Kaj.Wiik@hut.fi
Jarkko Vuori, OH2LNS
email: Jarkko.Vuori@hut.fi

$\aleph_0$

Overview

THIS manual describes various aspects about the $\aleph_0$ Alef Null DSP CARD 4. First we discuss the reasons for using digital signal processing (DSP) and what DSP really is. Then we present the DSP CARD 4, discuss what capabilities it has, and how it is implemented both in hardware and software. Then we reveal the current application software for DSP CARD 4. Then some hints on using, contructing and testing the DSP CARD 4 are shown, and finally the history of the DSP CARD 4 and some future speculation is given.

This document is for the DSP CARD 4 Rev. 2.3, which is the final release board. This series contains currently 50 boards. Please note that we are not Gurus and therefore this document contains errors and is far from complete.

We are extremely grateful to many radio amateurs all over the world who have been interested on our project. This have been a stimulating effect on our development of this little board. Special thanks we owe to *Rob Janssen PE1CHL*, *Henk Peek PA0HZP*, *Pawel Jalocha SP9VRC*, and *Reinhard Köhn DL7ARP* for pinpointing some nasty bugs in our hardware design, software and documentation.

This document has been made with Microsoft® WORD for Windows™ 2.0a in HP Vectra 486/33U environment. Making of this document has taken time of 92 hours. Pictures has been created with Micrografx® Designer 3.1 except schematics that have been designed with PADS schematics editor. All graphical plots has been created with MathWorks, Inc. MATLAB® 4.0a running on HP 9000 Series 730 workstation, real waveforms has been measured with HP 54600A scope connected via serial line to the HP Vectra, and spectrum measurements has been done with HP 8561 spectrum analyzer with HPIB interface to the PC.

[1] Alef Null consortium does not assume any liability arising out of the application or use of any product or circuit or software described herein; neither does it convey any license under the patent rights of others. DSP CARD 4 and other Alef Null products are not authorized for and should not be used within life support systems or nuclear facility applications without the specific written consent of Alef Null.

Life support systems are systems which (1) are intended for surgical implant into body, or (2) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user.

Examples of nuclear facility applications are applications in (1) a nuclear reactor, or (2) any device designed or used in connection with the handling, processing, packaging, preparation, utilization, fabricating, alloying, storing, or disposal of fissionable material or waste products thereof.

Alef Null reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Alef Null does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any licence under its patent rights or the rights of others.

There is no warranty. What you see is what you get.

Contents

Introduction

THE DSP CARD 4 is a product of the $\aleph_0$ Alef Null group. DSP CARD 4 is a *general purpose signal processing card*, dedicated mainly to the *amateur use*.

"*Art will always be art.*"

Johan Wolfgang von Goethe

Why to use digital signal processing

Digital Signal Processing (DSP) is the arithmetic processing of real-time signals that have been sampled and digitised in both time and amplitude.

Digital signal processing, DSP, offers many advantages over conventional analogue processing [33].

Stability	Unaffected by temperature or ageing
Repeatability	Not dependent on component tolerances
Power Consumption	Generally lower, especially CMOS devices
Cost	Lower system cost in many applications
Calibration	No calibration is required
Chip Count	Can be reduced in many applications
Algorithms	Many algorithms are difficult or impossible to implement in analogue technology

In amateur world, the main benefit of using DSP comes from the *easy modification*. A signal-processor based DSP hardware can be programmed for different circuits. For instance, the same hardware can generate all possible modems for all known amateur modulation standards (or in theory at least).

The DSP CARD 4 features

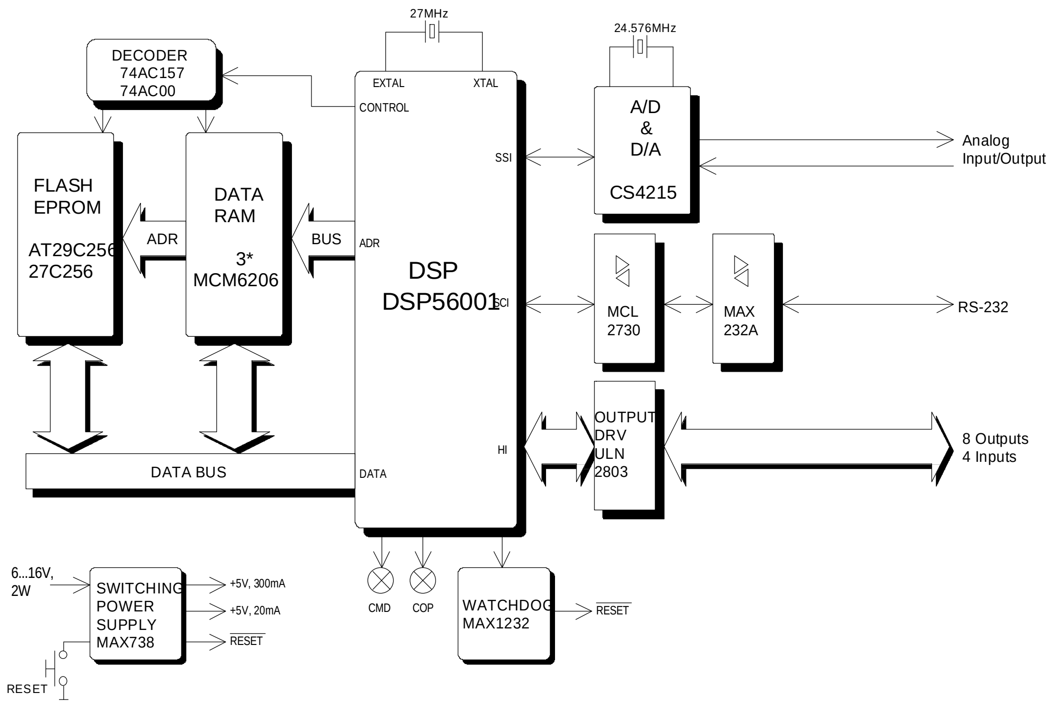

Figure 1: The block diagram of the digital signal processing card. DSP56001 has many I/O functions on the chip, which simplifies the overall hardware implementation.

The block diagram of the DSP CARD 4 is presented in the figure 1. DSP CARD 4 contains the Motorola 27 MHz DSP56001 digital signal processor with 13.5 MIPS processing power. The card has 32 kW (96 kB) high speed RAM, divided to two parts—8 kW for the program memory and 24 kW for the data memory. DSP56001 has internal 0.5 kW program memory and 0.5 kW data memory, so the total memory is 8.5 kW for program memory and 24.5 kW for data memory.

The DSP56001 has 24-bit word length, giving a dynamic range of 144 dB. Internally the accumulators are 56-bit, providing 336 dB dynamic range. The DSP56001 has a 32×24-bit bootstrap ROM for loading the internal program RAM via the host interface. Three functional units operate in parallel—the Data ALU, the Address ALU and the Program Controller. This allows a very high degree of internal parallelism that greatly enhances the overall processing power; an instruction prefetch, a 24×24-bit multiplication, a 56-bit addition, rounding two data moves and two address pointers updates can be executed in a single 74.1 ns instruction cycle. A high level of integration—with six on-chip resources, three on-chip MCU-style peripherals, a clock generator and seven buses—enable implementing the DSP system with very few external components [38,34].

DSP CARD 4 has the following interfaces

- host interface to the host computer via optoisolated high-speed RS-232 for downloading programs and high-speed data transfers
- line level analog input/output for direct audio or radio interface
- microphone preamplifier inputs and headphone outputs
- special monitor loudspeaker output

- eight general purpose high-current outputs and four protected general purpose inputs
- TTL-level time-multiplexed network interface for additional A/D or D/A converters

The whole system needs a single power supply, 6–16 VDC. Power supply is a switched mode type because of its greater efficiency and its ability to generate separate +5V supply for the isolated serial interface. Power consumption depends strongly on what the card is doing (i.e. what is the sample rate of the codec and what kind of program the codec is performing), but **typical** figures are[2]:

200 mW	Codec is turned off, processor is idling. Typical when DSP CARD 4 is doing nothing.
1 W	Codec is turned off, processor is maximally loaded.
1.2 W	Codec is turned on, sample rate is 8 kHz, processor is doing typical signal processing task. Typical power consumption.
1.7 W	Codec is on, sample rate is maximum (48 kHz), prosessor is maximally loaded (with external memory accesses). Absolute maximum power consumption.

Valid operating temperature range is 0 °C – 55 °C, storage temperature range is -55 °C – 125 °C. Weight of the card is about 150g.

DSP CARD 4 is contained on a single Eurocard (160mm×100mm), four layered, trough plated PCB. The card has a single EURO-64 connector, which contains all the needed connections.

What is Digital Signal Processing

Motivation

Up to about 20 years ago, the design of electronic circuits followed a bright vision—the function of the circuit was split into smaller units, each corresponding to already tested circuits or available components [24,25]. Easy to use 8-bit microcomputers solved many of the problems of digital circuit design, testing and maintenance, beside a further reduction of the cost of the hardware. The design time was shortened by introducing compact but reliable microprocessor and peripheral integrated circuits while test and/or debugging routines could be implemented at little if any additional cost in the final product.

Of course, continuous efforts were made to simplify the design and testing of the more difficult analog circuits. For example, a very successful innovation was the introduction of the operational amplifier: the design engineer could finally concentrate on the differential equations describing his/her problems instead of thinking how to bias his transistors.

The basic idea of Digital Signal Processing is to replace an electrical circuit with its mathematical equivalent and solving the equations describing the circuit numerically in real time. The latter can be done either with hardwired logic or with computers. Of course DSP circuits do not require any tuning or alignment, since the tolerances of the components are only limited by the accuracy of the numerical models used. Production line testing and maintenance is limited to check the operation of a microprocessor.

Besides cost advantages, DSP circuits allow the designer to use components that could hardly be implemented with analog electronic components: tuned circuits with arbitrary, even infinite but stable values of Q, easily variable component values during circuit operation

[2]There can be variations in power consumption about -0.1W – +0.4W depending on the exact types of the components used.

(adaptive signal processing) or very complex algorithms that would require a very large number of high accuracy analog components (Fast Fourier Transform).

The main drawback of DSP is a limited bandwidth of all the signals present in the circuit: all the computations usually have to be performed at a rate of at least twice the signal bandwidth. DSP circuits are therefore limited to IF, audio and video applications. DSP circuits also require very fast logic or powerful microprocessors to provide usable results. This is the reason why DSP has only become popular with advanced signal processor chips.

Some DSP circuit examples

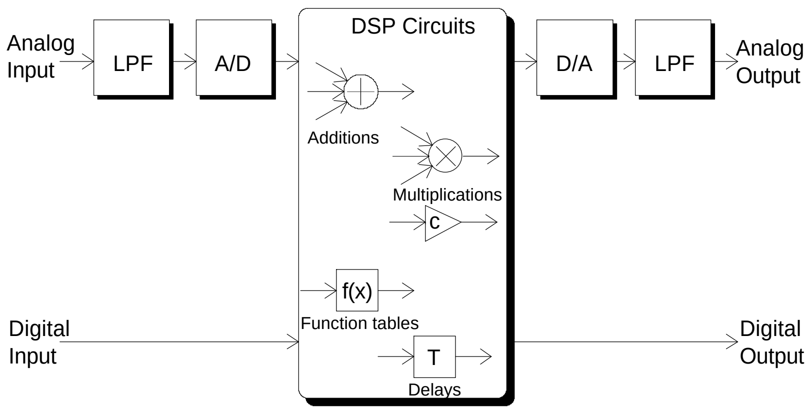

Figure 2: DSP circuit principles.

A DSP circuit can generally be split into an interface part including A/D and D/A converters and related circuits to interface to the analog ‘environment’ and a digital part performing some numerical operations on the digitised analog signals, figure 2.

The input analog signal is first band limited essentially to prevent interferences called aliasing: a finite number of samples taken by the following sample-and-hold stage can only represent a limited bandwidth signal. The sample-and-hold circuit and the following A/D converter are triggered periodically at regular intervals called the sampling period or its inverse, the sampling frequency. The latter should be at least twice the signal bandwidth: practical applications require a sampling frequency 2.5 to 3 times the signal bandwidth due to circuit imperfections.

The DSP circuit designed has to replace all those analog components—amplifiers, energy storing components (capacitors and inductors) and non-linear components (rectifiers, balanced mixers)—with mathematical algorithms that will be computed on each input signal sample coming from A/D converter and will provide a regular stream of output signal samples to the D/A converter.

Engineers usually describe the operation of linear analog circuits using linear differential equations. A DSP circuit will be a good replacement for an analog circuit if its operation can be described by a similar equation. Since DSP circuits work on uniform stream of samples and not on continuous signals, there is no way to compute derivatives nor to obtain a differential equation. One can however compute differences between successive samples: if the sampling period is sufficiently short these can be considered a good approximation for derivatives. Let us consider as a simple example LC resonator circuit in analog and DSP world.

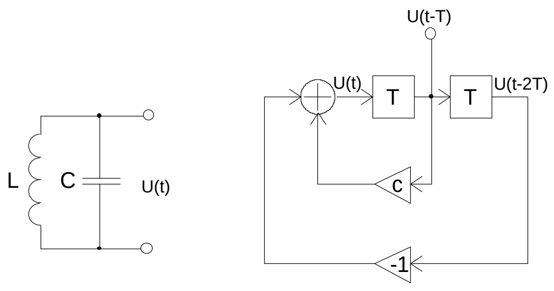

Figure 3: An LC tuned circuit and its DSP equivalent.

A simple LC-tuned resonator circuit, figure 3, contains two energy storing components and is described by a second order differential equation:

$$\begin{cases} I_c = C\dfrac{dU}{dt} \\ U = L\dfrac{dI_L}{dt} \\ I_c = -I_L \end{cases} \Rightarrow U(t) = -L\frac{d\left(C\dfrac{dU(t)}{dt}\right)}{dt} \Leftrightarrow \frac{d^2U(t)}{dt^2} + \frac{U(t)}{LC} = 0 .$$

The resonant frequency is well known to be $f = \dfrac{1}{2\pi\sqrt{LC}}$.

Derivative can be approximated by Euler's method $\dfrac{dx(t)}{dt} = \dfrac{x(t+T)-x(t)}{T}$, where T is the time between two successive samples:

$$\begin{aligned} &\frac{\dfrac{U(t+T)-U(t)}{T} - \dfrac{U(t)-U(t-T)}{T}}{T} + \frac{U(t)}{LC} = 0 \| T^2 \\ &\Leftrightarrow U(t+T) - 2U(t) + U(t-T) + \frac{T^2U(t)}{LC} = 0 \qquad . \\ &\Leftrightarrow U(t+T) = \frac{2LC-T^2}{LC}U(t) - U(t-T) \end{aligned}$$

From the previous equation we can see that the difference equation describing the DSP circuit can be re-arranged to become similar to the differential equation of the analog resonator: three successive samples are required to compute the approximation for the second derivative. When we assume $c = \dfrac{2LC-T^2}{LC}$, we get the DSP block diagram form shown in figure 3. The resonant frequency of this DSP implementation of LC resonant circuit is given by $f = \dfrac{1}{2\pi\sqrt{\dfrac{-T^2}{c-2}}}$.

Note that there are two energy storing components in the analog realisation of a resonator. In the digital realisation, there are two delay lines that are equivalent of energy storing components in analog realisation.

The digital signal processing block diagram in figure 3 can be programmed for Motorola DSP56001 signal as

```
; wait for one sample
loop   waitblk   r2,buflen,1

; then generate the sinewave
       move      #c,x1

       move      x:<t1,x0
       mpy       x0,x1,a      x:<t2,y0
       sub       y0,a         x0,x:<t2
       move      a,x:<t1

; and output the generated sample
       move      a,y:(r2)+n2

       jmp       <loop
```

The analog LC circuit shown and its equation corresponds to the ideal case of lossless resonator. It is well known that such a resonator can not build in practice, since there are always some loss mechanisms. On the other hand, a lossless resonator (with infinite Q) can readily be built as a (stable!) DSP circuit. Thus the DSP implementation of this LC resonator can be used as a sine wave generator. The output from the DSP CARD 4 when the previously shown program is running is shown in figure 4.

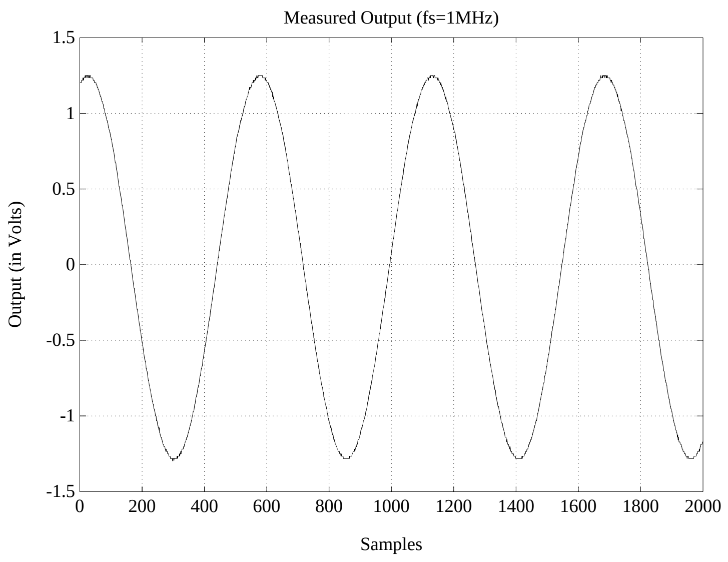

Figure 4: Output signal from the LC-resonator.

Is there a need for a Theory for DSP

A good theory should make it possible to understand how a system like the one in figure 2 works and how it should be designed. It seems clear that a sampled system would behave as a continuous-time system if the sampling period were sufficiently small. This is certainly true under very reasonable assumptions. Is there then any need for a special theory for DSP?

Some examples will be used to show that the system in figure 2 cannot be *fully* understood in terms of the theory of time-invariant, linear systems.

Time dependence

When a signal is fed through a DSP system, it encounters different delay depending on the moment of sampling. If the input is delayed, then the output is delayed by the same amount only if the delay is a multiple of the sampling period.

Higher-order harmonics

It is well known that a sinusoidal input signal applied to a stable, linear, time-invariant, continuous-time system will—after a transient—give signals in the system that are sinusoidal with the same frequency as that of the input. If for example sinusoidal signal with a frequency of 5.1 Hz is fed to the DSP system with a sampling rate 10 Hz, there will be an additional 0.1 Hz component on the output of the system. This cannot be explained in terms of linear, time-invariant systems.

Deadbeat control

It is possible to use special control strategies suitable only to DSP systems to obtain better results in performance than ordinary linear feedback systems can achieve. One of these control strategies is *deadbeat* control [17].

How Theory Developed

During and after World War II, a lot of activity was devoted to analysis of radar systems. These systems are naturally sampled because a position measurement is obtained once per antenna revolution. Since transform theory had been so useful for continuous-time systems, it was natural to try to develop a similar theory for sampled systems. The first steps in this direction were taken by Hurewicz [21]. He introduced the transform of a sequence {f(kT)}, defined by

$$\mathrm{Z}\{f(kT)\} = \sum_{k=0}^{\infty} z^{-k} f(kT)$$

The transform was later defined as the *z-transform* by Ragazzini and Zadeh [22].

Toward the end of fifties, the *z*-transform approach to sampled systems had matured, and several textbooks appeared. This theory, which was patterned after the theory of linear, time-invariant, continuous-time systems, gave good tools for analysis and synthesis of sampled systems. A few modifications had to be made because of the time-varying nature of sampled systems.

Where to learn DSP techniques

Does anyone know of a construction on audio signal filtering, equalisation, compression and more with a DSP circuit Texas Instruments, Intel etc. ?

If you have suggestions on references (books , source codes) i would be very thankfull! All expances will be paid (of course).

I woluld also like to get in touch with anyone that has knowledge on this field and is willing to teach me some of the secrets of DSP!

The need for information about DSP is today extremely intense as may seen from the above packet radio message. It is a very good thing that many good books on DSP have been published since 1970’s. There are both simple tutorials and advanced textbooks. For an example Ludeman’s simple book [1] is a good introduction to DSP and Oppenheim’s lucid and penetrating treatment of basic signals and systems [2] is highly recommended. Quite a new book from Proakis [15] is a very good textbook on DSP, oriented mainly for digital communication engineers. For intermediate level Jackson’s presentation [5] and the legendary Oppenheim and Schafer’s book [3] are good references. An excellent book on simple adaptive filters is Alexander’s one [11]. For advanced level [6,7,8,10] could be useful to read. Of course some of the IEEE serials (e.g. IEEE TRANS. ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, IEEE TRANS. ON AEROSPACE, IEEE TRANS. ON COMMUNICATIONS, IEEE PROCEEDINGS and IEEE SIGNAL PROCESSING MAGAZINE) and the IEE serials (e.g. IEE PROCEEDINGS-I, COMMUNICATIONS, SPEECH AND VISION, IEE PROCEEDINGS-F, COMMUNICATION, RADAR & SIGNAL PROCESSING) might be useful.

Of course DSP itself isn't quite useful. In amateur world DSP applications to the communication technology are the most interesting. Carlson's book [12] is the 'number one' introduction to the communications technology. It also contains a very complete reference section for the more interested reader. One quite good practical oriented book is [9]. Lee's and Messerschmitt's book [13] tells a little more about digital communications at quite tutorial level. Bingham's book [14] has many practical hints for modem designer, mainly for telephone modems. More theoretically oriented books are [15,16]. Of course the IEEE TRANS. ON COMMUNICATIONS, IEEE JOURNAL OF COMMUNICATIONS and IEE PROCEEDINGS-F, COMMUNICATION, RADAR & SIGNAL PROCESSING are good sources of advanced information.

There are also very fascinating DSP applications in the field of control theory. Basic book, dedicated mainly for prevailing design is Dorf's textbook [18]. Entirely devoted to digital control system design is Åström's elegant book [17]. Adaptive control is studied in his book for adaptive control [19]. For advanced control, the identification of the process is an absolute prerequisite. One good text about this field is Ljung's book [20].

Hardware

"Tietä pitkin tien on vanki,
vapaa vain on umpihanki."
Aaro Hellaakoski

The total chip count of the DSP CARD 4 is only twelve chips. This is possible because the DSP56001 has so many interface functions on it, and we used a special A/D and D/A converter chip that contained all the needed support functions for analog interfacing.

DSP and memory

The DSP56001 interface is very straightforward [3] [39]. The crystal oscillator circuit is a very typical fundamental frequency Pierce type oscillator. If the crystal is not a fundamental type, C1 and L1 are needed to suppress the fundamental frequency.

	Internal	*External*
P	0000–01FF	0200–1FFF
X	0000–00FF	0100–1FFF
Y	0000–00FF	0100–3FFF

Table 1: DSP CARD 4 memory map. All addresses are given in hexadecimal.

We used three 32K×8 type 25 ns high-speed static RAMs because they are cheap and easy to obtain (cache-memories in PCs often have this type of chips in them). The DSP and memory interface is very simple. All we need is a NAND gate (A2)[4] for enabling the RAM when DSP56001 wants to access program space (PS) or data space (DS), and multiplexer chip (A3) for handling memory space partitioning and write enable signal prolongation. Note that the data space is partitioned to 8 kW X-partition and 16 kW Y-partition, and program space is a

[3]See the schematics at the end of this document.

[4]The character and number in paranthesis points to the part id in the schematics.

single 8 kW cluster. This kind of memory partition was selected because DSP-programs tends to be quite small and in some applications there is a need for a larger data memory space. For memory map description, see the table 1.

The access time t_{AA} of the RAM is 25 ns, and the DSP demand for access time from the valid address is specified by parameter #130[5] as 42 ns (assuming 27 MHz clock). Signals derived from addresses are fed through two NAND gates that have maximum delay t_{PLH} as 8 ns. Therefore there is 42 ns - 25 ns - 2*8 ns = 1 ns reserve for reliable working or for some experiments with higher operating frequencies[6].

The DSP parameter #125 = 12.5 ns does not meet SRAM t_{DW} = 15 ns requirement, therefore the WR signal is directed via the multiplexer chip (A3) to lengthen WR signal.

DSP CARD 4 boots from the on-board EPROM (A7). Immediately after reset, DSP56001 fills its internal memory from the external EPROM and starts executing the program there. The access time of the external EPROM can be about 1 µs because there are automatically inserted 15 wait-states on the EPROM reading. This EPROM can be an ordinary type 256 kB device, or a special FLASH EPROM that can be erased and programmed without removing the chip from the board. This Atmel Corp. AT29C256 FLASH EPROM we use needs no external programming voltages [48], therefore FLASH or an ordinary type EPROM can be used on the same socket. This AT29C256 can also be programmed in 64 byte blocks that makes the partial updating of the FLASH possible.

Stereo A/D and D/A converter

We selected Crystal Semiconductor’s 16-bit, 48 kHz, Multimedia Audio Codec CS4215 [49,50] for the DSP CARD 4. CS4215 is very easy to use; it has both dual A/D and D/A converters, their anti-aliasing filters, references and amplifiers for microphone, headphones, line input, line output and for special monitoring loudspeaker in a single 44-pin PLCC package. The sampling rate can be selected to one of the following: 48, 32, 27.42875, 16, 9.6 and 8 kHz. CS4215 can be set to special power-down mode in order to reduce power consumption notably. Multiple CS4215 devices can also be connected to the same serial line interface with simple time multiplexing scheme.

Interface to DSP56001 is made by synchronous interface with 8 or 16-bit words and continuous clock with a separate frame synchronisation signal. After reset the CS4215 is sending the shift clock and framing pulses and is in 8 bit mode, after initialising it the DSP56001 is the source of shift clock and framing pulses and the interface is set to 16-bit mode. The interface to DSP56001 contains only two data lines, one clock line, one framing pulse line, one line for data/control mode driving and one line for power down mode setting. Because there is also a possibility to use more than one codec together, therefore all the necessary codec interface signals for multiple codec configuration are fed to the external connector.

The power supply for CS4215 is given via LC low-pass filters because the power-supply noise rejection (40 dB) is quite poor on the CS4215. The 3 dB corner frequency of those LC filters (L3 C30) and (L4 C32) is about 16 kHz. These two-pole low-pass filters have attenuation together about 2×-12 dB/octave. The main source of interfering noise if the switching power supply that has switching frequency 165 kHz. The maximum frequency that the codec can handle is 24 kHz, so the attenuation at this frequency must be enough to ensure 80 dB dynamic range the codec has. LC-filters attenuate -82 dB the 165 kHz signal at the frequency 24 kHz, and the codec has 40 dB internal power supply rejection, hence the total attenuation is 123 dB—enough to guarantee the 80 dB dynamic range of the codec.

CS4215 outputs are short circuit protected, so there is only simple one-pole RC low-pass anti-aliasing filters are needed on the inputs and outputs. The nominal voltage level on the line inputs and outputs are $1V_{rms}$ ($2.8V_{pp}$).

[5]Parameter numbers as in [36].

[6]Please note that when working at very high frequencies on the standard 27 MHz chip, the DSP56001 dissipates more power, so the DSP chip may need some additional cooling.

Power supply

Power supply generates two separate +5V outputs. First output, with 500 mA output capability, is for all the codecs, processors and whatever. Second output, with 30 mA output capability, is for the isolated serial port interface only. The power supply is switched-mode supply because of its higher efficiency and the easiness with which the isolated outputs can be made. There are dangers in using switching-mode supplies, but we thought that because the currents are small and the switching frequency is high, those typical switcher power supply noises can be filtered out.

The power supply is implemented with MAXIM MAX738A +5V Step-Down Current-Mode PWM Regulator [47]. This regulator has a high switching frequency (165 kHz), good efficiency (87%), and because of current-mode type feedback has a rapid response to load changes. This chip is also extremely easy to use, there are neither inductor nor capacitor design at all! The inductor should be in range 33µH–100µH and the capasitor is 100µF. Only the size of the toroid must be checked if other than those MAXIM recommends will be used.

The magnetic size of the toroid must be large enough to store the maximum energy peaks transferred through the toroid [46]. From Ampère's law we get the magnetic flux density (in Teslas, T) of the toroid as

$$B = \frac{\mu_r \mu_0 N I}{l},$$

where μ_r is the relative permittivity of the matter (dimensionless), μ_0 is the permittivity of vacuum ($4\pi \cdot 10^{-7}\ \mathrm{H/m}$), *N* is the number of turns on the toroid, *I* is the current through the toroid and *l* is length of the toroid.

Magnetic flux (in Webers, Wb, Tm^2) can be stated as

$$\Phi = BAN,$$

where *A* is the cross sectional area of the toroid. Using the well known definition for the inductance (in Henries, H, $\mathrm{Wb/m}^2$) and the previous result, we get

$$L = \frac{\Phi}{I} = \frac{BAN}{I} = \frac{\mu_r \mu_0 N^2 A}{l}.$$

A properly selected inductor must provide the right value of *L* without exceeding the maximum magnetic flux density for the core used. In other words, the core must not 'saturate' under conditions of peak current I_p. From the Ampère's law and the previous result, we get formula for the volume of the toroid needed for the given peak current as

$$V_c = Al = \frac{I_p{}^2 L \mu_r \mu_0}{B^2}.$$

We wanted to use Amidon FT50-61 toroid because of its good availability. Its main characteristics are:

- B = 2350G = 235 mT (one Gauss is 0.1 mTesla)
- $\mu_r = 125$
- V=0.402 cm^3
- A_L=68 mH/1000 turns

The maximum peak current is I_p = 700 mA and the inductance is L = 40µH. Therefore

$$V_c = \frac{(700 \cdot 10^{-3}\mathrm{A})^2 \cdot (40 \cdot 10^{-6}\mathrm{H}) \cdot 125 \cdot (4\pi \cdot 10^{-7}\mathrm{H/m})}{(235 \cdot 10^{-3}\mathrm{T})^2} = 0.0558\mathrm{cm}^3.$$

Because this is smaller than the volume of the toroid used, it is not driven into saturation—it can store and give the necessary amount of energy during one cycle of operation.

The number of turns in the toroid can be calculated from the equation

$$N = 1000 \times \sqrt{\frac{L \text{ (in mH)}}{A_L}}\,.$$

In case of FT50-61 toroid the previous equation gives 25 turns. The isolated output is simply taken from the secondary winding of the inductor (flyback mode) and it is not very well regulated. The maximum output current that can be taken from this winding is about 10% of the main output current. The secondary winding must be connected so that the energy is delivered to the secondary output when the PWM switch turns off. During the off-time, the voltage across the primary winding is regulated by the feedback loop, yielding a constant Volts/Turns ratio. The number of turns for secondary voltage is the same as in primary winding because the output voltages and the diodes are same for both outputs.

There is an additional transient suppressor (V5) in order to protect the board from the possible over voltage and destructive noise spikes. The filter (C14,L2,C15,C16) prevents the supply noise to get out of the board.

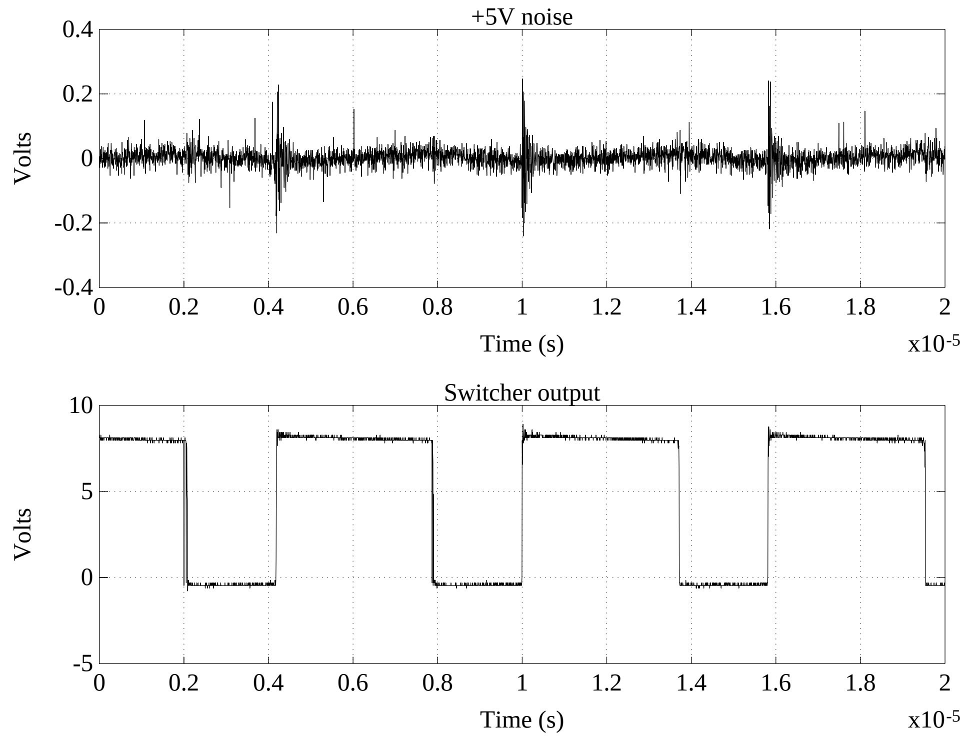

Figure 5: Measured switcher waveforms.

The outputs of the switching mode power supply are shown in the figure 5 when the input voltage to the board is 9V and there is sine oscillator program running. The power consumption is about 1W, the switching frequency is 172 kHz and the duty cycle is 64%. The upper part of the figure shows the measured waveform at the cathode of V3. The lower part of the figure shown is measured signal at the pin 14 of A2.

Watchdog and power-on reset

Because we thought that this DSP CARD 4 board might be used on places where there are no supervision, the reliability of the hardware and software must be good. One common way to ensure the operation of the processor is to use watchdog that needs some kind of regular attention from the processor. Watchdog resets the processor when it has not been responding for a given period of time.

We decided to use the MAXIM MAX1232 Microprocessor Monitor chip. It is the cheapest device and has many second source suppliers. It has all the necessary functions we needed in this application: good power-on reset, supply voltage monitor, reset switch input and watchdog function. DSP56001 toggles on of its output periodically (this output is also directed to the green ‘heart-beat’ led for visual monitoring), if there is no updates in 1.2s, the MAX1232 gives 250 ms reset pulse to the DSP. Also if the +5 supply goes under -10%, MAX1232 will issue system reset to the board.

Interfaces

High-speed serial interface

High-speed serial interface is implemented with the MAXIM MAX232A High-Speed +5 Powered RS232 Drivers/Receivers chip. This chip is an upgrade for the older MAX232 chip, it has higher maximum speed (116 kbit/s), and needs smaller capacitors (100nF). Linear Technology LT1180 or MAXIM MAX202 can be used, if there is not need for the maximum 116 kbit/s data transfer rate. Of course the older MAX232 can be used if those small 100 nF capacitors are replaced by larger 10μF capacitors and slower data rate is accepted.

The isolation is done with the HP HCPL-2730 Dual Low Input Current, High Gain Optocouplers and the isolated power supply. We selected HCPL-2730 because of its good availability and it has many second source suppliers.

This interface is mainly asynchronous, but the SCLK signal from the DSP56001 is fed to the output and can be programmed to give shift clock for the interface. With synchronous mode the maximum data transfer rate is 3.375 Mbit/s (27 MHz clock assumed).

For higher communication speeds the MAX232A can be removed and the signals can be jumpered directly to the output where some additional driver/receiver (e.g. RS-422 type) can be installed.

General purpose I/O

General purpose outputs are buffered by octal ULN2803A device. It can sink 500 mA current and has protection diodes for inductive loads. The diodes can be connected with CLAMP signal to the V+ source of the target.

General purpose inputs are protected with simple resistor and diode network for over voltage conditions. The diode used, 1N4148, can sustain continuous 400 mA DC forward current, thus in a theory the input protection can handle as large as $U=RI=2200\ \Omega \times 400\ \text{mA}=880\ \text{V}$.

Analog interface

Analog interface is extremely simple because the Codec chip CS4215 contains all the necessary amplifiers and protection networks. Only simple antialiasing networks are needed. It must be noted that the input has nearly zero impedance at the high frequencies, therefore there may be problems with some signal sources. This can be resolved by placing an additional amplifier between the signal source and the DSP CARD 4.

Connections

Every connection to the DSP CARD 4 is made via the 64-pin Euro-1 connector on the board edge.

Power	6V–16VDC, power consumption is typically 1W (depending on the mode of the DSP56001 and CS4215), and the maximum power consumption is about 2W.
RS-232	This is a standard RS-232D interface. Can be used on the maximum speed of PC (116 kbit/s). With simple modifications it gives high-speed TTL-level interface.
Additional Codecs	TTL-level signals for connecting more Crystal CS4215 codecs, or other devices compatible to the DSP56001 SSI interface, to the DSP CARD 4.
Discrete outputs	Eight general purpose outputs capable of sinking 500 mA. Provisions for inductive load driving.
Discrete inputs	Four general purpose inputs. Protected for over voltage (max. 48 VDC).
Analog	Line level inputs and outputs (2.8 V_{pp}). There are also microphone level inputs, headphone outputs (max. 8 Ω) and monitor loudspeaker output (max. 32 Ω).

Some power supply signals are also available from the card. Logic +5V (for additional logic circuits), isolated +5V and isolated boosted +9V and -9V (for additional communication driver/receivers and signal setting).

There is also available a separate reset switch signal (BRS) that can be grounded in order to get reset signal.

Parallel port recommendation

For radio amateur communication purposes, the usage of the output parallel port is recommended to be as:

O0 - PTT trx L	drives transmitter
O1 - DOWN trx L	pulse to move receive freq DOWN
O2 - UP trx L	pulse to move receive freq UP
O3 - CAT trx L	can control trx using some serial protocol
O4 - PTT trx R	drives transmitter
O5 - DOWN trx R	pulse to move receive freq DOWN
O6 - UP trx R	pulse to move receive freq UP
O7 - CAT trx R	can control trx using some serial protocol

Input parallel port usage is recommened as:

I0 - RTS trx L	request to transmit
I1 - CAT trx L	CAT input from trx for feedback
I2 - RTS trx R	request to transmit
I3 - CAT trx R	CAT input from trx for feedback

In these, L and R refer to the channel of the CODEC to which thetransceiver is connected. The RTS inputs are thought to be used when a microphone is connected to the DSP card which in turn drives the transceiver via the analog outputs and PTT lines. E.g. in a speech processing application where you want the DSP to be in control of the transmitter for things like VOX, but also like to have a manual transmit capability. Alternatively these inputs could be used for some simple mode selection using a switch.

This layout assumes a 2-transceiver configuration is practical. Should this not be the case (or when you want to use the 2 channels to generate the I and Q parts of the signal) only the L mappings can be used and the remaining signals are available for other use.

Wiring table for TNC-like DSP CARD 4 applications

For each connector, the pins on the DSP CARD 4 euro-connector to be used are given below:

EXTERNAL CONN		COMMENT	DSP CARD 4 CONN	
Power		**concentric power conn**		
1	GND		DGND	29,30,31,32
2	+POWER		POWER	62,63
RS232		**DB9 female**		
1	CD		DCD	58
2	RXD		TX	57
3	TXD		RX	25
4	DTR	n/c		
5	GND		-IO	28
6	DSR	via 4k7	+12	59
7	RTS		RTS	26
8	CTS	via 4k7	+12	59
9	RI	via 4k7	-12	27
Microphone LEFT				

	MIC		MINL	22
	MGND		AGND	24
	PTT	4k7 pullup	I0	2
Microphone RIGHT				
	MIC		MINR	23
	MGND		AGND	24
	PTT	4k7 pullup	I2	4
Headphone		**3.5mm stereo**		
tip	LEFT		HEADL	18
ring	RIGHT		HEADR	19
base	GND		HEADC	50,51
Transceiver LEFT		**8 DIN 270deg**	**compatible with 5pin 180deg TNC standard**	
1	TXaudio	via divider	LOUTL	17
2	GND		AGND	21,53
3	PTT		O0	34
4	RXaudio	via divider	LINL	54
5	CAT→DSP4	opt pullup	I1	3
6	DOWN	opt pullup	O1	35
7	UP	opt pullup	O2	36
8	CAT→TRX	opt pullup	O3	37
Transceiver RIGHT		**8 DIN 270deg**	**compatible with 5pin 180deg TNC standard**	
1	TXaudio	via divider	LOUTR	49
2	GND		AGND	21,53
3	PTT		O4	38
4	RXaudio	via divider	LOUTR	55
5	CAT→DSP4	opt pullup	I3	5
6	DOWN	opt pullup	O5	39
7	UP	opt pullup	O6	40
8	CAT→TRX	opt pullup	O7	41

Firmware

DSP CARD 4 has one EPROM chip that contains a simple monitor program, called LEONID, which boots immediately after reset. The role of this monitor is to ease the use of DSP CARD 4. There are functions for downloading programs, interfacing to the codec chip and serial line, handling transmitter on/off protocol, and treating the lowest level of AX25 protocol.

There are also some support software available for the PC to load programs to the DSP CARD 4 and to maintain EPROM software library.

Managing software with DSP CARD 4

The EPROM device on the DSP CARD 4 contains a special boot program to check the functionality of the card and to start-up all the necessary peripheral devices. This boot program also loads a monitor program from the slow EPROM to the high-speed RAM program memory to give some support functions (serial port I/O handling, Codec handling and real-time determination) for user's application programs.

It is possible to arrange things such that after all the essential initialisations are done, the boot program loads some predetermined program which is in the EPROM device and starts executing it. With this capability, the DSP CARD 4 can be operated without any host computer connected to it. It also possible to organise the system in a such way that after reset a different program starts from the pool of programs in a robin round fashion. This is useful when there is no host computer available and there are more than one DSP program that one is willing to use.

It is also possible to load a program from the host computer at any time. This is very useful when debugging programs—there is no need to program an EPROM to test how the program works on the real platform.

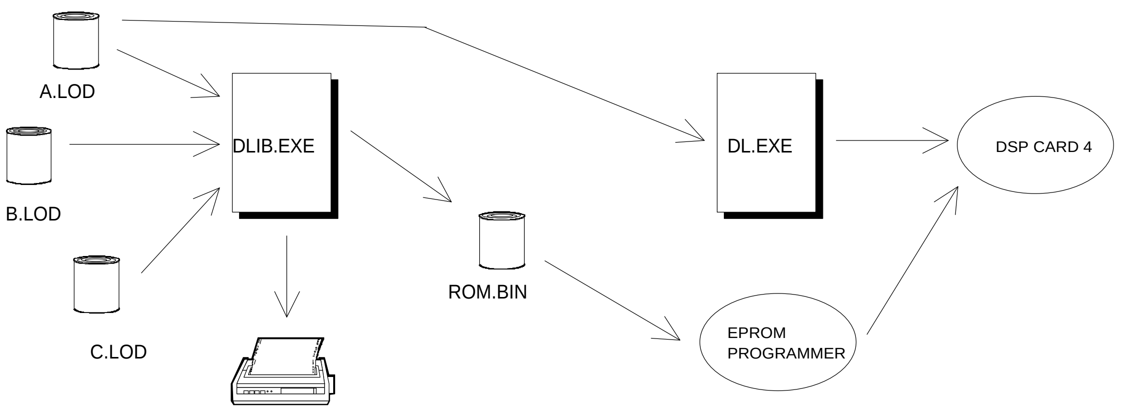

Figure 6: Software maintenance cycle. Library manager (`DLIB`) can combine several linker output files (`*.LOD`) together. The output of the library manager, EPROM image file (`ROM.BIN`), can be sent to the EPROM programmer. The loader (`DL`) can download linker output files to the DSP CARD 4 directly.

Tasks in order to produce loadable code are shown in Figure 6. Library manager combines the DSP56000 linker output files to the EPROM image file. This image can be programmed to an EPROM with EPROM programmer or sent directly via the serial line to the DSP CARD 4.

Start-up sequence of DSP CARD 4

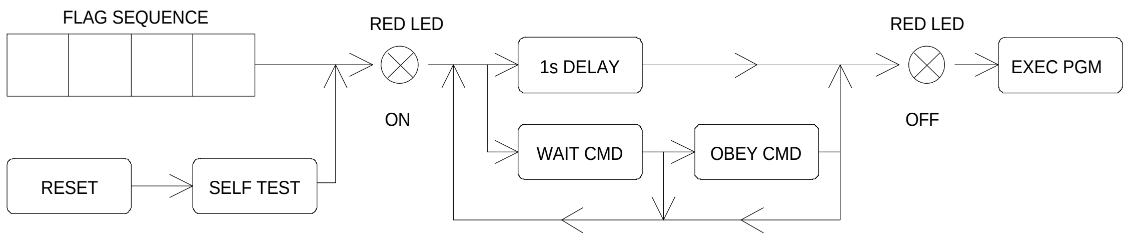

Figure 7: Start-up sequence of the DSP CARD 4. Flag sequence is a special character sequence detected on the serial input. Reset can be power-up reset, watchdog reset or user reset via reset button.

The start-up sequence of the DSP CARD 4 is shown in Figure 7. Directly after reset, the internal bootstrap ROM of the DSP56001 commands it to load 512 words from the EPROM to the internal P memory and start the program execution. This loaded program initialises all the needed peripherals (serial line, timer, i/o-pins) and then performs self checking functions. If there are some errors detected, these will be shown with the two leds on the board. If no hardware errors are found, then resident part of the monitor program is loaded to the upper external P memory. After that, the interrupts are enabled (green led starts blinking) and the red led will be set on in order to mark that the monitor is waiting for a command. The DSP waits 1s for a command from the serial line. If there is nothing from the serial line the DSP checks if this is the first time reset (this is determined by checking if a predetermined memory locations contains a valid checkword). If it is a first time reset, and there are programs in the EPROM to be loaded and executed, the DSP loads a program, and starts the execution of that program. If there are no programs available in the EPROM, then the DSP will be placed to energy saving sleep mode. If this is not the first time reset, the current program slot pointer is incremented and the next program in the EPROM will be loaded and executed. The red led is always on only when the main program execution is in LEONID monitor, e.g. when there is a command pending in the DSP CARD 4. The red led starts blinking when theere is a data communication error detected (interrupted data transmission, for example) between the host computer and DSP CARD 4.

The DSP CARD 4 can be placed to reset state by sending a special four byte flag sequence (its value is Int(100000000π)=314159265, and the sequence is therefore 12b9b0a1 in hexadecimal). This sequence places the DSP56001 in stop processing state, and the watchdog device reset the system after its time-out delay.

ROM library

There can be several programs in the EPROM that can be loaded when needed. There is also a special boot program to be loaded just after the reset condition. The boot program must be in a predetermined place in the memory space (DSP56001 hardware requirement). There is also a special 256 bytes long directory entry at the beginning of the EPROM to locate every program residing in the EPROM. At the end of the EPROM there is a two byte word containing a CRC-check word for checking the validity of the EPROM.

15:	LOAD ADR	LENGTH	DESCRIPTION	DATE	PROGRAM 15
	•	•	•	•	
2:	LOAD ADR	LENGTH	DESCRIPTION	DATE	PROGRAM 2
1:	LOAD ADR	LENGTH	DESCRIPTION	DATE	PROGRAM 1
	ROM ID	AUTO BOOT ID	DESCRIPTION	DATE	ROM info
	2	2	10	2	

Figure 8: EPROM directory structure. The are 15 entries for the programs. ROM ID is a word for checking EPROM type, AUTO BOOT ID is the number of program to be loaded and executed directly after start-up. DESCRIPTION is an ASCII string describing the program, DATE is the creation date of the program in MSDOS format. The number on the lower side of the table indicates the length on the fields in bytes.

Directory structure is shown in figure 8. This directory can contain at maximum 15 programs. The system can detect that a program is present in the slot when the LOAD ADR field is non zero. LOAD ADR and LENGTH fields are used to locate the file in the EPROM and help EPROM management when new entries are created. DESCRIPTION and DATE fields are just notes for the user describing the program. The first entry is a special entry for boot program. ROM ID is used for checking that the given object is a valid EPROM library (the check word contains programmers initials ‘JV’in ASCII, 564a in hexadecimal).

BLOCK ID	ADDRESS	LENGTH	DATA
1	2	2	N

Figure 9: The block structure of data in the EPROM library. The numbers are the length of fields in bytes.

Because DSP56001 has three types of memory (P,X,Y) and there can be large gaps in the memories to be initialised at the beginning, the data to be loaded is placed to blocks in the EPROM library. The structure of the block is given in figure 9. The block begins with a byte that describes to what type of memory (00=X, 01=P, 10=Y) the data will be loaded. ADDRESS contains the load address for the data, and LENGTH determines the length of the load block. Finally comes the data in three byte chunks (24 bit DSP56001 word needs three 8 bit words).

End of the program is noted by a special block that has a zero in the LENGTH field.

Interface protocol

DSP CARD 4 can be connected via a serial line to the host computer. The communication between DSP CARD 4 and the host computer is standardised with a special protocol described below.

After reset, which can be requested from the serial line with a special sequence, the DSP CARD 4 will give an ACK character (04 in hexadecimal) at the speed of 19200 bit/s on the serial line. This can be used to notify the host computer that the DSP CARD 4 has started from the reset state. After this ACK character two eight-bit characters will follow which contains the version date of the monitor program in MS-DOS date format. This version number can be usefull when trying to find out what monitor version the board have. After those three characters the appropriate command can be given to the DSP CARD 4 (currently there is only one command available , namely ‘load and go’, 03 in hexadecimal). If DSP CARD 4 receives a valid command it responses by sending ACK to the serial line. Otherwise there is no response.

After successfully resetting and offering the command, the host computer can begin to do whatever is necessary to complete the given command. The special ‘load and go’ command needs the program data to be loaded and then executed.

FLAG	PAC-ET ID	ADDRESS	LENGTH	CRC	DATA	CRC
1	1	2	2	2	N	2

Figure 10: Communication packet in the special protocol. The numbers are the length of fields in bytes.

Program data is transferred using a communication block described in figure 10. Packet begins by a special flag character (7e in hexadecimal) that can be used to indicate the start of the packet. Immediately after it comes same kind of block that is used in the EPROM library, in the figure it is marked with shading. After the block header, there is an additional checkword in order to protect the message header. Following the header is the block data itself. At the end of the communication packet comes two byte special check word. In this protocol we use the same Cyclic Redundancy Check (CRC) error detection method than is used in the amateur AX25 and CCITT X25 protocols. The error detection polynomial is $x^{16}+x^{12}+x^5+1$, and the checker is initialised with all ones, check word is send as inverted and the result is checked against a special check word f0b8 in hexadecimal.

After the data packet successfully received, DSP CARD 4 sends ACK character to notify host computer that it has received the packet correctly. If there was an error checkword detected, or if received character timeout has been received, the DSP CARD 4 sends a BADCRC character (05 in hexadecimal), and then the host can send the given packet again.

LEONID monitor

LEONID monitor program gives some simple functions to the programmer in order to ease his/her task to interface those devices on the DSP CARD 4. The functions are defined as macros in a special file `LEONID.ASM` that must be included in the user’s own program code.

There are also two special purpose constants `user_code` and `user_data` that determine the suitable starting locations of the code in P memory and data in X and Y memories.

`opensc`	Open serial line (flush buffers). Register `a` contains an address of a special KISS command handler subroutine. If a=0, then enter normal mode, if a≠0, then KISS protocol handling is enabled. Default mode is normal mode. Register `b` contains the address of transmitter on/off control routine. `C`-flag contains the information if the transmitter must be set on (`C=1`) or off (`C=0`).
`putc`	Put a byte in `x0` register to the serial output
`getc`	Wait for a character (returned at `x0` register) at the serial line.
`lookc` **t_d**	Test if there are characters waiting on the serial line. Sets carry flag if no data available, reset carry flag if there is a character available (returned in `x0` register). Maximum waiting time (in seconds) can be determined with t_d parameter.
`tstc`	Test if there are characters waiting at the serial input. Set carry flag if there are no data available, reset carry flag if there are data available. This function does not actually read characters from the buffer as `lookc` does.
`endc`	Terminate KISS frame. After this function call, all those characters put onto buffer (after previous `endc` function call) will be given to the serial line.
`rejc`	Reject current KISS frame. All characters (before previous `endc` function call will be discarded).
`putbit`	Put next bit in carry flag to the host transmit queue.
`getbit`	Returns next bit to be sent in carry flag, returns zero flag set if this is an end of the transmission.
`opencd` **f_s filt**	Initialise the CS4215 Codec and start sending and receiving samples. Register `r7` and `m7` must be set to point the desired sample buffer. f_S is the desired sampling rate in kHz (legal values are 8,9.6,16,27.42857,32,48). filt is `NOHPF` or `HPF` in order to switch codec's internal HPF off or on.
`closecd`	Stop sampling operation and place the CS4215 Codec to a power down mode.
`sleep` **t_W**	Wait for a specified time t_W (in seconds). At this time DSP is placed to power saving wait mode.
`putio`	Output a byte in `x0` to the general purpose parallel output lines
`waitblk` **reg len blen**	Wait for blen samples.
`ctrlcd` **mode reg len iset igains oset ogains**	Set control data oset & iset and gains to the codec.

Table 2: Services that LEONID monitor offers.

LEONID itself reserves 3 kW of top P-memory (1400H – 1FFFH) for serial line buffers and some monitor routines (monitor routines are located here in order to release maximum amount of internal P-memory for user application programs). Only a small part of the monitor resides in the internal program memory (about 64 words). This part contains only DSP56001 interrupt vectors. **Address register r3 (with n3 and m3) are reserved for LEONID.**

LEONID function calls

The main functions that LEONID offers are listed in the table 2. They can be divided to four groups: serial line handling (`opensc, putc, getc, lookc, tstc, endc, rejc`), bit-functions (`putbit, getbit`), codec handling (`opencd, closecd, waitblk, ctrlcd`), and miscellaneous functions (`sleep, putio`). Most of functions transfers data in `x0` register, and alters the contents of other ALU-registers (a,b,x,y). `opencd` and `getbit` use address register r0 and m0. They are defined as macros, so that they can be used at the place of operation code, e.g.

```
            include 'leonid'

            org   p:user_code

            move  #>$01,x0
            putio

      loop  getc
            putc
            jmp   loop

            end
```

In this simple example we first output simple bit pattern to the general purpose output lines and then forever wait for the characters on the serial line and then output them to the serial line.

Using Codec with LEONID

There are two support macros on the LEONID. One is called `waitblk` that waits for the desired amount of samples from the Codec and then continues the operation of the DSP program. Arguments for it are the register used to read samples from the sample buffer and length of the sample buffer and the number of samples processed at the one batch. Other is called `ctrlcd` that can be used to control the input and output settings of the CS4215 codec. Arguments for it are a flag that determines if the buffer is to be initialised (output data is set to zero), the length of the buffer used and the input and output settings.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HE	LE	LO5	LO4	LO3	LO2	LO1	LO0	0	SE	RO5	RO4	RO3	RO2	RO1	RO0

HE	headphones output enable
LE	line output enable
SE	speaker output enable
LO	left channel output attenuation (in 1.5 dB steps), 0 is no attenuation
RO	right channel output attenuation (in 1.5 dB steps), 0 is no attenuation

Figure 11: Crystal CS4215 output settings word.

Output settings word is shown in figure 11. This word controls miscellaneous outputs (on/off states) and the output level. There are predefined constants (in `LEONID.ASM`) `HEADP`, LINEO and SPEAKER which can be used instead of those bit patterns. Also gains for the output can be given directly with ctrlcd macro.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIO1	PIO0	OVR	IS	LG3	LG2	LG1	LG0	MA3	MA2	MA1	MA0	RG3	RG2	RG1	RG0

PIO	parallel input/output bits
OVR	overrange. When read as 1, indicates over-range has occurred. Bit remains 1 until cleared by writing 0. Writing 1 enables the overrange detection.
IS	input selection (0 - line level input, 1 - microphone level input)
LG	left channel input gain (in 1.5 dB steps), 0 is no gain
MA	monitor path attenuation (in 1.5 dB steps), 0 is no attenuation, 15 is mute monitor path
RG	right channel input gain (in 1.5 dB steps), 0 is no gain

Figure 12: Crystal CS4215 input settings word.

Input settings word is shown in figure 12. This word selects input source and gain of input amplifiers. There are predefined constants (in LEONID.ASM) LINEI and MIC which can be used instead of those bit patterns. Input gains can also be given directly with ctrlcd macro. To allow monitoring of the input audio signal, the output of the ADCs is routed through a monitor path attenuator, then digitally mixed into the input data for the DACs. Thus monitor path can be used to create an analog path loopback for diagnostics purposes.

X: (input)	…	LEFT CHANNEL	RIGHT CHANNEL	OUTPUT SETTINGS	INPUT SETTINGS	…
Y: (output)	…	INPUT SETTINGS	LEFT CHANNEL	RIGHT CHANNEL	OUTPUT SETTINGS	…

Figure 13: Crystal CS4215 data block. It contains four consecutive words, see [49] for details.

Input and output buffers use the same register r7, but the input buffer is on the X part of the data memory and the output buffer is on the Y part of the data memory. The CS4215 Codec communicates one block as four consecutive words as is shown in figure 13. Note that the input and output blocks are one sample apart in memory for facile data address pointer handling. With those output and input settings the gains for the input and output amplifiers and the input signal sources can be determined.

As an example:

```
        include 'leonid'

M       equ     160
buflen  equ     3*M

        org     p:user_code

        move            #buffer+2,r7
        move            #buflen*4-1,m7

        move            #buffer,r2
        move            #4-1,n2
        move            #buflen*4-1,m2

        ctrlcd  1,r2,buflen,LINEI,0.0,0.0,LINEO|HEADP,0.0,0.0
        opencd  8,NOHPF

; wait for one complete block
loop    waitblk r2,buflen,M

; then filter the left channel
        move            #buflen*4-1,m0
        move            #-4,n0
        move            #lotaps,r4
        move            #<lolen-1,m4

        move            r2,r0
        do      #M,_endlpf
        clr     a
        move            x:(r0)+n0,x0  y:(r4)+,y0
        rep     #lolen-1
        mac     x0,y0,a x:(r0)+n0,x0  y:(r4)+,y0
        macr    x0,y0,a (r2)+
        move            a,y:(r2)+n2
        move            r2,r0
_endlpf

        jmp     <loop


        org     x:user_data

buffer  dsm     buflen*4


        org     y:user_data

        dsm     buflen*4

; 900 Hz lowpass filter generated using Parks-McClellan algorithm
        dsm     256
lolen   equ     79
lotaps  dc       1.619394e-02,5.805688e-03,...
```

This example programs waits for M samples from the Codec, then filters those samples with a 900 Hz low-pass FIR filter and places the results to the output. Notice how the sample pointer is updated in two part to correctly keep in phase with the consecutive samples in the buffer.

LEONID likes KISS protocol

Serial interface can be placed to a special KISS protocol [31] handing mode with `opensc` function call. All the normal serial input/output functions behave as normal, and there are two additional functions `endc` and `rejc` available for KISS frame termination or rejection. `endc` function is normally used to inform LEONID that the end of the current KISS frame is detected because there is now other way for LEONID to know this. `rejc` informs LEONID that this current KISS frame should be discarded (e.g. when the CRC check at the end of the frame received from the air informs us that the previous bits have errors). Normally these functions are called internally when the application program uses only those bit-level functions described below.

It is also possible to send and receive databits one-per-one basis with `putbit` and `getbit` function calls. These functions handle all the necessary low-level AX25 protocol handlings.

As an example:

```
        nolist
        include  'leonid'
        list

        org      p:user_code

start   move     #reject,a1
        opensc

loop    getbit
        jeq      <loop

        putbit
        jmp      <loop


reject  rts
```

This example receives KISS packets from the serial line, and send them back to the host computer as KISS packets. Receiving and sending back one bit takes time about 9µs (with 27MHz DSP clock). This includes all the bit handling, KISS protocol handling and interrupt service routine time.

LEONID handles those KISS parameters that controls transmitter on/off delays. These parameters and their default values are listed in table 3. There can be other KISS parameters also if the user's application program implements them. When the host computer want to set KISS parameter that is not mentioned in the table 3, LEONID calls the subroutine whose address was given in the `outsci` function.

Command	Function	Comments	Default
1	TXDELAY	transmitter keyup delay in 10 ms	50
2	P	persistence parameter	63
3	SlotTime	slot interval in 10 ms	10
4	TXtail	hold up after frame send in 10 ms	1
5	FullDuplex	0 - half, 1 - full	0

Table 3: KISS parameters. Parameter P is scaled by 255, so the default value is actually 63/255=0.25.

Transmitter control

Transmitter control module takes care of transmitter on/off delays and CSMA protocol implementation. Flow diagram of the transmitter control module is shown in figure 14.

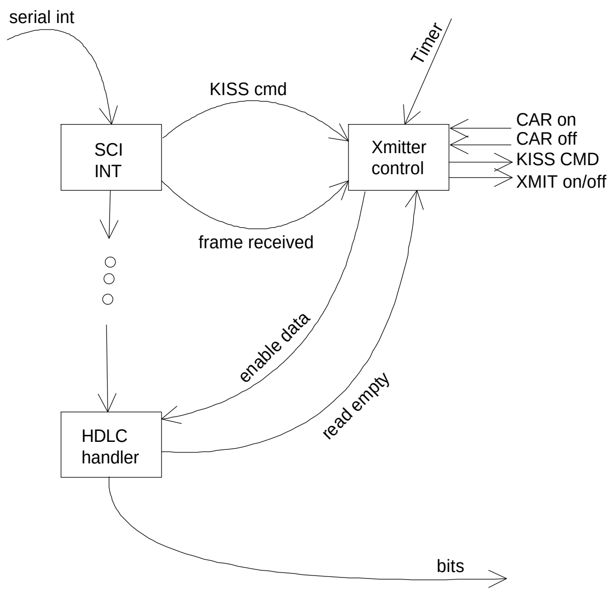

Figure 14: Flow diagram of the transmitter control module.

Transmitter control module gets information (via `caron`, `cardoff` function calls) from the user's application program if there is carrier detected on the air. Module calls user supplied function address (xmit on/off) when it is time to switch the transmitter on or off.

Transmitter control module gets information from the SCI interrupt handler if a new KISS frame has been received. If transmitter is off and there is no carrier detected, then control module calls user's xmit on function in order to scwitch the transmitter on. After some delay, HDLC handler is allowed to give databits to the user's application program.

Application Software

"Signaalivuokaavion viivojen verkko
merikortti tuntemattomaan,
kohtalo piirtää salaa
yön minälle kutsuaan.
Tien keskeen karien kuohun
sen kaavio viitoittaa.
Ei tarvis kuin purjeet nostaa,
vaan minne päin suunnistaa?
Miksi sydän pelosta hakkaa?
Mistä epäily levoton?
Merikortti on tallella tässä,
mutta kompassi, missä se on?"

Jacques Berg, 1993

Now there are some usefull programs available for the DSP CARD 4. There are programs for audio filtering and 1200 bit/s and 9600 bit/s data modems. In addition, all the old programs for the DSP CARD 3 can be ported to the new card. Thus we expect the following implementations to be ready in the near future

- 2400 bit/s LPC Vocoder
- 1200 bit/s BPSK modem (for satellite work)

Narrow bandpass filter

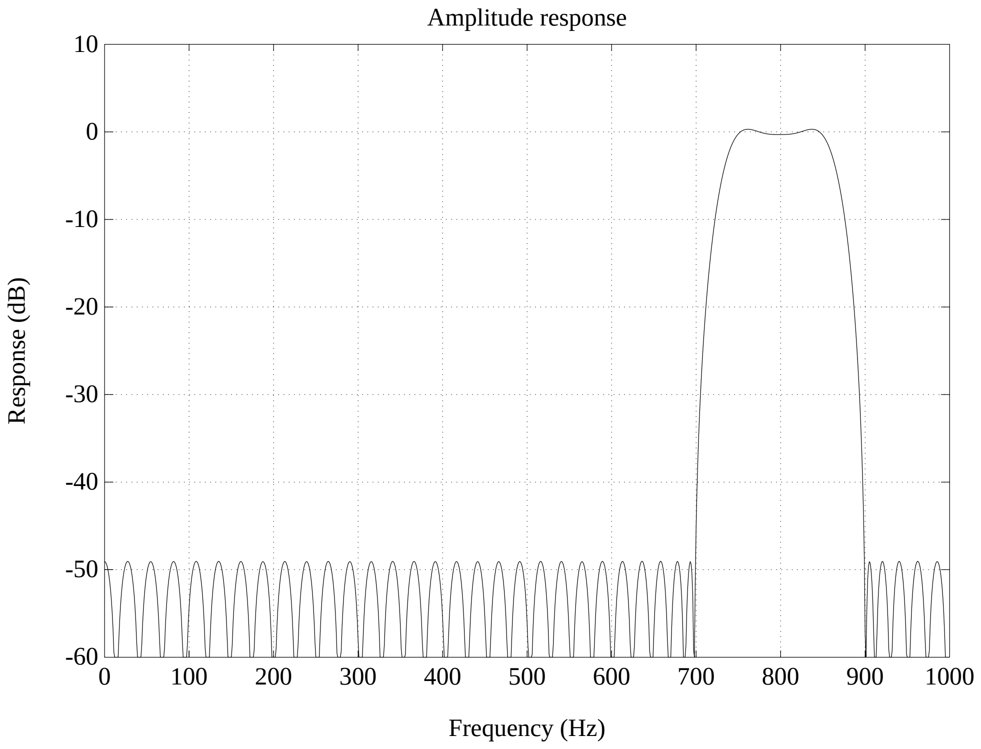

Figure 15: CW bandpass-filter frequency response.

In radio amateur CW operation there are many different stations very near in frequency that makes the differentiation of those stations very difficult. Thus there is a need for a very narrow bandpass type filter to reject unwanted signals. We implemented a fixed-bandwidth, fixed-tuned CW filter with a bandwidth of approximately of 200 Hz, centred at 800 Hz.

The CW-filter algorithm differs from conventional audio- or IF-bandpass filters in two ways. First the filter has linear phase response due to its FIR implementation. With linear-phase filtering, the filter bandpass can be made narrower for a given maximum CW speed.

Secondly, the FIR CW filter has a much better shape factor than most conventional filters, which, although they may have very high Q, are usually of low-order with just few poles. The high-order (320nd) DSP FIR filter has steeper skirts than many analog CW filters, and therefore better rejection of nearby signals. The calculated frequency response is shown in figure 15.

The filter was designed with Parks-McClellan Equiripple filter design method in Comdisco™ Signal Processing Worksystem™ (SPW) Filter Design System (FDS) [51]. The Parks-McClellan Equiripple FIR method creates a linear-phase FIR digital filter with a minimum weighted Chebyshev error approximating the desired ideal frequency response. The approximation theory can be found in references [3,5]. Filter coefficients were transformed automatically to the Motorola DSP56001 assembler format to be included in the passband filter source code.

Using bandpass filter

Bandpass filter programs is `BANDPASS.ASM`, and it must be loaded to the processor memory. This program filters signals in the left line-input channel, and produces the output on the both left and right channel line outputs.

QRM and QRN reduction filters

In various situations the speech audio signal is distorted in some way that listening of it becomes unpleasant mission. There are although clever signal processing methods to reduce much of that distortion of the signal without disturbing the actual speech signal itself [27].

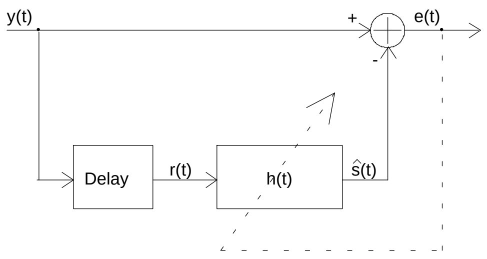

Figure 16: D-step noise-canceller predictor diagram

The basic idea of this is to note that the autocorrelations of the speech signal and the interfering signals are different. It is possible to build a system that separates those signals even they are on the same frequency band. The trick is that although they are on the same frequency band, their statistical properties are different. If the speech signal *s(n)* is contaminated with some kind of interference *n(t)*, the received signal can be stated as

$$y(t) = s(t) + n(t).$$

If we have some kind of structure that can identify the properties of the signal that can adapt to the given signal and the input to that device is delayed input signal *r(t)=y(t-d)*, we have the *d-step predictor* [6] structure of figure 16. The d-step predictor structure can also be seen as a 'correlation canceller'. This is because the output of the prediction filter, $\hat{s}(t)$, consists of the portion of the signal that is strongly autocorrelated at delay lag *d* or greater and this output is removed from the received signal *y(t)*. By adjusting the lag *d*, we can determine what components are removed and what are preserved in the received signal.

Adaptive signal processing

Some times there are problems for which the signal processing system parameters, filter coefficients for example, cannot be specified a priori. The only way these kind of problems are resolved is to make the parameters adjustable that can be optimised to minimise some measure of error. A filter with this kind of performance is called an *adaptive filter* [8,11].

Although both IIR and FIR filters have been considered for adaptive filtering, the FIR filters is by far most practical and widely used—mainly because of stability problems with IIR-filters.

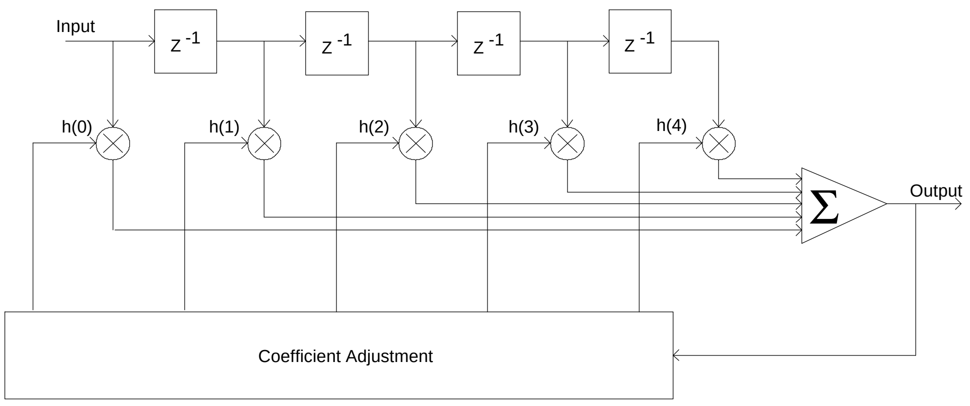

Figure 17: Direct-Form Adaptive FIR Filter

Of the various FIR filter structures that we may use, the direct form is one of the most often used filter implementation form. The direct form FIR filter structure with adjustable coefficients *h(0),h(1),...,h(N-1)* is illustrated in figure 17.

LMS algorithm for coefficient adjustment

Suppose we have an FIR filter with adjustable coefficients $h(k),\ 0 \le k \le N-1$. Let *x(n)* denote the input sequence to the filter and let the corresponding output be *y(n)*, where

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k).$$

Suppose that we also have a desired sequence *d(n)* with which we can compare the FIR filter output. Then, we can form the error sequence *e(n)* by taking the difference between *d(n)* and *y(n)*. That is

$$e(n) = d(n) - y(n).$$

The coefficients of the FIR filter will be selected to minimise the sum of squared errors. Thus, we have

$$\begin{aligned} \mathrm{E} &= \sum_{n=0}^{M} e^2(n) = \sum_{n=0}^{M} \left[d(n) - \sum_{k=0}^{N-1} h(k)x(n-k) \right] \\ &= \sum_{n=0}^{M} d^2(n) - 2\sum_{k=0}^{N-1} h(k)r_{dx}(k) + \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} h(k)h(l)r_{xx}(k-l) \end{aligned},$$

where, by definition,

$$\begin{aligned} r_{dx}(k) &= \sum_{n=0}^{M} d(n)x(n-k), 0 \le k \le N-1 \\ r_{xx}(k) &= \sum_{n=0}^{M} x(n)x(n+k), 0 \le k \le N-1 \end{aligned}.$$

We call $r_{dx}(k)$ the cross-correlation between the desired output sequence *d(n)* and the input sequence *x(n)* and $r_{xx}(k)$ is the auto-correlation sequence of x(n).

The sum of squared errors E is a quadratic function of the FIR filter coefficients. Consequently, the minimisation of E with respect to the filter coefficients *h(k)* results in a set of linear equations. By differentiating E with respect to each of the filter coefficients we obtain

$$\frac{\partial E}{\partial h(m)} = 0, m = 0,1,\ldots,N-1,$$

and, hence,

$$\sum_{k=0}^{N-1} h(k)r_{xx}(k-m) = r_{dx}(m), m = 0,1,\ldots,N-1.$$

This is the set of linear equations which yield the optimum filter coefficients.

In order to solve the set of linear equations directly, we must first compute the autocorrelation sequence $r_{xx}(k)$ of the input signal and the cross-correlation sequence $r_{dx}(k)$ between the desired sequence *d(n)* and the input sequence *x(n)*.

The *least-mean -square* (LMS) algorithm provides an alternative computational method for determining the optimum filter coefficients *h(k)* without explicitly computing the correlation