DSP CARD 3
Betaversion documentation

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

ℵ Alef-Null International
Signal Processing Division

September 4, 1991

Overview

THIS manual describes various aspects about the ALEF NULL DSP CARD 3. First we discuss the reasons for using digital signal processing (DSP) in amateur world and where to learn about DSP. Then we present the DSP CARD 3, what capabilities it has, and how it is implemented. After that the current software for DSP CARD 3 is presented. Then some hints on using the DSP CARD 3 and testing it are shown, and finally the history of the DSP CARD 3 and some future speculation is given.

Please note that this is the Beta version of the DSP CARD 3 Rev. 1.3. This documentation is far from complete and there may be some bugs floating around because we have not yet fully tested this revision. This first Beta test series contains eight boards.

We are extremely grateful to the betatesters—starring: Pekka "se toimii aina heti" Lempola OH2BPP, Sakari "spread spectrum" Mattila OH2AZG, Karl "transient" Tigerstedt OH2MBM, Peter Gülzow DB2OS and Lyle Johnson WA7GXD. Special thanks we owe also to Brad Eacker KB6FED who translated our HPGL files to PostScript form to be included in this document.

Errata

The DSPRAM is a **very** noisy chip, it generates severe peaks on +5 and ground lines. Therefore we havemade a so called ALEF NULL 'Full Satisfaction Supplementary Board' to overcome these serious transient problems. It will simply add a very good ground plane for DSP and DSPRAM.

ℵ ALEF NULL is not a registered trademark, it is used only as a nickname to our group.

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 3 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

1 General

THE DSP CARD 3 is a product of the ℵ ALEF NULL group. DSP CARD 3 is a *general purpose signal processing card*, dedicated mainly to the *amateur use.*

> "Art
> will always
> be art."
>
> *Johan Wolfgang von Goethe*

1.1 Why to use digital signal processing

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

Digital signal processing, DSP, offers many advantages over conventional analogue processing [23].

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).

1.2 Where to learn DSP techniques

Many good books on DSP have been published since 1970's. There are both simple tutorial textbooks and advanced books. For 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 [4] is a very good textbook on DSP, oriented mainly for digital communication engineers. For intermediate level Jackson's presentation [5] and the legendar Oppenheim and Schafer's book [3] are good references. An excellent book on simple adaptive filters is Alexander's one [9]. For advanced level [6,7,8] 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.

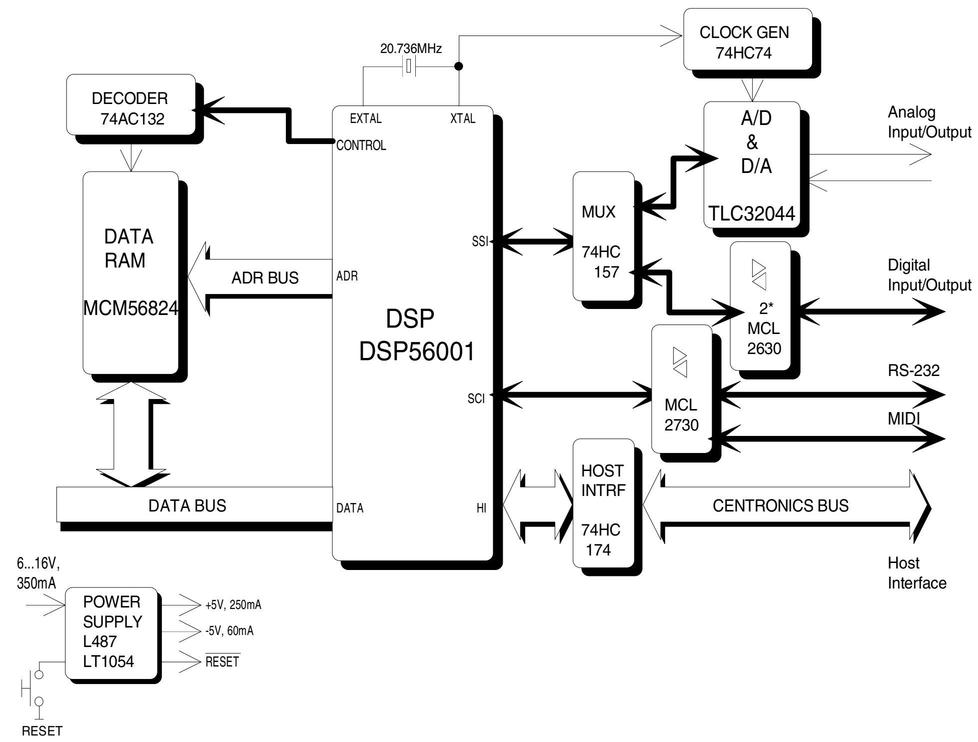


Figure 1: The block diagram of the digital signal processor card. DSP56001 has many i/o functions on the chip, which simplifies the overall hardware implementation.

Of course DSP itself isn't quite useful. In amateur world DSP applications to the communication technology are the most interesting. Carlson's book [10] is the 'number one' introduction to the communications technology. It also contains a very complete reference section for the more interested reader. Lee's and Messerschmitt's book [11] tells a little more about digital communications at quite tutorial level. Bingham's book [12] has many practical hints for modem designer, mainly for telephone modems. More theoretically oriented books are [13,14]. 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.

1.3 The DSP CARD 3 features

The block diagram of the DSP CARD 3 is presented in the figure 1. DSP CARD 3 contains the Motorola DSP56001 digital signal processor with 10.4 MIPS processing power. The card has 8 KW (24 KB) high speed RAM, divided to two parts—4 KW for the program memory and 4 KW for the data memory. DSP56001 has internal 0.5 KW program memory and 0.5 KW data memory, so the total memory is 4.5 KW for program memory and 4.5 KW for data memory.

The DSP56001 has 24-bit word length, giving a dynamic range of 144dB. Internally the accumulators are 56-bit, providing 336dB 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 which greatly enhances the overall processing power; an

	Internal	*External*
P	0000–01FF	0200–11FF
X	0000–00FF	0100–08FF
Y	0000–00FF	0100–08FF

Table 1: DSP CARD 3 memory map. All addresses are given in hexadecimal.

instruction prefetch, a 24×24-bit multiplication, a 56-bit addition, rounding two data moves and two address pointer updates can be executed in a single 96.5 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—enables implementing the DSP system with few external components [25,24].

DSP CARD 3 has many interfaces, host interface to the host computer via bidirectional Centronics port for downloading programs and high-speed parallel data transfers, synchronous serial interface (optoisolated) for interfacing high speed A/D or D/A converters, RS-232 and MIDI interfaces for various optional communications, analog input/output for direct radio interface and PTT, UP and DOWN outputs for radio control.

The whole system needs a single power supply, 6–16 VDC, 400 mA max.

Valid operating temperature range is 0 °C – 35 °C (when appropriate cooling on (A7) and (A8) is ensured, the upper temperature limit could be higher), storage temperature range is -55 °C – 125 °C.

DSP CARD 3 is contained on a single Eurocard (160mm×100mm) double layered, trough plated PCB. The card has a single EURO-64 connector, which contains all the needed connections.

2 Hardware

The total DSP CARD 3 chip count is only nine 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.

2.1 DSP and memory

The DSP56001 interface is very straightforward[1] [26]. The crystal oscillator circuit is a very typical fundamental frequency Pierce type oscillator. The non-standard frequency, 20.736 MHz[2], is selected because the converter chip we use needs 5.184 MHz input clock frequency that is generated by dividing the DSP clock by four with a flip-flop (A2)[3].

Other frequencies could be used, but it must be noted that the DSP crystal driving circuit needs fundamental type quartz and that the converter may then operate on some nonstandard sampling rate.

[1] See the schematics (figures on pages 32 and 33).

[2] Note that the used frequency is higher than the Motorola specs allow (20.5 MHz)—we did not encounter any problem using this only slightly higher frequency.

[3] The part id in paranthesis points to the part id in the schematics.

We used special Motorola DSPRAM (A4) [32] designed for easy interfacing to the DSP56001. So the DSP and memory interface is a very simple. All we need is a NAND gate (A3) for enabling the DSPRAM when DSP56001 wants to access program space (PS) or data space (DS). Note that we use the DSPRAM special feature, vector/scalar selection for partitioning the data space to two 2 KW X- and Y-partitions, and using single 4 KW partition for the program space. For memory map description, see the table 1.

The access time of the DSPRAM is 35 ns, and the DSP operating at 20.736 MHz needs acces time about 53 ns, and the NAND gate delay is about 7 ns, so there is about 53 ns-7 ns-35 ns=11 ns reserve for reliable working or for some experiments with higher operating frequencies[4].

2.2 A/D and D/A converters

We selected Texas Instruments special A/D and D/A converter chip TLC32044 [33] for the DSP CARD 3. TLC32044 is very easy to use; it has both the A/D and D/A converters, their anti-aliasing filters, references and amplifiers in a single 28 pin DIL package. The sampling rate is variable up to 19200 samples per second, and it has a special capability to advance or retard the sampling epoch. This makes it very easy to synchronize sampling rate to an external source.

Interface to the DSP56001 is made by synchronous 16-bit words with continuous clock. Synchronous mode is selected because it needs one signal line less, which can then be used for other purposes. The only signal needed excluding the two data lines and clock line, is the framing synchronization pulse from the TLC32044. Its polarity must be inverted (A3) before it can be fed to the DSP56001.

TLC32044's output stage is powerful enough to drive directly the outputs[5]. There is 300 Ω resistor and a couple of Zener diodes for protection. At the input, there is similar resistor/Zener protection network, they limit the input voltage to about -4 V – +4 V. The nominal input/output level is about 3 V_{pp}, but the dynamic range of the converter's is sufficient to accomodate much lower signal levels easily (e.g. 0 dBV = 1 V_{pp}, approx. 0.7 V_{rms}).

2.3 Power supply

The power supply generates four separate voltages, +5 V and -5 V for digital logic, +5 V and -5 V for analog circuits. Positive voltages are generated by SGS-Ates L487 low drop-out +5 V 500 mA regulator chip in pentawatt package. L487 has RESET output as an additional feature, RESET signal is driven low at power-up or when the output voltage drops lower than 4.85 V. +5 V for analog section is separated from digital +5 V power by a simple LC-filter.

Negative supply is generated by Linear Technology's LT1054 switched capacitor voltage converter chip. LT1054 has also voltage regulator, so the stabilized -5 V generator is very easy to build with LT1054. -5 V for analog section is separated by LC-filter. There is also a separate protection diode (V20) to prevent TLC32044 from latch up when the power goes on.

[4]Please note that when working at very high frequencies on the standard 20.5 MHz chip, the DSP56001 dissipates more power, so the DSP chip must be somehow cooled.

[5]It is possible to add earphones directly to the output via 10 nF coupling capacitor.

2.4 Interfaces

2.4.1 Centronics

The interface is primarily designed for IBM PC computers, but it may be used on other Centronics compatible ports also. The interface and its protocol are based on the DSP56001 host port. The only modification is a separate latch 74HC174 (A6) because the Centronics interface does not have enough lines to drive the DSP56001 host port address and control lines. The 74HC174 latch gets the address and the direction of information transfer from the Centronics data lines and feeds them to the DSP56001. The latch is driven by Centronics line `SELECT`. Centronics `STROBE` line is driven by Schmitt-trigger to prevent spurious transients from the DSP56001 HEN line. That line latches the information presented on data lines to the internal registers addressed by the 74HC174 latch (A6). The HREQ line from the DSP56001 is connected to the Centronics `ACK` line. In many computers the `ACK` line could generate interrupts, thus it could be used in interrupt based data transfers.

Centronics `RESET` line is coupled via low drop-out Schottky diode (V1) to the DSP56001 RS-line, so the host computer can reset the DSP CARD 3 every time it wishes to do so. After reset, the DSP56001 begins to read Host port for software downloading, so with the `RESET` line we can boot the DSP CARD 3.

One bit at 74HC174 latch is connected to the `BUSY` line on Centronics interface. It is possible for the host computer to detect the presence of the DSP CARD 3 by toggling this `BUSY` bit.

2.4.2 High-speed serial interface

The high-speed (max. transfer rate is 5.18 Mbit/sec) serial line is fully optoisolated on input and has one output. The interface is current based, 5 mA current means a logical '0', and no current logical '1'. The optoisolation is based on GE's MCL2630 dual channel high CMR high speed optocouplers. It has LSTTL compatibility and low input current requiremets, so the interface is very simple.

The DSP56001 serial interface can be programmed to variety of serial formats, so this interface could be used to interface easily very different kind of instrumens (e.g. high speed A/D and D/A converters).

2.4.3 RS-232 and MIDI-interfaces

The MIDI interface is a plain vanilla standard MIDI interface [34], except that the standard Sharp PC-900 optoisolator has been replaced by GE's dual high gain MCL2730 part. At physical level the MIDI interface is quite similar to the high speed serial interface. The interface is 5 mA current loop type, logical '0' is current on, and logical '1' is current off.

RS-232 interface's input is also constructed with the MCL2730 optocoupler. RS-232 output is build with a simple discrete circuit. There is an optional CTS-line for (KISS protocol) buffer handling. It uses the same DSP56001 output line (SCLK) as the discrete output, DOWN, so the CTS-flow control can't be used simultaneously with the DOWN-output.

2.4.4 Discrete I/O

These discrete I/O lines are provided for a simple radio control. All the discrete outputs are open collector type, capable of driving max. 100 mA and protected for inductive loads.

3 Connections

Every connection to the DSP CARD 3 is made via the 64-pin Euro-1 connector on the other side of the board.

Power 6 V – 16 V, approx. 400 mA (max). When input voltage is over 8 V, heatsink ($\theta \approx 17$ °C/W) to the A8 is needed. Lower than 7V could be used, depending the resistance of the inductor (L1) of the input filter. The lowest level possible is 5.8 V, limited by the drop-out voltage, 0.8 V, of the regulator chip (A8).

Centronics The interface is primarily designed for the IBM PC series computers, but it may be used on other Centronics compatible ports also. Pins (`DATA1` – `DATA7`, `STROBE`, `SELECT`, `INIT` (= `RESET`), `ACK`, `BUSY`) and several GROUND pins must be connected to the Centronics port. The port must be bidirectional[6] in order to fully utilize the DSP CARD 3, although the test program and DSP TNC program works on unidirectional port. The `INIT` and `BUSY` signals are optionals, and they are not needed in the rudimentary use of the DSP CARD 3.

RS 232 This is standard RS 232 interface.

MIDI This is standard MIDI interface, there are MIDI IN and MIDI OUT ports but no MIDI THRU port.

Serial This is special optoisolated current loop type serial link.

I/O These are special purpose open collector outputs suitable for radio controlling.

Analog Simple analog I/O. Max input/output is 3 V_{pp}, typical level is 1 V_{rms}.

4 Software

Currently there are the following DSP-software for the DSP CARD 3:

- Special test program for testing the card
- Bell 202 1200 bit/s FSK modem with KISS-TNC interface
- G3RUH 9600 bit/s PAM modem with KISS-TNC interface

For PC there is a program downloader for downloading the DSP56000 software to the DSP CARD 3 and simple graphical waveform display program for data analyzing.

[6]BYTE September 1988, pp. 303–312 contains an article discussing printer port bidirectionality

4.1 Program downloader

The program downloader (`DL`) loads `.LOD` files produced by the Motorola's LNK56000 linker to the DSP CARD 3 and starts execution. Its usage is:

```
usage: dl [-d | -l | -c | -p<portno> | -s<ssisource>] [<filename>]
    -d reads data from DSP CARD 3 and displays it
    -l reads data from DSP CARD 3 and stores it to the file LOG.DAT
    -c reads memory image from DSP CARD 3 and verifies it
    -p<portno> uses the specified printer port
       (otherwise dl searches it automatically)
    -s<ssisource> selects the specified ssi source
       (1 external, internal otherwise (default))
```

There are three special modes, debug (`-d`) mode for displaying data sent from DSP CARD 3, data logger mode (`-l`) for storing data sent from DSP CARD 3 in plain ASCII format[7] for later retrieval and a memory contents validity checker (`-c`). These modes are useful for debugging and analyzing the program performance. There is special program (`SHOW`) that displays the ASCII format data for easy signal analyzing. The memory cheking option makes `DL` to wait until DSP CARD 3 sends its memory image (by special macro) and then checks that image with the linker produced output. This is useful when there is any doubt that the suspected program alters unintentinally the static data in memory.

Normally `DL` automatically searches the paraller port where DSP CARD 3 is connected. With a special `-p` option the automatic parallel port selector is bypassed, and the user selected port is used instead. This option is useful when there is more than one DSP CARD 3 connected to PC.

There is a multiplexer on the DSP CARD 3 that allows the user to utilize the internal converter chip or external high-speed serial interface. With `-s` option the multiplexer on the DSP CARD 3 can switched to the desired position.

4.1.1 Host side

The host side is the more complex part of program downloader. Currently it is implemented only for IBM PC series computers. It is coded entirely in C-language, so it would be quite an easy task to port the system to an another system.

The host side can be divided into three main parts, `IO`, `RDLOD` and the main module. `IO`-module is responsible for reading and writing the data to the DSP56001 Host port. In IBM PC implementation this is made by handling the Centronics interface. This module provides simple read and write routines for Host port handling. `RDLOD`-module reads Motorola LNK56000 linker output file. It is implemented by a state machine method, and it provides one function for reading the next word from the Motorola linkers output memory image file.

The main part of the downloader at the host side uses previously described modules to read linker output file and put it to the DSP56001 Host port. First it resets the DSP56001. DSP56001 starts its internal bootstrap ROM code which waits for data to be present at the Host port, and puts the given data to its internal program memory. When 512 words are received or a special

[7] Simple ASCII format is a one very common format used in spreadsheets, presentation graphics or scientific calculation programs.

flag (HF0) is set, the execution of the previously downloaded program begins. When there is more program code, or some constants on the X or Y data RAM areas, then the internal bootstap program is incapable of doing downloading. So we first download our special downloader which then waits data from Host port to be placed to DSP CARD 3 RAM spaces. This downloader can load more than 512 words data or data to X ot Y data spaces.

When debug mode or data logger mode is requested, then after the code is downloaded to the DSP CARD 3 we read data from the DSP56001 to be presented or placed to the ASCII output file. At the DSP56001 side the data could be generated from various parts of program code (for debugging) or from some samples of internal variables (for performance analyzing).

After the code is downloaded and memory contents checker mode is enabled, we can read data from the DSP56001 to compare it with the memory image produced by the linker. This way we can check if there are any changes in memory locations that contain only constants or program code.

4.1.2 DSP CARD 3 side

The DSP CARD 3 side of the loader is very simple. The downloader is loaded to the internal memory of the DSP56001. Then the downloader will copy itself to the external P-memory at 11E0H and jumps there. After that the code reads data from the host port. When the HF0-flag is set, the next word tells the downloader what to do; load data to the X-,Y- or P-block or end the downloading process and jump to the downloaded code to start execution.

4.2 Graphical data displayer

The `SHOW` utility is for viewing the data that has been collected with the downloader. `SHOW` reads the ASCII file, scales the output appropriately, finds the zero line (if it is there) and draws the graph on the screen. Pressing '`T`' key causes the display to be expanded, and the sample moments to be enchanced by circles, this is useful for examining the smallest ingredients of the signal. Pressing the same key again reverts the display to the normal mode. Pressing '`.`' key returns to the DOS. Pressing any other key causes the display to advance to the next samples (there are 512 samples on one screen).

The ASCII input file format is also compatible with the `SIM56000` simulator, so `SHOW` can be used to visualize the results of the tested algorithms in the simulator.

Currently the software is implemented with Microsoft's Graphics library, and there is support only for VGA and Hercules (with special driver) displays.

4.3 Test program

This test program is intended for the initial DSP CARD 3 testing. It checks the memory, both the internal and external X-,Y- and P-spaces. It also tests the D/A-converter side of the A/D & D/A chip (and the communication link to it) by outputting simple sinewave. The serial interface is tested by printing at 9600 b/s the banner 'Hello World!', and reading a command from the RS-232 line to control the sinewave frequency. When the '+' character is read, the output frequency is incremented, when the '–' character is read, the output frequency is decremented.

The memory test is performed by writing a special test sequence to memory cells, and then, after the whole bank is written, checking by reading that the sequence was ok.

The sinewave generator uses a simple sinewave table lookup method. The generator is in SSI xmitter interrupt service routine.

There is also a special memory test program that displays on the RS 232 line the error data. This program, `MEM`, can be used to inspect onboard memory.

4.4 Modem skeleton

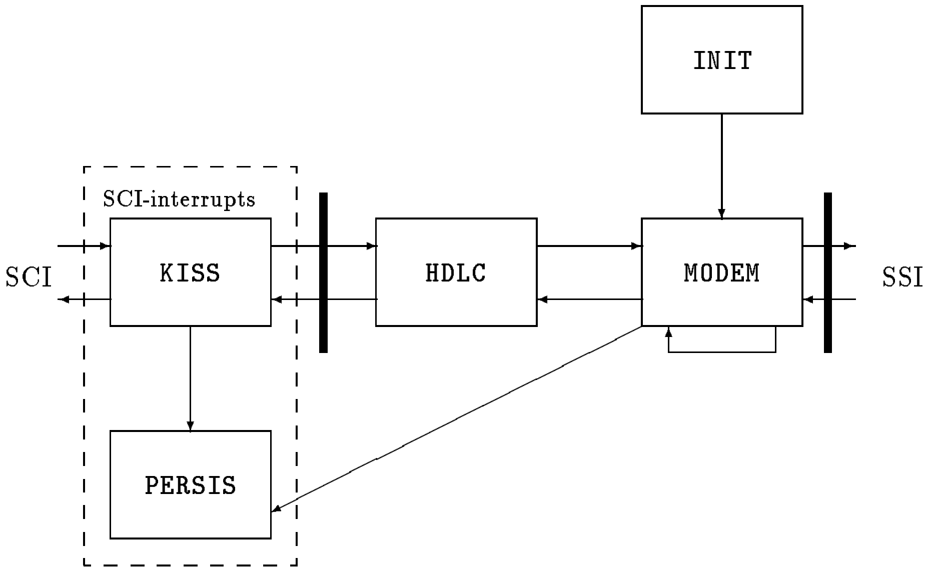


Figure 2: DSP Modem skeleton. The skeleton is totally interrupt based. Interrupts come from either the SCI or SSI, the buffer between these blocks is marked with a thicker line. Arrows between the modules mark the direction of the information flow. SSI interface is implemented with fast interrupts.

To be able to experiment with interesting modulation methods, one must communicate with others. In amateur world this often means that one must comply the AX25 protocol. So every succesfull modem experimentation nowdays must contain a full AX25 protocol implementation. Fortunately there are some helpful methods to remove the protocol burden from the modem part. One widely used method is to use the KISS protocol [21]. Then the host computer does most of the protocol work, and the modem has to take care only the lower level HDLC-type protocol handling.

To further ease the modem experimenters' work, we have made a general purpose modem skeleton. It handles the KISS-interface, low-level HDLC-handling [20] (CRC-generation/checking, flag generation/detection and bit stuffing/removing). The module schematics of the skeleton is shown in figure 2. The skeleton consists of five independent modules whose interconnections are strictly defined. The modules and their contents are

INIT Interrupt vector module. All the interrupts are first handled by this module. After RESET, the control goes here first. This module contains also the context save locations of the interrupt service routines. This module calls all other modules that they may initialize themselves, and it initializes all the needed peripherals. After initialization, the control goes to the main modem loop.

KISS SCI interrupt and KISS-protocol handling and buffering. The protocol handling is made by a simple state machine approach, so the structure of the code is quite clear. KISS-module introduces the functions to read and write bytes to and from the host computer, and the service to tell when it is time to start transmitting (a new KISS-frame is received after it has been waited for).

HDLC The HDLC-protocol handling. This module converts the bytes to bits and vice versa. At the same time the flag must be detected, zero bit must be inserted after five consecutive ones and the CRC checkword calculated. This module introduces functions to read and write bits, and it needs services from the KISS-module to read and write bytes.

MODEM This module is used to implement the currently needed modem. This is the main signal processing part. This module waits interrupts and check if new samples must be processed.

```
loop    wait
        jmp     <loop
```

The waiting is accomplished with the `WAIT`-instruction, so the power consumption is reduced. This module needs the functions of the HDLC module to read and write bits. In actual implementations this module is named after the implemented modulation method, e.g. the Bell 202 modem module is named after the used modulation method, CPFSK.

PERSIS P-persistence handling module. This module takes care of the start-up of the transmitter. The key-up procedure is made by P-persistence algorithm.

The modem status is stored to the global variable `x:status`, which has bits for the following states:

xmt Transmit data/preamble. This bit is used to determine when it is time to start data trasnmission after keying transmitter on.

dcd Carrier detect. Used to indicate when the carrier is on/off.

ptt PTT. Used to indicate wheter the transmitter is on or off.

Actually more bits are used in `x:status`, but those are used internally in modem skeleton (mainly in `PERSIS` module).

4.4.1 MODEM-module

The `MODEM` contains the appropriate modem implementation. There is a simple function API (APplication Interface), that the programmer must obey.

The following resources are available for the user:

putbit This function can be called from the `MODEM` module. It directs the received bit in the carry flag (C) to further protocol processing.

getbit This function could be called from the `MODEM` module. By return value, the zero flag (Z) is set when there are no more bits left to transmit. The bit to transmit is in the carry flag (C).

Also the `x:status` word has three useful bits (see section 4.4) for determining the current mode of the modem.

4.5 Debugging utilities

Currently there are only very simple debug facilities. The only debug aida are the buffered queue between the user's program in the DSP56001 and the host computer and the memory image checker.

In the DSP56001 side this can be used only via the `swi` intruction. By calling this instruction with the desired value in `a1` register, the routine will put the value in `a1` to the output queue. From that queue, the host computer can read the data. The downloader can store these datas to a special file, that can be postprosessed by many different ways, for example the data can be viewed with the `SHOW` utility.

In the `macros.asm` file there are some helpful macros; `bug` for sending a value to the queue, and `dump` for memory image dump. `dump` macro can be issued whenever the user wants to check the validity of DSP CARD 3 memory contents. This is especially important whenever the user suspects that DSP56001 memory gets corrupted.

4.6 Bell 202 modem

4.6.1 CPFSK signal

Bell 202 modem uses two different frequencies to carry the information, this modulation is known as *binary continuous phase frequency shift keying* (CPFSK). CPFSK signal can be described as [16]

$$s(t, \vec{a}) = \sqrt{\frac{2E}{T}} \cos\left(\omega_c t + \phi(t, \vec{a}) + \phi_0\right) \qquad 0 \leq t \leq nT, \tag{1}$$

where E is the energy of the on signal during one bit interval T, ω_c is the frequency of the carrier and ϕ_0 is the phase of the carrier at the beginning of the bit interval; in non-coherent detection we can assume that ϕ_0 is a random variable which is equally distributed on interval $[-\pi, \pi]$. The information bearing phase could be presented in the CPFSK case as

$$\phi(t, \vec{a}) = \pi a_i h \frac{t - (i-1)}{T} + \pi \sum_{r=1}^{i-1} a_r h \qquad (i-1)T \leq t \leq iT, \tag{2}$$

where $\vec{a} = (a_1, a_2, \cdots, a_n)$ is n-uncorrelated equal probality binary sequence ($a_i = \pm 1$) and h is the *modulation index.*

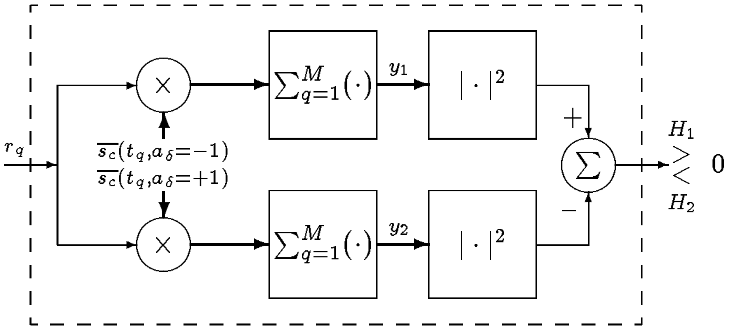


Figure 3: Optimal threshold type demodulator for CPFSK-signals. Complex signals are drawn on bold line. The demodulator correlates the received signal with two reference signals. The decicion of what symbol has been received is made by the stronger correlation result.

4.6.2 CPFSK demodulation

There are many ways to demodulate CPFSK signals. One very simple method is to use two filters centered on signal frequencies [10]. In our implementation we took another approach, namely *CPFSK demodulation using multiple bit observation interval* [29]. In this method we will utilize the continuous phase of the signal. The phase of the current bit depends on the phase of the previous bit and the phases of the future bits are dependent on the current signal phase and the symbol received [17,18]. Using this method we can get a little improved[8] performance in the presence of white or non-white noise [18].

We used a simple multiple bit observation interval *threshold detector* for demodulating weak CPFSK signals. The threshold detector compares the received signal with the synthetically formed reference signal, and the decision what symbol has been received is made when the correlation of the signals is higher than some predetermined level (threshold). Threshold detector is non-coherent by nature—it does not need to know the exact phase of the received signal, this makes the synchronization problem easy.

The detection problem at hand is a composite hypothesis statistical test which may be stated as [14]

$$\left.\begin{array}{l} H_1 : r(t) = b\cos\left(\omega_c t + \phi(t, a_\delta = +1, \vec{A_j}) + \phi_0\right) + n(t) \\ H_2 : r(t) = b\cos\left(\omega_c t + \phi(t, a_\delta = -1, \vec{A_j}) + \phi_0\right) + n(t) \end{array}\right\} 0 \le t \le nT, \quad j = 1, 2, \cdots, 2^{n-1}, \tag{3}$$

where b (signal attenuation) and ϕ_0 (carrier stating phase) are composite parameters with the latter uniformly distributed in $[-\pi, \pi]$ and the former having an arbitrary fading distribution. Further, it is assumed that b and ϕ_0 are independent. $\vec{A_j}$ is $(n-1)$ vector $(a_1, \cdots, a_{\delta-1}, a_{\delta+1}, \cdots, a_n)$ that represent another composite parameter whose distribution is easily determined by noting

[8] About 2 dB in E_b/N_o ratio [17].

that the data bits are independent and assume values from the set $\{+1, -1\}$. The quantity $\phi(\cdot)$ is the information-carrying phase defined in (2). The term n presents the additive noise.

The received waveform is uniformly sampled to obtain M samples. Assuming independent and identically distributed noise samples, the composite hypothesis test of optimally deciding between H_1 and H_2, is given by the likelihood-ratio test (LRT) [14]. Setting up LRT, making use of the Taylor series expansion for signal level near zero, discarding signal terms of order 3 and higher and averaging over all composite parameters, it can be shown that the threshold receiver is given via the decision rule

$$\left|\sum_{q=1}^{M} r_q \overline{s_c^p}(t_q, a_\delta = +1)\right|^2 \underset{H_2}{\overset{H_1}{\gtrless}} \left|\sum_{q=1}^{M} r_q \overline{s_c^p}(t_q, a_\delta = -1)\right|^2, \tag{4}$$

where

$$\overline{s_c^p}(t_q, a_\delta = \pm 1) = \sum_{i=1}^{2n-1} e^{j(\omega_c t + \phi(t, a_\delta = \pm 1, A_i))}. \tag{5}$$

In the equation (5) $\overline{s_c^p}(\cdot,\cdot)$ presents the refrence signals of the complex correlators: the other is composed from every n-bit length signals where $a_\delta = +1$, and the other from signals where $a_\delta = -1$. The demodulator that conforms to the equation (4), is presented in the figure 3.

So our receiver has two correlators whose outputs are compared and the decision of what symbol has been received is made by that comparison. The reference signals are given by [29]

$$\overline{s_c}(t, a_\delta = \pm 1) = \begin{cases} \cos\left(\dfrac{\pi h t}{T}\right) \cos(\pi h)^{i-1} e^{j\omega_c t} \\ \qquad 1 \le k \le n_1; (k-1)T \le t \le kT \\ \cos\left(\dfrac{\pi h(t - iT)}{T}\right) \cos(\pi h)^{i-1} e^{j(\omega_c t \pm \pi h)} \\ \qquad 1 \le k \le n_2; kT \le t \le (k+1)T \end{cases}. \tag{6}$$

in the CPFSK case.

4.6.3 Symbol synchronization

Symbol synchronization or timing recovery is one of the most critical receiver functions in synchronous communication systems. The receiver clock must be continuously adjusted in its frequency and phase to optimize the sampling instants of the received data signal and to compensate for frequency drifts between the oscillators used in the transmitter and receiver clock circuits. The timing information is usually derived from the data signal itself and is based on some meaningful optimization criterion which determines the steady-state location of the timing instants. A crude distinction can be made between three different kinds of methods [29]

1. The *threshold crossings* of the received baseband data signal are compared with the sampling phase. A correction of the sampling phase is initiated as a result of this comparision.
2. *Signals derivate* at the sampling instants is correlated with estimated data to produce the updating information required for the timing control loop.

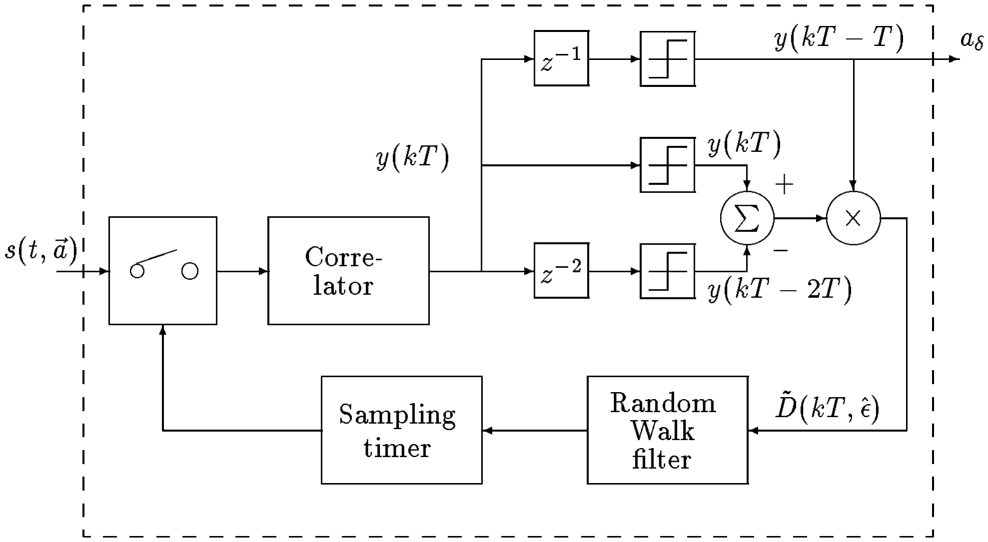


Figure 4: Symbol synchronization. The signal from the demodulator is used to make estimate for the timing error. The resulted estimate is filtered, and is made to control the sampler.

3. A *spectral line* at the clock frequency if filtered out by a narrowband loop. Since such lines are not ordinarily encountered in systems, some nonlinear processing of the signal is used to generate such lines.

In our system, we use the first method, mainly because then the realization becomes very easy.

The signal at the output of the demodulator is

$$\begin{aligned} y(t) &= e^{j\phi(t-\epsilon T,\vec{a})} + n(t) \\ &= r(t,\vec{a},\epsilon) + n(t), \end{aligned} \tag{7}$$

where ϵ is the symbol timing error and $n(t)$ is additive white Gaussian noise with doublesided spectral density of $N_0/2$ (W/Hz).

The symbol synchronization is a special closed-loop system that searches the minimum of ϵ and adjusts correspondingly the sampling system. The timing error is formed by a special function[9]

$$\tilde{D}(kT,\hat{\epsilon}) = \text{sgn}\,(y(kT,\hat{\epsilon}))\,[\text{sgn}\,(y(kT,\hat{\epsilon}+\Delta\epsilon)) - \text{sgn}\,(y(kT,\hat{\epsilon}-\Delta\epsilon))]\,. \tag{8}$$

When the sampling occurs too late $\tilde{D}(kT,\hat{\epsilon})$ is positive, and when the sampling occurs too early $\tilde{D}(kT,\hat{\epsilon})$ is negative. The given error sequence is filtered by a linear random walk filter to filter out the random and synchronizer self noise. The receiver implementation is shown in Fig. 4.

The random walk filter is formed by a simple up/down counter. In each symbol interval T, the content of the up/down counter $M(kT)$ is updated in accordance with

[9] $\text{sgn} = \begin{cases} -1 & \text{when } x < 0 \\ 1 & \text{when } x > 0 \end{cases}$

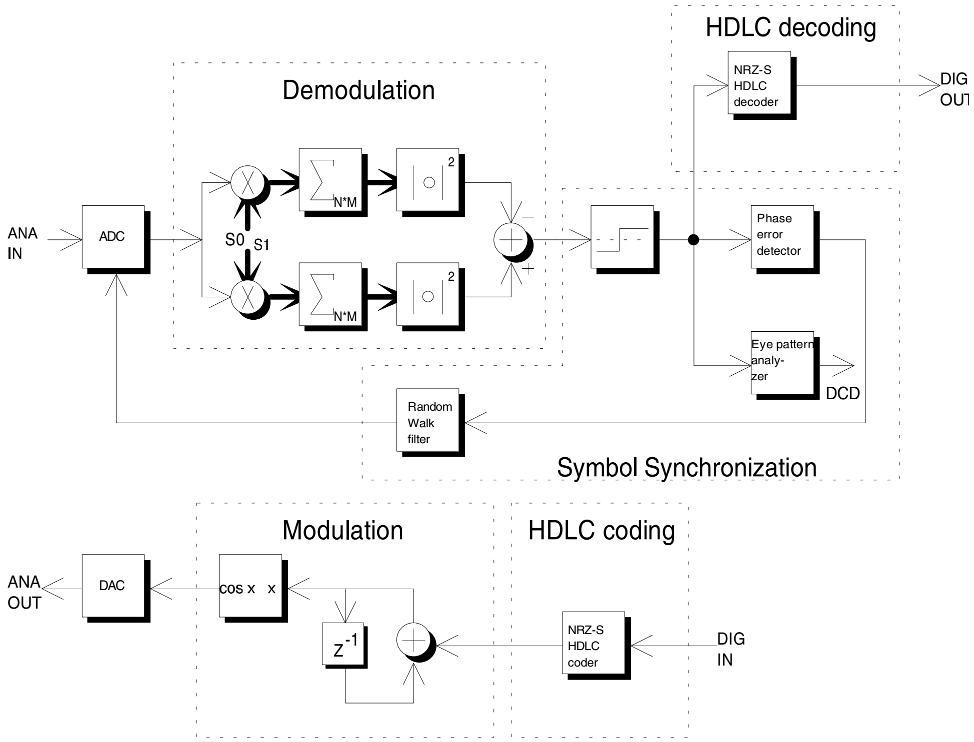


Figure 5: The block diagram of the modem signal processing.

$$M((k+1)T) = \begin{cases} M(kT) & \tilde{D}(kT,\hat{\epsilon}) = 0 \\ M(kT) - 1 & \tilde{D}(kT,\hat{\epsilon}) = 1 \\ M(kT) + 1 & \tilde{D}(kT,\hat{\epsilon}) = -1 \end{cases} \tag{9}$$

The initial content of the counter is M_0. Each time the $M(kT)$ reaches the value 0 or $2M_0$, a command for the timing phase correction is generated according to the rule

$$\hat{\epsilon} \Rightarrow \begin{cases} \hat{\epsilon} - \Delta\epsilon & M(kT) = 0 \\ \hat{\epsilon} + \Delta\epsilon & M(kT) = 2M_0 \\ \hat{\epsilon} & 0 < M(kT) < 2M_0 \end{cases} \tag{10}$$

and the up/down counter is reset to the initial value M_0 if $M(kT)$ reaches the value zero or $2M_0$.

4.6.4 Realization

The complete signal processing diagram is shown in Fig. 5. The implementation to the DSP56001 is straightforward. The needed complex correlation is formed by two real convolutions

```
; complex correlation (result in a1)
ccorr   macro
; s0 real -> b (real mode)
```

```
        conv    m*bits,r6,r0,a
        move    a1,x0                       ; square result
        mpy     x0,x0,a

; s0 imaginary -> a (real mode)
        conv    m*bits,r6,r0,b
        move    b1,x0                       ; square result
        macr    x0,x0,a                     ; add them togheter
        endm
```

The general purpose macro `conv` is defined in the file `macros.asm` as

```
; macro for convolution
conv    macro   n,xd,cf,acc
        clr     acc         x:(xd)+,x0  y:(cf)+,y0
        rep     #n-1
        mac     x0,y0,acc   x:(xd)+,x0  y:(cf)+,y0
        macr    x0,y0,acc
        endm
```

In complex correlation the correlation is made to the real and imaginary part of the reference signal. The correlated results are squared and added togheter.

By the two complex correlation, the demodulated signal is generated.

```
; correlate with s0 signal
        ccorr
        move    a1,x1

; correlate with s1 signal
        ccorr

; calculate result
        sub     x1,a
```

The received signal (over three symbol intervals) is correlated with '1' and '0' signals. The correlation results are subracted from each other, and the greater correlation signal is selected to present the estimate of the xmitted symbol signal.

The implementation of the symbol synchronization is very straightforward. The symbol synchronizer gets N samples from the demodulator and takes one sample to the decision process (to determine what symbol '0' or '1' has been received) and two nearby samples to detect symbol timing, see figure 6. The synchronization is based on theory in section 4.6.3. It tries to adjust the sampling so that the transition between symbols will be always at the same position. If the transition comes too early, the timing is advanced, and if the transition is detected too late, the sampling is retarded.

The carrier detect subsystem tries to analyze the *eye openings* [10]. When enough fully opened eye patterns are detected, the decision can be made that the carrier is present.

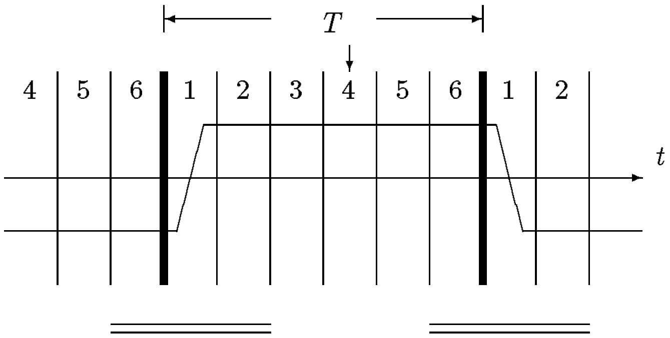


Figure 6: Symbol synchronization. Thin vertical lines represent the sampling intervals. Bolder vertical lines are between the symbol intervals. The horizontal thin line shows the signal, and the horizontal double lines show the transition analyzing places. The decision what symbol has been received is made in the arrow marked place.

4.7 G3RUH modem

"Kekkonen hiihtää,
Kekkonen juoksee.
Mutta kuinka käy Kekkoselta SDK?"

The original G3RUH modem was designed by James Miller. It provides 9600 bit/s data rate with simple FM radios [31]. G3RUH modem is connected directly to the modulator and discriminator of the radio, and to the modem disconnect pins on the TNC.

4.7.1 PAM signal

G3RUH modem uses baseband *Pulse Amplitude Modulation* (PAM) signaling, in which a sequence of time-translates of a basic pulse is amplitude-modulated by a sequence of data symbols. Such signals can be expressed by [10,11]

$$s(t) = \sum_k a_k p(t - kD), \tag{11}$$

where the modulating amplitude a_k represents the kth symbol in the message sequence, so the amplitude belong to a set of M discrete values. The index k ranges from $-\infty$ to $+\infty$. D is the duration of single data symbol. The unmodulated pulse $p(t)$ may be rectangular or some other shape, subject to the conditions

$$p(t) = \begin{cases} 0 & t = 0 \\ 1 & t = \pm D, \pm 2D, \ldots \end{cases} \tag{12}$$

This condition ensures that it is possible to recover the message by sampling $x(t)$ periodically at $t = KD, K = 0, \pm 1, \pm 2, \ldots$, since

$$x(KD) = \sum_k a_k p(KD - kD) = a_K \tag{13}$$

The rectangular pulse satisfies (13) if $\tau \leq D$, as does any timelimited pulse with $p(t) = 0$ for $|t| \geq D/2$.

4.7.2 Transmit and receive filters

In the real communication systems the transmission path distorts the pulse shape in some way, so in receiver we have a pulse as follows

$$y(t) = \sum_k a_k \tilde{p}(t - t_d - kD) + n(t_i), \tag{14}$$

where t_d is the transmission delay, $\tilde{p}$ stands for the pulse shape with transmission distortion and $n(t_i)$ represents some Gaussian noise.

Recovering the digital message from $y(t)$ is the task of the demodulator. An auxiliary synchronization signal may help the regeneration process by identifying the optimum sampling times

$$t_K = KD + t_d \tag{15}$$

If $\tilde{p}(0) = 1$ then

$$y(t_K) = a_K + \sum_{k \neq K} a_k \tilde{p}(KD - kD) + n(t_K), \tag{16}$$

whose first term is the desired information. The last term is the noise contamination at t_K, while the middle term represents cross talk or spillover from other pulses of the signal—a phenomenon called *intersymbol interference* (ISI).

One can easily assure oneself that the effect of $n(t_K)$ can be reduced by lowpass filtering the signal[10]. On the otherwise, lowpass filtering increases ISI. Consequently, some compromises must be made considering ISI, bandwidth and signaling rate.

G3RUH modem uses *raised cosine filter* [10,11] to shape the pulses to match the limitations of the transmission link [30]. Its frequency response is

$$P(f) = \begin{cases} 1 & 0 \leq f \leq \frac{5}{16}R \\ \text{cosine shape} & \frac{5}{16} \leq f \leq \frac{11}{16}R \\ 0 & \frac{11}{16} \leq f \leq \infty \end{cases}, \tag{17}$$

where $R = \frac{1}{D}$ is the bit rate (bit/s). Usually raised cosine type filter is specified by its *cutoff frequency* f_c and *rolloff* parameter α. The cutoff frequency is naturally half of the bit rate $R/2$, and the rolloff factor in this case can be obtained from the formula

$$\begin{cases} \frac{5}{16} & = & (1-p)\frac{1}{2} \\ \frac{11}{16} & = & (1+p)\frac{1}{2} \end{cases}. \tag{18}$$

In the G3RUH case, the filter parameters are $f_c = \frac{R}{2} = 4800$ Hz and $\alpha = \frac{6}{16} = 0.375$.

4.7.3 PAM demodulation

PAM demodulation is performed simply by sampling periodically the receiver filter output at right times. The performance of the modem can be little improved by adding a special shaping filter somewhere on the transmission path adapted for the used transmission path. In G3RUH modem this filter was implemented on the transmitter with a novel way to simplify the hardware

[10]Assuming here white-noise.

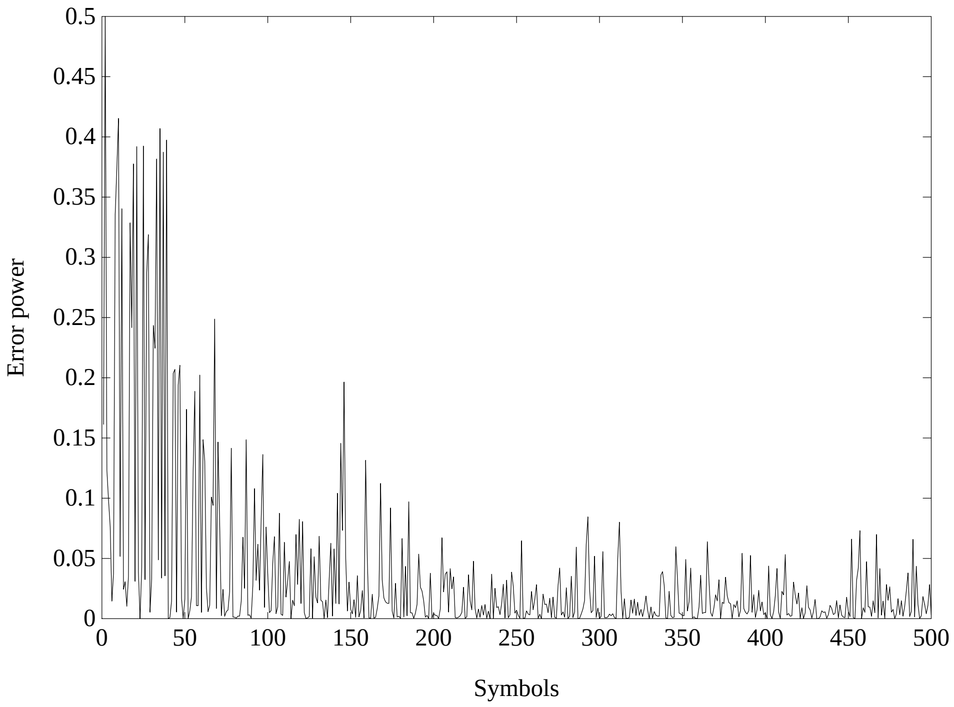


Figure 7: Adaptation error $e^2(n)$ versus symbols received. This curve shown the adaptation curve of the used LMS filter. Plot begins when start of packet is detected.

implementations, but it therefore requires that the transmitter must be adapted to the receiver which is difficult in multipoint links. Because filters are handled very easy with DSP, we decided to implement the shaping filter on the receiver side and to make it automatically adaptive to get better match with different transmitters.

In our implementation we used *Fractionally Spaced Equalizer* (FSE) [10,11] where the filter taps are spaced a fraction of the symbol interval apart. FSE can effectively compensate more severe delay distortion and deal with amplitude distortion with less noise enhancement than conventional equalizer.

The adaptation algorithm used is the very popular *Least Mean Squared* (LMS) algorithm [9,6, 7,11,13]. The popularity of the LMS algorithm stems from the simplicity of its computational structure, low storage requirements and the relative ease with which it may be mathematically analyzed. If we consider the general filter equation

$$\hat{d}(n) = \sum_{k=0}^{n} w_k x(n-k) \tag{19}$$

the LMS algorithm adjusts the filter coefficients w_i by

$$w_i(n+1) = w_i(n) + \alpha e(n) x(n), \qquad 1 \leq i \leq N, \tag{20}$$

where the error is $e(n) = d(n) - \hat{d}(n)$, desired response minus the output of the filter and α is the adaptation coefficient that determines the adaptation speed of the filter. The adaptation coefficient must be within some limits [9], for the LMS algorithm to converge the desired response.

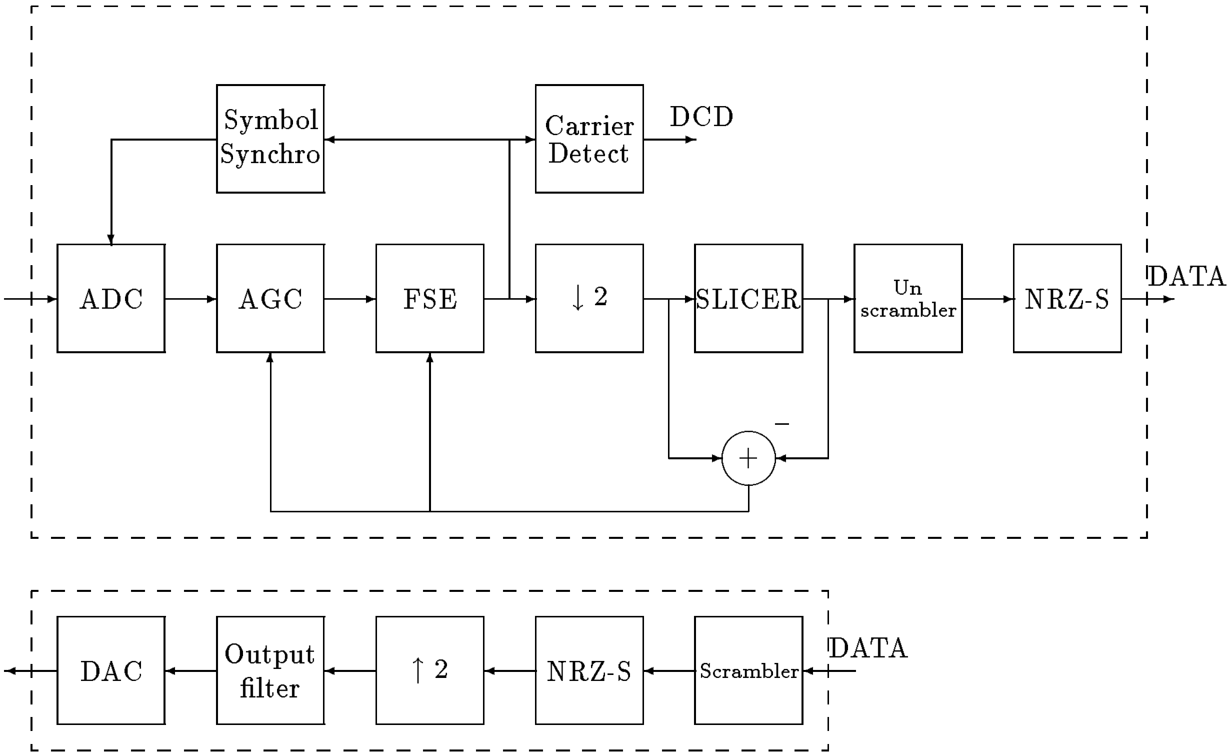


Figure 8: The block diagram of the G3RUH modem signal processing. Note that there are simple multirate structures on both the receiver and transmitter. $\uparrow 2$ and $\downarrow 2$ are interpolator and decimator structures [15].

When the adaptation coefficient $\alpha = 0.000769$, the adaptation curve is shown in figure 7. Data for this figure is obtained from the DSP G3RUH modem with `DL` data logging capability.

Carrier detect is implemented by determining the eye opening. That condition is checked by filtering the sampling and zero crossing instant values and then calculating their difference. When the difference is over some predetermined threshold, the DCD condition is met.

Symbol synchronization is performed the same way as in the CPFSK case (see section 4.6.3). The basic idea is to control the zero crossing time, and then to make the decision after $T/2$ delay.

4.7.4 Realization

The complete signal processing diagram is shown in Fig. 8. Sampling rate is 19200 Hz. After sampling, the signal level is compensated by automatic gain controller (AGC). The gain controller is actually decision directed [12], the filtered decision error is used to stabilize the signal level. This compensated signal is fed to the equalizer, its taps are adjusted by the LMS algorithm from the decision error. The filter output is used by symbol synchronization and carrier detector. Because the data rate is 9600 bit/s, the filter output is decimated by 2 to obtain the needed bit rate. Decimated signal is then used by the slicer, which makes the decision of the received symbol. Because there are only two symbols, the decision process is simple; negative value is '0' and positive value is '1'.

Received bits are fed to the self-synchronized descrambler [11]. It is a simple shift register with some exclusive-or gates to undo the scrambling used in the transmitter. The scrambling polynomial used is $1 + x^{12} + x^{17}$ [31].

The transmitter part is simpler than the receiver, partly because there are no need for symbol synchro, carrier detect and adaptive equalization. The output filter is the same as in the receiver input filter (though not adaptive).

5 Construction

5.1 PCB

Assembly drawing is shown in figure page 34.

The construction of the DSP CARD 3 is very easy. The exact resistor values, except R10 and R9 are not very critical. The resitors should be 200 mils type. The inductor values are not critical either, something between 4.7 μH – 10 μH is ok, it is important that the resistance of the inductors does not exceed about 0.5 Ω.

Because FACT CMOS 74AC132 Schmitt-trigger NAND chip may be somewhat difficult to find, 74F132 can be used instead, or when the reliability is not highly critical, 74HC132 (with slower switching times) could also be used instead.

5.2 Connectors

A suggested connector wiring is presented in fig. 9. MIDI ports are wired in a standard way [34], and radio interface is wired such that the plain vanilla TNC 5-pin DIN connector is compatible with this interface. In host connector, normal 25-pin D-connector may be used or a special 36-pin Centronics can be used instead.

5.3 Cabling

The cabling is simple except the Centronics line. The `STROBE` and `ACK` lines are expecially sensitive to noise. So special precautions must be taken on the cabling, `STROBE` and `ACK` signals must be properly isolated (we used separate shielded cable on those signals), the normal printer cables are not usually very good for this. The Centronics cable must not be very long, 1.4 m cable has been found to be work properly.

6 Using the DSP CARD 3

6.1 General setup

The DSP CARD 3 is intented to work with a software that implements the ISS-interface protocol software, mainly Phil Karn's NOS [22], but of course other software could also be used.

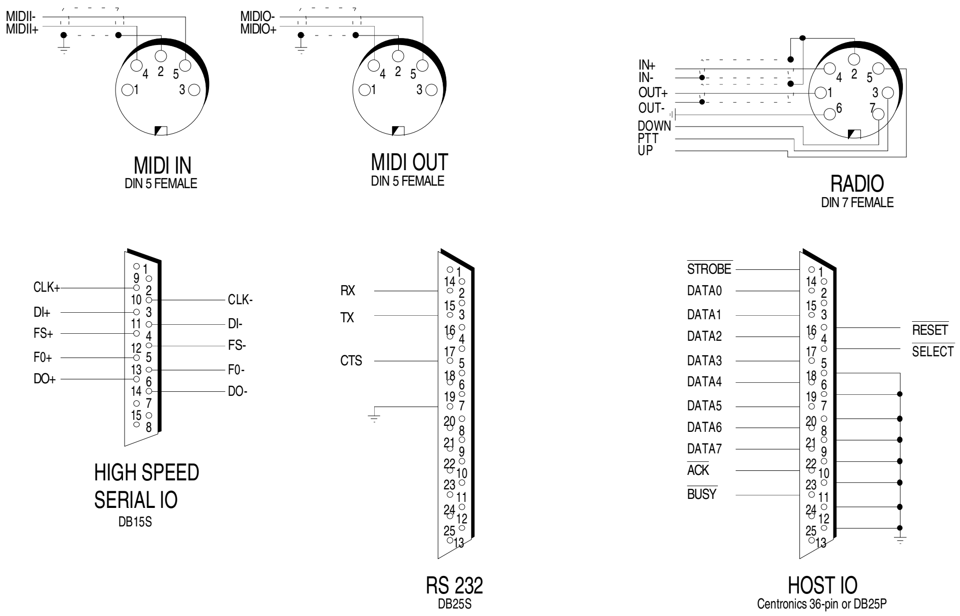


Figure 9: A proposal for DSP CARD 3 connector wirings.

The most usual setup is that the DSP CARD 3 is interfaced with Centronics and RS-cables to the PC, and audio and PTT lines to the radio. The PC boots the DSP CARD 3 and runs the needed protocol handling software.

6.2 Downloading programs

In order to download programs to the DSP CARD 3, a special downloader has been developed. It uses Centronics port LPT1 for communicating with the DSP CARD 3. It detects when the card is connected to the computer, so it can be used to test the Host/ DSP CARD 3 interface. The downloader program is called `DL.EXE`, and it downloads the given `.LOD` type file (the linker output file) to the DSP CARD 3 (see section 4.1 on page 7 for further details).

6.3 TST

These programs, `TST` and `MEM`, are special DSP CARD 3 test utilities. They are loaded with the command

```
DL TST
```

or

```
DL MEM
```

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 duplex, 1 - full duplex	
6	OutLevel	output signal level in 6/256 V_{pp} steps	3

Table 2: DSP TNC parameters. Parameter `FullDuplex` is not implemented, it is presented here for completeness only. Parameter P is scaled by 255, so the default value is actually $63/255 = 0.25$. Parameter `OutLevel` is scaled by $23.4(6/256)\,\mathrm{mV_{pp}}$, so the default value is $3\times(6/256)\mathrm{V_{pp}} = 70.3\mathrm{mV_{pp}}$.

6.4 Using various modems

Two modems are currently implemented and are ready to use: `CPFSK.LOD` is fully compatible Bell 202 TNC with serial KISS interface and `PAM.LOD` is fully compatible G3RUH and TNC with serial KISS interface. RS 232 (the speed is 4800 bit/s) interface should be connected to the appropriate node software (we have used Phil Karn's NOS). Analog input and output should be connected to the radio (no squelch) and PTT line should be connected to the radio PTT input. The programs are loaded with one of the following commands

```
DL CPFSK
```

or

```
DL PAM
```

after that the normal operation can be started, for example NOS can be started.

DSP TNCs has some standard parameters and one special parameter, see table 2. Note that the last parameter, `OutLevel`, is actually defined as `SetHardware` by standard [21]. The signal from the D/A converter is very strong. Thus the use of an external attenuator may be necessary.

6.5 Where can I get new software versions

Currently the new bits can be obtained from `ucsd.edu` directory `/hamradio/ka9q/dsp` or from `nic.funet.fi` directory `/pub/ham/dsp/packet`.

7 Testing the DSP CARD 3

7.1 Electrical testing

When the Centronics interface is connected to the host computer, and the power is connected to the DSP CARD 3, the card is ready for simple test. Download the testprogram to the DSP CARD 3 by command

```
DL TST
```

When everything is ok, the program completes with the message:

```
DSP CARD 3 program downloader (Sep 15 1990)
222,14,11 words to P,X,Y spaces downloaded
```

If the connection to the DSP CARD 3 is bad, the message is:

```
DSP CARD 3 program downloader (Sep 15 1990)
io error: No DSP CARD detected
```

In that case there may be one of the following malfunctions

- There is no power connected for the DSP CARD 3
- The cable is not connected or it is defective
- `BUSY` signal is not connected to the Centronics cable
- The latch chip (A6) or the DSP (A1) is defective
- There is a short circuit in the power rails

If the downloading process went ok, the the `TST.LOD` file is transferred to the DSP CARD 3 and it begins running. If everything is OK, the CTS-line should blink continuously and 'Hello World!!!' banner should appear on the RS-232 line (at 9600 bit/s) and the converter chip should produce sine wave. If no blinking is observed the problem could be one of the following

- The DSPRAM (A4) is damaged
- The DSP (A1) is damaged

When you are suspecting memory damage, you can use special memory tester, `MEM`. This program displays error data, so it may be possible to locate the error (e.g. some address line broken).

7.2 Mechanical testing

As a test procedure we have used a simple arrangement. The tests were conducted with periodic bicycle trips between Otaniemi and Tapiola test sites (distance approx. 2.5 km). Ambient temperature was around -10°C, effective wind speed was about 2 – 10 m/s and there was abt. 30 cm show on the test road. Despite a few fallings (fortunately only minor damage to the test person were reported after the test) the DSP CARD 3 was functioning perfectly.

Relationship	*Signed*	*Unsigned*
a $<$ x0	BLT	BLO
a $\leq$ x0	BLE	—
a $=$ x0	BEQ	BEQ
a $\neq$ x0	BNE	BNE
a $>$ x0	BGT	—
a $\geq$ x0	BGE	BHS

Table 3: This table shows which jump instructions will result in a jump taken when testing for a given relationship of a to x0 after a `CMP x0,a` instruction.

8 Programming with the DSP CARD 3

Devising and then implementing signal processing algorithms is a little different than 'normal' programming. First of all, *think analogwise when coding.* It very tempting, especially for 'real programmers', to use discrete like methods to accomplish the various needs for modems. For example, in order to remove the DC-level from the signal we first calculated the mean and then subracted it from the samples. This is the common way of doing when coding usual applications; devising a simple sequential algorithm. But in digital signal processing world, there is much more powerfull method—a simple high-pass filter [5]. Using first order IIR section and by selecting appropriate coefficients one can simply synthesize a filter that has infinity attenuation at the zero frequency (zero at the +1 position) and beautiful pass-band characteristics (pole at -1), so the filter actually removes the DC-component and is very easy to implement with DSP processors.

8.1 DSP56001

The 56001 User's manual [27] and application note considering the 56001 arithmetic [28] are essential when doing serious work with the 56001. This section is not a complete introduction to the 56001 However, in this section we present only some curious facts (and troubles) we have had when programming the 56001.

8.1.1 General

The syntax of two-operand instructions may be reversed from other machines one is used to. For instance, the 56001 instruction `move #1,a1` is equivivalent to `ld a1,#1` on some other machines. On the 56001, the destination register—the one affected by the instruction—is always second. The operand order for `CMP` instructions is also reversed from some other machines, so `cmp x0,a` means "compare a to x0".

8.1.2 Comparison operations

The 56001 provides the comparison operations shown in table 3. This includes all six possible relationships between two signed or unsigned numbers[11]. The distinction between signed and

[11]Comparing the word values 000006 and ffffe hexadecimal depends on how the numbers are interpreted. If they're signed numbers, 6 is greater than -2. But if they're addresses they are unsigned, and 000006 is lower

unsigned comparison comes up rarely, since they are the same unless one of the values involved has the high-order bit (the sign bit) set.

8.1.3 Addressing modes

Due to design difficulties and implementation tradeoffs, the 56001 doesn't have very rich addressing mode arsenal. There are only three main modes; register direct, (address) register indirect and special (actually an immediate and direct addressing modes) modes. There is no indexed mode, and one of the operands must be register. The special addressing mode is very curious, it contains both immediate and direct addressing modes. Actually one of us first believed that the 56001 doesn't have direct addressing mode at all, and always coded as follows

```
move    #$1234,r0
move    #0.34,x0
move    x0,x:(r0)
```

when with direct addressing can be coded like this

```
move    #0.34,x0
move    x0,x:$1234
```

In the special addressing mode group there are two important addressing modes, namely short forms of absolute address and immediate data. When the address or data is short enough (data 8-bit, address 6-bit, jump address 12-bit), it can be included in the program op-code and it doesn't need an additional word. This saves program memory and speeds execution considerably. Often it is needed to force the compiler to use the special short forms by '<' prefix, e.g. `jmp <loop`.

8.1.4 Jumps

The (conditional) jump instruction on the 56001 executes in two instruction cycles. The execution timing is the same whether the jump is taken or not. The program address generator fetches both jump targets (taken and not-taken addresses) but only decodes the chosen part[12].

8.1.5 Using parallelism

In order to make the 56001 really singing, it is very important to utilize its parallelism fully. Sometimes this is very easy, data movement goes side by side with ALU calculations, as is seen in the following example (CRC-calculation)

```
    move            #>1,x0
    and     x0,a    y:<rem,x0
    eor     x0,a
    lsr     a       #>poly,x0
    jcc     <_c1
    eor     x0,a
_c1 move            a1,y:<rem
```

address than ffffe.

[12] This is called *multiple prefetching.*

Sometimes it is advantageous to load register with parallel move even if it is not needed, as in the following

```
        cmp     x1,a    #>@cvi(@pow(2,15-1)),x1
        jne     <str4b
; yes, gain scaling (shift left 15 bits)
        mpy     x0,x1,a
```

8.1.6 Arithmetic and data representation

The most curious thing on the 56001 is its support for fractional arithmetic. Fractional numbers have a little different binary representation than the 'normal' integer numbers, and this may cause a little confusion. It is very important to note that 56001 assembler tries to use short immediate addressing mode whenever it can, and depending on the destination register the 56001 interprets this data as fractional and therefore stores it left-side justified. For example `move #56,x0` stores 380000 hexadecimal to register x0 which is clearly incorrect. If the immediate long force operator '>' is used, an extension word will be used, and the data will be right-side justified; `move #>56,x0` stores 38 hexadecimal to register x0 which is correct. When the destination register is other than x0,x1,y0,y1,a or b the 56001 doesn't interpret the data as fractional, so `move #3,a1` is correct, and in fact it is quite beneficial to use short forms of immediate data whenever it is possible.

It is possible to enter both integer form and fractional form data to 56001 assembler. When entering the number -1 it is very important to specify what kind of number it is, because fractional -1 is 800000 hexadecimal and integer -1 is fffffe hexadecimal. So whenever using integer -1, enter -1, and whenever using fractional -1, enter -1.0.

When multiplying two **integer** numbers togheter, it is important to note that the result must be right sifted one bit to the left, and the result is in the lowest order word.

One interesting fact about signal processors is the fact that the multiply instruction doesn't take very long time. So when doing multiple shifts, it is advantageous to use one multiply instruction rather than many shift operations.

8.2 Some methodology

The most common way to program the DSP56001 is by assembler. The assembler produced by Motorola, ASM56000, and the accompaning linker, LNK56000, and simulator, SIM56000 by Motorola, are in public domain.

The complete program can be made from separate modules which are linked together. This ensures modularity and maintainebility of the program. The generic modem skeleton is made for example from many different modules which are combined with the users own `MODEM` module. This saves also in compile time, because we need compile only the user's own modules.

So the programming with the modem skeleton begins by programming the `MODEM` module[13]. When the initial coding phase is over, we can start compiling. There is maintenance file `MODEMS` for Microsoft MAKE Utility, with which the program compiling and linking process can be automated a little. So we can simple say `MAKE MODEMS` and the appropriate files are compiled

[13]Actually there are no separate `MODEM` file, one can copy the `CPFSK` module instead.

and linked togheter. Some modifications may be needed in `MODEM.MEM` file that controls the linker memory allocation.

When the editing, compiling and linking phase is over and there are no syntax errors around, we are ready to test the program. We can do it with the simulator program. It is often quite benefical to use the simulator, because it has very powerful utilities to test and debug the programs.

After the simulator phase we can finally transfer the program to the DSP CARD 3. This is mady by the `DL filename` command.

8.3 So you want to build a modem

We have tried to warn you not to build a modem by your own. But we know that somewhere there are beings who don't make what they can, but what they want to[14]. So, when you want to build a completely new modem, the first thing to do is to study some theory because they don't teach us stuff like digital communication techniques in an elementary school. Do some theoretical framework to find out if there are any theoretical base for your new modem construction and how to possibly implement those ideas. After devising the needed algorithms it is time to simulate (if possible) those algorithms to ensure their performance. There are plenty of algorithmic simulation tools available (we used Comdisco's SPW, but Matlab can do also, for example). After the algorithms are OK, then it is time to code them to the DSP56001. Beware the aspects mentioned in section 8.1. Then simulate your code with `SIM56000`—you can use `SHOW` to display important parameters, timing error for example to detect symbol syncros performance. After all those steps it is the time to port your code to the modems skeleton and testing it with the real signals. Sometimes it is also useful in this phase to retrieve some data from the modem using the datalogger mode in `DL` for viewing the important parameters for fine tunig them. The data for figure 7 was obtained with `DL`[15]. From that figure it is easy to ensure that the adaptation process is working.

9 Overview of Alef Null ideology

> "Muuan mies Sysmästä,
> tunnettu oli taidosta.
> Signaalinselittäjä hän lie,
> ilosanomaa evankelioiden vie."

The mainstream of ALEF NULL ideology is minimalism. Minimalism first emerged in art on 1960's. Mostly those works were three dimensional, contained extremly simple modules, were aestethically simple and were anonymous in a sense that there were no worker identification. One of the most impressive musical composition is John Cage's 4'33" which contains only thirteen pauses. In general those minimalist compositions contain no beginning and no end, they are only meditative. In litterature the most famous novel is [35]. One interesting ensemble [36] (Zero Point) that contains only empty pages.

Minimalism in this DSP CARD 3 is clearly shown by its simplicity; the chip count is under ten. Despite of this simplicity, it is altough quite versatile.

[14] And that's good.

[15] Logged data was then read to MATLAB for plotting and then included to this document in PostScript form.

Year	*Actions*
1980	first general purpose DSP chip introduced
1983	DSP technology in widespread use
1986 spring	first time speculation of the DSP in Alef Null
1986 autumn	TMS32010 studying
1987	DSP CARD 1: M68HC11 & TMS320C20
1988	DSP CARD 2: TMS320C25
1989	DSP CARD 3: DSP56001 designing begins
1990 spring	first DSP-program working
1990 summer	DSP CARD 3 Rev 1.2 working
1990 autumn	DSP CARD 3 Rev 1.3 working
1991 summer	G3RUH modem working

Table 4: Alef Null history of DSP development.

10 History

For those interested, this section explains the history of DSP-dreams of the ALEF NULL Group.

The first general purpose DSP chip was μPD7720 that appeared on the market on 1980. Intel's 2920, introduced as early as 1979, lacked a hardware multiplier, so the 2920 did not qualify as a modern programmable DSP. In 1982, Texas Instruments introduced the TMS32010 as the first member of what was to become the most popular DSP family. In 1983 there were several articles in EDN magazine that discussed several topics in digital signal prosessing.

After six years the first general purpose DSP was introduced, and three years after the DSP methods were in widespread use, we heard first time the word DSP from some amateur sources and from those old magazines like EDN. So in 1986 we knew nothing about digital signal processing, but intensive, and fairly fanatic, DSP research began. First we read several books on the DSP-theory and designed (but we never implemented any of them) various systems with Texas chips. In 1989 we heard some noise from USA where hams were using Motorola DSP56001 processor. We studied it, and immediately liked the beautiful architecture and easy interfacing. So the first version we actually realized was DSP CARD 3 with Motorola DSP56001. First DSP-program we created was an complex correlator which with simulator was found to be working at 04/18/1990. At 06/18/1990 11:30pm the first alpha version (revision 1.2) of DSP56001 based card was found working. We made some modifications to it (MIDI-interface, E1 size board, etc.), and the beta series (revision 1.3) was completed. After struggling several dificulties with noisy high-speed static RAM, we got the Rev 1.3 working at 12/10/1990.

11 Conclusions and Future Speculation

Currently it seems that the chosen design paradigm—minimalism—has been quite succesful. The resulted board is very useful and quite cheap. The selected processor, Motorola DSP56001, has shown to be very powerful and easy to program.

Some worries are although around, because nothing new has been coming from Motorola in the DSP field and Texas Instruments has been announced its new C50 high-speed fixed point chip and Analog Devices has introduced its ADSP-2111 chip. Just now the Motorola 56001 seems to be a competitive chip, but in the future there may be a severe push from the Texas direction. Lately Motorola introduced the 56116 chip which has same capabilities as the ADSP-2111 and the C50, but we'll see in future how good the Motorola's support for that device is.

Currently there are only few simple modem programs available. But we have developed a little modem skeleton to which it is quite easy to add other modem types. Currently we are working on PSK modem.

References

[1] Ludeman, L., C., "Fundamentals of Digital Signal Processing", Harper & Row, 1986

[2] Oppenheim, A., V., Willsky, A., S., Young, I., T., "Signals and Systems", Prentice-Hall, 1983

[3] Oppenheim, A., V., Schafer, R., W., "Digital Signal Processing", Prentice-Hall, 1975

[4] Proakis, J., G., "Introduction to Digital Signal Processing", Macmillan, 1988

[5] Jackson, L., B., "Digital Filters and Signal Processing", Kluwer, 1986

[6] Candy, J., V., "Signal Processing, The Modern Approach", McGraw-Hill, 1988

[7] Haykin, S., "Modern Filters", MacMillan, 1989

[8] Anderson, B., D., O., Moore, J., B., "Optimal Filtering", Prentice-Hall, 1979

[9] Alexander, S., T., "Adaptive Signal Processing", Springer-Verlag, 1986

[10] Carlson, A., B., "Communication Systems", McGraw-Hill, 1986

[11] Lee, E., A., Messerschmitt, D., G., "Digital Communication", Kluwer, 1990

[12] Bingham, J., A., C., "The Theory and Practice of Modem Design", Wiley, 1988

[13] Proakis, J., G., "Digital Communications", McGraw-Hill, 1983

[14] Van Trees, H., L., "Detection, estimation and modulation theory", Wiley, 1968

[15] Vaidyarathan, P., P., "Multirate Digital Filters, Filter Banks, Polyphase Networks, and Applications: A Tutorial", Proc. of the IEEE, Vol. 78, No. 1, January 1987, pp. 56–93

[16] Aulin, T., Sundberg, C-E., W., "Continuous Phase Modulation — Part I: Full Response Signaling", IEEE Trans. on Communications, Vol. 29, No. 3, March 1981, pp. 196–209

[17] Osborne, W., P., Lunz, M., B., "Coherent and Noncoherent Detection of CPFSK", IEEE Trans. on Communications, Vol. COM-22, No. 8, August 1974, pp. 1023–1036

[18] Raveendra, K., R., Srinivasan, R., "Threshold detection of continuous phase-modulated signals", IEE Proceedings-I, Communications, Speech and Vision, Vol. 136, No. 6, December 1989, pp. 414–423

[19] Glišić, S., G., "Symbol Synchronization in Digital Communication Systems Using Partial Response CPM Signaling", IEEE Trans. on Communications, Vol. 37, No. 3, March 1989, pp. 298–308

[20] Stallings, W., "Data and Computer Communications", MacMillan, 1988

[21] Chepponis, M., Karn, P., "The KISS TNC: A simple Host-to-TNC communications protocol", Proceedings of the sixth ARRL computer networking conference, August 1987

[22] Garbee, B., "The KA9Q Internet Software Package", TAPR Inc., 1989

[23] Travis, B., "Single-chip DSPs advance in speed and versatility", EDN, 1989, October 12, pp. 125–134

[24] Shear, D., "EDN's DSP Benchmarks", EDN, 1988, September 29, pp. 126–148

[25] Kloker, K., L., "The Motorola DSP56000 Digital Signal Processor", IEEE Micro, 1986, December, pp. 29–48

[26] "DSP56001 56-bit General Purpose Digital Signal Processor, Technical Data", Motorola, 1988

[27] "DSP56000/DSP56001 Digital Signal Processor User's Manual", Motorola, 1989

[28] Chrysafis, A., Lansdowne, S., "Fractional and Integer Arithmetic Using the DSP56000 Family of General-Purpose Digital Signal Processors", Motorola Application note APR3, 1988

[29] Vuori, J. "Design and construction of a packet radio modem with a digital signal processor" (in Finnish), Master's Thesis, Helsinki University of Technology, 1990

[30] Miller, J., R., "The Shape of Bits to Come", Oscar News, 1991, pp. 29–37

[31] Miller, J., R., "9600 Baud Packet Radio Modem PCB", February 1989

[32] "Product Preview, DSPRAM: 8K×24 Bit Fast Static Ram", Motorola, 1989

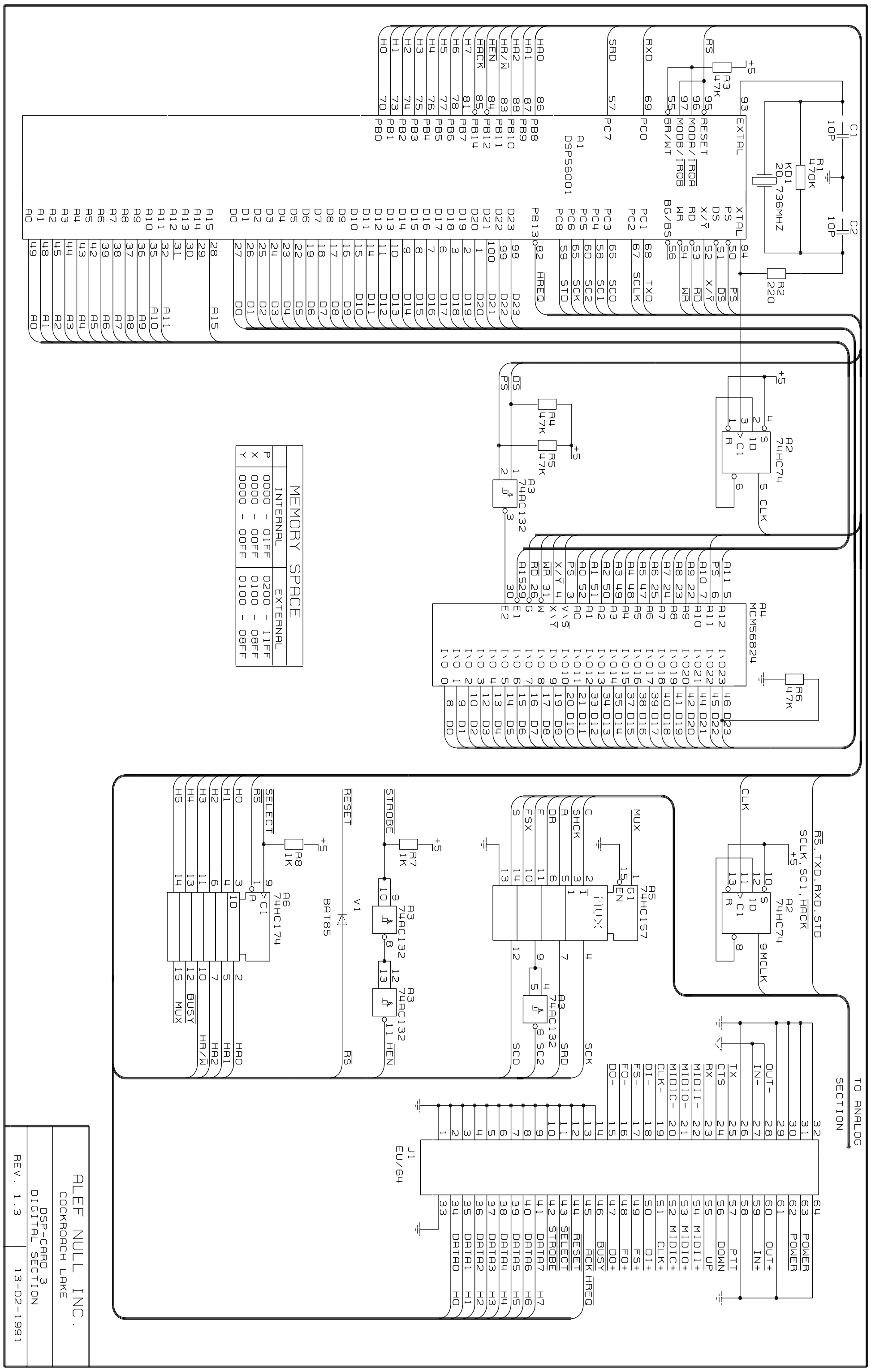
[33] "Linear Circuits, Data Acquisition and Conversion", Texas Instruments Data Book, Vol. 2, 1989

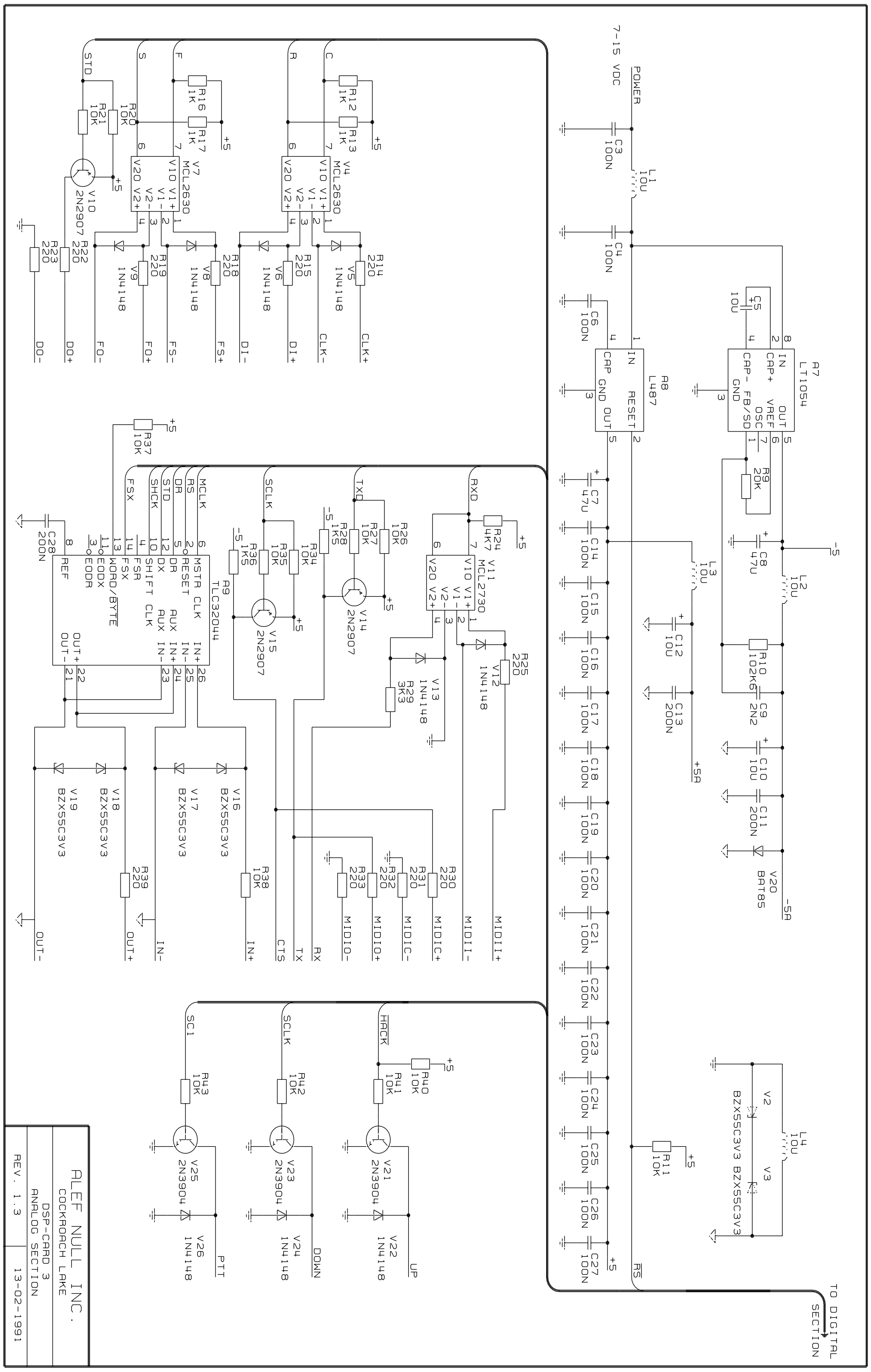
[34] "MIDI 1.0 Specification", International MIDI Association (IMA), 1983

[35] Easton Ellis, B., "Less Than Zero", 1985

[36] Jokinen, O., "Nollapiste", 1965

A Schematics




ALEF NULL INC.
COCKROACH LAKE
DSP-CARD 3
ANALOG SECTION
REV. 1.3
13-02-1991
POWER
7-15 VDC
TO DIGITAL SECTION

B Assembly Drawing