CONSTRUCTION AND USE OF THE DSP COMPUTER

1. Construction Overview

Although Digital Signal Processing was introduced to simplify the hardware required and thus reduce costs, some hardware still has to be built. Assembling a computer type circuit is certainly not as demanding as building a radio-frequency analog circuit, especially not from ready-made building blocks, like microprocessors, peripherals or memories.

However, these ready-made building blocks do not always conform to the published specifications, especially when there are more manufacturers offering apparently interchangeable products. Even with products from the same manufacturer there are sometimes sudden changes due to a new manufacturing process or simply a spread of important parameters well in excess of what guaranteed on the data sheets. During circuit design, every effort was made to try as many different products as possible and the results are presented as a recommended "component selection".

After assembly, the single modules have to be checked for correct operation. Some modules may require a few alignments of the on-board trimmers. Then the complete circuit has to be installed into a suitable case. To help correctly choose the latter, the mechanical, thermal and electrical constraints will be discussed.

Installing the software includes the adjustment of the various parameters to other analog hardware available, like transmitter modulation levels or packet-radio delays. Since source files will be made available, it is also possible to "personalize" the software adding minor but important features.

A few hints how to write DSP software are also described in this article. However, most users will probably find it necessary to refer to literature describing the microprocessor and peripherals used. Unfortunately there is not much literature discussing DSP techniques on an amateur level yet. Hopefully this will change in the near future.

2. Selection of Components

Throughout the DSP computer design and development I tried to use readily available components. Overall cost was also an important consideration.

Most computer components are integrated circuits, either complex function blocks like the microprocessor or memories or simple TTL and CMOS SSI gates and flip-flops. The latter belong essentially to three logic families. The bipolar LS series is used for most functions. The new HC CMOS series is used where both speed and low power drain in stand-by are required. The old 40xx CMOS series is used where speed is not critical. Finally, there are also a few ICs from other logic families where required.

Considering the decreasing prices of the HC family it is already convenient to replace the old LS series with the better HC series. If all LSs are changed to HCs, no additional pull-up resistors are required and many 3.3 khom pull-up resistors can be omitted. In fact, prototypes built with all HC circuits operated better than those with LS circuits. HC ICs have a higher noise margin and a stable output impedance compared to the LS series: both contribute to suppress crosstalk and ringing of the computer bus. An "all HC" design is recommended for CPU clock frequencies above 10 MHz.

The 68010 microprocessor is manufactured by Motorola, Signetics, Thomson-CSF, Mostek and probably some other manufacturers. The MC68010 is usually available selected according to the maximum guaranteed clock frequency of 8 MHz, 10 MHz and 12.5 MHz. I only had available 11 samples from Motorola with different specified clock frequencies, different packages and different manufacturing dates and mask-set identifiers. In practice I noticed a very large spread of the maximum clock frequency: guaranteed 10 MHz samples reached from 9.5 MHz up to 15 MHz before crashing the software. For comparison, the 8 MHz sample reached 12.5 MHz and the 12.5 MHz sample reached 14.5 MHz. Some experimenting may therefore be very worthwhile! The maximum clock frequency decreases quickly with increasing chip temperature and also depends on the clock waveform. The best results were obtained with a Valvo 74HC00 as the clock oscillator.

The video board uses special dual-port memories. In the prototypes I successfully tested both uPD41264 (NEC) and HM53461 (Hitachi) with identical results. Both 120ns and 150ns selections worked at 10 MHz CPU clock. The 120ns version is of course recommended for higher CPU clock frequencies. No problems were ever noticed using a refresh rate of 20ms (TV frame) in place of the 4ms required by the manufacturers.
The DAC0800 D/A converter is available from many suppliers.

The main computer memory uses static CMOS memories. Over 120 NEC uPD43256 120ns chips were tested without problems. The memory boards also worked with similar memories from other manufacturers. For processor clock frequencies above 10 MHz 100ns access time memories are recommended.

The NEC 710xx series peripherals are CMOS versions of the famous 82xx series of microprocessor peripherals. The uPD71055 parallel I/O port on the processor board can be replaced with a NMOS 8255, the latter sometimes requires a 330pF capacitor to delay the CS\ signal for proper operation. The uPD71054 programmable counter can be straightforward replaced with a 8253 chip. On the other hand, the uPD71051 USART can not be replaced with a NMOS 8251, since the latter is unable to operate at a 6.144 MHz clock frequency. A 82C51 could work, but was not tested. The uPD4990 real-time calender/clock chip is also manufactured by NEC.

Suitable CODEC chips are manufactured by Mostek: MK5156 and by SGS: M5156. The SGS samples had a lower DC offset in the A/D section than the Mostek parts. No differences were noted between the switched capacitor filters manufactured by National Semiconductor: TP3040 and its Thomson-CSF equivalent: ETC5040.

The floppy disk controller IC is manufactured by Western Digital: WD2797 and by Siemens: SAB2797. The serial communications controller is available from Zilog: Z8530 and from Sharp: LH8530. The same companies also manufacture a CMOS Z80CPU (used in the rotator interface) marked Z84C00 and LH5080 respectively. The ADC0804 is manufactured by National Semiconductor and equivalents are available from other manufacturers too. For accurate tracking it is recommended to use the more accurate versions of the same chip marked ADC0802 or ADC0801.

The computer modules include many crystal oscillators. Only two oscillators really need to be accurate and have trimmers for fine frequency correction: the real-time clock chip with a 32.768 kHz crystal and the 6144 kHz crystal on the analog I/O board that defines the sampling rates. Both these crystals are fundamental mode, 20pF parallel resonance. All the remaining crystal oscillators at 10 MHz (CPU), 12 MHz (video), 16 MHz (FDC), 6.144 MHz (serial I/O) and 10 MHz (rotator interface) need not be very accurate. "Computer-grade" fundamental mode crystals are adequate and no fine adjustment trimmers are provided.

Finally, the computer also requires a few larger units: a keyboard, a floppy drive and a TV monitor. The keyboard should be a standard ASCII keyboard with a parallel output including 7 data bits and a negative-going strobe. A keyboard that only requires a +5V supply is preferred.

The 3.5" floppy drive should be suitable for double-sided 80 track floppies. The spindle motor speed should be 300rpm. The raw capacity is 1 Mbyte at 250 kbps MFM. I successfully tested three different floppy drives: NEC FD1036 and FD1037 and Matsushita JU363. Older models, like the FD1036 or the JU363, require two supply voltages, +5V and +12V. New models, like the FD1037, operate from a single +5V supply but do not provide a READY signal (move the corresponding jumper on the floppy controller board). Warning! Although all three floppy drives use the same 34 pole connector, the connections on one drive were found to be a "mirror image" of another drive!

The video board will drive any (European) standard TV monitor. Only the vertical sync and size may need to be adjusted to the slightly different frame frequency. Due to the resolution of the picture displayed, a 12" or larger screen is recommended. If possible, the monitor should be installed in a metal case to avoid interferences to sensitive communications receivers.

3. Installation of Components

The processor board, the video board, the memory, the analog I/O and floppy controller are all built on double sided printed circuit boards, 170mm long and 120mm wide. All modules have a 64 pole "Euro-card" male right angle connectors with rows A and C equipped with pins and row B empty. The diameters of the two mounting holes on each connector have to be slightly enlarged to accept two 10mm long M3 screws to hold the connector in place on the printed circuit board.

The detailed component location plans of the above modules are shown on Fig.1, Fig.2, Fig.3, Fig.4 and Fig.5 respectively. "*" symbols denote feed-through holes: no component leads are installed in this holes, their only purpose is to connect traces on different sides of the printed circuit board.

All integrated circuits can be installed on (good quality) sockets. Use the type with round lathed contacts with gold plated springs inside. For an eventual troubleshooting it is only important to have the complex function ICs on sockets, like the microprocessor or peripherals, it does not help much to have all the TTLs or 40xx series CMOS on sockets.

All integrated circuits used in the DSP computer may be damaged by electrostatic discharges. While CMOS and NMOS circuits have protection networks to reduce the probability of damage, new bipolar circuits with small geometries are also sensitive to static discharges. The best way to prevent these problems is to understand the damage mechanism. A certain amount of energy is required to destroy an integrated circuit. Since the induced electrostatic voltage is limited by effects like corona discharge, the only variable left is the capacity of the capacitor holding the charge. For example, simply touching a pin of an unconnected IC can never be harmful: the capacity of the relatively small IC package towards its surroundings is very small. On the other hand, if the IC is installed in a circuit and the circuit is grounded or connected to another high capacity object, the energy may be sufficient to destroy a bipolar TTL IC. Therefore, I do not recommend using conductive ground planes for working surfaces or grounding wrist straps since in the case of a mistake, the latter may be dangerous for the operator too. Simply avoiding "dangerous" situations, like a high capacity to high capacity discharge through a sensitive component like the one described above, I never had even a single failure of a MOS IC to be attributed to a static discharge!

IC sockets are also used as jumper plugs (to select the interrupts) and as connectors. A very simple but reliable connector joint can be built from two identical IC sockets. One socket is soldered into the printed circuit board and is used as a female connector. The other socket is used as a male connector with wires soldered in the holes for the IC pins.

The power supply is built on a single-sided printed circuit board 120mm long and 100mm wide. Since it includes a few heavier components, like the NiCd batteries or the ferrite pot core and requires a few connections, only 4 wires, with the computer bus, the unit is not equipped with an "Euro-card" connector. It is to be installed separately using four M3 screws in the corners.

All electrolytic capacitors are vertical types. The NiCd cells are installed horizontally The connections are made soldering two pieces of 1mm thick copper wire to each battery cell terminals which also keep the cell in place. The BDX34 power Darlington transistor is installed on a small heat sink with a M3 screw.

The (approximately) 100uH 5A chokes include 25 turns of 1mm thick enameled copper wire wound on a 5mm diameter ferrite screw. The storage coil L is wound on a 30mm diameter pot core with an approximately 0.5mm thick air gap. It has 10 turns of four times 0.5mm enameled copper wire.

The bus board only includes female "Euro-card" connectors and supply bypass capacitors. The location of the latter is shown on Fig.7. The board itself is made of 2.5mm thick laminate for mechanical strength. The connector pins should be 4mm long for easy soldering. The connectors are held in place with two 8mm long M3 screws. It is very important to install these screws first, before soldering the pins, otherwise the sensitive springs inside the female connectors will be damaged resulting in intermittent contacts!

The rotator interface board, shown on Fig.8, is a stand-alone board but otherwise the installation of the components is identical to other computer modules. If all the integrated circuits used are CMOS versions as recommended, then the power drain is so low that the 7805 regulator does not require a heat sink.

4. Module Checkout and Alignment

It is not recommended to assemble the complete computer with all the modules and apply the supply voltage. With such a complex circuit chances that everything will work immediately are very small. Further, a malfunction in one of the modules, especially the power supply, could damage many expensive components on other modules. Finally, troubleshooting may be impossible in these conditions.

After assembling the single modules, an accurate visual inspection of the printed circuit boards is highly recommended. My personal experience is that most mistakes are made during soldering: either missed connections or accidental shorts between adjacent lines. After this the modules should be checked in a reasonable order: using already tested modules to check another module.

The first module to be tested is obviously the power supply. Its input should be connected to an adjustable voltage DC source to test both the switching regulator and the reset circuit. Due to manufacturing tolerances of Zener diodes it may be necessary to change the nominally 820 ohm resistor (*) in series with the Zener diode to obtain the correct output voltage of 5V (under load!).

The reset circuit should have the required hysteresis and delay. The voltage across the NiCd battery should also be verified. Finally, the 1N5822 Schottky diode used to charge the battery should be verified for leakage current (main power off) It should not be more than 20uA measuring the voltage fall across the 120ohm resistor. Do not unsolder the diode just for measuring purposes since the leakage current is very temperature dependent.

Next, the bus mother board has to be checked. The latter has a very large number of close solder joints. It is recommended to test it with an ohmmeter for shorts before connecting the four wires to the power supply.

Whenever inserting modules in the bus board, keep in mind that some supply voltages are always present, even with the main power off, due to the NiCd battery. The best way to insert a module with an "Euro-card" connector is to make the ground connection first. Since the latter is on the opposite side of the supply contacts, it is sufficient to remember to always plug-in the modules under such an angle that the ground connection is made first. Before plugging-in a module, a static discharge, touching both circuit grounds at the same time with naked hands is recommended too.

The video board can now be plugged in the bus. Connect a TV monitor to the video output. Due to the random content of the dynamic RAMs after power-up, the video board should produce a rectangle containing a random pattern and surrounded by a black border on the TV screen. Adjust the horizontal position trimmer to center the picture. Also adjust the various commands on the TV monitor to obtain a reasonable quality TV picture. Also check the operation of the on-board flyback inverter, which should produce a negative supply voltage of about -4V.

Once the video board is working the processor board can be added. Of course turn the power off before plugging in or out any module! Do not connect a keyboard to the processor board, nor the partial/total reset switch. The latter connection should be left open to enable a total reset. If everything is working properly, the random pattern on the TV monitor should disappear about one second after power-up (reset delay) and be replaced by a help message text covering almost all the screen. A blinking cursor should also appear.

If the random pattern remains unchanged there is a problem on the processor board or on the video board that has to be repaired before you can proceed with other modules. In this case it is useful to check the behavior of the HALT\ line on the bus: a double bus error makes it go low. Please remember that for a correct start-up the software required is contained in the 27128 (or 27C256) EPROM. The operation can not be checked without a programmed EPROM!

When the processor board is working correctly the keyboard can be connected. Under the operating system command mode, all the printable characters should be printed on the screen up to the maximum allowed line length. Any control character is interpreted as a carriage return. Any letters will only be printed in upper case in the command mode. The code required for the delete function is 7FH. After this, try to issue some simple commands as described in the operating system manual. Avoid commands that use peripherals that are not installed yet, like the floppy disk or the RS-232 port.

The operation of the real-time calender/clock can be checked using the corresponding operating system command "U". It is a little more difficult to adjust the 32.768 kHz crystal to the correct frequency, since the oscillator circuit is a very high impedance circuit. The simplest way is to leave the trimmer in the middle position and correct the adjustment if a clock drift is detected a few days later.

The processor board carries a nominally 10 MHz crystal oscillator and the MC68010 should also be a 10 MHz version. However, it is very useful to test your processor for its maximum speed. This can simply be done by replacing the crystal in the clock oscillator. Of course, the computer can not be operated reliably at the maximum clock frequency. The latter will be further reduced when loading the bus with additional modules. A safety margin of about 2 MHz below the maximum frequency was found appropriate in most occasions. All the software was designed for a minimum clock frequency of 9 MHz, but a higher clock frequency is very desirable.

The remaining modules are easy to test once the processor is operating, since the latter can also be used as a powerful troubleshooting tool. The easiest to test are the memory modules. When installing the memory modules do not forget to program each of them for a different address range. Further, all the modules should form a contiguous memory area to be used efficiently by the operating system software.

To test a memory module, try the "W" instruction first to write to or read from a few memory locations. Most defects are discovered in this way. However, it is a little impractical to test 1 Mbyte of memory in this way. The final check can be done later, for instance by loading a few files from a floppy disk.

The analog I/O board first requires a few checks on its own. The on-board voltage inverter should supply a negative voltage of -5V. The crystal oscillator has to be adjusted exactly to 6.144 MHz with a frequency counter only connected after a buffer gate. If it is not possible to adjust the frequency with the given tuning range of the trimmer capacitor, then the fixed capacitor in parallel with the trimmer should be replaced.

The operating system software does only include routines to use the RS-232 port on the same module. The analog I/O could be tested issuing a number of "W" commands to initialize the appropriate registers, but it is usually easier to load a DSP program that uses the analog I/O port. The A/D converter offset can be adjusted when running a program that is sensitive to A/D offset errors, like the APT picture receiving program.

Before installing the floppy disk controller module, the floppy disk drive manual has to be consulted. All the connections between the drive and the controller should be verified and the jumpers adjusted for the drive used.

The floppy controller itself requires three adjustments: VCO frequency, read pulse width and write pre-compensation. All three adjustments are to be performed by connecting the TEST\ input (pin 22) to GND only AFTER reset has been applied to the computer. Grounding TEST\ while reset is active will program the 2797 FDC for a different mode of operation! The divided VCO frequency, 250 kHz, can be measured on pin 16 (DIRC) with a frequency counter and adjusted with the VCO's capacitive trimmer. Then the RPW duty cycle can be observed with an oscilloscope on pin 29 (TG43) and has to be set to 12.5% using the 47kohm trimmer.

The write pre-compensation pulse width can be observed on pin 31 (WD). Since however the floppy drive manuals seldom specify this value an experimental adjustment is required. Leave the 10kohm trimmer in center position and disconnect the

TEST\ line from GND. Insert a new, empty floppy disk and try to format it as described in the operating system manual. An improper write pre-compensation will result in formatting errors appearing on the innermost tracks, usually on tracks 75 to 79. In the case of errors, try another setting of the 10kohm trimmer and retry formatting tracks 75 to 79 until all the tracks format properly.

The rotator interface only carries a single trimmer. However, there are two trimmers in the rotator control unit and eight variables in the tracking program to be adjusted. All these adjustments are to be made with the actual rotator with the antennas installed. To understand all these adjustments an insight in the operation of the automatic tracking program is required.

5. Housing Constraints

After assembly, the competed computer has to be installed in a suitable case. There are however mechanical, thermal and electrical constraints. The main mechanical requirement is to build a suitable guide structure to keep the modules in their respective connectors. Even when the computer is being transported to another location it is very inconvenient if all the memory content is lost due to an intermittent contact on the bus.

The electrical performances of all integrated circuits used in the computer degrade quickly with increasing ambient temperature. In particular, the operating speeds of all logic families decrease with temperature. It is therefore highly desirable to keep the ambient temperature and the temperatures of the single chips as low as practical.

It is very difficult to install heat sinks on all integrated circuits. The printed circuit boards can however be installed in a vertical position to allow an efficient air flow. Finally, the power supply unit (another heat source) should not be installed close to the processor board (the most sensitive to heat).

A 16 bit microcomputer has at least twice the number of connections compared to an 8 bit microcomputer. A 16 bit computer also operates at a speed an order of magnitude faster than an 8 bit machine. The most immediate result is that a 16 bit microcomputer radiates 20 to 30dB stronger radio interferences when compared to an 8 bit microcomputer. Since a DSP computer is designed to work with communications equipment including sensitive receivers, a very accurate shielding is absolutely necessary.

The computer has to be installed in a metal (aluminum) box. It is important that the box is completely closed and that the covers make good electrical contacts with the box frame on all edges. The latter should therefore not be painted or anodized. All the connections should be bypassed and/or should use shielded cables. Bypass capacitors should be installed immediately on the connectors and ground conductors should be grounded on the connectors too, otherwise they will work as coupling loops for the disturbs. The keyboard connections, the video output, the RS-232 port and the high-speed serial port can not have bypass capacitors and the use of shielded cables is mandatory.

ASCII keyboards now use a single chip microcomputer to scan the keys and generate the codes. The latter could also cause some radio frequency interference since keyboards are usually packaged in non-conductive plastic cases, although the switching frequency is low and the disturb level is low too. A metal case keyboard is the best solution. A valid alternative is an old, surplus keyboard, with a dumb diode matrix logic, which can not generate RFI either.

A more important source of interferences is the TV monitor, if packaged in a plastic case. It will generally only cause interference when fed with a computer generated video signal. The latter is amplified to about 20 to 30 Vpp to drive the CRT. A very efficient but cheap solution to limit interferences at VHF and higher frequencies is to filter the video signal before sending it to the TV monitor. Since the video signal does not contain useful information above 10 MHz, the spectrum above 10 MHz can simply be filtered away with a low-pass filter greatly reducing the interference without degrading the picture at all.

6. Software Installation

Software installation is relatively simple compared to commercial computers. The operating system is completely contained in an EPROM. The actual version V7.1 is about 15 kbytes long and was written directly in MC68010 machine code. The EPROM can be a 27128 or a 27C256 programmed to fit in the same socket without modifications, since the 27128 and in particular the 27C128 are more expensive and not always available. For future upgrades the CPU printed circuit board already carries an additional address line to three solder pads close to the EPROM socket. The circuit can be thus modified to use all the 32 kbytes of a 27C256. Note however that the 27C256 EPROMs supplied with the V7.1 software are programmed to fit into an unmodified circuit (A14 always high) and will not work if this modification is made.

After power-up the action taken by the computer depends on the position of the partial/total reset switch, which should be installed on the front panel. In the case of a first power-up (or after a catastrophic software crash...) the computer requires a total reset. During a total reset, the whole content of the EPROM is copied into the nonvolatile system RAM. All system variables are thus reset to their default values. The operating system software is always executed in the RAM, since the latter has a much lower access time than the EPROM and does not introduce any wait states.

The partial/total reset switch is then moved to the partial reset position for normal operation. In this case the computer assumes the RAM already contains valid software and only a small part of the content of the EPROM is copied into the corresponding part of the RAM. Most of the software in the RAM is left unchanged including the operating system parameters.

After a total reset some operating system parameters should be set. The most important is to tell the computer the location and amount of RAM available using the operating system command "N". This and other operating system commands are described in detail in the "Operating System Manual". The latter can not be published since it is too long but will be made available separately. A short description of all available commands can be obtained by issuing the "H" (Help) command. The latter is also programmed as the default autostart command.

Application programs are supplied on floppy disks. Both source program files and compiled executable files are usually made available. Application programs have extensive menus and/or help messages activated only when a wrong command is issued. To use a program it is only necessary to load the compiled (.EXE) file from the floppy disk into the RAM (command "L"). A program can then be executed using the command "R".

DSP programs contain many variables - parameters to be set by the user. In the supplied application programs these variables have been set according to my own analog hardware. Some programs also include an option in the main menu to reset all variables to their default values.

Source program files are well commented to provide a better understanding of the operation and should enable users to modify the programs. Source program files are ASCII files that can be handled by any text editor program, including the text editor built in the operating system. The high level language syntax is explained in the "Operating System Manual".

Source files usually only contain symbolic variable names. After compiling a program, the compiler will also assign memory space to store the actual variable values inside the compiled executable file. The initial content of the variables is thus completely random. However, if an executable file with the same name already existed in the RAM directory, the compiler will try to copy all variable values into the new executable file. This is very useful since for example, after a small modification of the satellite tracking program it is not necessary to type in the Keplerian elements for 40 satellites again (only a few hours of typing...).

Finally, if an application program tries to use an inexistent peripheral or runs out of memory, the operating system will stop the execution in an orderly way indicating the bus error address, type of access and program counter value.

7. Writing DSP Software

DSP programming usually includes both programming in machine code for the DSP algorithm itself and programming in a high-level language to support and use the DSP routines. To recall the DSP routines periodically interrupts have to be used. The following discussion describes how this can be done efficiently on the DSP computer described.

When a MC68010 microprocessor receives an interrupt request with a priority level higher than the current processor priority (described with the corresponding bits in the status register) a call to the interrupt handling routine is made. Start addresses of interrupt handling routines are stored together with other exception vectors in the exception vector table in the computer memory. The exception vector table is located in the system RAM during normal program execution (except at power-up). Its start address is stored in the Vector Base Register (VBR).

Hardwired logic on the processor board makes the MC68010 to operate in the auto-vector mode: each interrupt level is assigned a single vector in the exception vector table. After reset, all interrupt vectors are pointing to an error handling routine (except interrupt 7 - the operator keyboard) and the processor priority is set to 7. To use an interrupt, the following has to be made exactly in the order given:

1) Replace the current interrupt vector with a vector pointing to the interrupt handling routine.
2) Initialize the peripheral that will request the interrupt.
3) Adjust the processor priority level if required.

After the interrupt is no longer being used, the following has to be made to restore the original conditions:

1) Disable the peripheral requesting the interrupt.
2) Return the default interrupt vector (pointing to the error handling routine).
3) Return the processor priority level if modified.

The address of the memory location containing the interrupt vector is computed by adding the corresponding offset to the start address stored in the VBR. The offset is 64H for INT1 level auto-vector, 68H for INT2 ... up to 7CH for INT7.

From the user point of view it is very convenient to build interrupt routines into high-level language programs that can be handled or compiled as any other program. The compiler produces relocatable .EXE program files: the operating system considers that an .EXE file can be executed on any memory address. However, the exception vector table contains absolute addresses. The program should therefore include a routine to compute the interrupt routine address dynamically each time the program is executed on a new memory location.

In the case of the MC68010 it is convenient to use the Load Effective Address instruction, in particular LEA d(PC),An to find the actual program counter. The address-finding routine is usually located just in front of the interrupt routine so that the unknown start address can be computed easily.

Interrupt routines are usually DSP routines written almost entirely in machine code. However, high-level language expressions are necessary to interface to the rest of the program. Whenever a call to an interrupt routine is made, it is necessary to save the content of the microprocessor internal registers on the stack and restore it before exiting from the routine. Considering the many MC68010 internal register this operation may be very time consuming. Since most DSP machine code routines only need a few registers, only the latter are saved and restored before exit to save CPU time. High-level language expressions however use all MC68010 registers!

Programs designed for use in an interrupt driven multitasking environment require some additional features. Since different interrupt routines may belong to different main programs and the interrupt nesting is arbitrary, the content of all microprocessor registers is completely unknown when entering an interrupt routine. If high-level language expressions are used in interrupt routines, the values of registers containing base addresses, usually A4 and A5, have to restored for correct operation.

8. Conclusion

As already mentioned in the first, theoretical part of the article, the DSP computer described is only a very successful prototype able to solve many practical problems. We will probably not have to wait long to see much more powerful computers built with new devices designed especially for DSP. We probably still have to invent most applications of DSP in amateur radio too.

It is very difficult to describe a complex electronic circuit in the limited space of a few magazine article without omitting important details. I would recommend to all users to order a copy of the "Operating System Manual" (about 15 sheets A4) which includes a detailed description of all operating system commands, text editor and high-level language compiler. I preferred not to publish the manual in the magazine since it would be very boring to all readers not directly interested in this DSP computer.

Those interested in writing DSP software will certainly need more information on the hardware used, especially the MC68010 microprocessor. During the development of this computer I used the following publications from Motorola:
1) 16-bit Microprocessors Data Manual, 1983.
2) M68000 16/32-bit Microprocessor Programmer's Reference Manual, 1984.

Data on memories and peripheral devices were obtained from several data-books and data-sheets from NEC, Western Digital, Zilog, National Semiconductor, Mostek, RCA, Fairchild and Matsushita. All the above information on the components used includes more than 1000 pages A4.

Both design and construction articles mainly discussed the hardware and the operating system software. The application software was only briefly mentioned in the theoretical introduction article. A detailed description of the application software would require at least another article. However, all the application software includes extensive menus and help messages and maybe an additional description is not necessary at all. Software is also changing quickly and such a description would become soon obsolete.

Besides the five application programs mentioned in the theoretical article, the hardware described offers several other possibilities. Some of these are now only planned. Others are tested, but the final version is not implemented yet. Finally, some are already fully implemented. The most

interesting programs include:
1) AO-13 400bps PSK telemetry reception (implemented)
2) Meteosat WEFAX reception (tested)
3) SSTV RX/TX (planned)
4) Audio recorder with variable playback speed (tested)
5) FFT audio spectrum analyzer (planned)
6) AX.25 communication modems up to 2400bps (planned)
7) Advanced protocols with error correction coding (planned)
8) Ranging for satellite orbit determination (planned)

Many other programs and some hardware (an EPROM programmer module) were developed just as tools to work on the hardware and on the operating system. I decided not to describe them since they are probably not interesting for the majority of readers.

At the end I hope this series of articles will stimulate more amateurs to experiment in the DSP field. Maybe the capabilities of the DSP computer shown are not comparable to what are the leading semiconductor manufacturers advertising on their data-sheets, but do not forget that there is a long way from the data-sheet to a working DSP project. In any case, the described DSP computer was found much more versatile than the originally planned successor to the APT scan-converter and in some cases it performed even better than dedicated DSP computers.

****

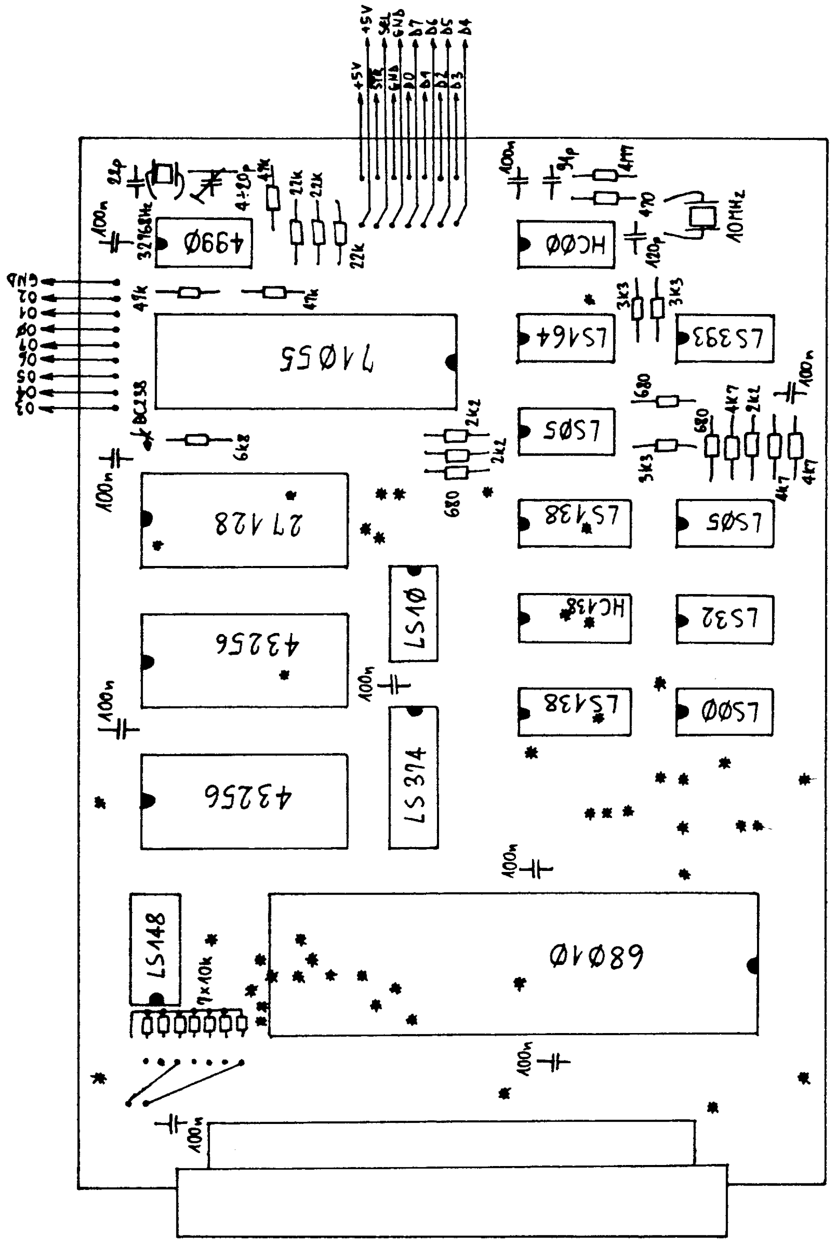


Fig. 1 - CPU board, component location (top view).

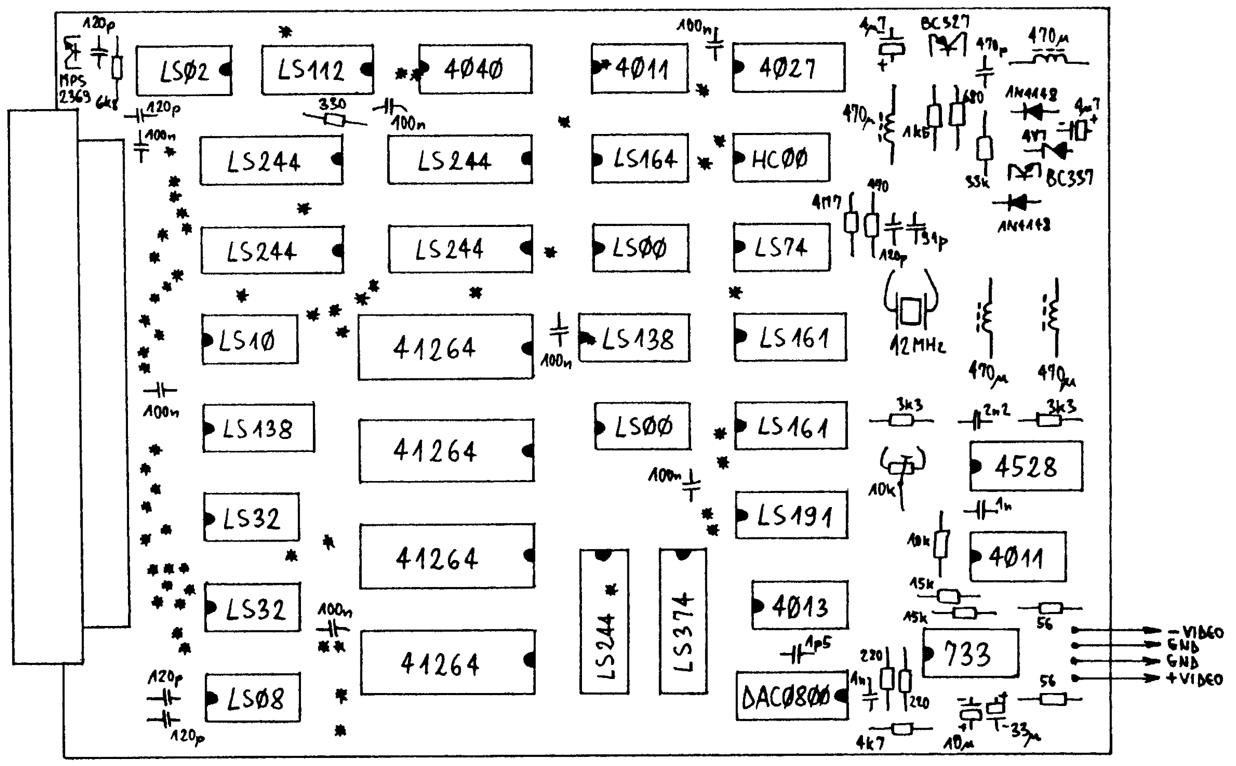


Fig. 2 - Video board, component location (top view).

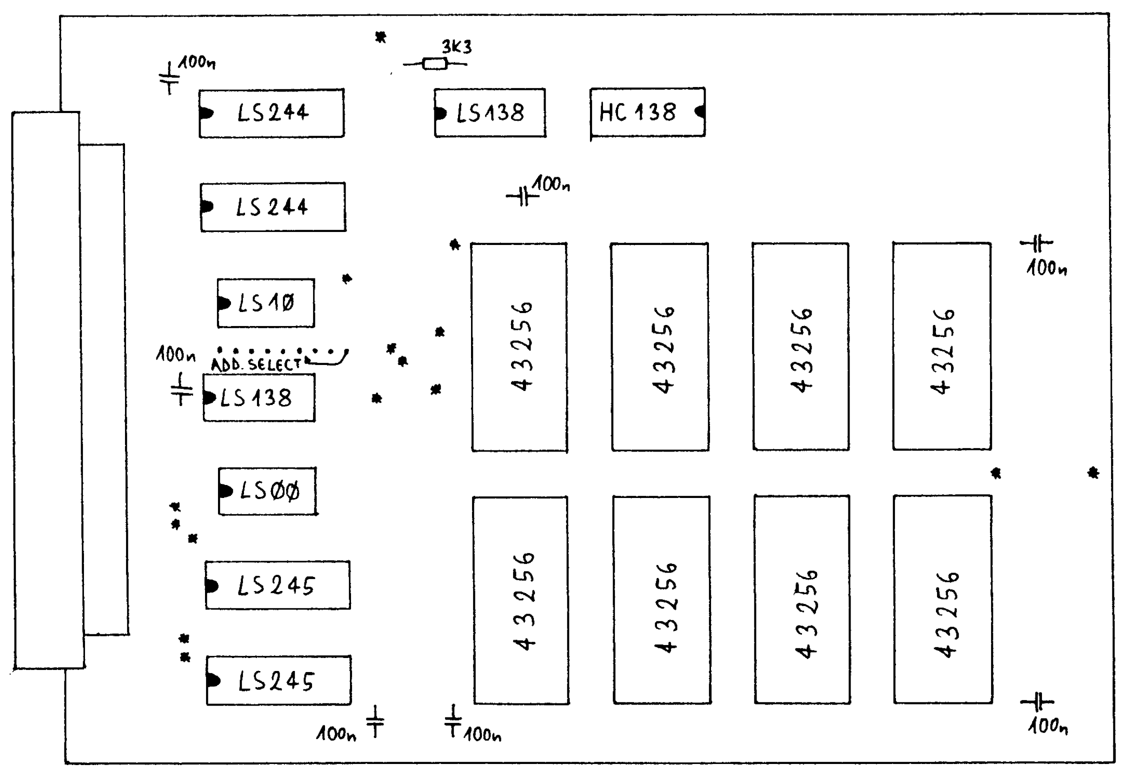


Fig. 3 – 256 k CMOS memory board, component location (top view)

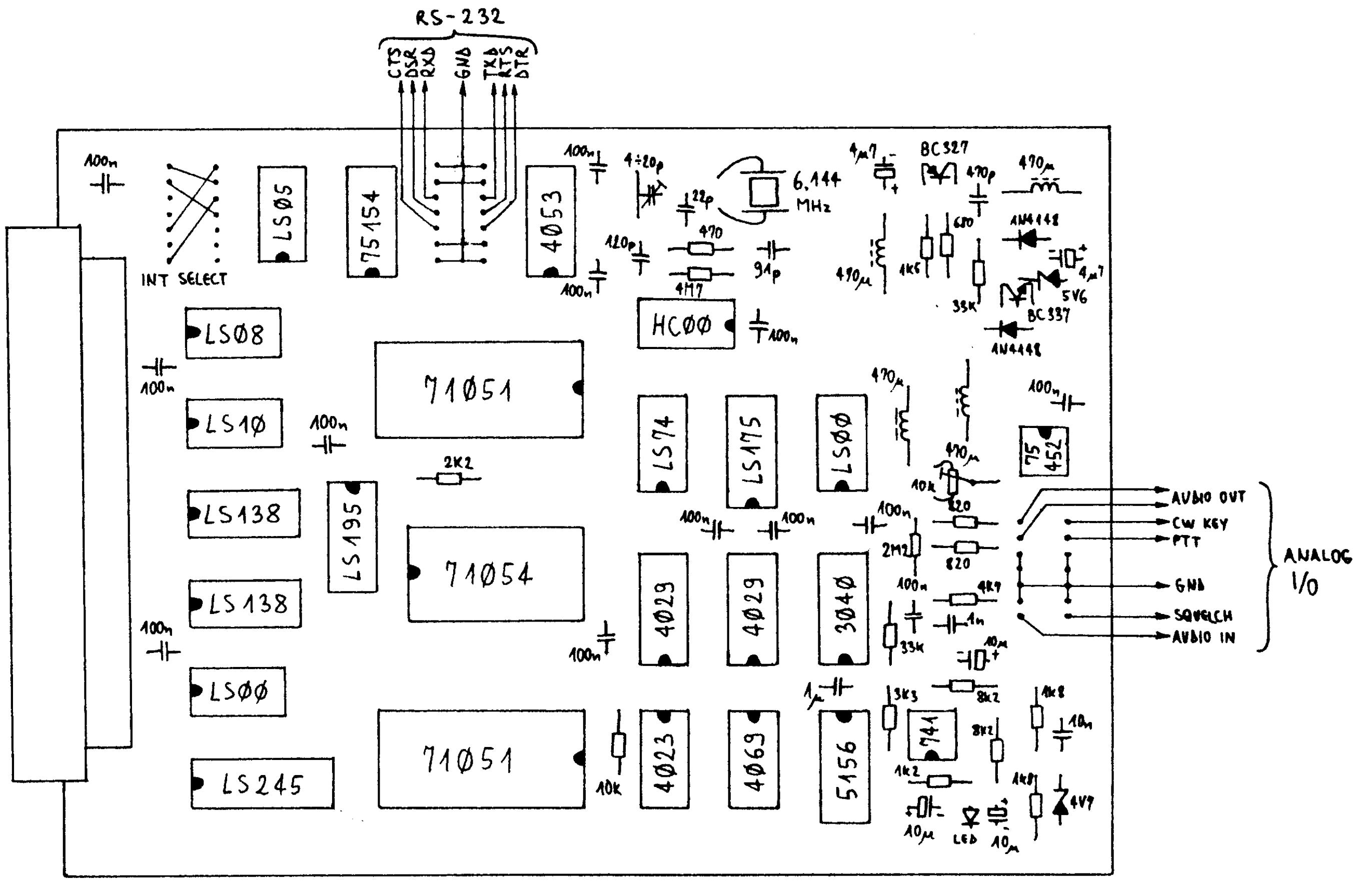
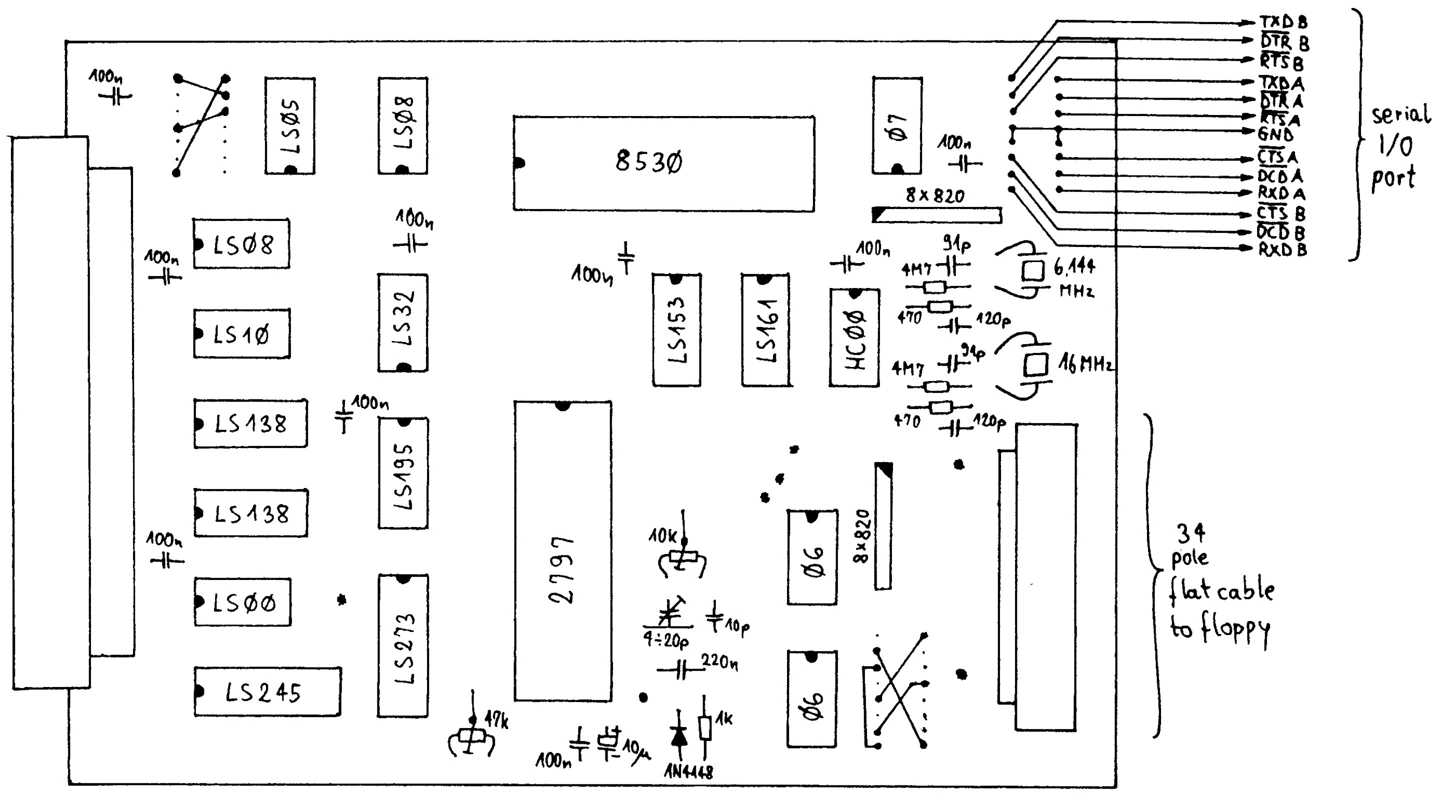


Fig. 4 - Analog I/O and RS-232 board, component location (top view).



TXD B
$\overline{\text{DTR}}$ B
$\overline{\text{RTS}}$ B
TXD A
$\overline{\text{DTR}}$ A
$\overline{\text{RTS}}$ A
GND
$\overline{\text{CTS}}$ A
DCD A
RXD A
$\overline{\text{CTS}}$ B
$\overline{\text{DCD}}$ B
RXD B
serial I/O port
34 pole flat cable to floppy
8530
2797
LS05
LS08
LS08
LS10
LS138
LS138
LS00
LS245
LS32
LS195
LS273
LS153
LS161
HC00
07
06
06
8x820
8x820
6,144 MHz
16 MHz
4M7
470
91p
120p
34p
100n
10K
47k
4÷20p
10p
220n
1k
1N4148
10µ

Fig. 5 - Serial I/O and floppy interface board, component location (top view)

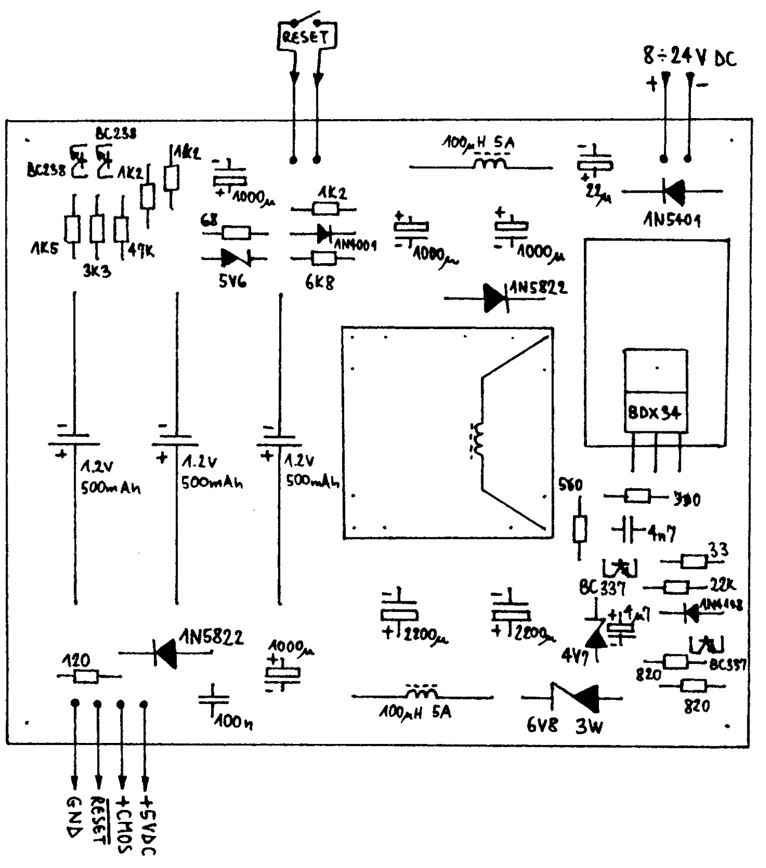


Fig. 6 - Power supply board, component location (top view).

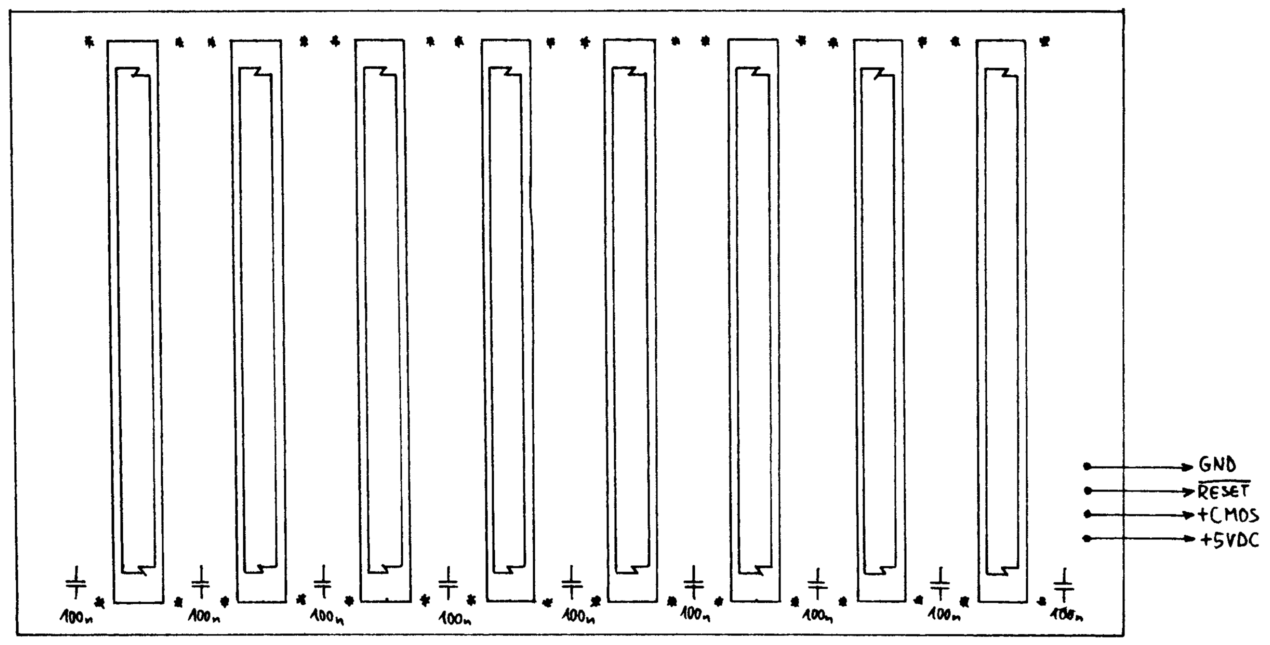


Fig. 7 - Bus board, component location (top view).

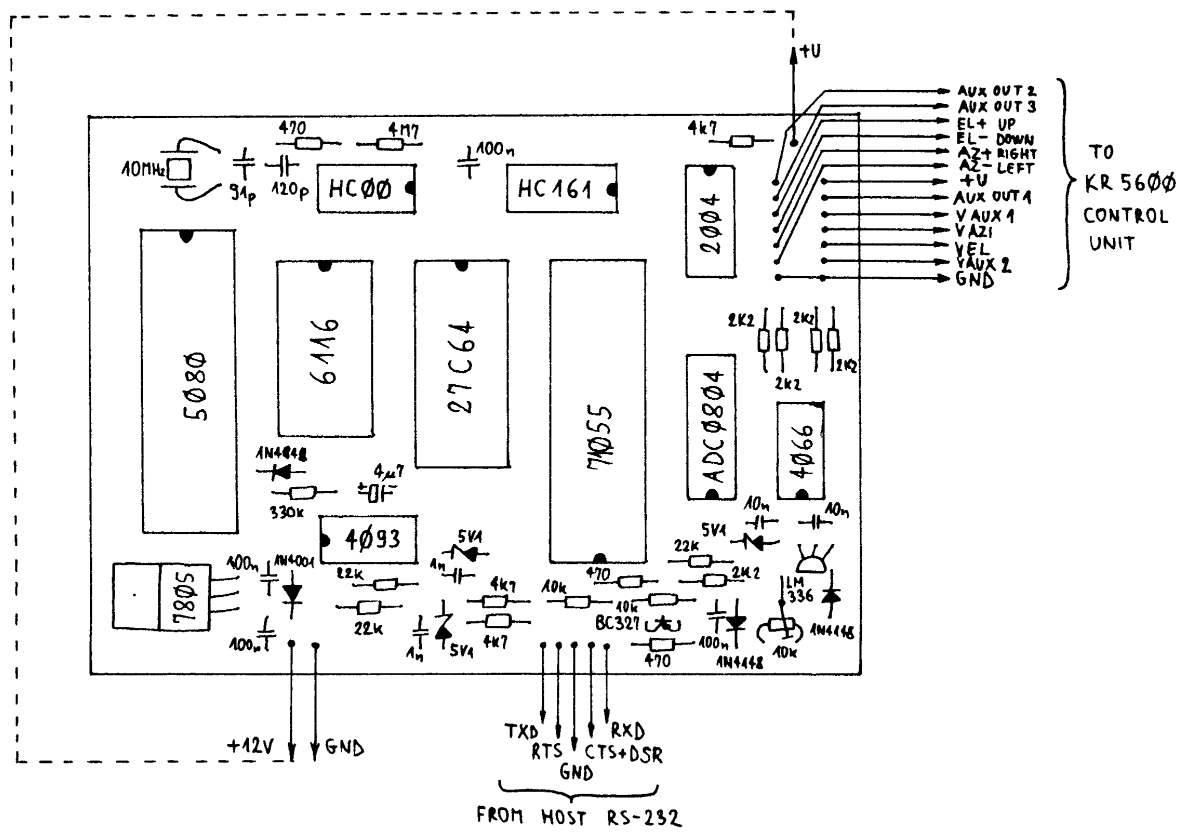


Fig. 8 - KR5600 Rotator interface, component location (top view).

*** YT3MV - DSP OPERATING SYSTEM MANUAL - V7.3 ***

1. INTRODUCTION

The YT3MV DSP operating system is intended to be used with the corresponding DSP computer including a MC68010 16 bit microprocessor, usually 1Mbyte of static nonvolatile CMOS RAM and peripheral devices including parallel and serial I/O ports, analog I/O ports (A/D and D/A converters), a floppy disk controller/driver and a real time clock.

Since almost all of the available RAM is nonvolatile, the operation of the computer is no longer based on mechanical storage devices, like floppy or hard disks or tape drives. The operating system uses the nonvolatile RAM at the same time as a long term volume storage device (hard disk) and as the CPU working memory. Floppy disks are only used as a memory backup in the case of a system crash and/or to ease the transfer of programs and data.

The nonvolatile RAM is organized as a directory of files. All the RAM files are stored in contiguous memory locations: program files can be executed in the same memory locations where are stored. This is a much simpler and better solution than the popular "RAM disks", which retain the same data organization as floppy disks with files split in a number of equal size blocks, requiring an additional RAM area and precious CPU time to assemble program files into executable programs.

Most operating system commands only work with RAM files. A limited number of commands is provided to support the operation with floppy disks, since the latter are not used frequently. Operations with files on floppy disks are not just more slow and complex, they are also very time consuming for the CPU. In a computer devoted to Digital Signal Processing, the CPU is loaded with the very time demanding DSP routines almost to the theoretical speed limit of its processing power. Operation with disks or other mechanical storage devices at the same time would severely degrade the DSP computer performance.

Routines to use the serial RS-232 port and the real time clock chip are also provided. Files can be received and/or transmitted through the RS-232 port using different speeds and data formats.

The operating system includes a screen text editor program and a high-level language compiler. The screen text editor can be used to edit ASCII text files and in particular to prepare source programs for the high-level language compiler. The compiler program generates a machine code program for the MC68010 microprocessor from the source ASCII file. The high-level language allows single and indexed variables, floating-point arithmetics including transcendent functions, input/output routines and a simple and efficient way to insert and communicate with machine code routines. The latter is particularly important for DSP since the basic DSP routines have to be written directly in machine code since they must be very fast. On the other hand, the various support routines require complex mathematical operations that are easier to perform in high-level language.

2. OPERATING SYSTEM COMMANDS

All the operating system commands include a single letter command that may be followed by one or more filenames or parameters. Typing errors can be corrected using the <DELETE> key (code 7FH form keyboard) and the command is executed after depressing the <CR> key (code 0DH from the keyboard). A filename is made of up to 12 characters followed by an up to three characters long extension. When more than one filename is required by a certain command, the required filenames are separated using space characters. Complete filenames have to be typed: wild-cards (*) are not allowed. Some commands are however able to find the right extension on their own.

Some commands do not always require that one or more filenames are supplied, filenames typed with the previous commands are used in the case no filename is specified. Of course, dangerous commands like kill (K) or format (F) always require the exact number of filenames or parameters to follow.

The operating system includes two directories of files: a RAM file directory (command M) and a floppy disk file directory (command D). The RAM file directory may include more than one file with the same name and all the commands act on the first file with the specified name encountered in the directory. To access the other files rename (command J) or kill (command K) the first file carrying the same name. Two or more files with the same name are not allowed on a floppy disk. Files can not be killed or renamed on a floppy disk, the only way is to erase the complete disk by formatting it.

The operating system provides three different error messages:

? means an operator error, usually an inexistent or duplicated file name.

* means a floppy disk error. In the case of a disk read operation, the file loaded from the disk into the RAM should be discarded (killed) since it contains errors. In the case of a disk write operation, the disk should be formatted.

A program crash, either due to a programming error or a hardware breakdown usually causes a bus error or an address error (or another unused exception vector). A call to the monitor program is generated and an error message is printed in the form:

```
PC:<prg_counter> F:<stack_format> S:<SSW> A:<access_address>
```

The last two items are included only in the case of a long stack format (bus error and address error).

The execution of a program or operating system command can be interrupted or aborted by pressing <CTRL-C> (code 03H from the keyboard, which calls the operating system immediately. The computer can now be commanded using the operating system commands: use command A to reset the computer, command Q to resume the execution or other commands to diagnose the problem.

Implemented V7.3 commands include:

A   Abort - Software RESET Command
     Resets the computer in the same way as the reset pushbutton or at power-up. The actual action depends on the position of the partial/total reset switch.

B   Enter Autostart Command
     After the ":" symbol, the autostart command can be typed in as any other operating system command.
     The autostart command is executed immediately after each computer reset, its syntax being identical to the operating system commands originated from the keyboard.
     Any valid operating system command can be executed after a partial computer reset as the autostart command.
     After a total computer reset, the autostart command is first set to its default (the H (help) command) and then executed.

C <source> <source> <source> <destination>   Copy RAM Files
     Copies the specified source files into a destination file.
     Up to 7 (source) files can be pasted together and the resulting file receives the last specified filename (destination).

D   Floppy Disk File Directory
     Prints the floppy disk file directory in the following format:

<filename>	up to 12 characters
<extension>	up to 3 characters
<start_track>	between 0 and 79, two decimal digits
<start_side>	0 or 1, one digit
<start_sector>	between 1 and 5, one decimal digit
<file_length>	up to 5 hexadecimal digits

E <filename>   Screen Text Editor - 63 Column Version
     Calls the screen text editor program, which has a number of its own commands described later. If no file with the specified name can be found in the memory file directory, a new, empty file is created. If there already exists a file in the memory file directory with the specified name, a copy is generated and the original file is renamed: its extension is modified to .BAK.
     If the specified filename carries the extension .EXE, the editor automatically looks for a file with the same name but with the extension .SRC.
     All the editor commands are control characters. The exit command is <FF> or <CTRL-L> (code 0CH from the keyboard).
     The E command automatically sets the screen display routine in the 64 column mode, thus text with line lengths up to 63 characters can be edited. See also the Y command for editing text with lines up to 84 characters long.

F <start-track> <stop-track>   Format / Verify Floppy Disk
Formats the floppy disk from the specified start-track up to and including the specified end-track. One track includes 5 sectors on each side of the floppy disk. Each sector is 1024 bytes long. A fully formatted 80 track floppy disk thus includes 800 sectors or 800 kbytes.
The usual formatting command is: F 0 79
The formatting routine prints out the number of the track actually being formatted. Defective tracks, if any, are marked with the symbol "*". Floppy disks containing defective tracks should be discarded immediately, since the operating system does not allow bad sectors. In the latter case it is recommended to try the formatting once again, reinserting the floppy in the drive. Most errors with 3.5" floppy disks were found to be caused by an incorrect mechanical insertion rather than a defective floppy disk.
If the floppy disk is write protected, the command will only perform a verification of the formatting.

G <filename>   Compile .SRC >>--->> .EXE / .OLD
Compiles a source file in high-level language (extension .SRC) into an executable machine code program file (extension .EXE). Both the .SRC and .EXE files share the same name (first 12 characters) and the command looks automatically for the correct extensions.
If a file with the specified name and extension .EXE already exists, a copy is made first and the original is renamed to .OLD. The source file is then compiled in the same memory locations that carried the copy of the original .EXE file. In this way the numerical values of the variables in the old .EXE file are transferred to the new .EXE file if the latter uses the same variables.
An error in the source program is indicated with a "?" followed by the offset (in hexadecimal) in the .SRC file where the error was found.
WARNING! Do not try to run the resulting .EXE file, if an error was found by the compiler!

H   HELP
Prints out a short help file including a list of all the operating system commands. The display routine is reset to obtain bright characters on a dark background.

I <destination>   Input File From RS-232 Port To RAM
Data received by the serial RS-232 port according to the format specified with the command P is assembled into a RAM file. The length of the recorded file is displayed in hexadecimal format. The recording can be stopped by pressing any key on the keyboard.
The RS-232 signals are used in the following way:
TXD not used (remains negated).
RTS remains negated.
DTR becomes asserted when the command is started and is negated when the command is stopped.
RXD accepts the serial data.

DSR is used as a carrier detect: must be asserted to accept data.
CTS is ignored.

K <filename>   Kill RAM File

Kills a RAM file. All the files located above the deleted file are shifted down in the memory, to create a contiguous empty space at the top of the RAM.
WARNING! The computer should not be reset during the execution of the kill command, otherwise the files located above the deleted file may be damaged!

L <filename>   Load From Floppy Disk To RAM

Loads the specified file from the floppy disk to the RAM. The operating system will answer with a "?" if no such file can be found on the floppy disk or if a file with the same name already exists in the RAM.
WARNING! If the operating system answers with a "*" (indicating a floppy disk error) or the computer is reset during the execution of the command, a corrupted file may still appear in the RAM file directory with no additional warning messages!

M   RAM File Directory

Prints the RAM file directory in the following format:

<filename>	up to 12 characters
<extension>	up to 3 characters
<start_address>	usually 6 hexadecimal digits
<file_length>	up to 6 hexadecimal digits

At the end of the file directory, the number of the remaining free RAM space bytes is displayed (hexadecimal).

N <start> <length>   New Memory Space

Tells the computer where and how much RAM is available.
After a total reset, the start address is set to 39600H and the length to 6A00H corresponding to a small amount of free RAM on the CPU board, which is always present in the system, regardless of its configuration.
The start address and length have to be set according to the available memory at system start-up.
A N command without (or followed by an incorrect number of) parameters will only display the current parameters.

O <source>   Output File From RAM To RS-232 Port

Transmit a RAM file through the serial RS-232 port according to the format specified with the P command.
The length of the remaining part of the file to be transmitted is displayed in hexadecimal format.
The RS-232 signals are used in the following way:
TXD supplies the serial data.
RTS becomes asserted at the beginning of the transmission and is negated after the execution of the command.
DTR remains negated.
RXD is not used (is ignored).
DSR is ignored.
CTS is used to temporarily stop the transmission when

negated, the transmission resumes when CTS again becomes asserted.

P <speed> <length> <parity>   Parameters RS-232 Port

Sets and displays the parameters of the RS-232 port to be used with commands I and O.
<speed> is in bits per second (bauds) and the value entered is rounded to the nearest available bit rate. All standard data rates between 50 and 19200 bps can be selected.
<length> is the number of data bits, between 5 and 8.
<parity> 0 or 2 mean no parity bit, 1 means odd parity and 3 means even parity.
If less than three parameters are supplied, only the first parameter(s) will be modified. A P without any parameters will just display the current port parameters.
After a total reset, the parameters of the RS-232 port are set to the default values: 9600 bps, 8 bits, no parity.

Q   Quit

Quits the operating system if called from another program or using <CTRL-C>. If the operating system was not called from another program or using <CTRL-C>, then the command Q resets the computer exactly like the command A.

R <filename>   Run .EXE

Runs the specified program file. The extension is set to .EXE automatically. To return to the operating system, the program file should include a RTS instruction at the end. The latter condition is fulfilled automatically if the .EXE file is generated by the high-level language compiler (command G).
Program files usually include some memory space to store the values of the variables. Since the latter are modified during the execution of the program, the whole program file will be modified as well! Therefore, two previously identical .EXE files will no longer compare if they were executed with different data.

S <filename>   Save From RAM To Floppy Disk

Saves the specified file from the RAM to the floppy disk. The operating system will answer with a "?" if no such file can be found in the RAM or if a file with the same name already exists on the floppy disk.
WARNING! If the operating system answers with a "*" (indicating a floppy disk error) or the computer is reset during the execution of the command, a corrupted file may still appear in the floppy disk file directory with no additional warning messages!

T <filename>   Type ASCII File

Types the specified file on the screen as ASCII characters. Some control characters are interpreted by the display routine and in particular <ESC> sequences may modify the display parameters. Use the command H (help) to reset the display routine.

U <yymmddhhmmss>   Set / Read Real Time Clock
     Sets the real time clock chip to the specified date and
     time. The <yymmddhhmmss> string should contain exactly
     12 figures with no spaces or other characters between the
     numbers. If no date and time is specified, the command U
     will just print out the actual date and time supplied
     from the real time clock chip.

V <source> <source>   Verify - Compare Two RAM Files
     Compares two RAM files. First, the difference in length
     in bytes (hexadecimal) is printed out. After that all the
     differing bytes are printed out in the following way:
     <offset>:<file1byte>^<file2byte>
     up to the end of the shorter file, the excess bytes at the
     end of the longer file are ignored.

W <address> <data>   Write / Read Memory Location
     Writes the given 16 bit data word to the specified memory
     location. If no data is supplied it just performs a read
     from the specified address. This command provides a direct
     access to the hardware and is used essentially to debug
     the hardware and/or for system initialization (start-up).
     WARNING! The W command may corrupt files or crash the
     operating system if used incorrectly!

X <filename>   Execute .CMD File
     Executes the specified operating system command file. The
     extension is set automatically to .CMD.
     An operating system command file is an ASCII file that
     includes one or more operating system commands identical
     to the commands typed directly to the operating system.
     Each command is executed after a <CR> encountered in the
     command file. An operating system command file can be
     assembled using the screen text editor (commands E or Y)
     like any other ASCII file.
     WARNING! It is not recommended to use commands B, K, N
     and Q in command files!

Y <filename>   Screen Text Editor - 84 Column Version
     Calls the screen text editor program in the same way as
     command E. A more detailed description of the editor
     commands follows later. The Y commands sets the screen
     display routine to 85 columns allowing the editing of
     ASCII files with up to 84 characters per line, including
     the 80 character per line computer standard. Otherwise,
     the Y editor works exactly as the E editor.

3. SYSTEM START-UP

When powering-up the computer for the first time, a total computer reset has to be performed and some operating system parameters have to be adjusted. The same procedure is also recommended after a program crash that might have damaged the

content of some memory locations before the computer stopped.
The hardware can be reset in four different ways:

- Applying the supply voltage.
- Pressing the RESET pushbutton.
- Using the operating system command A (Abort).
- A software jump or call to the location 000100H.

The action taken by the computer after a reset is applied is identical in all four cases, it only depends on the position of the TOTAL/PARTIAL RESET switch. If the latter is open, a total computer reset will be performed. If it is closed, only a partial computer reset will be performed.

During a total computer reset, the complete operating system software is copied from the EPROM to the nonvolatile system RAM, all the operating system parameters are set to their default values, all the peripheral devices are reset, the modified character generator table is generated. All the memory which is not used by the operating system is however left unchanged!

After a total computer reset the HELP message should appear on the video screen, since the default autostart command is command H. The TOTAL/PARTIAL RESET switch has to be closed (enabling a partial reset) and the available RAM space has to be specified using the command N. If there is one megabyte of RAM available starting at address 200000H, the command is:

```
N 200000 100000
```

To clear the content of the RAM, if any RAM files are displayed by the command M, it is sufficient to clear the first RAM location using the command W, in the above case:

```
W 200000 0
```

Further initialization commands may include commands B, P and U to set respectively the autostart command, the serial port parameters and the real time clock but they are not essential like the N command.

When the nonvolatile RAM contains useful data, like the operating system software with updated parameters and application program and data files, it is reasonable to perform a partial computer reset at power-up. During a partial computer reset, only the exception vector table is copied from the EPROM to the nonvolatile RAM, all the peripheral devices are reset, the video screen is cleared and the autostart command is executed.

After a partial reset, the computer is immediately ready for use, including the application program and data files in the nonvolatile RAM which operates almost like a "hard-disk"!

4. HARDWARE CONFIGURATION

The hardware required to run the DSP operating system

includes:
(A) CPU board with the MC68010 microprocessor, operating system EPROM, 64 kbytes of nonvolatile CMOS RAM, keyboard interface and real time clock.
(B) Video board with 128 kbytes of dual port video RAM.
(C) One or more 256 kbytes CMOS memory boards.
(D) Analog I/O and RS-232 board.
(E) Serial I/O and floppy interface board, including a 3.5" floppy drive.
(F) Bus board and power supply.

The operating system can run with just the memory on the CPU board (A), the latter is however far too small to load most application programs and additional memory boards (C) have to be added. The analog I/O and RS-232 board (D) is only used by the operating system with instructions I, O and P, however all the DSP application programs need the analog I/O. Similarly, the serial I/O and floppy interface board is only required with instructions D, F, L and S.

The CPU board (A) includes a timeout logic that generates a bus error if no memory or peripheral device terminates a bus cycle within 192 CPU clock cycles. Missing or defective peripheral devices can thus be detected, as well as inexistent memory locations.

The MC68010 CPU allows a 16 Mbyte address range. The first megabyte is reserved for the operating system memory, video memory and peripheral devices and the remaining 15 megabytes can be used for general purpose memory. The first (system) megabyte is allocated as follows:

000000H to 007FFFH	EPROM containing the operating system software
010001H	71055 port A - parallel keyboard input
010003H	71055 port B - spare parallel output
010005H	71055 port C - keyboard strobe & RTC chip control
010007H	71055 command register
020001H	Analog I/O 71051 data register
020003H	Analog I/O 71051 control register
024001H	71054 CTR0 A/D sampling rate
024003H	71054 CTR1 RS-232 baud rate
024005H	71054 CTR2 D/A sampling rate
024007H	71054 control register
028001H	RS-232 71051 data register
028003H	RS-232 71051 control register
030000H to 03FFFFH	64 kbytes nonvolatile system RAM
040000H to 05FFFFH	128 kbytes dual port video RAM
080001H	8530 B command
080003H	8530 B data

080005H	8530 A command
080007H	8530 A data
084001H	LS273 latch
088001H	2797 control register
088003H	2797 track register
088005H	2797 sector register
088007H	2797 data register

Care should be taken if other peripheral devices are to be added, since the above addresses are not fully decoded. In addition, the locations immediately below 030000H should be left unused to allow a safe end in the case of a program crash making the stack to grow indefinitely: the resulting double bus error will stop the CPU and prevent it from destroying all the memory content.

Function code 7 is decoded and hardwired to VPA\ to generate interrupt auto-vectors, the remaining function codes are not further decoded. The interrupts are assigned as follows:

INT1	RTC 4990 TP	
INT2	RS-232 71051 TX ready	
INT3	RS-232 71051 RX ready	(FDC 2797 INT)
INT4	FDC 2797 DRQ	
INT5	D/A 71051 TX ready	(8530 INT)
INT6	A/D 71051 RX ready	
INT7	KBD 71055 INTA	

The computer hardware includes a nonvolatile memory supply and protection circuit that preserves the content of the nonvolatile RAM regardless of the power-up and/or power-down sequences. The protection circuit applies the reset signal to the microprocessor and inhibits the access to the nonvolatile RAM chips and real time clock chip immediately after the supply voltage at the input of the 5V switching voltage regulator falls below the minimum value of 6.5V. The reset and inhibit signals are only removed about one second after the regulator input voltage has risen above 7V.

5. OS V7.3 ROM ROUTINES

The DSP operating system V7.3 consists of 16 kbytes of MC68010 machine code stored in a 27128 or a 27C256 EPROM. After a total computer reset the operating system software is copied from the EPROM into the nonvolatile system RAM. The operating system is always executed in the RAM since the access

time of the RAM is much shorter than that of the EPROM.
The organization of the software contained in the EPROM is follows:

000000H to 0000FFH	Initial exception vector table
000100H to 0001FFH	Start-up routine
000200H to 003FFFH	Programs and data to be copied into the system RAM between locations 035800H and 0395FFH

The nonvolatile system RAM is organized as follows:

030000H to 033FFFH	System stack (16 kbytes)
034000H to 0357FFH	Modified character generator table
035800H to 035AFFH	ASCII character generator table
035B00H to 035DFFH	Keyboard and video routines (TRAP0, TRAP1, TRAP2, TRAP5, TRAP10 and INT7)
035E00H to 035EFFH	Exception vector table (64 vectors)
035F00H to 0395FFH	Operating system program
039600H to 03FFFFH	Spare RAM area (6A00H bytes free)

The following general purpose routines can be accessed through TRAP instructions:

TRAP0 Prints the content of D0.B as an ASCII character on the screen. The screen format is 32 lines of either 64 or 85 characters each. In the 64 column mode, the character font is 7x7 pixels while in the 85 column mode it is 5x7 pixels. The routine performs automatic line feeds if the number of characters exceeds 64 or 85 and automatically scrolls the screen up as well. The following control characters are implemented:

<BS> (08H)	Moves the cursor one character position back.
<HT> (09H)	Moves the cursor one character position forward and performs a line feed and/or scroll if necessary.
<LF> (0AH)	Moves the cursor one line down performs a scroll if necessary.
<VT> (0BH)	Moves the cursor one line up.
<FF> (0CH)	Clears the screen.

<CR> (0DH)   Moves the cursor at the beginning of
the line.

<ESC> (1BH) + two characters
Sets the display parameters with the following two characters. The first determines the brightness of the background (modulo 32) and the second the brightness of the characters (also modulo 32). Further, bit 5 of the first character selects either 64 column mode (0) or 85 column mode (1). The switching between display modes can be performed at any time: it is not necessary to start a new line or clear the screen.

Any other control characters and codes above 126 (7EH) print the cursor symbol.

TRAP1	Supplies in A0 the start address of the screen display routine parameter table: cursor position (2 words) & display mode (3 words).
TRAP2	Keyboard entry routine. Displays a blinking cursor. Any character typed on the keyboard causes an exit and is available in D0 but it is NOT printed on the screen.
TRAP3	Not used.
TRAP4	Calls the operating system program.
TRAP5	Keyboard interrogation routine. Supplies in D0 the last typed character and the number of times any key was depressed from the last call of TRAP5.
TRAP6	Prints the content of D0.L in hexadecimal format on the screen.
TRAP7	Sets A0 to the beginning of a parameter table including the RAM start address, the RAM length and the autostart command string.
TRAP8	Executes an operating system command string supplied as -(A6).
TRAP9	Searches a file in the RAM with the name supplied as (A0)+. The start address of the file is made available in A1 and the file length in D0. If the file is not found, D0 is set to FFFFFFFFH and A1 points to the first free RAM location. If no free RAM is available A1 is set to 0.
TRAP10	Supplies in A5 the reference address of the floating--point arithmetics routines.
TRAP11, 12, 13 and 14	Not used.

The operating system commands work with files. Any file includes a header of 20 (14H) bytes followed by an even number of data bytes. The file start address and file length are always referred to the data bytes WITHOUT the header. The 20 header bytes are assigned as follows:

1 byte	File start flag AAH
12 bytes	File name
3 bytes	File name extension
4 bytes	File length

In the nonvolatile RAM the header of the first file in the RAM directory starts on the first available RAM location. The header of the next file starts immediately after the last data byte of the first file. After the last file there is an empty space usually filled with zero bytes. There are no empty spaces between RAM files. After a K (Kill) command, the files above the killed file are shifted down and the resulting empty space at the top of the RAM is filled with zeros.

Files are arranged in a similar way on the floppy disk. The first file starts on sector 1, side 0 and track 0. When writing a long file on the floppy disk, the sector number is incremented first. When the track is full, the floppy drive heads are switched to the other side. When both sides are full the heads are moved to the next track. There are no empty sectors between files, however, each file uses an integer number of sectors. The last sector may include filler bytes so that the next file starts exactly on the beginning of the next available sector.

6. THE SCREEN EDITOR

The screen editor can be entered with the E or Y command to edit a text file made of ASCII characters. The original file, if any, will be renamed .BAK. The two versions of the screen editor differ only in the maximum number of printable characters allowed in a line, but are otherwise identical. The E command will set the screen to 64 columns and will allow lines up to 63 characters long (to prevent the automatic line-feed and allow to display the cursor at the end of the line). The Y command will set the screen to 85 columns and correspondingly allow 84 characters in a line.

If the file does not correspond to a text file: if it does not contain a <carriage-return> + <line-feed> combination at least every 63 (E) or 84 (Y) printable characters (the maximum allowed number of characters in a line), then the file will be truncated just in front of the detected error. Stand-alone <CR> characters are not allowed and cause a truncation of the file. The file may however contain other control characters which are printed as inverse characters (dark characters on a bright background).

All the screen editor commands are ASCII control characters. Valid commands are:

^@ (00H)	delete line.
^A (01H)	scroll the display window one line up.
^B (02H)	scroll the display window one line down.
^C (03H)	call the monitor program immediately, exit after return with Q.
^D (04H)	scroll the display window 8 lines (1/4 page) up.
^E (05H)	scroll the display window 8 lines (1/4 page) down.
^F (06H)	scroll the display window 1 page (32 lines) up.
^G (07H)	scroll the display window 1 page (32 lines) down.
^H or <BS> (08H)	cursor one character left (backward).
^I or <HT> (09H)	cursor one character right (forward).
^J or <LF> (0AH)	cursor one line down.
^K or <VT> (0BH)	cursor one line up.
^L or <FF> (0CH)	exit from the screen editor.
^M or <CR> (0DH)	insert a new line.
^N (0EH)	change an upper-case character into lower-case or a lower-case character into upper-case.
^O (0FH)	allows the insertion of control characters: the next printable character typed will be transformed into a control character.
<DEL> (7FH)	delete the character left of the cursor.

All the other ASCII control characters (10H to 1FH) cause an exit. All printable characters (20H to 7EH) will be simply inserted in the text.

Characters that disappear at the right margin are lost when the cursor is moved to another line or the screen window is moved along the text.

The screen text editor updates the file in the nonvolatile RAM each time the cursor is moved to another line or the screen window is moved along the text. Therefore, in the case of an unexpected power failure or another source of a reset of the computer, only the last line typed may be lost, all the remaining text is saved in the nonvolatile RAM! When typing a long text it is therefore not necessary to periodically exit from the text editor to save the file like on computers that use dynamic memories and hard discs to avoid losing several hours of typing due to a power surge!

When the editor is called for the first time, both the cursor and the display window will be positioned at the beginning of the text. However, when the editor is recalled, it will try to remember the position of the cursor and display window as they were set last time. The two editors E and Y have separate buffers for these variables.

7. THE HIGH-LEVEL LANGUAGE COMPILER

The high-level language compiler accepts an ASCII source file (extension .SRC) and produces a relocatable machine code program for the MC68010 microprocessor (same file name, but with an extension .EXE). If an .EXE file with the same name already exsists, it will be renamed to .OLD and the numerical values of the variables will be copied into the new .EXE file. When an error is found in the source file, the resulting .EXE file should not be used!

The compiler only supports floating-point or REAL type variables and constants. A variable name is made of up to six characters, the first must be a letter and the following may also be numbers. Upper and lower case letters are considered different symbols in variable names.

A single variable requires six 16 bit words of memory and includes a 6 character variable name, a 16 bit binary exponent, a 31 bit mantissa and a sign bit. A 31 bit mantissa allows a computation accuracy of about 10 decimal digits for the basic operations (addition, multiplication, division) and about 8 to 9 decimal digits for transcendent functions. A 16 bit exponent allows numbers to range between about 10^-9863 and 10^+9863. This range is probably enough for any kind of computations. In the remote case of an overflow, the functions are programmed to set the result to zero with no warning messages! Correspondingly, a division by zero produces a zero output.

The floating-point format can support ACCURATE operations with integers, provided that these are small enough. For addition and subtraction, both operands and result have to be absolutely smaller than 2^32-1 or 4294976295. For multiplication and division, at least one of the operands or result have to be smaller than 2^16-1 or 65535.

Real number results are always truncated to the nearest absolutely smaller value that can be represented with the given real number format. Note that most decimal numbers with nonzero figures behind the decimal point do not have an exact representation in the binary format: for example, 2.3 would be printed as 2.299999 if the result is not additionally rounded before printout.

Arrays may have up to 20 dimensions and must be declared. Each index may range between 0 and 65535. Each array element requires three 16 bit words of memory and some additional memory is required to store the array name and information about its dimensions.

Arithmetic expressions are executed in the same order as they are written: all the arithmetic operations including functions have the same priority. Parentheses are not accepted. For example, the expression:

```
MMY/45+12*ERT=FGH
```

means, that the actual value of the variable MMY is first divided by 45, then 12 is added to the result of the division, later the sum is multiplied by the value of the variable ERT and finally the result of the operation is assigned to the variable FGH. Variable names are case sensitive: fgh, fgH, fGh, fGH, Fgh, FgH, FGh and FGH are considered 8 different and completely independent variables.

Similarly, the expression:

```
-TYP(J)*TYP(J)+1$Q=IOP(N)
```

means that the negative value of the J-th element of the single dimension array TYP() is multiplied by itself, 1 is added to the result and the square root (function $Q) of the sum is assigned to the N-th element of the array IOP(). The compiler allows the existence of a single variable with the same name as an array: for example TYP(J) and TYP are completely independent, the first being an array and the second a single variable.

More than one arithmetic expression can be written in a single program line, separators are simply space characters or any ASCII control characters, including <CR> and <LF>.

Functions include both mathematical and other functions. They may or may not require an operand and may or may not produce a result. Mathematical functions require one operand and produce one result. In the case of a math function with a limited domain, like arcsin, no warning or error messages are generated if the operand lies outside its domain. Instead, the result is set to the nearest sensible value. All the trigonometric functions and their inverses work with angles expressed in radians. The exponent and logarithm functions work with the base of the natural logarithm e=2.718....

All the functions are denoted by the symbol "$" followed by one letter; both upper and lower case letters are considered the same here. Valid functions are:

$: Declare an array. For example, $:TGU(12,3,7) means that TGU() is a three dimensional array, the first index ranges between 0 and 12, the second between 0 and 3 and the third between 0 and 7. The dimensions of the array are therefore 13*4*8 since 0 is a valid index!
WARNING! The complier will provide no warning or error messages if two arrays or more are declared with the same name.

$< Print the following ASCII characters on the screen. The string must be terminated with a ">" character, which is not printed.

$A ARCTAN mathematic function. Domain unlimited. Range between -PI/2 and PI/2.

$B	Absolute value function. Domain unlimited. Range all positive numbers.
$C	Cosine function. Domain unlimited. Range between -1 and +1.
$D	ARCCOS mathematic function. Domain unlimited. Range between 0 and PI. For arguments less than -1 the function returns the value PI and for arguments greater than 1 the output is 0.
$E	Exponent function with the base e. Domain unlimited. Too large arguments produce a zero result. Range all positive numbers.
$F	Fractional part function. Domain unlimited. Range between -1 and +1. The function computes the fractional part of the absolute value of the argument and retains the sign.
$H	ARCSIN mathematical function. Domain unlimited. Range between -PI/2 and +PI/2. For arguments less than -1 the function returns the value of -PI/2 and for arguments greater than +1 the output is PI/2.
$I	Integer part function. Domain unlimited. Range all integer numbers. The function computes the integer part of the absolute value and retains the sign. For example, the result of -1.5$I is -1!
$J	Unconditional jump to a label. The label number must follow immediately!
$J+	Jump only when the result of the previous instruction was positive. A label number must follow immediately!
$J-	Jump only when the result of the previous instruction was negative. A label number must follow immediately!
$K	Call the keyboard input routine which produces an input string to be used with the functions $M and $N. This function displays a blinking cursor on the screen and any characters entered through the keyboard are shown on the screen. The latter can be edited using the <delete> (7FH) key. <CR> or other control characters cause a return to the main program that called $K.
$L	Natural logarithm function with the base e. Domain unlimited. Range limited to real numbers that can represent natural logarithms of real numbers. Negative arguments return a zero result.
$M	Transforms an ASCII character from the keyboard input string into a REAL (6 byte format) number. The string pointer is incremented by one character unless at the end of the string (returned value 13 or character <CR>).

$N   Transforms a number represented by an input ASCII character string into a REAL (6 byte format) number. The string pointer is incremented to point to the next character after the number. Only integer and floating point decimal numbers are accepted. Entries in mantissa plus exponent format require to use the function $N twice and a short conversion routine.

$O   Prints a number on the screen in the format specified by the following two constants. For instance:

```
FGH$O7,3
```

means that the value of the variable FGH will be printed with 7 decimal places in front of the decimal point and 3 decimal places after the decimal point. An overflow will be indicated with a pattern 9999999.999 and an underflow with a pattern 0.000. In any case, 12 characters will be printed: sign, up to 7 figures for the integer part, the decimal point and three figures for the fractional part. If the second constant is 0, the decimal point will not be printed.

$P   Sign function. Its output is 1 for any positive number, -1 for any negative number and 0 for 0 input.

$Q   Square root function. Domain unlimited. Negative inputs produce a zero result. Range all positive integers that can be square roots.

$R   Return from subroutine (called by "$Z" plus label).

$S   Sine function. Domain unlimited. Range between -1 and +1.

$T   Tangent function. Domain unlimited. Range unlimited.

$U   Clock function. It produces an input string in the form "yy mm dd hh mm ss" to be read by the $M and $N functions, just like if the date and time were input from the keyboard using $K.

$W   Prints the ASCII character corresponding to the given number.

$Z   Call to subroutine. A label must follow immediately!

$Z+   Call only when the result of the previous instruction was positive. A label must follow immediately!

$Z-   Call only when the result of the previous instruction was negative. A label must follow immediately!

Jumps, conditional jumps and calls to subroutines are performed to labels. Labels are denoted by the symbol "#" followed by a label number. For example:

```
#345
```

is a valid label. The expression:

```
NJ-W$j+345
```

will cause a conditional jump on the above label only when the difference of NJ minus W is positive. If the result of the operation is zero, the operation is undefined: the jump may or may not occur!

Comments should be written between the symbols "<" and ">" and are ignored by the compiler program. For instance, the expression:

```
<This is a valid comment and will be ignored by the compiler>
```

is simply skipped by the compiler program. Note that the function "$<" uses a similar syntax, however the latter does have an effect on the execution of the program!

Machine code routines can be inserted in hexadecimal format using the expression "&<" to begin and ">" to end a hexadecimal listing.

The high-level language compiler produces a machine code program that includes the following parts:

- Program header
- Table of array variables
- Table of single variables
- Table of labels
- Compiled program

The compiled program uses all of the 16 general purpose registers of the CPU. The content of some of the registers should not be modified in an uncontrolled way to avoid crashing of the following compiled code: These are:

A7 - stack pointer
A6 - base address of input string buffer
A5 - base address of arithmetic routines (TRAP10)
A4 - base address of variables and labels
A3 - input string buffer pointer

Some registers can be used immediately to communicate with the high-level language when inserting machine code routines. These are:

A2 - Offset of the variable address
D1 - 16 bit exponent of the current math expression
D0 - 32 bit signed mantissa of the current math expression

Registers A1, A0, D7, D6, D5, D4, D3 and D2 are only used by arithmetic expressions and functions to hold intermediate computation results and their content can be modified without affecting the execution of the compiled high-level language program.

Further directions for using the high-level language compiler, either high-level language only or used together with machine code and operating system routines, can be found in the many application programs supplied for the DSP computer, and in particular their source code listings. The compiler only includes functions that were found necessary in writing applications software for the DSP computer. Although the latter may seem very restricted when compared to a commercial compiler running on a commercial computer, the trade-off is computation speed. The code produced by the high-level language compiler running on a 10MHz MC68010 can do about 20000 basic arithmetic operations (addition, multiplication or division) per second or compute about 1200 trigonometric functions (sine or cosine) per second.

8. PROGRAMMING HINTS

Regardless of the software support available, efficient real-time programming of any computer requires using its peripherals directly and thus writing small portions of the software directly in machine code. This is especially true if new peripherals are added to the computer, since no software support is available.

Considering the above fact, the software support in the high-level language compiler is intentionally kept to a minimum: it only includes the basic routines for keyboard input and screen output. On the other hand, the compiler is designed to allow very simple insertion of machine-code programs of any size and rather simple communication between machine code and high-level language software.

In order to write machine code programs it is necessary to study the hardware first, and in particular the microprocessor used. A complete description of the MC68010 microprocessor can be found in the following two books:

1) Motorola: 16-bit Microprocessors Data Manual, 1983.
2) Motorola: M68000 16/32-bit Microprocessor Programmer's Reference Manual, 1984.

Transferring parameters between machine-code routines and the high-level language main program can be accomplished in may different ways. To transfer single variables (not arrays) registers D0 and D1 can be used. For example, the expression:

```
AAA&<  ...HEX machine-code program... >
```

supplies the mantissa of the variable AAA in D0.L (32 bits) and the exponent of the variable AAA in D1.W (16 bits). Both can then be used by the machine code program. Similarly, the expression:

```
&<  ...HEX machine-code program...  >=AAA
```

assigns D0.L to the mantissa of the variable AAA and D1.W to the exponent of the variable AAA. Since machine-code programs require signed or unsigned integer data, the high-level language program has to care about the necessary conversions from and back to the floating-point format. A very simple way to perform these conversions is to use a special constant, defined by the floating point format used:

```
65536*32768=npr    ...or...    &<7000 323C 801F>=npr
```

Conversion from floating point to unsigned integer is then a simple addition operation. For example, the program:

```
AAA+npr&<13C0 0001 0003>
```

converts the value of the variable AAA to unsigned integer (of course npr has to be defined before!). The unsigned integer is available in D0.L. The following machine-code program takes the low-order 8 bits of D0.L and writes them to the immediate address 00010003H (spare parallel output port on the CPU board). The above program is thus equivalent to the BASIC "POKE" instruction.

This simple conversion rule also works in the opposite direction. For example, the expression:

```
npr&<1039 0001 0001>-npr=AAA
```

first sets D0.L and D1.W to npr, then reads a byte from the immediate address 00010001H (keyboard) into the low-order 8bits of register D0, then converts the result from unsigned integer to floating point and finally assigns the result to the variable AAA. The operation of the above expression is thus similar to the BASIC "PEEK" instruction.

Using different constants, conversions to and from signed integer format of any length up to 32 bits can be performed in a similar way.

On the other hand, high-level language variables can be used to store or transfer binary data between machine code routines. In this case no conversions are necessary, since these variables are not being used by the high-level language functions. For example, suppose that a machine-code DSP routine needs 1kbyte of memory space for a lookup table. This memory space can be assigned by declaring a high-level language array variable of suitable dimensions: The expression:

```
$:buffer(170) buffer(0)&<200A D08C>=address
```

first declares an array of 171 elements. Since every element requires 3 words or 6 bytes, this totals up to 171*6=1026bytes. The expression "buffer(0)" computes the relative address of the first element and stores it in A2. The following machine code adds A4 to A2 thus computing the absolute address of the first element of the array in D0.L and finally stores the result in the variable "address". The start address of the 1kbyte memory area is thus computed dynamically during program execution and conveniently stored in a variable that can be accessed at any time.

DSP programming usually includes both programming in machine code for the DSP algorithm itself and programming in a high-level language to support and use the DSP routines. To recall the DSP routines periodically interrupts have to be used. The following discussion describes how this can be done efficiently on the DSP computer described.

When a MC68010 microprocessor receives an interrupt request with a priority level higher than the current processor priority (described with the corresponding bits in the status register) a call to the interrupt handling routine is made. Start addresses of interrupt handling routines are stored together with other exception vectors in the exception vector table in the computer memory. The exception vector table is located in the system RAM during normal program execution (except at power-up). Its start address is stored in the Vector Base Register (VBR).

Hardwired logic on the processor board makes the MC68010 to operate in the auto-vector mode: each interrupt level is assigned a single vector in the exception vector table. After reset, all interrupt vectors are pointing to an error handling routine (except interrupt 7 - the operator keyboard) and the processor priority is set to 7. To use an interrupt, the following has to be made exactly in the order given:

1) Replace the current interrupt vector with a vector pointing to the interrupt handling routine.
2) Initialize the peripheral that will request the interrupt.
3) Adjust the processor priority level if required.

After the interrupt is no longer being used, the following has to be made to restore the original conditions:

1) Disable the peripheral requesting the interrupt.
2) Return the default interrupt vector (pointing to the error handling routine).
3) Return the processor priority level if modified.

The address of the memory location containing the interrupt vector is computed by adding the corresponding offset to the start address stored in the VBR. The offset is 64H for INT1 level auto-vector, 68H for INT2 ... up to 7CH for INT7.

From the user point of view it is very convenient to build interrupt routines into high-level language programs that can be handled or compiled as any other program. The compiler produces relocatable .EXE program files: the operating system considers that an .EXE file can be executed on any memory address. However, the exception vector table contains absolute addresses. The program should therefore include a routine to compute the interrupt routine address dynamically each time the program is executed on a new memory location.

In the case of the MC68010 it is convenient to use the Load Effective Address instruction, in particular LEA d(PC),An to find the actual program counter. The address-finding routine is usually located just in front of the interrupt routine so that the unknown start address can be computed easily.

Interrupt routines are usually DSP routines written almost entirely in machine code. However, high-level language expressions are necessary to interface to the rest of the program. Whenever a call to an interrupt routine is made, it is necessary to save the content of the microprocessor internal registers on the stack and restore it before exiting from the routine. Considering the many MC68010 internal register this operation may be very time consuming. Since most DSP machine code routines only need a few registers, only the latter are saved and restored before exit to save CPU time. High-level language expressions however use all MC68010 registers!

Programs designed for use in an interrupt driven multitasking environment require some additional features. Since different interrupt routines may belong to different main programs and the interrupt nesting is arbitrary, the content of all microprocessor registers is completely unknown when entering an interrupt routine. If high-level language expressions are used in interrupt routines, the values of registers containing base addresses, usually A4 and A5, have to restored for correct operation.

9. FUTURE UPGRADES

The DSP operating system V7.3 is the result of a long evolution. Therefore no major upgrades are planned for the operating system itself: any further upgrades are therefore planned just as applications software.

****