

Distant Multiaccess to NM6403 Processor

E.V.Chepin (Tchepine)
MEPhI System Co
Moscow, Russia
chepin@dozen.mephi.ru

S.V.Landyshev
MEPhI System Co
Moscow, Russia
yegresnal@mtu-net.ru

A.B.Vavrenjuk
MEPhI System Co
Moscow, Russia
wawren@dozen.mephi.ru

Abstract

The purpose of this work is the design of software for distant multiaccess to NM6403/04 processors. The given processor is the rather expensive hardware, therefore both for the objectives of learning, and for work of collective of the professional programmers it is necessary to have the software ensuring multiaccess to such hardware resources.

1. Introduction

Processor NM6403 is a single-purpose processor, which possible refer to DSP class. Processor NM6403 has original architecture and designed by the Russian company "Module". On the given processor are effectively decided problems an neuroinformatic, as well as problems of signal and image processing and number of others.

The given processor is the rather expensive hardware, therefore both for the objectives of learning, and for work of collective of the professional programmers it is necessary to have the software ensuring multiaccess to such hardware resources.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CSIT copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Institute for Contemporary Education JMSUICE. To copy otherwise, or to republish, requires a fee and/or special permission from the JMSUICE.

Proceedings of the 3rd International Workshop on Computer Science and Information Technologies CSIT'2001

Ufa, Yangantau, Russia, 2001

In Moscow State Engineering Physics Institute (Technical University) power of programmers MEPhI System Co this problem was a speech. For the base development will take a software programs package, ensuring access to the emulator an NM6403 charge. Correspond modulas of emulator are replaced on client modulas realizing interaction with the server, having in its composition real neuroprocessor charge. Being in accustomed for itself ambience Visual C++, user can develop and debug parallel programs in that mode, either as an user, working just on the computer with installed in it charge.

2. Architecture of DSP NM6403 and the Neuro Matrix NM2 PCI board [1,2].

From the outset processor NM6403 was designed as hardware accelerator for computation of direct signal propagation along neural homogeneous network with successive links [3,4]. It's capability to dynamically change width of the weights and neuron values marks it out of the other digital signal processors. Besides changing of weight and neuron values widths has an influence upon NM6403 performance. Primordial predestination of this DSP defined its name – Neuro Matrix 6403. Nevertheless some capabilities of this processor relate him to high performance digital signal processor (DSP), for example [5]. NM6403 architecture shown in Fig. 1.

RISC CORE – central processor unit that performs arithmetic and logical operations and shifts on 32-bit scalar data. It also forms 32-bit addresses of commands and data for loading them from memory. This is the main unit that manages processor's computation.

Vector Unit – performs arithmetical and logical operations on 64-bit vector data of programmable

capacity. It contains 3 memory blocks. Each of these blocks consists of 32 64-bit registers. Vector Unit contains also special purpose registers and functional block for matrix operations.

Control Unit – manages interrupt processing, controls pipeline commands. It also synchronizes access to internal and external buses.

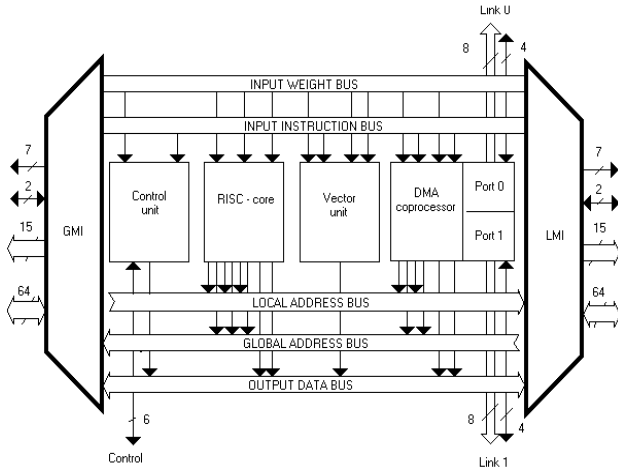


Figure 1. Architecture of DSP NM6403.

Local Memory Interface and Global Memory Interface – these programmable interface blocks with local and global 64-bit external buses can be connected to external memory which contains up to 2^{31} 32-bit words. It is possible to exchange with external memory by either 32-bit words or 64-bit words. In case of 64-bit words processor extracts two neighbor words. In this processor it is used paging. The same 15-bit address bus is used for high and low address bits. High address bits are used only while working with new memory page.

NM6403 has five internal buses through which data and commands are transmitted between processor blocks:

Local Address Bus and Global Address Bus are used for transmitting commands and data formed by RISC CORE in program mode or data addresses formed by communication ports in DMA.

Output Data Bus is used for transmitting data to local or global memory from RISC CORE, Vector Unit and communication ports.

Input Weight Bus and Input Instruction Bus are used for transmitting data and commands from local or global memory into any of main units of NM6403 connected to it. Both these buses can be used for either command transmitting or data transmitting.

Input Bus#1 and Input Bus#2 and Output Bus can transmit 32-bit and 64-bit data words during 1 processor cycle. So it is possible to combine in register pairs such operations as “register-register” and “memory-register”.

Commands are extracted from external memory as 64-bit words. Each of these 64-bit words is either one 64-bit instruction or two 32-bit processor instructions.

While exchanging data with external memory processor uses 32-bit address bus. Available processor’s address space is 16 Gigabytes. Most significant bit of the address chooses local (is equal 0) or global (is equal 1) memory. Less significant bit chooses low or high 32-bit half of 64-bit word.

Advanced interrupt system. NM6403 has 1 external interrupt and 9 internal interrupts.

But the most outstanding feature of this processor is capability to multiply data vector on data matrix. Programmer can specify the capacity of data fields in vector and matrix. This feature of NM6403 makes it possible for program to find compromise between performance and precision.

NM6403 is supplied on board MC4.02, which contains following components:

Global static memory bank. Its capacity is 512 kilobytes. This memory bank is accessible for both reading and writing by IBM PC. This bank is usually used for loading program into processor, weight matrix, binary images and so on.

Local dynamic memory bank. Its capacity is 32 Megabytes. This memory bank can be only accessed by NM6403. It is inaccessible from IBM PC.

The main features of MC4.02:

1. Processor

Number of processors	1
Clock rate	40 MHz

2. Input-output

Two communication ports with capacity	Up to 20MB per second each of them
Bus capacity	Up to 132 MB per second

3. Performance

Scalar operations on 32-bit data	Up to 120 MOPS
Vector operations on 8-bit data	Up to 960 MMAC

PCI controller. It connects boards to PCI.

3. Principle of working a server

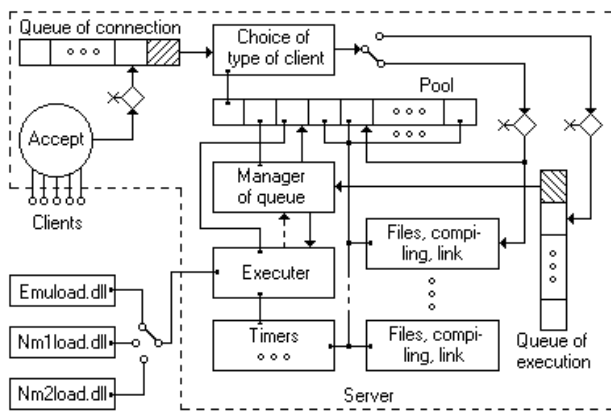


Figure 2. NM6403 Server Scheme.

NMServer is a program, ensuring access of users to PC plug-in cards with processor NM6403, as well as to the programme emulator from local and global network. Given version of server was developed for working under OS Windows9x. As clients are to emerge Windows and UNIX machines. Primary task of server - to give a chance work with the processor NeuroMatrix users, not having on their own computers of this processor. Server can be use for organizations of scholastic classes on educating a programming for NM6403.

Server - multithread program: the whole main work on servicing the requests of clients is realized by corresponding threads. Network interface is organized by means of sockets, ensuring work with the family of protocols TCP/IP. NMServer works in different modes, depending on tasks, decided by the client.

3.1. Explanations to scheme of working a server

On the scheme (Fig. 2) by arrows are shown way of data base communications. Data present itself a descriptor of socket. Rhombuses are mark blocks of checking an overflow of queues or pool before will write there data. Crosshair is marked refusal of work with the client because of the overflow of queues or pool.

After the connection all clients regardless of the type fall into the queue of connection, are whence choose by

the thread of choice of type of the client. All clients are to belong to one of two types:

- Clients, working with clients library (boot and performing the programs).
- Clients, working with clients application.

Users sockets of first type are make for the queue on the performance. Whence they are choose by the manager of queue of programs, and will be send in the thread - performer (executer) of programs. Thread - manager exists only then, when in queues is present at least one element, and while works a thread - performer. These two threads interact between itself as follows: from the thread - manager is generated thread - performer, in which will be send descriptor of socket for working with the client. Thread - manager is block before the arrival from the thread - performer of signal on the completion of work with the client, or prior to the expiration of the certain temporary interval (deblocking on timeout). Interrupt arrow shows a transmission of signal on the completion of work. At The Beginning of Initially its work each thread - performer starts a timer, which count a time lag, conducted on work with the client.

In the second event (works clients application) descriptor of socket will be send in the thread - request handler of clients application, which is created in the thread of choice of type of the client. Thread - handlers are mark on the scheme "files, compiling, link". In the name is display circle of questions, decided by these threads. At each moment of the time are to work several thread - handlers. Amount of such threads is limited by sizes of pool. In the pool, as from the fourth cell, are saved thread identifiers - handlers and corresponding to it descriptors of sockets. Thread - a request handler from clients application, as well as thread - performer, at the beginning of initially its work starts a timer.

Thread - performer works with dynamic libraries (DLL), ensuring access to NM6403 processor or emulator. Depending on installing a server in the program is load one of the three libraries. Calling the library functions are realized through pointers just from the library.

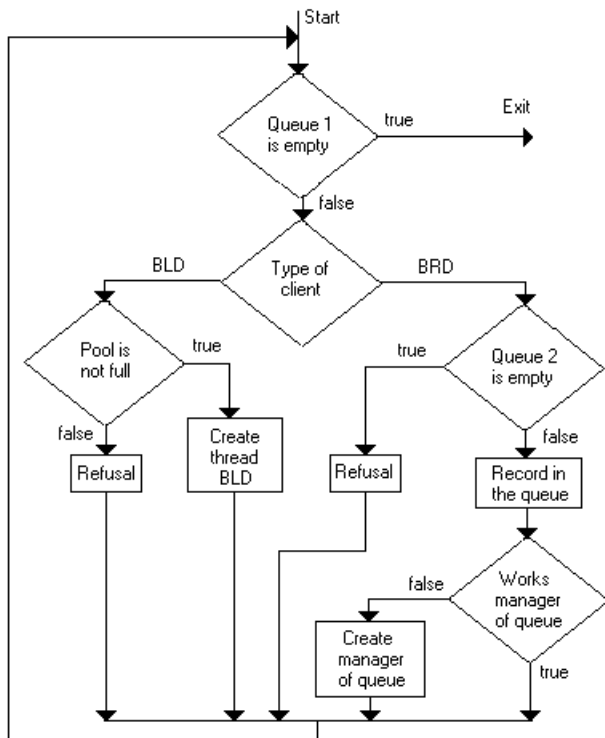


Figure 3. Functioning scheme a thread of choice of client type

During start a server loads required for its work a library. If server starting aptly, in the main window of application will appear a line "server started". At the arrival of signal on join from the client main window of application will get WSA_ACCEPT message (this message is determined in the program as user) and from its window functions will cause a method of class CServer for processing a received message. Hereinafter pointer on the copy of class CNMSocket (already having socket descriptor and line with IP-an socket address, which is defined in the constructor) is inserted in the queue 1 (queue of connecting). Insertion in the queue occurs after checking by methods a class CQueue overflow of queue. On the rice. 2. actions on checking a queue is marked by the rhombus. In the event of impossibility of insertion a server abandons to join (on the drawing a failure is placed cross). It is Necessary to note that in this queue get all created in the handler sockets (functioning scheme a thread of sample of client type seen on Fig. 3). Later, clients sockets, working with the charge, will move over to the second queue.

For the relationship with the client is created socket, having such parameters, as socket, created at the start of server, but with other descriptor number.

Reasons of highlighting of work on samples of type in the separate thread are a possible malfunction in

functioning (working) a server when using the blocking functions for functioning (working) with sockets that not possible when functioning (working) a main thread of program. Primary task of new thread is a reading signature, sending client of right after connection.

Client can call a server on functioning with the plug-in card ("BRD"), or on the dialogue with the server for viewing available on the server of programs with possible compiling and assembly of its program ("BLD"). For the simplicity of explanation of principle of functioning (working) all connecting are divided into two types. Actually under the type "BLD" are implied different requests, which are processed in correspond threads.

At the request of client to boot and performing a program for NM6403 processor ("BRD") client immediately falls into the queue 2. In the method of queue for the insertion of element is first of all checked not crowded queue and, if queue is packed - a client will be refused in connecting and thread of sample will go over to processing a following first queue element. Otherwise, program analyses, exists at given time thread for processing a queue 2 (thread - a manager of queue 2). If thread no, it is created and in the pool is brought identifier of right before creating thread.

If client has sent an inquiry to the assembly, or viewing the programs ("BLD") then, first of all, is realized searching free place in the pool. Absence of the free place in the pool means that with the server already works a maximum amount of clients (maximum number of clients is defined by the manager of server), and current client will be refused in connecting, but thread of sample terminate, complete its functioning. If in the pool be free cells, the pointer on the client's socket is brought in the pool, is created thread for functioning (working) with the client, and its identifier is also remembered in the pool.

Here should say several words on reasons, on which in the program are used two queues. This is because on the charge with installed by the processor NM6403 at each moment of the time can be loaded only one program, but inquiries to the assembly and viewing the programs can be executed with use several threads simultaneously. So all sockets from the first queue, working with the charge, are written in the second queue and are later chosen from it thread - manager of second queue. All tasks of other types begin its execution from the first queue just.

Hereinafter short described mainstreams, which must process a server. For some from them brought diagrams, explaining principle of their work.

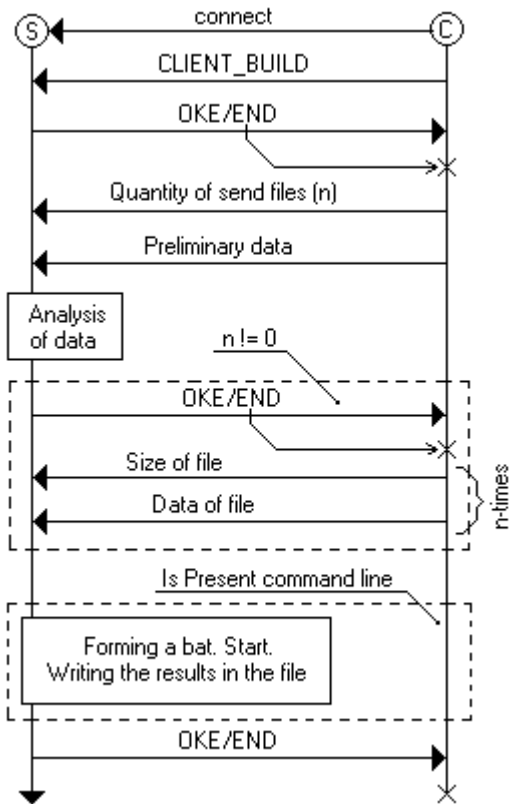


Figure 4. Protocol of exchange by information when receiving the files and assembly of programs

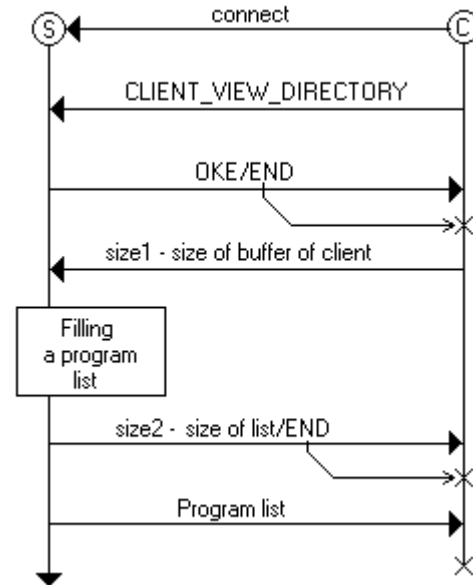
From all threads, processing requests of client application, largest volume of functioning executes a thread - a picker of programs. The whole functioning (working) this thread possible to divide into two part: first, in the thread are taken files, chosen by the user on removed machine for the transmission on the server; secondly, is caused program nmcc for compiling and assemblies performed on NM6403 modula. On Fig 4 brought protocol of exchange by information a server and client application when processing an inquiry to the assembly of programs and transmission of files.

In the beginning functioning from clients application are taken amount of sending files (n) and byte array, having following structure:

Name of program	Parameters nmcc	File name 1	...	File name n
14	256 bytes	on 14 bytes		

Thereby, this array possible to receive a visit at the buffer by the size: $256 + 14 (n + 1)$ byte. A Line without fall must be present In this array with the name of program (all components of array - a zero terminal lines), the directory name, in which will fit send files and in which will be compiled and be going to programs. Line - parameters for nmcc and name of files can be started ($n = 0$).

Thread - a manager of queue to boot and performing the programs for NM6403 processor. As a parameter in the function of thread will be sent pointer on the copy of class CNMSocket, with which works a current client. Thread falls into the cycle, condition of output from which is an unset determined in functions of flag. In the



cycle from the client is taken a number of function host-machines (functions with the prefix PL. By means of the plural choice operator is executed inquired function.

Figure 5. Protocol of exchange by information at program list transmission

Request of clients application on viewing existing on the server of programs is ensured by the thread of viewing the programs (Fig. 5). For getting a program list is determined protocol of exchange by information between the server and client.

Task of this thread is sending directories, basing in the directory by the certain server manager. At the assembly of program all source files are going to in the directory, name which always complies with the name performed on the processor NM6403 program, so sufficiently send a client names of directories that it had a presentation on all available on the server programs. The manager of server defines maximum amount of programs (or directories). At the task by the user a name of program by the server is superimposed restriction on its length. Name of program can not exceed 13 of symbols. Thereby, possible counts a length of symbol array, containing available program list:

$$\text{Len} = \text{Max_programs} * 14.$$

Here multiplier 14 is program name defined by the maximum length of with the account of terminating zero.

After installing a join client application sends a server inquiry number for the program list reception and expects in the answer a line "OKEY" - under the normal move of functioning, or "END" - at the malfunction (for instance, failure because of the overflow of queue). Parcel "END" brings about the join breakup. Hereinafter, client informs a server on the size of its buffer, for receiving a line with programs. Server forms a list available beside it directories and on checking an inequality $size1 \geq size2$ (Fig. 5) sends on the server or size of tinning line with the program list, if condition is executed, or "END", if condition is not executed. Either as in the first event "END" brings about closing a join. Hereinafter, server sends a client

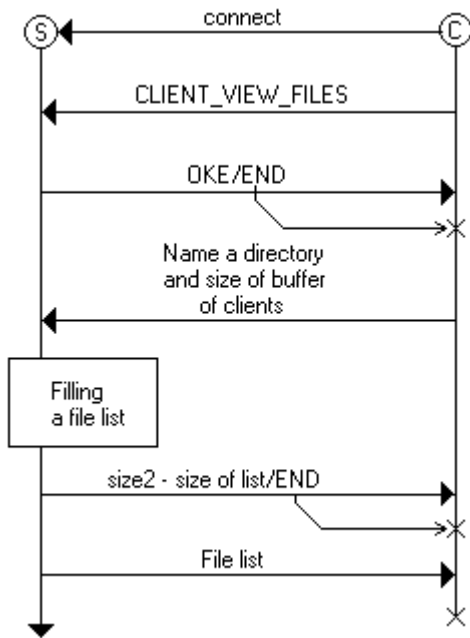


Figure 6. Protocol of exchange by information at file list transmission

program list and on this communication link is terminated together with the correspond thread.

Client Application can send on the server an inquiry for viewing contents what or directory (here there are in view of directories of programs, which base in the directory, determined by the manager). New thread will be organized for processing this request.

For getting a file list is determined protocol of exchange by information between the server and client, submitted for Fig. 6. Functioning (working) this thread looks like functioning (working) a thread, described above.

When functioning (working) with the server to the client is given a chance get required file. This functioning is executed by the thread on sending the files (Fig. 7). After beginning of its functioning in thread are expected data from client application: name of catalogue and filename. Is it hereinafter checked presence of inquired

file on the server. If file does not exist, the client will get from the server a line "END", otherwise - a size of file. Bytes of file are sent then.

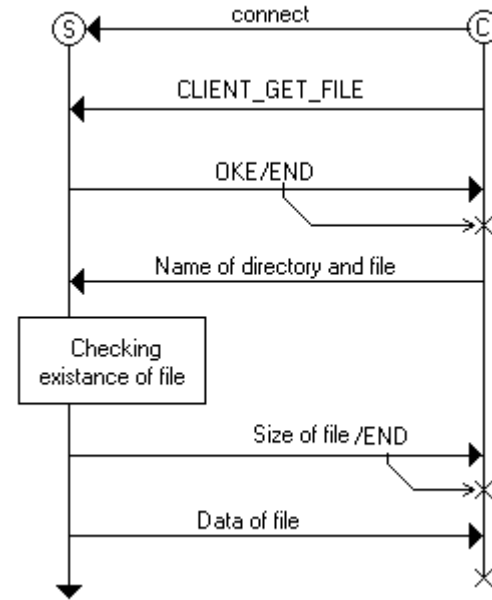


Figure 7 Protocol of information sending at file transmission

4. Results

4.1. Server Software Decision

The stopping on some aspects of realization. Access to the server by means of Internet possible due to the fact that the interaction with distant users is realized with using a stack of protocols TCP/IP. Server's Part marketed in the manner of applications under OS WINDOWS'95/98/NT. Client's part marketed in the manner of DLL-a modula completely compatible with modulas of emulator. For a time of starting its program each user monopoly seizes neuroprocessor's a facility. To limit this time several seconds, it is necessary to exclude interactive interaction during performing a program (reaction of user always slow). For this all necessary input given prepare in some file beforehand, whereupon task is dispelled in the packet mode. Herewith inevitably appears a queue of tasks, if size of queue exceeds certain value, next user be refused with the request to repeat a starting a task through several minutes.

After starting application a server moves over to the passive condition. For starting a server to the manager needed start from the menu. After the start a program conducts a row of initializing actions. Are they first of all read data from Windows system roll and is initialized corresponding variable server. Is it hereinafter created socket, characteristics, which will inherit all created jacks. Socket is translated in the Wait State of join by means of extensions of programs interface Windows Socket, intended for performing the

anisochronous operations. Program affords broad possibilities for administering a server by means of windows of adjustment. Installed type of the charge with options, which will be support in current work, amount of users, simultaneously working with the program. For clients possible to assign a maximum time of work with the server, on the outflow which user will be unplugged. Possible accommodation of directories of programs in any place on the disk.

For notices of the condition of server, on current work and connections server removes reporting in its window. Also leads a constant writing all reporting in the file. The Program Development with calling the functions with the prefix PL for host-machines, using facility of plug-in card, but not having this devices, is realized absolutely for the programmer transparent, that is developer does not come to change a source code of program for carrying it on the real device. The substitution dynamic library for working with real charges (or with the emulator) the client library ensures transparency. By means of this library is ensured interaction with the north, but signifies and access to NM6403 processor.

User is give a chance compiles and collect performs on the charge a program by means of special client application. Server collects all under development programs and derived files in corresponding to directories, whence program are load and execute on the signal processor. By means of clients program possible to view these directories and files from the distant computer.

4.2. Library of functions host-computer for working with server

Purpose of development an client Library - to ensure an access to the server for users - developers of programs for NM6403 processor.

All programs, using facility of plug-in card (charge with the processor NM6403) consist of two parts. The First part is kept a program, started on the charge, second is kept usual performed file for IBM PC, from which are caused functions of driver of device. The First part always must be present at the machine with the processor NM6403, second part can work at any other machine (without NM6403 processor) and address to the driver of charge on network. Exactly this idea bes the base of server.

Client Library is an additional section in the chain file.exe - a driver - NM6403 and changes direct addressing to the driver (themselves through the operating system mediation) on turning through the relationship channel. That is to say functions NM SDK are substituted by functions, which organizine a relationship channel, send on the server necessary data and take a result from the server. Real functioning (working) a program is executed by the server.

"Transparency" is ensured By means of libraries for the developer when making the programs, which address to the charge. The Programmer uses same header files and same functions, which used at the real program development.

Client Library was developed in the ambience Microsoft Visual C++ 6.0, and presents itself dynamically connected library (DLL). Main function of library a system can cause with some parameters, so when boot main function is caused with the parameter DLL_PROCESS_ATTACH - signifies that dynamic library is displayed in the address space of process. What or initializing actions possible to produce On this signal. Parameter DLL_PROCESS_DETACH means that library to the process anymore needed, at programmer can produce necessary actions for correct terminating a functioning with the library.

4.3. Client application for working with server

Client application ensures clients a possibility to view files, which inhere on the server, send its files on the server, compile and collect their own programs, as well as writes in the system roll data required for functioning (working) an client library.

Program was developed in the ambience Microsoft Visual C++ 6.0 with using an object-oriented approach. For the program-client designed several classes, by means of which is afforded suitable interface for the user and is realized interaction with the server.

Client application is executed in the manner of windows with bookmarks, which allow an user be quickly switched between different windows.

Program first has no relationships with the server.. Relationship Channel will be formed only in response to determined actions of user. Such interaction with the server allows avoiding an excessive load on it. The whole most further functioning (working) application is built on shaping the requests, which are then refered on the server. That is to say application works in the mode of transmission of commands for the server and relationship channel exists while goes a performing a command on the server and returning the results of processing the requests to the client program.

Interface of program for functioning (working) with the user is organized thereby that user can quickly view the contents of the directory of programs of server, as well as work with its tree of directories (files an client machine).

5. Summary and Conclusion

The main results are:

- Designed requirements and models on the base an sockets for the multi-user access to the single-purpose DSP processor NM6403.
- Developed server for the processor NM6403 in the ambience MS Visual C++, herewith designed number of C - classes.
- Designed by Library of functions host-computer for working with server and Client application.
- Now developed software passes an operation testing.

Prospects of development of server:

Life cycle of any program expects its most further improvement, in the course of which must be taken into account wish of users and marketed new ideas developers. Will here be stated offers on the improvement of server and specified spots of application of waking efforts of developer.

Given version of server on is poured some defects, one of which and, probably, the most essential is an absence of the mechanism to registrations and authentications of users-clients with granting each of them file space on computer-server (personal directory).

Noninteractive is a single essential defect. Are they at present led work on system modernizations for the reason optimization of main its parameters. Note that given system possible to use in scholastic purposes not only. Group of developers(MEPhI System Co), concerning with parallel programming for the given charge, also needs for periodic debugging their own programs. Our system will allow it will be limited by aquisition of one copy of charge or even lease of server

in Internet, equipped given by the charge. Some information about realization of distant multi-access to NM6403 processor may see at <http://www.mepsys.com>.

Acknowledgments

The authors thanks specialists from “Module” for technical and informational support. Special thanks to Kashkarov V. and Grusdev M.

References

1. Introduction to Neuro Matrix NM6403 architecture, version 1.1b [nm6403ao11b-r.pdf](#), RC «Module», www.module.ru.
2. NeuroMatrix® NM6403: User manual,RC «Module».
3. P.A. Chevtchenko, D.V. Fomine, V.M. Tchernikov, P.E. Vixne. "Neuroprocessor NeuroMatrix NM6403 Architectural Overview", 9-th Workshop on Virtual Intelligence/Dynamic Neural Networks, SPIE Proceedings Vol. 3728, 1999, pp.253-264.
4. P.A. Chevtchenko, D.V. Fomine, V.M. Tchernikov, P.E. Vixne. "Use of the Microprocessor NM6403 for Neural Net Emulation", 9-th Workshop on Virtual Intelligence/Dynamic Neural Networks, SPIE Proceedings Vol. 3728, 1999, pp.242-252.
5. A.V.Arshavsky, O.A.Levchenko, A.N.Nikitin E.V.Tchepine. Time parameters of some parallel image processing algorithms of hardware complex – TMS320C40+NM6403, CSIT-2000, v.2, pp 29-37.