To be published in
IEEE Trans. on Nucl. Sci.

BNL 20627

Conf. - 75116--15

Invited paper presented at
1975 Nuclear Science Symposium,
Sheraton-Palace Hotel, San
Francisco, November 19 - 21, 1975.

**THE BROOKHAVEN REACTOR EXPERIMENT CONTROL FACILITY
- A DISTRIBUTED FUNCTION COMPUTER NETWORK - ***

**D. G. Dimmler, N. Greenlaw, M. A. Kelley,
D. W. Potter, S. Rankowitz, and F. W. Stubblefield**

**Brookhaven National Laboratory
Upton, New York 11973**

November 1975

MASTER

*This work was carried out under the auspices of the Energy Research
and Development Administration.

THE BROOKHAVEN REACTOR EXPERIMENT CONTROL FACILITY
- A DISTRIBUTED FUNCTION COMPUTER NETWORK - *

D. G. Dimmler, N. Greenlaw, M. A. Kelley,
D. W. Potter, S. Rankowitz, and F. W. Stubblefield

Brookhaven National Laboratory
Upton, New York 11973

NOTICE
This report was prepared as an account of work sponsored by the United States Government. Neither the United States Government nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.

ABSTRACT

A computer network for real-time data acquisition, monitoring and control of a series of experiments at the Brookhaven High Flux Beam Reactor has been developed and has been set into routine operation. This Reactor Experiment Control Facility presently services nine neutron spectrometers and one x-ray diffractometer. Several additional experiment connections are in progress.

The architecture of the facility is based on a distributed function network concept as introduced and defined elsewhere¹. This paper is a statement of implementation and results.

Introduction

A computer network for real-time data acquisition, monitoring and control of a series of experiments at the Brookhaven High Flux Beam Reactor has been developed and has been set into routine operation. This Reactor Experiment Control Facility (RECF) presently services nine neutron spectrometers and one x-ray diffractometer. Several additional experiment connections are in progress.

The architecture of the facility is based on a distributed function network concept. This concept as well as the plans for the Reactor Experiment Control Facility have been introduced and defined at the 1973 Nuclear Science Symposium¹. The following, therefore, is a statement of implementation and results. Frequent references to the above referenced publication are made; thus, prior familiarization is recommended.

Some Specifications

The users of the facility introduced several mandatory requirements during the planning stage. The following is a discussion of the requirements and the results.

The facility should have a low operating cost. Besides the hardware maintenance personnel, the facility support staff should not exceed one person. This requirement prompted the explicit design and automation of almost every interaction between the user and the facility. This includes application program generation, magnetic tape management, and other functions usually provided by computer operators and system analysts.

The emphasis has been placed on the provision of real-time functions in the facility. Capacity and facilities for data reduction and analysis are provided as far as the results are necessary for the experimental activity. Off-line data reduction and analysis are provided by the BNL Central Scientific Computing Facility, a computer center with 2 x CDC6600 and 1 x CDC7600. Continued emphasis on this specification will be given during future expansions in order to avoid deterioration of the facility into a general purpose, operator intensive computing device.

The facility uses one application program language only. FORTRAN has been chosen because of its general acceptance in the scientific community. A one-level overlay structure has been chosen for FORTRAN overlay program structures. One FORTRAN program structure may consist of a total of 62 overlays. This represents, depending on conversion factors applied, approximately 15,000 to 20,000 FORTRAN statements.

There is essentially no possibility of accidental interactions between experiments. Subsystems in the facility are incrementally expandable with prevailing hardware and software. Local expansion at one experiment does not influence in any way the operating schedules and other conditions of the other experiments. This specification also implies that at some future time, parts of the facility may become technologically obsolete; the entire system, however, should not have a scheduled or otherwise expected obsolescence date.

Economy of Architecture

The distributed function architecture favors dedication of many programmable processors and many dedicated random-access memory modules. It favors sharing of high-quality (expensive) electromechanical peripherals and large-volume secondary storage devices. This architecture stays economically desirable and feasible as long as the following trends in technology continue to be true:

- (a) computer processors become inexpensive items. Sharing of processors by different functions for reasons of economy becomes increasingly desirable;
- (b) random-access computer memories continue to drop in cost. Presently a one-microsecond core memory costs approximately 1 cent/bit. Substantial decreases in cost can be expected if and when semiconductor memory is available on a large-scale basis. Traditional operating systems using memory sharing algorithms become economically unattractive;
- (c) high-quality electromechanical peripherals and secondary storage devices will continue to be too expensive for dedication. Sharing remains economically desirable;
- (d) on-line sharing of data structures and files is a standard requirement for modern networks. This requirement furnishes a logical reason for sharing of secondary storage devices.

* This work was carried out under the auspices of the Energy Research and Development Administration.

There seems to be a high probability that technology will proceed along this course in the foreseeable future.

Network

The facility presently includes 12 nodes* each containing at least one FDP11 processor and at least 24 kilowords of core memory. The nodes are interconnected via the transaction communication subsystem as shown in Fig. 1. The nodes are classified into three types according to the function they perform in the network.

- (a) One program development node manages the execution of all program development functions. Editing, compiling, linking and source library management are those functions.
- (b) Each experiment has assigned one private experiment control node. A private node may be considered an execution node within the network. The application programs are executed here.
- (c) All above mentioned nodes are connected via the transaction communication subsystem to one common node. The common node provides all shared on-line functions within the network.

Program Development Node

All program development functions, in particular compiling, linking and source as well as object library managements are provided by the program development node. This node most closely resembles in the hardware and in the software structure a traditional computer system. The node operating system is a standard BATCH/DOS system² provided by the Digital Equipment Corporation. All necessary peripherals are confined to this node. They are connected to a FDP11/40 with 28 kilowords of memory, a system disk with a capacity of 500 kilowords, a line printer, two file-oriented DECtapes, a magnetic tape unit, papertape equipment and a keyboard/printer. The procedures for development of system programs as well as for the development of FORTRAN written application programs are automated to a degree feasible with function specific batch procedures which are permanently stored on the system disk. Also system libraries are permanently stored and protected on the system disk. User source and object libraries are maintained on file-oriented DECtapes. The tapes containing source modules have the label SRC. The tape containing all object modules for one FORTRAN overlay program structure is labelled OBJ. The OBJ tape is keyed with a group code which is assigned once at the time the system directories are initially created at the tape. The group code identifies:

- (a) the experimentalist who owns the tape; and
- (b) one or several private nodes on which the overlay program structure is to be executed as explained later.

Those batch procedures which have to be known to the application programmer are self-descriptive.

Typical procedures are:

FMAIN - Compile a FORTRAN main program.

FOVL - Compile one of up to 62 FORTRAN overlay programs.

FLIB - Compile a FORTRAN subprogram and insert the object module into the appropriate library level on the OBJ tape. The user library is organized hierarchically in eight levels. The user assigns an appropriate level so that a subprogram has references only to lower level subprograms.

FLKMN - Link a FORTRAN main program with FORTRAN subprograms and all system libraries and submit the resulting load module to the common node for insertion into the load module library of the overlay subsystem.

FLKOV - Link a FORTRAN overlay and execute the above mentioned procedures.

FLKMNP, FLKOVF - Punch after the linking process the load module on a papertape for standalone operation as explained later.

The batch procedures maintain, update and report on the line printer for each FORTRAN program a directory describing the present status of the main program and all corresponding overlays. A sample directory is shown in Fig. 2. The first column shows a coded disposition of the program. The dispositions are:

- 0 = Overlay does not exist;
- 1 = Source code has been assembled;
- 2 = Object code has been linked;
- 3 = Load module has been inserted into the overlay subsystem library of the common node.

The second column shows the name of the source program as it is recorded on the appropriate SRC tape. Columns 3 and 4 show the corresponding names used in the object module library of the OBJ tape and in the load module library of the common node. The other columns show self-explanatory load module specific parameters.

Common Node

The common node* functions within the network as a shared service node. It manages high-quality peripherals as well as file storage on a 20 megaword disk. Interactions between any node and the common node is accomplished by a transaction as the basic unit of communication. A transaction is comprised of a dialogue with five interactions, as shown in Fig. 3. Three of those are messages transmitted via a memory-to-memory transfer. The requestors of a service, a private node or the

*For the definition of a node see Ref. 1.

*In Ref. 1 the term "central node" is used.

program development node, transmits a 16-word transaction request block to the common node. The responder, the common node, validates the request, retrieves, if necessary, information from the disk and returns a 16-word acknowledge block to the requestor. If the request specified a data transfer and the common node returned a positive success flag in the acknowledge block, a variable data transfer of a maximum of 8192 words will follow. In the example of a FORTRAN overlay request, the contents of the various blocks would be as follows:

The request block contains essentially:

Transaction type: data transmission requested;
Transaction destination: overlay subsystem;
Function: read overlay;
Name: name of the main program presently active in the private node;
Overlay number: integer 1 to 62.

The common node directs the request to the overlay subsystem, reads and verifies the requested overlay module into a data buffer and submits the acknowledge block containing:

the entire request information;
the start address of the load module in the core memory of the private node;
the load module size;
the program transfer address;
the success flag; and
various error indicators if the request was unsuccessful.

Then the load module will be transferred to the requestor. The elapsed time between the origination of the request and the completion of the transaction is a few hundred milliseconds.

The hardware of the common node consists of a of a FDP11/40 processor, 80 kilowords of 16-bit memory, a 20 megaword disk and two magnetic tape units.

The common node software structure is divided into two global partitions:

- (a) the common node operating system CNDOPS; and
- (b) a collection of tasks comprising subsystems.

The task-oriented common node operating system operates in the "kernel mode" of the FDP11/40 processor. Detailed descriptions of this software structure are given elsewhere^{3,4,5}. The various subsystems are programmed as tasks and executed in the "user mode" of the processor. Two task level subsystems have been implemented and are in operation. The overlay subsystem manages all program load modules residing within

the common node. The fixed-record device subsystem manages shared record-oriented output devices as described in more detail elsewhere⁶. Presently two magnetic tape units are managed which are selectively used for recording of data obtained at the experiments. The fixed-record device subsystem manages ASCII coded 80-character card images within an ANSI labelled file. The facility is used by many visiting scientists. Thus, a high probability of media and file protocol compatibility of the recorded output with other university computer centers is essential. The file management subsystem is in development.

All software structures within the common node have been developed and implemented specifically for the common node.

Private Node

A private experiment control node may be considered an execution node within the network. The hardware includes a FDP11 processor, 24 kilowords of memory, a paper tape reader, a keyboard/printer and a private node terminal of the transaction communication subsystem. The software structures include two network specific subsystems and two experiment specific subsystems. The network specific subsystems are the private node operating subsystem PNDOPS and the solicited transaction subsystem. The latter facilitates communication between the private node and the common node via the transaction communication subsystem. The experiment specific subsystems are the application subsystem and the experiment service subsystem. The experiment service subsystem consists here of a software subsystem managing the experiment interface hardware. It is considered here an "incomplete node" as defined elsewhere¹, residing in the private node.

Application Subsystem

As mentioned before, application programs are coded in FORTRAN within an overlay structure allowing up to 62 overlays of a maximum of 8192 words. The private node core memory contains at any given time the main program, one overlay program and the FORTRAN COMMON area. A typical overlay structure which has been developed and is in routine operation for data acquisition and automatic control of an x-ray diffractometer and several neutron spectrometers consists of one main program and 23 overlays⁸. A private node has potential access to up to 224 different main programs which reside within the overlay subsystem of the common node. Communication of the application subsystem to the experiment service subsystem and the solicited transaction subsystem is established by using CALL statements which reference system subprograms.

Private Node Operating Subsystems

The private node operating subsystem, PNDOPS, provides support functions for the FORTRAN application subsystem as described below. The subsystem communicates with the system subroutines of the application subsystem via emulator traps as described elsewhere⁹. Each trap represents a discrete function. Managing of local Input/Output devices is accomplished by drivers, which are part of the subsystem. A specific private node operating subsystem can be generated at the program development node by supplying a table of the desired functions and the desired local devices to the batch procedure GUSOPS (generate universal operating subsystem). A maximum of 8192 words are provided in the private node memory for the resulting load module.

Experiment Service Subsystem

A specific experiment service subsystem is developed at the time an experiment setup is constructed or an experiment is switched to the facility. Every potential function of the experiment hardware is supplemented by a FORTRAN CALLable subroutine. The CALL statements have the general format:

CALL XXXXX(AL,...IS)

where IS is a standardized success flag associated with every call. In order to avoid homonyms with the application subsystem, 6 character subprogram names are reserved for the experiment service subsystem.

It is assumed, as justified elsewhere¹, that the experiment service subsystem is subject to frequent changes due to changing requirements and to technological advancements. Thus, the strategy has been pursued to design the CALL statements as functionally defined and as hardware and software structure independent as feasible in order to avoid major application program modification in cases of update of the experiment service subsystem.

The neutron spectrometer service subsystem may serve as a typical example. A neutron spectrometer includes a series of computer controlled motors operating in a range of 720 degrees as well as several detector-counter combinations. The motor positions are determined by using incremental high-precision encoders. The control circuitry had been designed about 12 years ago for the Multiple Spectrometer Control System¹⁰ which was replaced by the Reactor Experiment Control Facility. In order to achieve an orderly development plan within given budget and manpower constraints it was decided to interface to the old control circuitry for initial deployment of the facility and to develop new and modern circuitry later. Many functions which are presently designed in software will then be implemented in hardware. This implies that the present software structures and the hardware structure of the experiment service subsystem will then become obsolete entirely. The CALL statements to the experiment service subsystem have been designed so that this effort can be implemented without modification of the application software as long as the functions do not change. The motor control commands are for example:

CALL MOVNMT (IU, P, IM, IS)

which initiates movement of motor IU to the target position P either within a 720 degree range (IM ≠ 0) or in a circular mode (IM = 0).

CALL TSTMOT (IU, IT, IS)

reads the present status of motor IU, where the possible states are:

the motor is in idle status;

the motor is in motion;

the motor has successfully arrived at the target position;

the function was prematurely terminated.

CALL ZERMOT (IU, IS)

initiates movement of motor IU to the mechanical zero position;

CALL ABOMOT (IU, IS)

stops unconditionally movement of motor IU and returns the status to idle;

CALL SETENC (IU, PN, PO, IS)

sets the value of the present position of the motor IU from the old position PO, which is here a return parameter, to be the new position PN.

CALL REDMOT (IU, P, IS)

reads the present position P of motor IU.

CALL REDCNT (IN, P, IS)

reads the content P of the detector-counter combination IN.

Hardware Modularity and Bus Structure

Experiment service subsystems for different experiments are in development. In association with this activity a modern hardware structure is being developed for modular assembly and modification of experiment hardware. The importance of the selection of one bus structure has been recognized.

Modern bus structures have at least the following conceptual specifications:

- (a) minimum restrictions on the node hardware structure and mechanical measurements;
- (b) uncommitted data and control flow direction within a bus (master/slave control organization);
- (c) uncommitted interconnection architecture for the connection of multiple busses to nodes;
- (d) provision for separation between data and control flow;
- (e) request/response relationship of the control and data flow at all levels.

The UNIBUS^{*} of the Digital Equipment Corporation satisfies these requirements. Therefore, this bus structure has been selected. Modules are designed as closely to the specification of the Digital Equipment Corporation as feasible¹¹. There are some additional advantages to the UNIBUS structure. It is in widespread use, far beyond the somewhat specialized field of process control. Many industrial companies offer compatible modules, components, connectors, sub-assemblies and interfaces. It seems that the UNIBUS is rapidly becoming a widely accepted standard interconnection structure.

* Registered trademark of Digital Equipment Corporation.

Standalone Operation

A private node may operate at the option of the experimentalist as a standalone system as long as services from the common node are not required. A limited local overlay subsystem is available using a papertape reader as an input device for appropriately requested overlay load modules. There is, of course, a practical limit to a few overlays. A dual floppy disk device is presently in the final tests for use in this application. The two disks which would operate on such a device would hold a structure of 27 overlays. If the reliability of the devices and the media turns out to be satisfactory then it is contemplated to replace the papertape overlay subsystem by a much more convenient floppy disk overlay subsystem. The load modules for the local overlay subsystem are, as mentioned earlier, generated at the program development node.

Standalone operation is, in conjunction with the distributed function concept, particularly useful during some phases of the initial development and construction of an experiment. Often new devices, such as a new type of a motor or a detector, need to be used. These devices, then, prompt the development of new hardware modules and software modules within the experiment service subsystem. These new designs as well as the new devices can be developed and thoroughly tested out within small and defined functional subsystems. Later integration of several subsystems into a private node and integration of the private node into the network takes place without disturbance of the functionally defined boundaries of the new modules and without redesign of already existing functions. This process provides reasonable assurances that:

- (a) the probability of design errors remaining in the new subsystems is low; and
- (b) it is unlikely that the integration process introduces errors in the existing subsystems and thus minimizes difficult tests of the overall system.

Transaction Communication Subsystem

Transactions are submitted via point-to-point communication lines between the private nodes and the common node. Each line consists of a common node terminal, a private node terminal and a cable with 60 individually shielded twisted pairs. Each pair carries a unidirectional signal. In order to avoid possible interferences with the high electrical noise typical for experimental areas, each pair is terminated at its receiving end with a high-speed optically coupled isolator. Such an isolator which is manufactured by Hewlett Packard provides a DC insulation of 2500v and guarantees a common mode rejection of 30v at 2 MHz.

The hardware control logic has the scope of a transaction. As mentioned earlier a transaction consists of five consecutive interactions. Word transmission within the interactions as well as message transmissions within the transaction are asynchronous with respect to each other. All control signals generate bidirectional exchanges. The achievable speed of the communication subsystem, then, depends on:

- (a) the distance between the two terminals. A word transfer requires at least 2 times the cable delay time;

- (b) the bandwidth of the terminal to memory UNIBUS in the common node.

Presently, the distance between the common node and the furthest private node is 200 feet. In order to avoid an interlock of the asynchronous transmissions in case of an error, the three messages as well as the complete transaction are protected by time-out circuits. For economic reasons it was decided to connect all transaction communication terminals and all peripherals to one UNIBUS at the common node. Two peripherals generate synchronous transfer rates. The disk generates 153,000 discrete UNIBUS transfers per second and a magnetic tape unit generates 36,000 discrete UNIBUS transfers per second. In order to avoid possible conflicts with those transfers, each point-to-point transaction communication line has been limited to a rate of approximately 100 kHz.

Example

It is here the intent to show the convenience of the facility to those experimentalists who want to be bothered with computer specific information only to the extent that enables them to design, document and manage large FORTRAN overlay program structures. Let's assume an experimentalist wishes to modify within the overlay structure NEXDAS a FORTRAN subprogram ATOX which is referenced by overlay number 34. He will proceed with the following sequence of actions:

EDIT: This experimentalist considers the on-line editor supplied by the Digital Equipment Corporation too complicated and too cumbersome to use. He, therefore, prefers to edit a punched card source code deck and to transfer the corrected source module via the batch procedure CRTSRC to the SRC tape.

COMPILE: The experimentalist mounts at the program development node as shown in Fig. 4, the appropriate SRC tape and the OBJ tape. He then initiates the procedure FLJB. The two questions within the procedure are:

NAME? - (answer) ATOX

LEVEL? - (answer) 7

The procedure now compiles the subprogram and inserts the object module to the OBJ tape. A source listing and an updated list of all subprograms residing at this level are printed.

LINK: The experimentalist initiates the procedure FLKOV. The two questions within the procedure are:

MAIN PROGRAM NAME? - (answer) NEXDAS

OVERLAY NUMBER? - (answer) 34

The first question is redundant, because a tape only contains one main program. The information is used here to verify correct tape mounting. The procedure now links the object module of overlay 34 with the eight user libraries residing on the OBJ tape as well as a series of system libraries which reside on the system disk. The resulting

load module is transmitted to the common node and inserted into the overlay subsystem. A load map and the updated overlay structure directory as shown in Fig. 2 is printed. The experimentalist is now finished with the procedures at the program development node.

EXECUTION: At the appropriate private node, as shown in Fig. 5, the experimentalist responds to the questions:

WHICH PROGRAM? - (answer) NEXDAS

GROUPCODE? - (answer) 200

The latter question is again redundant and used for verification. The appropriate program is retrieved from the common node and is initialized at the private node.

Remarks and Future Plans

Within the approximately nine months, in which the network has been in operation, the distributed function architecture has proven to be very suitable for experimental control applications. Further developments proceed presently in three directions.

Firstly, some additional private nodes are in the process of being connected to the Reactor Experiment Control Facility. One of the experiment service subsystems in development will service a two-dimensional position-sensitive detector for thermal neutrons¹². Secondly, single, small experiment data acquisition and control systems are designed as standalone private nodes as far as the hardware structure and the system specific software structure are concerned. An example of such a system is given elsewhere^{13,14}. This approach offers several advantages:

- (a) major predesigned and tested subsystems, particularly software subsystems, can be used;
- (b) the automated batch procedures of the program development node can be used as a standardized design tool; and
- (c) the standalone system can be functionally extended without redesign within the node any time a functional expansion becomes desirable. This implies also that the standalone system can be at any time integrated into a network such as the one described above in order to gain access to additional functions.

Thirdly, the suitability of the architecture to control systems for large and complex machines is studied^{15,16}.

Acknowledgements

The Construction Group, under the direction of D. Stephani, was responsible for a large portion of the layout and the construction of the printed circuit boards. The Computer Service Group, under the direction of P. Gillaeny, was responsible for installation and testout of the system hardware. The invaluable service of both groups is gratefully acknowledged.

References

1. D. G. Dimmler; Functional Distribution - An Architecture for Multi-User Computer Networks in Instrumentation; IEEE Trans. Nucl. Sci., NS-21, No. 1, 832 (February 1974).
2. DOS-11 Monitor Handbook, Digital Equipment Corporation.
3. F. W. Stubblefield and D. G. Dimmler; Transaction Processing in the Common Node of a Distributed Function Laboratory Computer System. IEEE Trans. Nucl. Sci., Vol NS-22, 1 (Feb. 1975) 473-479.
4. F. W. Stubblefield and D. G. Dimmler; A Task Scheduler for the Common Node of a Distributed Function Laboratory Computer Network: to be presented at IEEE Nucl. Sci. Symp., Nov. 19-21, 1975.
5. F. W. Stubblefield; Logical and Physical Resource Management in the Common Node of a Distributed Function Laboratory Computer Network: to be presented at IEEE Nucl. Sci. Symp., Nov. 19-21, 1975.
6. F. W. Stubblefield; Continuous Sharing of a Record-Oriented Output Device Within a Distributed Function Laboratory Computer Network: to be presented at IEEE Nucl. Sci. Symp., Nov. 19-21, 1975.
7. D. G. Dimmler; Network Specific Software Structures in a Private Node of a Distributed Function Laboratory Computer Network (in preparation).
8. R. K. McMullan, private communication.
9. FDP11 Processor Handbook, Digital Equipment Corporation.
10. D. R. Beaucage, M. A. Kallay, D. Ophir, S. Rankowitz, R. Spierad, and R. Van Norton; Multiple Spectrometer Control by Computer; Nucl. Instr. and Meth. 40, 26 (1966).
11. FDP11 Peripheral Handbook, Digital Equipment Corporation.
12. J. L. Alberi, J. Fischer, V. Radeka, L. C. Rogers, and B. Schoenborn; A Two-Dimensional Position-Sensitive Detector for Thermal Neutrons; Nucl. Instr. and Meth. 127 (1975) pp. 507-523.
13. D. G. Dimmler; Real Time Acquisition of Oceanographic Data; Proc. of Conf. on Computer Support of Environmental Science and Analysis, Albuquerque, July 1975.
14. D. G. Dimmler, N. Greenlaw, S. Rankowitz; A Controllable Automated Environmental Data Acquisition and Monitoring System; to be presented at IEEE Nucl. Sci. Symp., Nov. 19-21, 1975.
15. D. G. Dimmler; The ISABELLE Control and Monitoring System; BNL-CRISP 75-5 (April 1975).
16. Proc. 1975 ISABELLE Summer Study Group; BNL 20550, Vol. II, Computer and Control (in press).

Figure Captions

- Fig. 1. Architecture of the Network.
- Fig. 2. Directory of a FORTRAN Overlay Program Structure.
- Fig. 3. Flow of a typical transaction
- Fig. 4. Program Development Node.
- Fig. 5. Private Experiment Control Node.

AUTOMATIC OUTPUT ENVIRONMENT

CONTROL & MONITORING ENVIRONMENT

PROGRAM DEVELOPMENT ENVIRONMENT

EXPERIMENT ENVIRONMENT

#CODE	SRC	OBJ	ID	SIZE	START	XFER	GR. COL
3	NEXDAS.SRC	NEXDAS.OBJ	0	32434	105024	105024	
#3	OVER1.SRC	OVL01.OBJ	1	3443	75714	75714	
#3	OVER2.SRC	OVL02.OBJ	2	10645	63310	63310	
#3	OVER3.SRC	OVL03.OBJ	3	11235	62330	62330	
#3	OVER4.SRC	OVL04.OBJ	4	11437	61724	61724	
#3	OVER5.SRC	OVL05.OBJ	5	11067	62644	62644	
#3	OVER6.SRC	OVL06.OBJ	6	11060	62662	62662	
#3	OVER7.SRC	OVL07.OBJ	7	6124	70552	70552	
#3	OVER8.SRC	OVL08.OBJ	8	4757	73064	73064	
#3	OVER9.SRC	OVL09.OBJ	9	1173	102434	102434	
#3	OVER10.SRC	OVL10.OBJ	10	2420	77762	77762	
#3	OVER11.SRC	OVL11.OBJ	11	5206	72406	72406	
#3	OVER12.SRC	OVL12.OBJ	12	4757	73064	73064	
#3	OVER13.SRC	OVL13.OBJ	13	4757	67272	67272	
#3	OVER14.SRC	OVL14.OBJ	14	6654	67272	67272	
#3	OVER15.SRC	OVL15.OBJ	15	13402	56016	56016	
#3	OVER16.SRC	OVL16.OBJ	16	7513	65574	65574	
#3	OVER17.SRC	OVL17.OBJ	17	13126	56546	56546	
#3	OVER18.SRC	OVL18.OBJ	18	0	0	0	
#3	OVER19.SRC	OVL19.OBJ	19	0	0	0	
#3	OVER20.SRC	OVL20.OBJ	20	0	0	0	
#3	OVER21.SRC	OVL21.OBJ	21	3107	76604	76604	
#3	OVER22.SRC	OVL22.OBJ	22	15403	52014	52014	
#3	OVER23.SRC	OVL23.OBJ	23	0	0	0	
#3	OVER24.SRC	OVL24.OBJ	24	0	0	0	
#3	OVER25.SRC	OVL25.OBJ	25	0	0	0	
#3	OVER26.SRC	OVL26.OBJ	26	0	0	0	
#3	OVER27.SRC	OVL27.OBJ	27	0	0	0	
#3	OVER28.SRC	OVL28.OBJ	28	0	0	0	
#3	OVER29.SRC	OVL29.OBJ	29	7540	65522	65522	
#3	OVER30.SRC	OVL30.OBJ	30	0	0	0	
#3	OVER31.SRC	OVL31.OBJ	31	2156	100466	100466	
#3	OVER32.SRC	OVL32.OBJ	32	13537	55524	55524	
#3	OVER33.SRC	OVL33.OBJ	33	0	0	0	
#3	OVER34.SRC	OVL34.OBJ	34	3632	75336	75336	
#3	OVER35.SRC	OVL35.OBJ	35	4571	73440	73440	
#3	OVER36.SRC	OVL36.OBJ	36	0	0	0	
#3	OVER37.SRC	OVL37.OBJ	37	0	0	0	
#3	OVER38.SRC	OVL38.OBJ	38	0	0	0	
#3	OVER39.SRC	OVL39.OBJ	39	0	0	0	
#3	OVER40.SRC	OVL40.OBJ	40	0	0	0	
#3	OVER41.SRC	OVL41.OBJ	41	0	0	0	
#3	OVER42.SRC	OVL42.OBJ	42	0	0	0	
#3	OVER43.SRC	OVL43.OBJ	43	0	0	0	
#3	OVER44.SRC	OVL44.OBJ	44	0	0	0	
#3	OVER45.SRC	OVL45.OBJ	45	0	0	0	
#3	OVER46.SRC	OVL46.OBJ	46	0	0	0	
#3	OVER47.SRC	OVL47.OBJ	47	0	0	0	
#3	OVER48.SRC	OVL48.OBJ	48	0	0	0	
#3	OVER49.SRC	OVL49.OBJ	49	0	0	0	
#3	OVER50.SRC	OVL50.OBJ	50	0	0	0	
#3	OVER51.SRC	OVL51.OBJ	51	0	0	0	
#3	OVER52.SRC	OVL52.OBJ	52	0	0	0	
#3	OVER53.SRC	OVL53.OBJ	53	0	0	0	
#3	OVER54.SRC	OVL54.OBJ	54	0	0	0	
#3	OVER55.SRC	OVL55.OBJ	55	0	0	0	
#3	OVER56.SRC	OVL56.OBJ	56	0	0	0	
#3	OVER57.SRC	OVL57.OBJ	57	0	0	0	
#3	OVER58.SRC	OVL58.OBJ	58	0	0	0	
#3	OVER59.SRC	OVL59.OBJ	59	0	0	0	
#3	OVER60.SRC	OVL60.OBJ	60	0	0	0	
#3	OVER61.SRC	OVL61.OBJ	61	0	0	0	
#3	OVER62.SRC	OVL62.OBJ	62	0	0	0	

FIGURE 2

FIGURE 3

FIGURE 4

FIGURE 5