intel®

APPLICATION NOTE

AP-244

October 1985

Distributed Job Control the Key to Increased Network Productivity

SRIVATS SAMPATH
DSO APPLICATIONS ENGINEERING

Order Number: 231480-001

INTRODUCTION

Large software projects and shorter production schedules generate the need for a more flexible and productive development environment, which allows users full access to all available resources.

Recognizing this need, Intel designed the Distributed Job Control (DJC) Facility into the NDS-II system. DJC allows currently idle networked development systems to be supplied to the network as public resources. This is essentially a remote job execution unit to which jobs can be sent by other users on the network. Remote job execution offers higher throughput and increased efficiency, since more than one computer on the network can be controlled and used by a single user. This ability to manipulate idle systems on a network and convert them into productive systems for other users directly translates into increased project productivity.

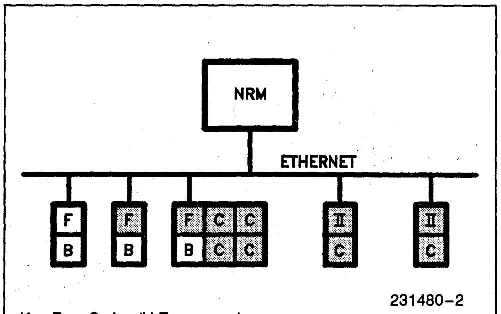
DJC consists of a set of system utilites that enable the NDS-II system manager to more efficiently run the network. When all idle systems on the network are allocated to other active users, the throughput and efficiency of the network dramatically increases. The Network Resource Manager (NRM) is the nerve center for the distributed job control system (DJC). All jobs are scheduled and queued by the NRM. The NRM also coordinates job cancellation and maintains a system log of job queue activity. DJC, with its powerful set of options, positions itself as an invaluable tool for increased network productivity.

WHY DISTRIBUTED JOB CONTROL?

The need for distributed job control (remote job execution) is apparent in a networked environment where a number of teams are working on different projects. With DJC, all idle systems can be channeled towards the particular time-critical project. As a result, the engineers have control over more than one system and increase their efficiency and productivity.

Figure 1 shows a typical NDS-II system. This network includes the NRM configured with two 84 MB Winchester drives, a 600-LPM line printer, three Series IV Microcomputer Development Systems (one of which has four cluster boards), two Series IIs with one cluster board each, and an assortment of ICETM and I2ICETM modules. Although the development systems are functionally similar, they are logically different as viewed by the NRM. Figure 2 illustrates the difference. Two teams are working on this network. Team 1 is an 8-bit development team, and Team 2 is a 16-bit development team. Both teams are meeting tight deadlines and need all the system time that they can get. One engineer working on the 8086 project is on vacation, as a result, one Series IV is underused. The other two Series IVs do not have anything running in their background. On an average of the 14 computers available to this network, (the Series IVs being counted as two each with foreground/background capabilities), only 10 are being used. The percentage use rate is only 60 percent when it should be close to 100 percent. Percentage use rate can be defined as:

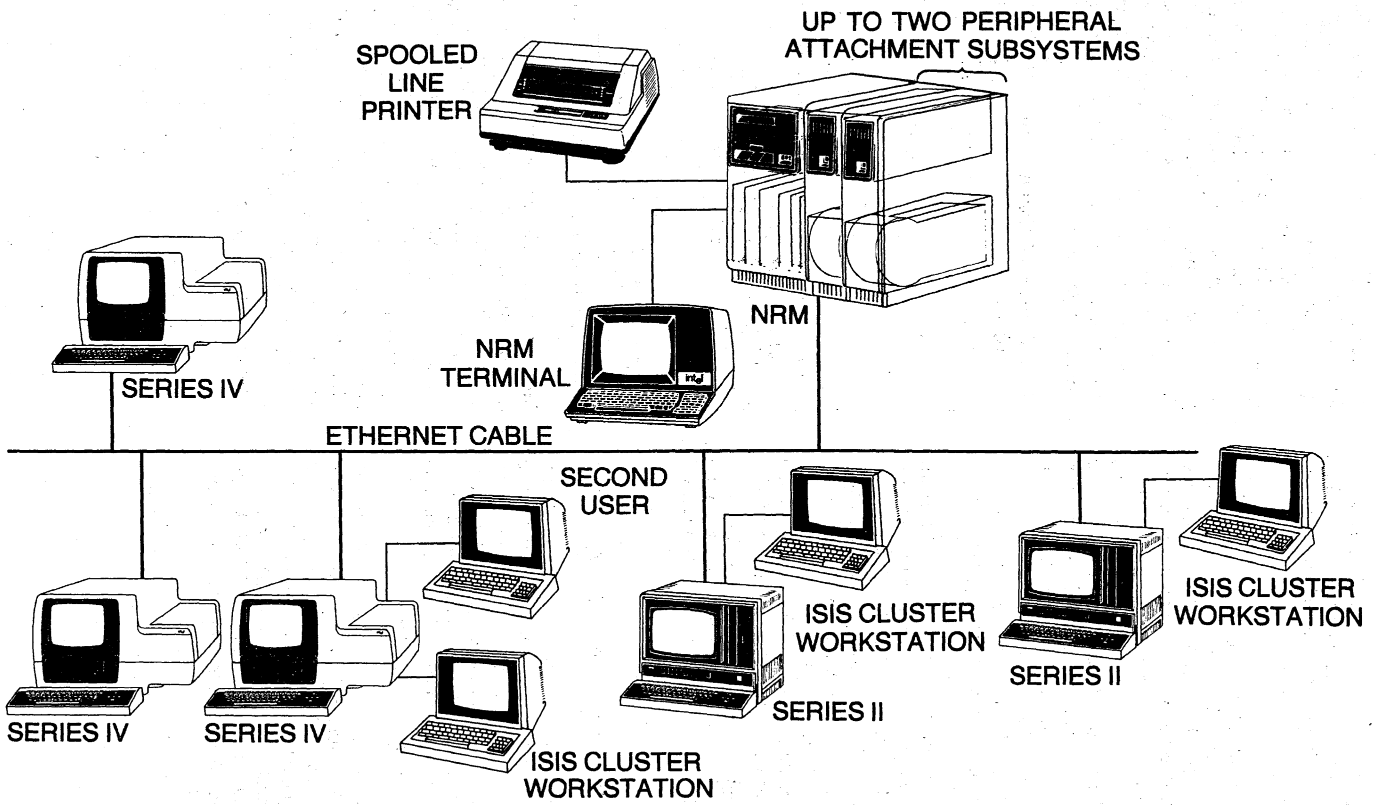
(Total Number of Nonidle Systems/Total Number of Systems)*100



Key-F = Series IV Foreground
B = Series IV Background
C = ISIS cluster board

Figure 2. Non-Idle Computers are Shown Shaded

Meanwhile, the 8-bit team is trying to meet a very tight schedule and needs all the system time possible. This team requires a dedicated compile engine that will free its systems for interactive work, such as debugging and editing. DJC can help the 8-bit team by converting the idle machine and the backgrounds of the other two Series IV systems to productive work doers. This enables Team 1 to have all their compiles remotely executed while they concentrate on editing and debugging other modules. These remote execution units can serve both teams, since the Series IV can operate in both 8-bit and 16-bit modes. This results in definite increase in overall team and network productivity.



231480-1

Figure 1. A Typical NDS-II Network

A CLOSER LOOK AT DISTRIBUTED JOB CONTROL

The DJC system recognizes that the network has three types of stations: the NRM, private workstations, and import workstations. The NRM is the nerve center of the DJC system, and it maintains all the state information of remote jobs and status of workstations. A private workstation is one that can send jobs to the NRM and have them executed at other workstations on the network. However, it does not accept jobs from the NRM. These are clasified as work generators. Examples of work generators are Model-800, Series II, Series III, Series IV and ISIS clusters.

An import workstation that can accept jobs from the NRM is called a work doer. Examples of work doers are Model-800, Series II, Series III, Series IV and ISIS clusters. Work generators and work doers use the same hardware. The software executing at the workstation determines if it is a generator or doer. The mix of generators/doers may be flexibly altered through the day/week/project to best suit the user's needs. Normally, when a workstation is first powered up or reset, it configures itself as a private workstation. (The workstation can also be configured to power up as an import station. See Appendix A).

A private workstation can be turned into a work doer for the network withth e IMPORT command. The import command informs the NRM that the private workstation is now capable of doing some type, or types, of jobs. A station remains an import station until the keys CONTROL and C are pressed from its keyboard. If an import station is executing a remote job when the CONTROL and C keys are pressed, it continues executing the job until the job is finished. Only then, does it return to private workstation mode. A Series IV stationthat supports both foreground and background partitions can import into either foreground, background, or both. Thus, a physical station can appear as two import stations to the DJC system.

DJC UTILITIES

Understanding Job Queues

Since the network consists of heterogeneous workstations, some type of mechanism is needed to match the job with the type of workstation it can execute on. This generated the concept of a job queue. A job queue can be envisioned as a waiting place for all work doers and a depository for workgener ators. Job queues are created and deleted using the QUEUE utility, and their statis is monitored using the SYSTAT utility. Each network can have up to 10 queues. The system does not support any predefined queues. While any name may be used, descriptive names based on the work doer's capabilities are recommended. 8-bit.q, 16-bit.q, and print.q are good names, while ONE.queue and compile.queue are not.

The system does not guarantee that a job is sent to a queue capable of doing the job. A Series II or ISIS cluster is only capable of doing 8-bit work. Therefore, the work generator is responsible for ensuring that the work doer chosen can execute the job. Multiple work generators can export to the same queue, and more than one work doer may import from a queue to get jobs done quicker.

Queues are maintained using files at the NRM. The DJC system uses these queue files to maintain job and queue status. These files are shared files and should not be tampered with.

Remote Job Execution

A job is scheduled for remote execution using the EXPORT command. An in depth discussion on the syntax and use of job queues is discussed in Chapter 5 (DJC Utilities). The export command must specify a job queue capable of executing the job. Export checks if the queue exists and displays an exception message if the job is not queued. It also warns the user if there are no importers (work doers) currently serving the chosen queue. The job will wait indefinitely at the job queue until a work doer is assigned to import from this queue.

The exported job may have to wait for some time before it can be executed, since other jobs arrived at the queue earlier might not have been executed. The queue is a first-in-first-out (FIFO) file.

In this way, the DJC system keeps a track of all jobs at the queue and executes them on a FIFO basis.

Example:

```
QUEUE NAME: 16BIT.Q              QUEUE NAME: 8BIT.Q

JOB NAME       STATUS            JOB NAME       STATUS

FOUR.CSD       WAITING           PRINT.CSD      WAITING
THREE.CSD      WAITING           LOCATE.CSD     EXECUTING
TWO.CSD        EXECUTING         LINK.CSD       DONE
ONE.CSD        DONE              COMPILE.CSD    DONE
```

When it successfully finds an importer for the specified job queue, the NRM will send the job over to that station. At the station, an implicit logon takes place using initialization options that are identical to a normal user logon. For example, the station will take the user's INIT.CSD file and execute it first and then execute the job. The environment set up at the import station is exactly as that at normal logon. The import station, a"reincarnation" of the user who exported the job, has access to all of files the user has. It looks just as if a user is inputting information at the import station. The only difference is that, in this case, input is from a file specified by the user exporting the job.

DJC Commands

DJC system on the NDS-II system has a number of commands that help theuser in effectively configuring an efficient remote job execution system. These are:QUEUE, IMPORT, EXPORT, CANCEL, and SYSTAT.

Each of these commands perform unique and important tasks to make the network distributed job control system a very productive and efficient solution.

QUEUE

QUEUE is a command for managing and displaying job queues at the NRM. The QUEUE command displays the name, number of jobs outstanding and the number of servers. After this information is displayed, the user is prompted to:

ADD DELETE LIST EXIT

ADD	option creates new queues. Up to 10 queues can exist at the NRM
DELETE	option deletes a queue
LIST	redisplays previously displayed information
EXIT	terminates QUEUE

For example:

>QUEUE <cr> will bring up the following display

```
NAME OF QUEUE       # OF SERVERS        # OF WAITING JOBS
16BIT.Q             2                   1
8BIT.Q              1                   0
PRINT.Q             1                   2
```

Anyone can delete these queues. Since there is no protection offered, the use of the QUEUE command should be restricted to a SUPERUSER. This may be accomplished simply by removing world access rights on QUEUE.86. However, a queue that has jobs waiting cannot be deleted; in this example, only 8BIT.Q can be deleted.

IMPORT

The syntax for the IMPORT command is the following:

```
IMPORT FROM queue ,queue ....... TO
BACKGROUND
```

where

queue	is a character string up to 14 characters long, which names the queues from which the import station can execute jobs. Up to five queues may be specified in the command line.
TO BACKGROUND	is an option that will execute the job in a background mode. This is a Series IV option only.

The IMPORT command declares the given workstation to be a public resource on the network, converting it from a work generator into a work doer. This public resource can now receive jobs from the various queues in the NRM. If the user enters the name of a queue that does not exist at the NRM, an exception message will be displayed. The queues are searched for jobs according to the order in which they were listed in the command line (left to right). If jobs are available on any queues, the import station starts processing them. The importing station starts by performing an implicit logon for the user, whose job is first at the head of the queue. Then it processes the commands within the command file. At the end of the command file, the importing station logs off the user and looks for jobs from the queues to process (left to right).

For example, the import station is configured to execute jobs from 16bit.q, 8bit.q, and iNDXutility.q. Initially, there is only one job on 8bit.q, so execution of it commences at the import station. During theexecution of this job, three more jobs arrive at 8bit.q and one at 16bit.q. The job on 16bit.q will be the next to execute, since the command line in import mode is always scanned left to right.

All output messages from the remote job, displayed on the screen of the import station, may be put in a log file if the LOG option is specified with the EXPORT command. When a station is in import mode, no local processing is possible. To reconvert the import station back to a private station, the user must enter CONTROL-C by pressing both the CONTROL and C keys.

EXPORT

The syntax for the EXPORT command is the following:

```
EXPORT pathname [parameters] TO queue
[{LOG/NOLOG}]
where
```

pathname	is a valid pathname for a command file
parameters	is a list of up to 10 parameters
queue	is the queue to which the job is to be sent
LOG, NOLOG	specifies whether a log is to be kept of all console activity on a mass storage device.

The EXPORT command allows a command file composed at one workstation to be executed on another workstation. The command file must be on a public volume, so that the import workstation can access it. An example of a public volume is a volume resident at the NRM and not a local mass storage device. If the queue does not exist at the NRM, an exception message is displayed and the job does not get queued. LOG, NOLOG determines whether a log file is to be maintained of all console activity, at the import station during the execution of that particular job.

The optional parameters specified in the command line are actual parameters to be substituted for the formal parameters embedded within the command file. In the example below, %0 will be replaced by the name of the source file specified in the command line. This way, one compile command file can handle programs with different names.

In this example, the command file links, and binds a "C" program.

```
Listing for:COMPILE.CSD
cc86     %0.c      debug
link86   %0.obj,                &
         C/sqmain.obj,      &
         C/sclib.lib,       &
         C/small.lib,       &
         C/87null.lib       &
         to %0.86               &
         bind                   &
         ss(stack(+800h),memory(+1200h))
>EXPORT COMPILE(/C--SOURCE--DIR/ISTIME) TO 16BIT.Q LOG
>EXPORT JOB NUMBER :0027H
```

This will export the job to the specified queue (in this case 16BIT.Q), print an export job number, and return control to the user, so that he or she may continue with productive work. Meanwhile, the import station acting as a server for 16BIT.Q will log on as the user, process his or her initialization file, and process the command file COMPILE.CSD. After all commands inCOMPILE.CS D have been processed, the import station goes back into waiting mode and waits for other jobs to be sent from the NRM.

CANCEL

```
CANCEL [BACKGROUND/REMOTE] queue
 {(job name) (# job number)}
```

where

queue	is the queue where the job has been queued for execution
job name	is the final component name of the remote job to be cancelled (in the previous case, the job name will be COMPILE)
job number	is the assigned value of the remote job (this can be displayed by the SYSTAT command discussed next).

The CANCEL command is used to cancel a background or remote job. If the user wants to abort a remote job, the job name and job number must be entered. If the job name is selected and multiple instances of the job name are in the queue, the first one encountered is deleted (this may not be the first one queued). To avoid this, the unique name job number may be used,

Example:

```
> CANCEL REMOTE 16BIT.Q (COMPILE) will
result in
iNDX-W41 (V2.8) CANCEL VERSION V2.8
""COMPILE'' CANCELLED
```

The job name can be substituted with the job number. In this case, it will be 0027H (see example under EXPORT). Once the job is cancelled, the import station will execute the next job in the queue it is serving. If no jobs exist in the queue, it will go into a waiting mode for the next job.

SYSTAT

The syntax for the SYSTAT command is the following:

```
SYSTAT [{QUEUE/MY JOB } (queuename
[,....])] TO PATHNAME [EXPAND] [ALL]
```

where

queuename(s)	designates the name(s) of the queue(s) for which jobs are to be listed
pathname	designates the file where the information is listed
QUEUE	displays information for all queues, or for only those queues explicitly listed after the queue specifier. If this option is specified, the queuenames must be separated by commas.
MYJOB	parallels the queue option but lists information about jobs belonging onlyEXPAND specifies that complete information is displayed for each job. If expand is not specified, condensed information will be displayed for each job.
EXPAND	specifies that complete information is displayed for each job. If expand is not specified, condensed information will be displayed for each job.
ALL	displays appropriate information for all jobs in the specifiedqueue(s). If ALL is not specified, information is displayed only for waiting or executing jobs.

The SYSTAT command is used to display information about the DJC subsystem to the user. There are many options which are best discussed by examples.

Examples:

```
<SYSTAT <cr>
SYSTAT VERSION V2.8
QUEUE           # OF JOBS # OF IMPORT
NAME            WAITING   STATIONS
16BIT.Q         0         1
8BIT.Q          0         1
iNDXUTILITY.Q   1         0
```

This command displays the status of all queues and information on the number of jobs waiting and number of import stations serving any queue. No detailed information of actual job status is shown here.

```
<SYSTAT QUEUE
SYSTAT VERSION V2.8

JOB STATUS FOR: 16BIT.Q

JOB NAME  JOB #      OWNER     DATE       TIME         STATUS
```

No jobs are waiting or executing in this queue.

```
JOB STATUS FOR: 8BIT.Q

JOB NAME  JOB #      OWNER     DATE       TIME         STATUS
```

No jobs are waiting or executing in this queue.

```
JOB STATUS FOR: iNDXUTILITY.Q

JOB NAME  JOB #      OWNER     DATE       TIME         STATUS

PRINTFILE #0028      JOHN      11/30/84   16:20:22     WAITING
```

This command lists by queue all jobs waiting in a queue. This helps in quickly determining the status of jobs in a queue.

```
<SYSTAT QUEUE ALL
 SYSTAT VERSION V2.8

JOB STATUS FOR: 16BIT.Q

JOB NAME  JOB #      OWNER     DATE       TIME         STATUS

COMPILE   #1003      SRIVAT    11/30/84   12:12:30     DONE
COMPILE   #1002      SRIVAT    11/30/84   12:05:19     DONE
COMPILE   #1001      SRIVAT    11/30/84   11:30:20     DONE

JOB STATUS FOR: 8BIT.Q

JOB NAME  JOB #      OWNER     DATE       TIME         STATUS

COMP      #2008      WAYNE     11/28/84   18:12:30     DONE
COMP      #2007      WAYNE     11/28/84   15:10:20     DONE
LINK      #2006      NORI      11/27/84   10:10:23     DONE

JOB STATUS FOR: iNDXUTILITY.Q

JOB NAME  JOB #      OWNER     DATE       TIME         STATUS

PRINT     #2002      JOHN      11/30/84   18:10:20     WAITING
PRINT     #2001      SRIVAT    11/29/84   12:10:22     DONE
PRINT     #2000      WAYNE     11/29/84   10:10:10     DONE
```

This command lists the status of all jobs done or waiting in the queue since the queue was created. This is useful to the system administrator to study queue use.

This command lists the status of all of the jobs that users have submitted. Queue files are circular files 256 jobs long. For example, SYSTAT will display the last 255 jobs done or waiting. If the number of jobs exceeds 256, the first entries (jobs) into the queue file are deleted to make room for the new entries. The expand option, which displays all these jobs, is useful for system administration purposes. Information containing average wait time for each job, the average length of a job, may be obtained. The system administrator may use this information to install another work doer on a particular job queue, thereby optimizing the system for his or her particular environment. This queue can be deleted and then recreated once this information is recorded to clear this log of queue activity.

RECOMMENDATIONS FOR AN EFFICIENT DJC SYSTEM

The following discussion outlines recommendations for a useful DJC system for a network. A number of considerations should be made before your DJC system is implemented on the network.

A minimum of three queues should exist at the NRM:one queue for 8-bit work, one for 16-bit work, and the other an indxutility queue. Normally, one server is enough to serve these queues. However, if the load on any particular application increases, having a dedicated server for that queue will be more efficient.

In the example following, it is assumed that the high 16-bit workload requires a dedicated server for the 16-bit work being done on the network. Therefore, a dedicated server for 16BIT.Q has been generated using the IMPORT command. The other server imports from all three queues. Private workstations can also be converted into import stations whenever they are not being used. The background of one of the private workstations should come up in automatic import mode on powerup. This is discussed in Appendix D.

APPENDIX A
Looping in Export Files

Often, a job needs to be run continuously to do a predetermined task like checking mail. The versatility of the DJC system allows the user to do this in just one submit file. For example, an import station can export a job to itself or any other server on the network.

Example:

```
Mail Box(%0)
Save 1 msg.file
EXIT
checkexist msg.file
if %status = 0 then
     report YOU HAVE MAIL IN BOX %0
end
export mailcheck (%0) to indxutility.q
nolog
end
```

This is an example of an export file that constantly checks for mail in a user's box. If a mail message exists, a message is sent to the user. Checkexist is a program that looks for a specified file and sees the value of %status to 1 if the file exists and 0 otherwise. Report is a utility that sends a message to the user's console. These utilities are explained in depth in the Application Note AP-245:"Using Command Files to speed program development."

The submit file is exported using the command:

```
EXPORT MAILCHECK(SRIVAT) TO
iNDXUTILITY.Q
```

The import station will execute this command file and later reexport the job back to the queue. This job will be put at the end of the job queue behind all others waiting at this queue. It will not totally dominate the job queue. The only way to stop MAILCHECK once it is running is to use the CANCEL command. There is no limit to the number of times an export job can be looped.

Conditional exports can also be done from within an exported job. The IF, THEN, ELSE constructs of command files are used. The above example is just one of the different ways DJC can be used. This feature is very useful if some remote job has to be done continuously.

APPENDIX B REPORT.86

Since all exported jobs are remotely executed, the only method of monitoring their status is by using the SYSTAT utility. The need for a more interactive status reporter becomes more pronounced. REPORT.86 has been designed to answer this need. REPORT is a utility that should be included in all export files. The syntax for REPORT is the following:

```
REPORT <any message>
```

The following command file example shows how REPORT is used:

```
cc86      %0.c debug                       ; Compile the program
if %status <> 0 than                       ; If error in compile
     REPORT Error in compile of %0.c       ; Send message to user
else                                       ; and exit.
     REPORT Successful compile. Proceeding with LINK
          link86 %0.obj, &
          l/sqmain.obj, &
          l/sclib.lib, &
          l/small.lib, &
          l/87null.lib &
          to %0.86 &
          bind &
          ss(stack(+800h),memory(+2800h))
          if %status = 0 then                           ; Check for error in link
               REPORT Successful Link. End of Job.      ; If no error inform user
          else
               REPORT Error while linking.....          ; If error inform user and
          end                                           ; and exit.
```

REPORT.86 writes the message specified into the user's home directory in a file called REPORT.DAT. All the messages get appended on to this file. The ISIS and iNDX command line interpreters (CLI) have been extended to check for the existence of the file REPORT.DAT in the user's home directory. If the file exists, the contents of the file are displayed on the user's screen. The CLI then deletes this file. This gives the user the ability to constantly monitor the execution of a remote job. In the above example, if there was an error in compilation of the program, REPORT will write the message "Error in Compile of filesheck.c" and the remote job will terminate. This message will then come up on the user's terminal anywhere on the network, and the user can take corrective action. All messages are held until the user returns to the command level. They are not displayed instantaneously in the middle of an AEDIT session, for example.

The REPORT function used throughout the submit file will keep the user constantly informed on the success of all required operations. This results in greater productivity, since the user does not have to wait until the whole submit file is over and then examine the log file. The extensive use of the variable %STATUS in this submit file requires explanation. All Intel utilities, such as PL/M86, C86, and LINK86, exit with a UDI call DQ$EXIT(0) if the operation is successful and DQ$EXIT(n) if the operation was not successful (N is any number). This value passed into the DQ$EXIT call is stored in a variable called STATUS. This variable can be accessed from any submit file. Conditional operations can be done by accessing this variable.

APPENDIX C CHECKTIME.C

Often, a program must be executed at a particular time. CHECKTIME.86 is a utility that allows a program to be executed at a particular time from within a submit file. The concept of STATUS and looping in submit files are used here again. This program obtains from the user a particular time, which can be set to be less or greater than system time. When the defined condition is satisfied, the program will exit with a return code of 1. Otherwise, it will exit with a return code of 0. For example, a match condition will exit with DQ$EXIT(0). This return code is passed on to the %STATUS variable that can be accessed by a submit file.

This program has been designed for doing jobs at a particular time of day in an export file.

The syntax for CHECKTIME.86 is the following:

```
    CHECKTIME greater   22:23:45   or
    CHECKTIME greater   22:23      or
    CHECKTIME greater   22         or
    CHECKTIME g         22:23:45   or
    CHECKTIME g         22:23      or
    CHECKTIME g         22
```

This will return with a return code of 1 if the system time is greater than the time specified and 0 for all other cases.

```
    CHECKTIME less      22:23:45
    CHECKTIME less      22:23
    CHECKTIME less      22
    CHECKTIME l         22:23:45
    CHECKTIME l         22:23
    CHECKTIME l         22
```

This will return with a return code of 1 if the system time is less than the time specified and 0 for all other cases.

For example, backup needs to be done only at a particular time, preferably during the night, when the system load is lighter. This can be done in an export file using the CHECKTIME.86 utility.

```
File: BACKUP.CSD

CHECKTIME g 22:59:00
if %STATUS = 1 then
     TREE BACKUP /APS--WO/USER.DIR/
SRIVAT.DIR/* to /APS1/SRIVAT.DIR
else
EXPORT BACKUP to iNDXUTILITY.Q
end
```

In the above submit file, CHECKTIME compares the given time with the system time. If a match is found, it will exit with STATUS set to 1; otherwise it will exit with STATUS set to 0. If STATUS is set to 0, a match has not been found and the submit file will export itself to the queue. In this way, the jobs get stacked up on the queue. When the CHECKTIME condition does get satisfied, the export file will back up all files in the volume

APS—WO/USER.DIR/SRIVAT.DIR to the /APS1/SRIVAT.DIR.

APPENDIX D Configuring a Station to Come Up as an IMPORT Station

A workstation (Series IV) can be configured to power up as an import station through the SYSGEN command at the NRM. SYSGEN, restricted only to the SUPERUSER, will not allow any other user to modify the system configuration. Invoke SYSGEN by typing:
`SYSGEN`

SYSGEN will then clear the screen and display all the workstations on the network and their Ethernet addresses. Select the soft key labelled "Options". Next, select the node that has to come up as an import station. SYSGEN will then display another screen with one of the options being:

(7) Automatic Import to Partition 1 Partition 2

Select the partition needed to to come up in import mode. Both partitions can be selected. SYSGEN will next ask for queue names that will serve that import station. List the queues (maximum of 10) and then exit from SYSGEN. Reset the network and the Series IV will come up as an import station on powerup. To terminate import mode, do a Control-C at the import station keyboard by pressing the Control and C keys simultaneously.