GH20-1291-1

Systems

IBM System/360 and System/370 ASP Version 3 Asymmetric Multiprocessing System Application Programmer's Manual

Program Number 360A-CX-15X

The ASP system is a multiprocessing operating system that provides a compatible extension to the Operating System (OS). Designed for the user with a large computer job-shop environment, ASP provides increased automation of the computing operation. The ASP system functions as a programmed operator of OS. It provides advanced scheduling facilities for optimizing total installation production.

This manual contains information concerning system concepts, programming considerations, control cards, deck setup, and output.

IBM

PREFACE

This manual contains information concerning system concepts, programming considerations, control cards, deck setup, and output. OS is the primary operating system used for ASP; therefore, to avoid repetition of documentation, the material in this manual has been prepared on the assumption that the programmer is familiar with OS programming and has previously read the ASP Version 3 General Information Manual GH20-1173. Details of OS utilization are found in IBM Operating System/360 Job Control Language Reference (GC28-6704), and in its prerequisite and recommended publications. Details of OS Time Sharing Option are found in the IBM System/360 Time Sharing Option Command and Language Reference (GC28-6732).

Other publications currently available for ASP Version 3 are:

- ASP Version 3, General Information Manual GH20-1173
- ASP Version 3, Messages and Codes Manual GH20-1290
- ASP Version 3, Operator's Manual GH20-1289
- ASP Version 3, System Programmer's Manual GH20-1292
- ASP Version 3, Reference Card GX20-1927
- ASP Version 3, Logic Manual GH20-1403

Second Edition (September 1973)

This edition, GH20-1291-1, is a major revision obsoleting GH20-1291-0. It applies to Version 3, Modification Level 1, of the program product ASP, Asymmetric Multiprocessing System (360A-CX-15X) and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

Changes are continually made to the information herein. Therefore, before using this publication, consult your System/360 and /370 Bibliography (GA22-6822) for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments has been provided at the back of this publication. If this form has been removed, address comments to: IBM Corporation, Department J74, 1930 Century Park West, Los Angeles, California 90067. Comments become the property of IBM.

CONTENTS

FIGURES

CHAPTER 1. INTRODUCTION

The ASP system is a multiprocessing operating system that provides a compatible extension to the Operating System (OS). Designed for the user with a large computer job-shop environment, ASP provides increased automation of the computing operation. The ASP system functions as a programmed operator to OS. It provides advanced scheduling facilities for optimizing the total installation production.

To the application programmer ASP can be transparent, or he may choose to utilize the many features provided by ASP. The programmer needs only to supply his job with the standard OS JCL, and ASP will handle the scheduling of the job. The scheduling algorithms can be tailored to an installation's system configuration by the system programming staff. The application programmer should confer with this staff to determine how his job's performance can be enhanced.

Features of ASP allow the programmer to select a specific Main Processor for execution; cause pre-execution setup; control the processing of his output; select installation-written support functions; and use ASP Utilities. To take advantage of these and other features, the programmer may submit ASP control cards with his normal JCL. These cards and examples are discussed in Chapter 3.

Jobs submitted to ASP are in two basic categories: standard and nonstandard. Standard jobs are controlled by the functions provided by ASP; Reader/Interpreter, Main Processing, Print, Punch, and Purge. These functions will be covered later. A nonstandard job allows the programmer to specify a different sequence of processing such as print only for listing a deck of cards.

ASP performs its functions with Dynamic Support Programs (DSPs). These are programs which reside on direct access storage and can be brought into main storage as they are needed.

A standard job is read into the support system via an ASP reader and placed in the ASP queue. When selected from the ASP queue the first function is to convert the JCL for the job into internal OS control blocks, the format required by OS initiators. Selection of the Main Processor and the setup and verification of the required volumes are completed before sending the job to the Main Processor for execution. SYSOUT information from the job during the time it is active on the Main Processor is returned to ASP for later printing and/or punching by ASP. When the functions are all complete the job is purged from the system. This is the basic job flow which the programmer can expand to take advantage of other features.

ASP supports remote job submission from Binary Synchronous Communications terminals via Remote Job Processing (RJP).

ASP also provides, via Network Job Processing (NJP) under operator or ASP control card control, the ability to transmit jobs between two or more ASP installations remote from one another.

A TSO terminal user may submit jobs, and have ASP schedule them on any Main Processor in the system and have selected output returned to his terminal or to a destination of his choice.

The execution of a job may be dependent upon the completion of one or more other jobs. Dependent Job Control (DJC) allows the user to create a network of jobs with dependencies.

The ASP internal reader, whose use is referred to as Internal Job Processing (IJP), is provided as an additional aid to application programs, such as an Information Management System (IMS) application running for a considerable period of time in a multiprogramming environment. It is the design of OS/MVT that until a job terminates, the output data is not available for a writer. Therefore, no SYSOUT output from this IMS job can be seen until the IMS job terminates. With the ASP internal reader, it is possible to get intermediate output printed without having to terminate the non-ending job.

The preceding features and the ASP control cards available to the application programmer are discussed in this manual. The user should consult his installation system programming staff for additional guidelines in using ASP facilities. General information about ASP can be found in the ASP Version 3 General Information Manual. The General Information Manual should be read before reading this manual.

CHAPTER 2. SYSTEM FLOW

JOB PROCESSING

A fundamental concept of ASP is that many jobs are handled simultaneously by the system. To facilitate control over the many jobs being processed concurrently, each job is divided into three phases: preprocessing, processing, and postprocessing. The system is controlled by two supervisory programs: OS with ASP in the Support Processor, and OS in the Main Processors. The three phases of processing are illustrated in Figure 1.

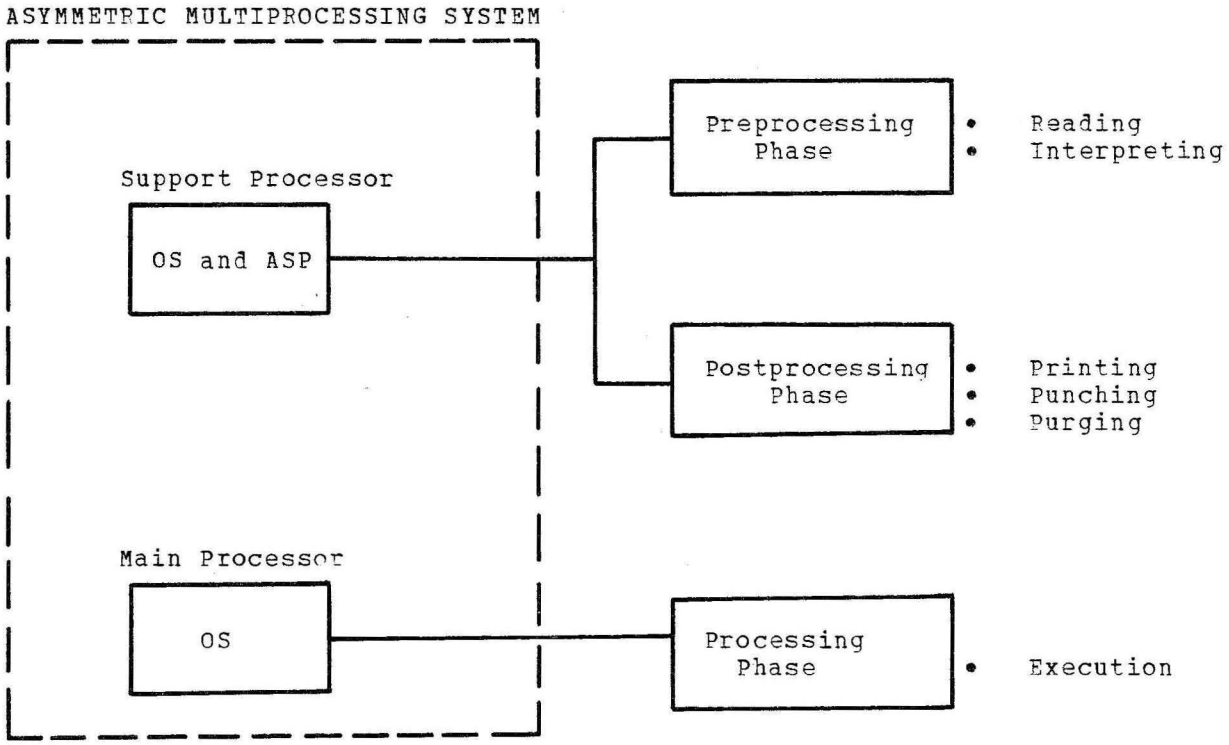


Figure 1. Phases of Processing.

OS resides in the Main Processor and controls the processing phase. OS and ASP reside in the Support Processor and control the preprocessing and postprocessing phases. Each system performs its functions asynchronously. The overlap of job processing with the preprocessing of other jobs significantly reduces turnaround time.

JOB FLOW

A job in the ASP system, as identified by the OS JOB card, is a unit of work consisting of one or more job segments also known as Scheduler Elements (SEs). For the standard jobs Job segments are automatically defined by the ASP system.

1. Input Service - reads the input stream on the Support Processor, recognizes control cards, and takes the appropriate action.

2. Reader/Interpreter Service - interprets OS job control language to determine system resources required by a job before the job is sent to a Main Processor.

3. Main Service - handles operator communication of jobs in execution, manages the flow of data (for example, system input, system print and punch data sets) across the CTC to and from the Main Processor.

4. Print Service - prints the data sets.

5. Punch Service - punches the data sets.

6. Purge - removes each job from the system, returning to the system all direct access storage space allocated to that job.

For a nonstandard ASP job the //*PROCESS cards specify the job segments to be processed. All ASP control cards are extracted from the input stream by Input Service and are transparent to the Main Processors.

STANDARD JOB FLOW

A standard job normally requires input/output units on the Main Processors to be allocated and special volumes (either disk or tape) to be mounted before execution. These jobs are specified by an OS JOB card, followed by a normal OS JCL deck. A standard job is composed of five scheduler elements.

1. Reader/Interpreter. Interprets OS JCL.

2. Main Service. Transmits the system data sets via the Channel-to-Channel Adapter to and from the Main Processor.

3. Print Service. Prints the output print data sets.

4. Punch Service. Punches the output punch data sets.

5. Purge. Purges the job from the system.

The job flow of a standard job is shown in Figure 2. This figure, read sequentially from top to bottom, illustrates the job flow of a single job.

The steps in the flow of a standard job are as follows:

1. The operator communicates with the ASP system via the console. To initiate the reading of the input stream, the operator enters:

   ```
   *X,CR
   ```

2. Console Service accepts the message and routes it to the ASP Supervisor.

3. The ASP Supervisor interprets the message and initiates the Input Service program.

4. Input Service reads the input stream and enters the job into the system with the necessary routing information. This routing consists of the pre- and postprocessing steps.

   Note: Job submission from remote workstations uses the standard ASP reader. The reader does not distinguish between local and remote submission. The same applies to output

to remote workstations. Print Service and Punch Service make only minimal distinction between local and remote devices.

5. When the job is selected from the ASP queue, the Reader/Interpreter segment will get control. The OS JCL submitted is translated into OS control blocks and placed back into the ASP queue.

6. Once the OS control blocks have been created the job is scheduled for processing by Main Service. As units become available for mounting on the Main Processor, the Main Device Scheduler (MDS) program assigns the units to jobs in priority sequence. As soon as all of the unit allocation and setup requirements for a job have been satisfied, the job is released for scheduling for the Main Processor. Main Service sends the OS control blocks to the Main Processor. The Main Processor executes the job and then sends the output data sets to the Support Processor over the CTC. The Support Processor places the data sets on the ASP queue devices for output processing. In addition, Main Service logs time-on and time-off for the job to indicate the total wall clock time spent on this job by the Main Processor. After execution on the Main Processor MDS can perform its breakdown phase. At this time, the devices that were assigned to the job are returned to the system for subsequent jobs.

7. After processing is completed on the Main Processor, the job is placed in the queue for the Print and Punch Service programs. When all scheduling criteria for printing are satisfied for this job segment, the job is sent to Print Service.Print Service is not a sequentially dependent process; that is, it can be processed in parallel with other job segments of the same nature, such as Punch Service, for the same job. When all scheduling criteria for punching are satisfied for this job segment, the job is sent to Punch Service.

8. Print Service reads the SYSMSG and output print data sets from direct access storage devices and prints them. Print Service logs time-on and time-off to indicate the total time spent by this job on the printer.

9. Punch Service reads the output punch data sets from direct access storage devices and punches cards for each data set until it either exhausts the data set or encounters a double end-of-file. Punch Service also logs time-on and time-off to indicate the total time spent by this job on the punch.

10. Once processing is complete for both Print Service and Punch Service, the job is sent to Purge SE. Purge is a sequentially dependent process and is always the last function to be performed.

11. Purge returns all tracks on direct access storage devices that were allocated for this job. At this time, an optional accounting routine may be entered to process the accumulated job accounting information.

12. Finally, with all processing segments complete for the job, the job is removed from the ASP Job Control Table, which completes its flow through the system.

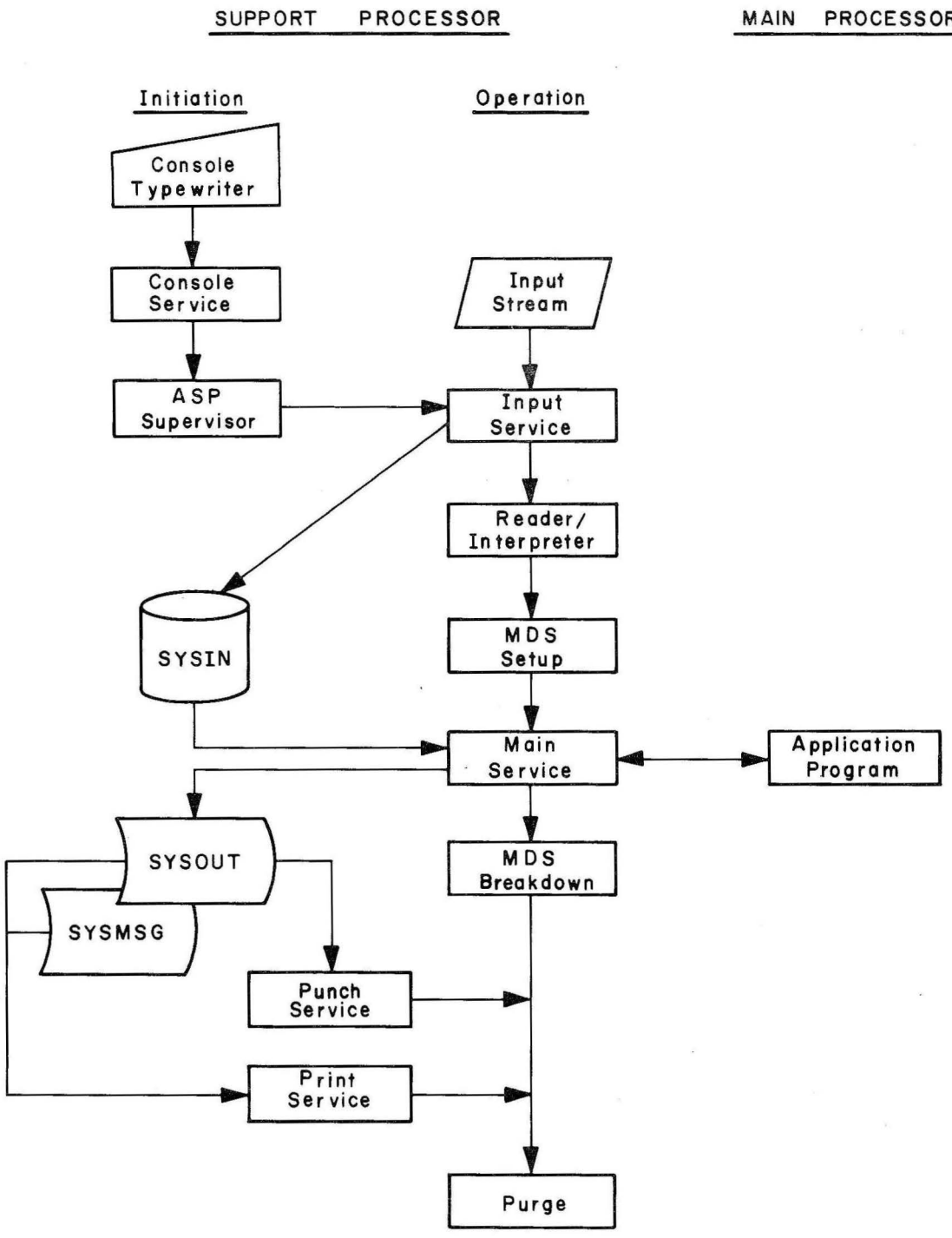


Figure 2. Standard Job Flow.

NONSTANDARD JOB FLOW

A nonstandard job is a combination of job segments that deviates from the standard combination of job segments (that is, Reader/Interpreter, Main Service, Print Service, Punch Service, and Purge). Some examples of nonstandard job segment combinations are:

- A preprocessing job segment plus the standard job segments

- Fewer than the five standard job segments (for example, Reader/Interpreter, Main Service, Print Service, and Purge, without Punch Service)
- One or more postprocessing job segments plus the standard job segments
- One or more job segments, of which only Purge is a standard job segment
- Overlap printing of a single data set at many locations simultaneously.

A nonstandard job is specified to the system by a JOB card, followed by a //*PROCESS card for each job segment, followed by the normal OS deck. In a nonstandard job, there must be a //*PROCESS card for each job segment except Purge.

The steps in the job flow of a nonstandard job are similar to those for a standard job (see Figure 3):

1-3. These steps are identical to the first three steps for a standard job.

4. This step differs from Step 4 for the standard job only in the sequence in which support functions are scheduled. The sequence of the //*PROCESS cards determines the order in which job segments are executed. Each //*PROCESS card represents one job segment. Purge and the Main Device Scheduler (MDS) are not included on //*PROCESS cards. Purge is implied for all jobs in the system, and the Main Device Scheduler is automatically scheduled for all jobs requiring setup. All other job segments in a nonstandard job must be specified on a //*PROCESS card.

5. Scheduling for the job takes place in the sequence specified by the //*PROCESS cards. Scheduling continues until the last job segment for the job (Purge) is complete.

6. When all segments are complete, the job is removed from the ASP Job Control Table. At this time, the job has completed its flow through the system.

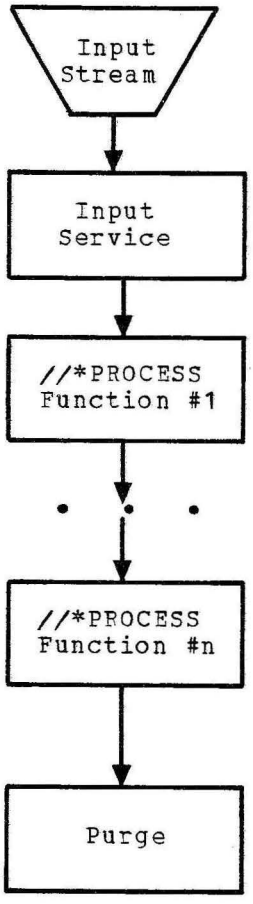


Figure 3. Nonstandard Job Flow.

Examples of standard and nonstandard jobs are given in Chapter 3, ASP Control Cards.

CHAPTER 3. CONTROL CARDS

The ASP system offers several control cards that permit the programmer to define job processing. These control cards are submitted with the JCL for the job and are transparent to the Operating System (OS). Flexibility can be gained by use of the control cards, or the installation standards can be used for determining job processing. Control cards provide:

- Special instructions for forms, trains, and carriage tape control on the printer or forms control on the punch
- Specification of volumes that are to be mounted prior to execution
- Special functions that are required for job processing
- Specification of Main Processor dependencies

Errors in ASP control cards are noted in the SYSMSG data set; the job is canceled, and the SYSMSG data set is printed.

When ASP is started and gains control it goes through an initialization process. The initialization of ASP defines the system configuration and processing options. The initialization process is controlled by OS JCL and ASP initialization control cards.

The application programmer should confer with the system programming staff when completing ASP control cards for his job. Some of the control card parameters will require knowledge of the standards established within the installation.

The following chart summarizes many of the functions available to the application programmer from ASP. For more detail, see appropriate control card explanation in this chapter.

Feature	ASP Control Card	Parameter
Print multiple copies	FORMAT	COPIES=n
Punch multiple copies	FORMAT	COPIES=n
Suppress printing of DATASET	FORMAT	COPIES=0
Suppress punching of DATASET	FORMAT	COPIES=0
Submit data mode 2 data	DATASET	MODE=C
Specify single-space print	FORMAT	CONTROL=SINGLE
Route a data set to a remote location	FORMAT	DEST=location
Specify special forms	FORMAT	FORMS=name
Specify printer overflow off	FORMAT	OVFL=OFF
Specify special print train	FORMAT	TRAIN=name

Specify special carriage tape or FCB load	FORMAT	CARRIAGE=name
Route TSO output to TSO terminal	FORMAT	USER=userid
Select a CPU for execution	MAIN	SYSTEM=name
Specify a job class of 2 or more characters	MAIN	CLASS=name
Specify job execution limits	MAIN	CARDS=/LINES=
What to do if CPU fails	MAIN	FAILURE=options
Submit a job from another location	MAIN	ORG=loc-name
Specify DEADLINE for execution	MAIN	DEADLINE=list
Build a job network	NET	See Chapter 4
Write a message to the operator	OPERATOR	text
List a deck of cards	PROCESS/FORMAT	PRINT with FORMAT
Copy a deck of cards	PROCESS/FORMAT	PUNCH with FORMAT

CONTROL CARD SYMBOLOGY

All ASP control cards are identified by //* in card columns 1 through 3 followed by a keyword. In those cases in which the parameter list associated with an ASP control card may exceed the capacity of a single card (such as //*FORMAT and //*MAIN), the list may be extended to a continuation card by punching a nonblank character into column 72 of the first card, punching //* in columns 1, 2, and 3 of the continuation card, and resuming the text in column 4 of the continuation card. A keyword and its parameters must not be split between two cards; that is, the initial text on the continuation card must be a keyword. Comments should not be punched in ASP control cards.

The symbols listed below are used in the command formats to show the user how and when to write certain operands.

	Vertical stroke. Means "or". For example, A|B means either the character A or the character B. The user does not write this symbol.
/	Slash. Means "not". For example, /(SY1,SY2) means do not schedule SY1 or SY2.
{ }	Braces. Indicate that one of the enclosed alternative items must be written by the user. For example: MODE={E|C} means that either E or C must be written by the user. The user does not write this symbol.
[ ]	Brackets. Indicate that the item, items, or groups of items within the brackets are optional and may be omitted at the user's discretion. For example:

[FILES=nn]

means that the entire operand may be omitted if the user does not wish to designate the number of files. The user does not write this symbol.

[{ }] Braces within brackets. Indicate that, although the entire item is optional, if the user elects to make an entry, he must choose one of the values enclosed within the braces. For example:

[DEN={800|556|200}]

means that the user may either omit this operand entirely or he may choose to use it, in which case he must select one of the specified values. The underline means that if the user chooses to omit the item, the program will automatically select the underlined value.

, Comma. Used to separate operands. A comma must be written in place of any omitted optional positional operand unless no other positional operand follows. A comma must precede a symbolic operand only when another operand appears before the symbolic operand.

( ) Parentheses. Used to mark a group of similar items such as a series of suboperands, and must be written by the user exactly as shown. In most cases, suboperands are positional. Parentheses must be used even if there is only one item in the group.

ASP CONTROL CARDS

DATASET

```
//*DATASET  DDNAME=ddname[,J={NO|YES}][,MODE={E|C}]
```

Provision is made in ASP to permit additional input data sets from the input stream. This dynamically created data set can be referenced by the DDNAME in the jobs JCL anytime during the life of the job, or a //*FORMAT card can reference this data set. The //*DATASET card defines the beginning of an additional data set that can contain OS JCL and/or data. A data set should be terminated by a //*ENDDATASET card.

DDNAME=ddname

Defines the ddname to be used to reference the data set.

J={NO|YES}

Determines how the data set is to be terminated. NO indicates that reading an OS JOB card will terminate the data set. YES specifies that OS JOB cards can be included in the data set and a //*ENDDATASET card must be provided to terminate the data set.

MODE={E|C}

Defines the card reading mode. E specifies that the cards are to be read as EBCDIC with validity checking. C specifies that the cards are to be read in card image form (column binary or data mode 2).

Note 1: MODE=C is invalid for jobs to be read by the Disk Reader (DR) and Tape Reader (TR) DSPs, and for jobs submitted via RJP.

Note 2: J= parameter is ignored when MODE=C is specified. The data set must be terminated with a //*ENDDATASET card.

Note 3: If a jobstream contains a //*DATASET MODE=C, the operator must include the C operand in the CALL for the Card Reader (CR) DSP, that is, *X CR,C.

Note 4: When a data set specified in a //*DATASET card is to be used as input to OS, the DD card referencing that data set must include the following parameters:

```
//ddname    DD  UNIT=(CTC,,DEFER),
//              VOL=SER=dummy-name,
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
```

However, if MODE=C was specified, then LRECL=160 and BLKSIZE=160.

ddname - as specified in the DDNAME= parameter.
dummy-name - a dummy serial number. A unique name must be formulated for each data set in the job; the name chosen also must not duplicate that used for a real volume (tape on disk).

Example 1:

```
//DD5    DD  SYSOUT=A
//DD6    DD  UNIT=(CTC,,DEFER),VOL=SER=DUMMY,
//           DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//*DATASET  DDNAME=DD6
123      679 ABC
124      679 INPUT
           etc.
954      321 NOW
//START JOB MSGLEVEL=(1,1)
//STEP  EXEC
```

In Example 1 a data set will be created that can be referenced by DD name DD6. The data set will contain all of the information between the //*DATASET card and the OS JOB card.

Example 2:

```
//DD5    DD  SYSOUT=A
//*DATASET  DDNAME=APPLES,J=YES
//J2     JOB MSGCLASS=9
//STEP   EXEC    PGM=WORK
//DS     DD  UNIT=2314,VOL=SER=123456,DISP=NEW
           etc.
//*ENDDATASET
//*DATASET  DDNAME=PEAR
           data
//EXAM2      JOB
```

Example 2 shows creating a data set, DDNAME=APPLES, that contains the JCL between the //*DATASET and //*ENDDATASET. This example also includes a second data set that is to be referenced by DDNAME=PEAR.

Example 3:

```
//DD5    DD  SYSOUT=A
//*DATASET  DDNAME=GOOD,MODE=C
        Column Binary (Data Mode 2 format) cards.
//J3     JOB
//STEP   EXEC
//DEXM   DD  *
        Column Binary (Data Mode 2 format) cards.
//*ENDDATASET
//J41    JOB
```

In Example 3, all of the cards between the //*DATASET and //*ENDDATASET cards will be read in card image mode.

Example 4:

```
//DD5    DD  SYSOUT=A
//SYSIN DD   DATA
//*DATASET  DDNAME=SOSO,J=YES
//J2     JOB
//STEP   EXEC
          etc.
//*ENDDATASET
/*
//J3     JOB
```

In Example 4 the cards following the SYSIN DD card, down to the /* card, will be included as SYSIN data to the program and no additional data sets will be created because of the //*DATASET card.

Example 5:

```
//DD5    DD  SYSOUT=A
//*DATASET  DDNAME=NONO
          data cards
//SYSIN DD   DATA
//A      JOB
//S      EXEC
//D1     DD
//*ENDDATASET
```

Example 5 will produce unpredictable results. The data set, defined by DDNAME=NONO will contain the following data cards plus the //SYSIN DD card. Jobname A will be a separate job and the //*ENDDATASET card will be ignored.

ENDDATASET

The //*ENDDATASET card terminates the creation of an input data set. It must appear immediately after the last card for that data set. Refer to the //*DATASET card for examples of usage.

```
//*ENDDATASET
```

FORMAT

The //*FORMAT card communicates special destination and format related instructions to the system for processing the print and punch data sets. //*FORMAT cards are required only for nonstandard data set processing such as routing, multiple copies, etc. The system default data set handling is specified to ASP on the SYSOUT initialization control card. The application programmer should consult his system programming staff for installation standards that could dictate use of a //*FORMAT card.

Multiple //*FORMAT cards may be used for any data set to specify different special requirements for each copy of the data set.

```
//*FORMAT {PR|PU|NJP|AC},text
```

The keyword PU signifies that this //*FORMAT card is associated with a punch data set. PR indicates that this card is associated with a print data set. NJP indicates that this card is associated with and directly follows a //*PROCESS NJPIO card. AC is used in conjunction with TSO to describe data sets destined for TSO terminal users.

//*FORMAT and PU, PR, NJP, or AC are separated by one or more blank columns. The text portion of the card is different for each type of data set and is explained below.

If //*PROCESS cards are used, the //*FORMAT card or cards must be placed immediately following the associated //*PROCESS card.

Note: //*FORMAT cards referencing the same ddname should be placed in descending order; that is, the least qualified ddname last.

Print Text (PR)

The text for the //*FORMAT card for Print consists of several keyword parameters. The keywords and their subparameters are:

```
//*FORMAT   PR,DDNAME=[ddname]
             [,DEST={ANYLOCAL|device-name|group-name}]
             [,CONTROL={PROGRAM|SINGLE|DOUBLE}]
             [,COPIES={1|nn}]
             [,FORMS={STANDARD|form-name}]
             [,CARRIAGE={STANDARD|carriage-tape-name}]
             [,TRAIN={STANDARD|AN|GN|HN|PN|QN|QNC|RN|SN|TN|XN|YN|PCSAN|
                      PCSHN}]
             [,OVFL={OFF|ON}]
```

DDNAME=ddname

Specifies the ddname of the data set to which this card applies. This ddname should be qualified to the level required (procname.stepname.ddname). When the DDNAME= is given and no ddname follows the = sign, the parameters specified become the defaults for the job and apply to all print data sets that have no //*FORMAT cards.

DEST={ANYLOCAL|device-name|group-name}

Specifies the printer to be used for output. If the parameter is omitted, the first available printer in the origin group (the group of printers defined for the local or remote locations) that fulfill all processing requirements will be assigned. If the job originated at a remote RJP terminal, the output will be returned to the originating terminal group. If ANYLOCAL is specified the data will

be processed on any local printer that fulfills all processing requirements.

CONTROL={PROGRAM|SINGLE|DOUBLE}

Specifies the type of forms control to be used. PROGRAM indicates that the carriage control character is the first character of each logical record in the data set, if the problem program DCB (or DCB parameter on a DD override) specifies either "A" or "M" carriage control; otherwise, no carriage control is assumed. This character may be either the extended USASI code supported by OS, which includes those codes defined by USASI FORTRAN (preferred), or the actual channel command code for the System/360 and System/370 channel. SINGLE indicates forced single spacing. DOUBLE indicates forced double spacing.

COPIES={1|nn}

Specifies the number of original copies to be printed for this data set (maximum is 99). If zero is specified, printing for this data set is bypassed.

FORMS={STANDARD|form-name}

Specifies the printer forms to be used. STANDARD indicates that standard installation forms, as specified in the ASP initialization STANDARDS card, are to be used. Otherwise, the form name or the form number of special forms that are to be mounted is given. This field has a maximum length of eight characters. There is no checking of character restrictions.

CARRIAGE={STANDARD|carriage-tape-name|fcb-load-name}

Specifies the carriage tape or Forms Control Buffer (FCB) load to be used. If the parameter is omitted or is specified as STANDARD, the installation standard carriage tape or fcb is used. This field may have a maximum of eight alphameric characters.

For 3211 type printers, a module must be included in the ASP program for each fcb-load name. The module name is 'FCB2' plus four characters of the user's choice.

TRAIN={STANDARD|AN|GN|HN|PN|QN|QNC|RN|SN|TN|XN|YN|PCSAN|PCSHN}

Specifies the printer train to be used. If the parameter is omitted or is specified as STANDARD, the train specified in the ASP initialization STANDARDS card is to be used. Since these are not standard machine features, the programmer should verify that the installation has the required printer train before he specifies one of these parameters.

Note: The TRAIN= parameter should not be used for output destined for a remote RJP terminal.

OVFL={OFF|ON}

Specifies whether or not the printer program should test for forms overflow. ON specifies that the printer program should eject whenever the end-of-forms indicator (channel 12) is sensed. OFF specifies that forms overflow control is not to be used.

Note: For RJP the overflow test is a responsibility of the terminal package for the remote RJP terminal.

Examples of //*FORMAT cards for Print are:

```
//*FORMAT PR,DDNAME=STEP1.SYSPRINT,COPIES=2
```

THIS PAGE INTENTIONALLY LEFT BLANK

The above card specifies that two original copies are required for the STEP1.SYSPRINT data set, and that any printer that has standard forms, train, and carriage tape mounted may be used.

```
//*FORMAT PR,DDNAME=STEP1.SYSPRINT,FORMS=2-SECRET
```

The above card specifies (by local convention) that two-part paper with preprinted SECRET classification is required for the STEP1.SYSPRINT data set.

```
//*FORMAT PR,DDNAME=REPORT2,DEST=PR1,CONTROL=SINGLE,COPIES=3
```

The above card specifies that the data set REPORT2 should be printed on printer PR1 under single-space control and that three original copies are required.

```
//*FORMAT PR,DDNAME=,DEST=ANYLOCAL
```

The above card specifies that all data sets without FORMAT cards are to be printed on any local printer.

Punch Text (PU)

The text for the //*FORMAT card for Punch Service consists of keyword parameters. The supported keywords and their definitions are:

```
//*FORMAT  PU,DDNAME=ddname
            [,DEST={ANYLOCAL|device-name|group-name}]
            [,COPIES={1|nn}]
            [,FORMS={STANDARD|form-name}]
            [,INT={YES|NO}]
```

DDNAME=ddname

Specifies the qualified ddname of the data set to which this card applies. If the DDNAME= is given without the ddname, the parameters specified apply to all punch data sets without FORMAT cards.

DEST={ANYLOCAL|device-name|group-name}

Specifies the punch unit to be used for output. If the parameter is omitted, the first available punch unit in the JOB origin group will be assigned. If the parameter is omitted, and the job originated from a remote RJP terminal, the output is returned to the originating terminal group. If ANYLOCAL is specified the output is processed locally.

COPIES={1|nn}

Specifies the number of copies to be punched for this data set (maximum is 99). If zero is specified, punching for this data set is bypassed.

INT={YES|NO}

Specifies whether or not the output is to be interpreted. Specification of YES will cause an attempt to obtain a device type PUN3525I. If DEST does not include a 3525I, INT=NO is substituted.

FORMS={STANDARD|form-name}

Specifies the card forms to be used. This field may have a maximum of eight characters. Omission of this field or the use of STANDARD indicates that standard installation card forms are to be used.

An example of a Punch //*FORMAT card is:

```
//*FORMAT PU,DDNAME=PUNCHOUT,DEST=PU1,FORMS=RED-STRP
```

The above card causes Punch Service to punch one copy of the data set named PUNCHOUT on unit PU1. Before processing, Punch Service is required to request the operator to insert "RED-STRP" cards into the designated punch.

Network Job Processing Text (NJPIO)

The text for the //*FORMAT card for NJP consists of keyword parameters. These keywords and their definitions are:

```
//*FORMAT  NJP,FROM=terminal-name
              ,DEST=terminal-name
```

FROM=terminal-name

Specifies the name of the NJP terminal from which the job will be transmitted.

DEST=terminal-name

Specifies the name of the NJP terminal to which the job will be transmitted.

The terminal name may be determined by reference to the NJPTERM initialization control card. The terminal name for the local Support Processor is located on the STANDARDS initialization control card. For information on the initialization cards, see your system programming staff.

An example of an NJP //*FORMAT card is:

```
//*FORMAT  NJP,DEST=NYORK,FROM=LAX
```

Example 1 - NJP program:

```
//SAMPLE    JOB 123,ASP,MSGLEVEL=(1,1)
//*PROCESS  RICONTL
//*PROCESS  MAIN
//*PROCESS  NJPIO
//*FORMAT   NJP,FROM=LAX,DEST=DETROIT
//*PROCESS  NJPIO
//*FORMAT   NJP,FROM=DETROIT,DEST=NYC
//*PROCESS  PRINT
//*FORMAT   PR,DDNAME=SYSPRINT
//*ENDPROCESS
//STEP  EXEC    PGM=SAMPLE
    OS JOB DECK
```

In the sample NJP program SAMPLE will be processed on any Main Processor. The job and its output will be sent to DETROIT which will, because of the //*PROCESS NJPIO card (see //*PROCESS cards in this manual), send the job and its output to NYC. NYC will print the data set described by DDNAME SYSPRINT. This is an example of transmitting to a location where there is no direct connection from LAX to NYC.

Example 2 - NJP Program:

```
//SAM2      JOB
//*PROCESS  NJPIO
//*FORMAT   NJP,FROM=LAX,DEST=NYC
```

```
//*PROCESS  RICONTL
//*PROCESS  MAIN
//*PROCESS  NJPIO
//*FORMAT   NJP,FROM=NYC,DEST=LAX
//*PROCESS  PRINT
//*FORMAT   PR,DDNAME=FT06F001
//*ENDPROCESS
//STEP  EXEC    PGM=SAM2
    OS JOB DECK
```

In Example 2, job SAM2 will execute in New York and will be returned to Los Angeles for printing.

ASP Created Data Sets Text (AC)

This FORMAT card is used by TSO users or local users wishing to route data sets to TSO users. TSO users should use the installation-defined TSO output class to retrieve their output. This will reduce the need for //*FORMAT AC control cards.

```
//*FORMAT  AC,DDNAME={ddname|SYSMSG}
           [,ERDEST=printer-name]
           [,USER=userid]
           [,PRINT={NO|YES}]
```

The DDNAME parameter is the ddname on the DD card that represents the output which the user desires to access. This DD statement must be of the form:

```
//name DD SYSOUT=class
```

The SYSMSG ddname will provide the user with OS SMBs (System Message Blocks).

The ERDEST parameter defines the device name of a printer that may be used to print the user-specified job output in the event this output cannot be sent back to the user. The printer name must be a valid ASP printer name.

The USER parameter allows the user to specify a USERID other than his own. The USERID is used to inform the TSO user when his output may be accessed.

The PRINT parameter indicates to the ASP Created Data Set (ACDS) DSP whether or not the DDNAME output is to be printed after successful creation of the TSO EDIT accessible data set. If this parameter is not specified or PRINT=NO is specified, the ACDS DSP will ensure that the affected data set is not processed by ASP Print Service after completion of ACDS processing. If PRINT=YES is specified this data set will be printed.

The following are examples of //*FORMAT AC ASP control cards:

1. `//*FORMAT AC,DDNAME=SYSMSG`

2. `//*FORMAT AC,DDNAME=FT06F001,ERDEST=RM001PR1`

3. `//*FORMAT AC,DDNAME=SYSPRINT,USER=IBMUSER,ERDEST=PR2`

In Example 1 the user has indicated that SYSMSG output is to be sent across the CTC to be processed by ADSGEN (an ASP module which creates and catalogs TSO output data sets) at which time he will be notified. Example 2 is similar to Example 1 except that if ADSGEN processing cannot be accomplished the FT06F001 job data set output is to be sent

to a printer on remote terminal RMC01PR1. Example 3 is similar to 1 and 2 except that a specific user (IBMUSER) is to be notified when the processing of the SYSPRINT output is complete.

MAIN

The MAIN card is used to define the Main Processor dependencies for the current job. Many of the parameters are used to override parameters of the STANDARDS Initialization control card.

```
//*MAIN [SYSTEM={ANY|LOCAL|REAL|main-name|(main-name,main-name,...)|
                 /(main-name,main-name,...)}]
       [,TYPE={ANY|VS2|MVT}]
       [,LINES=(nnn[,{WARNING|CANCEL|DUMP}])]
       [,CARDS=(nnn[,{WARNING|CANCEL|DUMP}])]
       [,HOLD={NO|YES}]
       [,SETUP=(ddname,ddname,..)|/(ddname,ddname...)]
       [,CLASS=class-name]
       [,FAILURE={RESTART|CANCEL|HOLD|PRINT}]
       [,JOBSTEP={NOCHKPNT|CHKPNT}]
       [,NJPCLASS=class-name]
       [,HOTJOB={NO|YES}]
       [,ACMAIN=main-name]
       [,ACHOLD={NO|YES}]
       [,IORATE={MED|HIGH|LOW}]
       [,ORG=group-name]
       [,DEADLINE=(time,type[,{date|rel,cycle}])]
       [,FETCH={ALL|NONE|SETUP|(ddname,...)|/(ddname,...)}]
       [,JPRTY={ASP|JOB}]
       [,LREGION=nnnnk]
```

SYSTEM=(listed-above)

The SYSTEM parameter defines the Main Processor name(s) or type of system to be used for this job.

ANY defaults to any system (real or local) that will satisfy the job requirement.

LOCAL job is to be run on a local Main Processor only. The LOCAL Main Processor is the main in which ASP is resident.

REAL job is to run on any real (non-local) Main Processor that will satisfy the job requirements.

main-name or (main-name,main-name,...) defines the only Main Processor(s) to be considered for this job.

/(main-name,main-name) defines the Main Processor(s) to be excluded from consideration for this job. Any Main not excluded will be a possible candidate for execution of this job. This job will be flushed with an error if all Main Processors are excluded.

TYPE={ANY|VS2|MVT}

The control program to be used. If the type control program requested is not active on the system requested with the SYSTEM parameter, the job will wait indefinitely until that control program is active.

LINES=(nnn[,{WARNING|CANCEL|DUMP}])

The estimated maximum number of lines of data, in thousands, to be printed for this job. The entered information overrides the installation standards established by the ASP initialization STANDARDS control card. The second subparameter specifies the action to be taken when the line estimates are exceeded:

WARNING (or W) Issue operator warning and continue processing

CANCEL (or C) Cancel the job

DUMP (or D) Cancel the job with OS ABEND dump

CARDS=(nnn[,{WARNING|CANCEL|DUMP}])

The estimated number of cards, in hundreds, to be punched for this job. Processing of this parameter is identical to that of LINES, above, except that if zero is specified, the Punch Service Scheduler Element is set complete, and no punching occurs for this job.

HOLD={NO|YES}

If HOLD=YES is specified, the job will be entered into the system in operator hold status and will be withheld from processing until the operator requests its release. This parameter is used for submitting a job that has dependency upon external operator action.

SETUP=(ddname,ddname,...)|/(ddname,ddname...)

The SETUP parameter is used to modify the standard setup algorithm used by the Main Device Scheduler in assigning devices to a job prior to its execution on a Main Processor. The absence of this parameter causes device requirements for mountable tape and disk volumes to be calculated by the system for the entire job. Additional information on the use of SETUP is contained in Chapter 7 of this manual.

(ddname) - specifies the DD statements (that is, stepname.procstepname.ddname, if applicable) that are to be setup before a job enters execution. If this parameter is used, enough devices must be referenced to allow allocation for the step using the largest number of devices. If any setup cannot be allocated and enters OS allocation recovery, the job will be canceled. The ability to specify fewer devices than required in total applies only to tape; all disk references must be processed by setup.

Note: There is no distinction made between concatenated JCL cards and the card to which they are concatenated. Therefore, any reference to a concatenated DD should be avoided. Jobs requiring such a reference should allow OS to set up its devices by specifying SETUP=/(ddname).

/(ddname) - removes the specified DD names from consideration for setup. If setup is required, it will be handled by OS.

Example:

```
//EX1   JOB
//*MAIN SETUP=(DD1,DD2)
//STP1  EXEC
//DD1   DD  UNIT=2314,VOL=SER=VOL1,DISP=OLD          X
//           DSN=EXAMPLE
//STP2  EXEC
//DD2   DD  UNIT=2314,SPACE=(TRK,1),VOL=SER=VOL2     X
           DSN=EXAMPLE2
```

In this example (job EX1) two units will be set up, one each for the DD1 and DD2 statements.

The following is an example of a job that will be canceled because of invalid use of SETUP.

```
//EX2   JOB
//*MAIN SETUP=(DD1,DD2)
//STP1  EXEC
//DD1   DD  UNIT=2314,VOL=SER=VOL1,DISP=OLD                    X
//           DSN=EXAMPLE
//STP2  EXEC
//DD2   DD  DSN=ASP.SYSPRINT,DISP=OLD,VOL=REF=*.STP1.DD1
```

This job (EX2) will be canceled because the SETUP parameter specifies two units are to be set up and the OS JCL specifies one unit because of the refer back (*.STP1.DD1) in the DD2 statement. This situation could be corrected by specifying either SETUP=DD1 or SETUP=DD2.

CLASS=class-name

The CLASS parameter defines the job class for this job. The name may be one- to eight-characters. If a single character class name is used it may be specified on the JOB card. If this parameter is omitted and no class is specified in the JOB card the installation standard class is used. A valid CLASS parameter on the //*MAIN card overrides a valid CLASS parameter on the JOB card.

FAILURE={CANCEL|HOLD|PRINT|RESTART}

Specifies the job recovery option to be used in case of system failure for non Hot Jobs (Real or Local). This option overrides the installation standard. CANCEL cancels the job on the Main Processor. HOLD holds the job for restart on Main. PRINT prints the job and then puts the job in hold for restart on Main. RESTART restarts the job on Main.

JOBSTEP={CHKPNT|NOCHKPNT}

Specifies the job step checkpoint option. CHKPNT causes a checkpoint to be taken at the end of each job step on Main. This checkpoint contains the current status of all ASP output data sets for the job. If an ASP system failure occurs on the Support Processor before the job is completed on Main, all output up to and including the last complete job step will be saved and made available for output processing. (No output data is lost when a system failure occurs on a real Main Processor.) Output processing will occur after an ASP restart when FAILURE=CANCEL or FAILURE=PRINT is used. If the job is restarted on Main (which would occur when FAILURE=RESTART, FAILURE=HOLD or FAILURE=PRINT is used), no data from the previous run is saved (that is, the checkpoint information and all of the output data sets will be purged). NOCHKPNT stops the checkpoint facility.

NJPCLASS=class-name

Specifies that this job is to be placed into an identifiable group of jobs that may be transmitted via NJP. During processing, the operator may initiate transmission of this entire group of jobs with one operator command rather than entering a separate command for each individual job. The class name may be alphameric and consist of from one to eight characters. It may be specified as an actual terminal name if desired.

HOTJOB={NO|YES}

Hot Jobs, or non-ending tasks, can be scheduled by ASP. If an ASP system failure occurs while a Hot Job is active, ASP can be restarted without interrupting its execution. In addition, for Hot Jobs utilizing the ASP setup facility, the allocated devices will be reserved over an ASP restart. As a result there will be no rescheduling requirements for the Hot Jobs after an ASP restart. Hot Jobs cannot use the CTC for SYSIN or SYSOUT.

ACMAIN=main-name

Defines a Main Processor which has TSO resident. For more information, see the section, "Using ASP TSO Support" in this manual. This parameter must be used in conjunction with the //*FORMAT AC card if ASP job output is to be sent to TSO terminal users other than the submitter.

ACHOLD={NO|YES}

This parameter should be used by the TSO user who wishes to submit a job to ASP for processing. It allows him to temporarily suspend output processing while he inspects his results. Provides the ability to place the job in hold status after the ACDS DSP has processed the data sets defined by the //*FORMAT AC cards. A use of this parameter is to allow the user to examine his SYSMSG data set to determine successful job completion and then either release or cancel the job, via the appropriate TSO commands.

IORATE={LOW|HIGH|MED}

This parameter describes the I/O to CPU ratio for a job as being Low, High, or Medium I/O. Main Service attempts to balance the mixture of jobs executing on a Main Processor based upon this IORATE If this parameter is omitted, the installation default for this job class will be used.

ORG=group-name

The ORG parameter is used to override the group name of the device from which this job entered the ASP system. Normally any output from a job is directed to the same group of devices from which it originated. This parameter will cause any output to be directed to the specified group, unless specifically directed by a //*FORMAT card. This may be specified by the user who temporarily must use an alternate RJP workstation but would like all his output to be processed as it was entered through his normal sources.

DEADLINE=(time,type[,{date|rel,cycle}])

Time may be specified in minutes, hours, or wall clock time. It specifies the time that the job is due to be completed. Specifying time in minutes requires the letter M, (1M), one minute; time in hours the letter H, (1H), one hour. Time as wall clock time requires four digits, 0800, eight AM.

The type field is a single character and must match one of the installation deadline types. If an invalid deadline type parameter is specified, it is ignored by ASP, and the job is processed without deadline support.

The date is an optional parameter that specifies the date on which the time parameter will expire. The date is specified as MMDDYY. If date is omitted, the current date is assumed provided that the current time is less than the deadline time. If the current time is greater, the next day's date is assumed. The rel and cycle parameters are for production runs which are run on a periodic basis. Cycle is either WEEKLY, MONTHLY, or YEARLY. The rel parameter modifies cycle by specifying what day within the cycle the deadline falls. To specify Sunday of every week, code 1,WEEKLY Saturday would be 7,WEEKLY. The last day of every month would be 31,MONTHLY. Year end jobs might be 365,YEARLY. Weekly values are truncated to 7 if greater than 7, monthly values are truncated to the last day of the month if greater than that day, and yearly values are truncated to the last day of the year if greater than that. Thus, 8,WEEKLY becomes 7,WEEKLY, a non-leapyear February specification of

31,MONTHLY, becomes 28,MONTHLY, and a non-leapyear specification of 366,YEARLY becomes 365,YEARLY.

If the current date is specified and the job was submitted after the deadline time, all of the priority changes will be applied to make it the same priority level it would have been if it had been submitted prior to the deadline and not completed.

The following two examples will show a job submitted before and after its deadline. The //*MAIN card used for the examples and the Deadline type defined at ASP initialization are shown below.

```
//*MAIN DEADLINE=(0800,A,032173)
```

DEADLINE A=(10,1H,+1,30M) specifies deadline type A. Type A will cause priority 10 to be set 1 hour before a job's deadline and will increment the priority by 1 every 30 minutes until the job is complete.

Example 1:

```
Job submitted   0400    PRTY=5
                0700    Set PRTY=10
                0730    increment+1 PRTY=11
Deadline        0800    increment+1 PRTY=12
                0830    increment+1 PRTY=13
Job complete    0840
```

Example 2:

```
Deadline        0800
Job submitted   0840    priority calculated and set to 13
                0900    increment+1 PRTY=14
Job completed   0910
```

FETCH={ALL|NONE|SETUP|(ddname)|/(ddname)}

The ASP R/I will issue Fetch messages to the console designated for tape and disk setup. This parameter will override the installation defined standard for this job.

ALL

Messages will be issued for all volumes in DD statements using ASP setup devices except permanently resident volumes.

NONE

No Fetch messages.

SETUP

Only messages for the volumes in the DD statements specified in the SETUP parameter on the //*MAIN card. If no SETUP parameter is supplied, the FETCH parameter will default to ALL.

(ddname)

Only the volumes in the DD statements specified will be fetched.

/(ddname)

Issue fetch messages for all vclumes in setup DD statements except these that are listed.

JPRTY={ASP|JOB}

If JOB is specified, the job will be run on Main using the PRTY parameter from the JOB card. If this parameter is omitted or JPRTY=ASP is used, the execution priority will be assigned by ASP. Job priority is changed after job selection but before job execution. Therefore the original priority is used for job selection and for any post-execution processing.

```
Example:    //*MAIN SYSTEM=REAL,LINES=(2,W),                X
            //*CLASS=ABLE,JOBSTEP=CHKPNT
```

This job can execute on any real Main Processor. All DD cards for a job will be set up before the job will schedule on a Main Processor. Checkpoints will be taken at the end of each job step.

LREGION=nnnnk

LREGION should closely approximate the largest step's working set in real stcrage during execution. The LREGION parameter applies to those jobs that may execute on a VS2 Main Processor either specifically or through the ANY designation of the TYPE= parameter. The LREGION (logical region) is used internally by ASP to optimize scheduling on a VS2 Main. If this parameter is omitted ASP will determine the value of LREGION based on the LSTRR assigned to this job's class during ASP initialization. It is recommended that the applicatiCn programmer consult the system programming staff prior to using the LREGION parameter.

NET

The //*NET control card identifies to the ASP system the relationship of dependent jobs that are members of a given job-net. Specifically, this control card defines the link between a predecessor job and its successor job(s). In turn, successor jobs may be predecessors to other jobs of a job-net.

Only one //*NET card may be defined for each job of a job-net.

Two forms of notation are allowed in defining keywords. Listed first, in the keyword definitions, is the long form of notation. Listed second is the shorthand form. These notations may be mixed on the //*NET card.

The format of the //*NET card is:

```
//*NET {NETID|ID}=name
      [,{NHOLD|HC}=n]
      [,{RELEASE|RL}=(jobname1,jobname2,....,jobnamen)]
      [,{NORMAL|NC}={D|F|R}]
      [,{ABNORMAL|AB}={R|F|D}]
      [,{OPHOLD|OH}={NO|YES}]
      [,{RELSCHCT|RS}=n]
      [,{NETREL|NR}=(netid,jobname)]
```

The keywords and their parameters are defined as:

{NETID|ID}=name

NETID specifies the name of the job-net of which this job is a member. The name must be alphameric and may consist of one to eight characters, the first of which must be alpha. All jobs put into the system with the same netid form a Dependent Job Control (DJC) job-net. NETIDs must be unique within the ASP system.

Note: Duplicate job-nets cannot exist since a job that has the same NETID as an existing job-net is added as a member to that job-net.

{NHOLD|HC}=n

This parameter specifies the number of immediate predecessor job completions required before this job can be released for scheduling. When this parameter is defined, the job is placed into DJC hold status when it enters the system. N has a range of 1 to 32,767. If n is zero or not specified, then this job has no predecessors and is immediately eligible for scheduling.

Note: If an incorrect NHOLD count is specified, two situations can occur: (1) if n is greater than the actual number of predecessor jobs, then this job will not be released from DJC hold when all of its predecessors complete execution; (2) if n is less than the actual number of predecessor jobs, then this job will be prematurely released from DJC hold.

{RELEASE|RL}=(jobname1,jobname2,...,jobnamen)

The release parameter is used to denote jobnames of successor jobs to this particular job. From 1 to 50 successor jobnames may be specified. Jobnames may be from one to eight alphameric characters, the first of which must be alpha.

Note: This is the only //*NET parameter that may be split on a //*NET continuation card.

{NORMAL|NC}={D|F|R}

This parameter specifies the action to be taken for this job when any predecessor successfully completes execution.

- F= flush this job from the system as well as its successors. The job will be canceled with print and all successors are canceled regardless of their NC or AB specifications.
- R=retain this job in the system and do not decrement the NHOLD count. This suspends this job and its successors from scheduling until positive action is taken either by resubmitting its predecessor or via operator action.
- D=decrement the NHOLD count of this job. If the NHOLD count goes to zero, this job is released from DJC hold and becomes eligible for scheduling.

{ABNORMAL|AB}={R|F|D}

This parameter is the converse of the normal parameter and is used when a predecessor abnormally completes execution. F, R, and D are defined above.

{OPHOLD|OH}={NO|YES}

Specification of this parameter causes this job to be placed in DJC-operator-hold. Operator hold prevents scheduling of this job until it is explicitly released from hold via operator action. DJC operator hold is independent of the normal ASP operator hold function.

{RELSCHCT|RS}=n

This parameter is called the release-scheduling-count parameter. It can be used in conjunction with those members of a job-net which are ASP SETUP jobs. Its function is to allow a dependent job to be scheduled through the SETUP function of ASP before all of its predecessors have completed execution. At the point in processing a net job where the NHOLD count becomes less than or equal to its release schedule count, the job is made eligible for scheduling through the ASP SETUP function. N has the same range as the NHOLD parameter. If N is zero, it is the same as no parameter specification.

Note: This parameter must not be specified for a job which may have catalog dependencies. That is, it is mutually exclusive of catalog dependent jobs.

Specification of this parameter is invalid for nonstandard DJC jobs.

{NETREL|NR}=(net-id,jobname)

This parameter is called the net release parameter. It provides a function in DJC which allows a job of one job-net to be a predecessor to a job of another job-net. To identify the alien successor to this job, the successors jobname and NETID must be defined as the parameters of the NETREL keyword. The NETREL parameter may be specified once for each job of a given job-net.

Refer to the section of this manual entitled "Using Dependent Job Control (DJC)" for additional information and examples. For more detailed information on DJC, refer to the System Programmer's Manual.

THIS PAGE INTENTIONALLY LEFT BLANK

OPERATOR

The //*OPERATOR card is used to transmit any desired message to the operator. Columns 1 through 80 of the card are sent to the LOG console when the job is entered into the ASP queue.

```
//*OPERATOR text
```

Example: //*OPERATOR CALL EXT. 641 WHEN THIS JOB STARTS

PROCESS

Jobs submitted to the Support Processor locally through an attached card reader, tape unit, or disk unit normally are routed first to the Reader/Interpreter, then to the Main Processor for execution, then to Print and Punch Services, and finally to the Purge program.

Standard processing is sufficient for most program execution; however, nonstandard jobs require special steps to be performed on the Support Processor either before or after execution on the Main Processor. This nonstandard job routing is specified by inserting a series of //*PROCESS cards, in the desired sequence, into the job deck behind the JOB card. When a //*PROCESS MAIN card is used it must be preceded by a //*PROCESS RICONTL card.

If a particular function requires //*FORMAT parameter cards, these cards must immediately follow the associated //*PROCESS card. The maximum number of //*FORMAT cards per //*PROCESS card is dependent upon ASP buffer size. For information on the use of //*PROCESS cards in scheduling callable DSPs, see Chapter 8 in the section "Background Utility Programs", of the ASP Operator's Manual.

The format of the //*PROCESS card is:

```
//*PROCESS  function-name
```

The function name, which follows the //*PROCESS verb is separated from it by one or more blanks, and is from one to eight characters in length. This name must be listed in the ASP Dynamic Support Program Dictionary. (See Appendix A for a list of DSP names.) Reference to a function not contained in the ASP DSP Dictionary will cause the job to be terminated.

An example of a //*PROCESS card is:

```
//*PROCESS PUNCH
```

ENDPROCESS

When //*PROCESS cards are used, they must be terminated by an OS control card, or by the //*ENDPROCESS card. When used, the //*ENDPROCESS card terminates the control card processing phase of Input Service and must be the last ASP control card of the sequence.

The format of the //*ENDPROCESS card is:

```
//*ENDPROCESS
```

READER CONTROL CARDS

//**command cards

Cards having //** in columns 1 to 4 may have special meaning to ASP if placed before the first JOB card in a submitted job stream. Operator commands may be submitted via a reader by punching the command into a card, starting in column 4. Also, an input reader may be halted temporarily by punching the pseudo command PAUSE, starting in column 5.

The use of these cards normally would be directed by an installation's systems programming staff for system status control and system checkout.

Operator's instructions on the use of reader control cards is contained in Chapter 5 of the ASP Version 3 Operator's Manual, GH20-1289.

OS JCL

The JCL parameters used by the application programmer are not restricted, but some of the parameters may take on additional meanings or usage; for example, for job scheduling, determining use of the CTC, etc.. The cards affected in an ASP environment are the JOB, EXEC, and DD cards.

JOB CARD

ASP uses the JOB card parameters to determine how the job is to be scheduled. The REGION, CLASS and PRTY parameters are used in conjunction with the installation defined scheduling algorithms to determine the scheduling. The //*MAIN card, if supplied, has parameters that can be used to override or further define part of the scheduling algorithms. The application programmer should consult the system programming staff about using JOB card parameters to best effect. It should be noted that the REGION parameter on the JOB or EXEC card is handled in the normal manner. No special processing occurs for Region determination in the job steps.

Main storage hierarchy support is not available in ASP. REGION specifications originally made for hierarchy support do not have to be recoded. The system will sum the values coded and use the sum as the job's region requirement.

MT may not be used as the job name on a JOB card.

TYPRUN=HOLD has no significance and is ignored by ASP.

If the job's REGION parameter is too large to execute on every Main Processor in the ASP complex the specific Mains should be selected on a //*MAIN control card.

The OS/VS2 NULL statement will be supported as a job terminator. If the following statement is not another NULL statement or an OS/VS2 comment statement, it will be assumed to be an OS/VS2 JOB statement.

EXEC CARD

If any EXEC step of the job specifies a program name of JCLTEST, the ASP R/I will flush the job with its interpreted JCL written to the ASP SYSMSG print data set accompanied by a JCLTEST message. This facility allows one to test only JCL, without using a Main Processor or any setupable devices.

DD CARD

The DD card usage by the application programmer remains unchanged from that of a non-ASP environment. As in the JOB card some parameters can take on additional meaning or usage. ASP will assign DD cards describing SYSIN or SYSOUT to the CTC, all other DD cards are unaffected.

The DD cards describing SYSIN are determined by checking for an asterisk (*) or the word DATA following the DD operation field. These cards are assigned to the CTC by replacing the parameter, * or DATA, with UNIT=(CTC,,DEFER),DISP=(OLD,DELETE), DSNAME=&&ASPInnnn,DCB=(LRECL=80,BLKSIZE=80,RECFM=F) (nnnn starts at 0001 and is incremented by one for each SYSIN type data set in the job.) OS allocation will allocate this DD entry to the CTC.

DD cards containing the parameter, SYSOUT=class, will be assigned to the CTC, providing the "class" has been defined by the installation as a class that is eligible.

DD statements referencing tape devices that are processed by the Main Device Scheduler will have tape volume file protect ring status monitored. The correct ring status will be required as determined by the DISP parameter in the DD statement. DISP of NEW or MOD will require a ring in the tape volume and OLD must have no ring.

Following are examples of the DD cards that designate data sets for the Support Processor via the Channel-to-Channel Adapter:

- System input:

```
//SYSIN     DD  *
```

- System output that is normally printed:

```
//SYSOUT    DD  SYSOUT=A
```

Restrictions

- Use of the CTC is restricted to SYSIN or SYSOUT.
- The SYSIN cannot be blocked.
- SYSOUT blocksize must be a multiple of LRECL and less than ASP buffer size minus 24 bytes. ASP buffer size is determined by the system programmer in the BUFFER initialization card.
- Concatenation of CTC data sets is not permitted unless the DCB in use has the unlike-attribute bit set. In addition, a BPAM OPEN to a CTC data set concatenated to a disk data set is not supported (this is an error condition).
- Chained scheduling is not supported for DD entries assigned to the CTC.
- The Data Delimiter Parameter (DLM=), if used on a SYSIN DD card, must appear on the first card of the DD statement, not on a continuation card.
- Only one DCB can be open to any CTC-device DD statement at one time.
- Reserved ASP DD names:

```
SYSMSG
JCBIN
JCBTAB
JCLIN
ASPInnnn
```

- ASP catalog support for the VSAM replacement of an ISAM data set will not recognize multiple device types within a single DD card request.
- VSAM JOBCAT and STEPCAT private library requests are not supported.
- MDS is more critical than OS regarding the use of the 'DISP' parameter for tape mounts (RING/NORING). For example, 'DISP=NEW' must be used for mini-reel correction tapes because there is no ring to remove.

- Under ASP, jobs bearing like names will not be scheduled concurrently. They will, instead, be scheduled to run one at a time.
- Any job which requests a device by a specific unit address must include a //*MAIN card specifying the specific system name to which the device is attached.
- The ASP R/I does not perform any checking for the presence of the COND parameter on the EXEC card of a step. Therefore, the assumption is made that all steps of a job will be executed serially, and setup will be performed accordingly.
- The Channel Separation parameter (SEP=) on a DD card is not supported by ASP.
- OS/VS1 may request some ASP premounted tape volumes be moved to units other than those upon which they are currently mounted. This may occur when 'deferred' tape mounting is requested.

DECK SETUP

This section describes the deck setup requirements for submitting application programs to the ASP system. ASP permits the programmer to incorporate preprocessing and postprocessing features implemented on the Support Processor. Using ASP control cards, the programmer specifies job requirements to ASP via the input stream, thereby eliminating many of the functions ordinarily performed by installation personnel. (See the section on ASP control cards for a detailed description.) As presented in the input stream, the job may be considered as either standard or nonstandard. A standard job implies that the routing of the job is determined by Input Service and includes the job segments for Reader/Interpreter, Main Processing, printing, and punching, with or without external volumes that require mounting. A nonstandard job requires all preprocessing, processing, and postprocessing phases to be stated explicitly by the programmer, with only the Purge segment and MDS scheduling implied. Note that all jobs must have an OS JOB card as the first card.

STANDARD JOB

Following is a group of examples and explanations for standard job execution:

Example 1:

```
//EXAM1 JOB
//STEP  EXEC    PGM=EXAM1
//DD1   DD      SYSOUT=A
//DD2   DD      *
        data
/*
```

In Example 1, ASP control cards were not included in the job's JCL. The changes which ASP will automatically make to the JCL will be to assign DD1 and DD2 to the CTC by adding the UNIT=(CTC,,DEFER) parameter, and the installation defined default values for priority, region and class will be affixed to the JOB card.

Example 2:

```
//EXAM2     JOB
//*MAIN     SYSTEM=(MAIN1,SY2,TSO1),FAILURE=HOLD,        X
//*ACMAIN=TSO1,IORATE=HIGH
//*PROCESS  RICONTL
//*PROCESS  MAIN
//*PROCESS  ACDS
//*FORMAT   AC,DDNAME=DD6,USER=TSO12
//*PROCESS  PRINT
//*FORMAT   PR,DDNAME=DD6,COPIES=3,DEST=PR2
//*PROCESS  PUNCH
//*FORMAT   PU,DDNAME=DD7,DEST=MACHROOM,                 X
//*FORMS=5081
//STEP      EXEC    PGM=EXAM2
//DD1       DD      user-supplied parameters
//DD4       DD      user-supplied parameters
//DD6       DD      SYSOUT=N
//DD7       DD      SYSOUT=P
        remaining JCL
```

Example 2 shows use of //*FORMAT cards to describe special handling to the system or operator. The //*FORMAT card for Print indicates that three original copies of the data set referenced by DD6 are to be

printed on PR2. The train, carriage tape, and forms used will be determined by the installation as a default value.

The //*FORMAT AC card is used to send a copy of the referenced data set to a TSO terminal user whose ID is TSO12. The ACMAIN parameter on the //*MAIN card is used to specify the Main Processor to which the TSO12 user is connected.

Punch service will reference the //*FORMAT PU card to determine special handling for DD7. In the example given, DEST=MACHROOM, where MACHROOM is defined as a specific group of devices defined at ASP initialization. One copy of the referenced data will be punched using 5081 cards.

The program EXAM2 can be executed on any one of the three systems defined in the //*MAIN card. In the event the support system should fail while this job is in execution, ASP will place this job in hold status. The job has a high input/output activity. The DD statements used for setup will be covered later.

Example 3:

```
//EXAM3     JOB
//*MAIN     SYSTEM=MAIN1,HOTJOB=YES,CLASS=LONG
//STEP      EXEC    PGM=FOREVER,TIME=1440
        remaining JCL
```

EXAM3 is specified as a HOTJOB and has a job class named LONG. Hot jobs are normally long running jobs that are not affected by a support system restart. The CLASS parameter is used in an installation-defined job scheduling algorithm. The CLASS name must match a CLASS name defined at ASP initialization.

Example 4:

```
//EXAM4     JOB
//*MAIN     SYSTEM=/(MAIN1,MAIN2),LINES=(2,WARNING),     X
//*DEADLINE=(1600,D,2,WEEKLY),NJPCLASS=LACC
//STEP      EXEC
        remaining JCL
```

This example specifies that MAIN1 and MAIN2 are not eligible as Main Processors for this job's execution. The job will use the installation-defined Deadline Scheduling algorithm type D and will be scheduled prior to 4 PM (1600) the following Monday (2,WEEKLY). The operator may send this job via the NJP network by referencing the NJPCLASS name.

The use of DEADLINE means that this job is to be scheduled anytime prior to the Deadline time. If EXAM4 was entered into the system on Saturday at a low priority, it may be executed on Sunday, depending upon the number of jobs of higher priority. This meets the requirement for the deadline. Ancther technique would be to enter the job at a priority that has been held by the operator. This way the job will be held until the DEADLINE algorithm increments and changes the priority to a priority level not being held. At this time the job will enter contention for scheduling.

After the job has completed execution it will have to be resubmitted for the next week's run; this is not automatic.

Example 5:

```
//EXAM5     JOB
//*MAIN     SETUP=(S1.DD2,S1.DD3,S2.PS1.DD1)
//S1        EXEC    PGM=EXAM5
//DD1       DD      SYSOUT=A
//DD2       DD      setupable disk reference
//DD3       DD      setupable tape reference
//DD4       DD      *
            data
/*
//S2        EXEC    PROC1
//S3        EXEC    PGM=
//DD5       DD      setupable tape reference
//J2        JOB
```

PROC1 is defined as:

```
//PS1       EXEC    PGM=
//DD1       DD      setupable disk reference
//DD2       DD      SYSOUT=C
/*
```

Example 5 depicts a job using setup for specific DD names. The device types described by the three DD statements will be set up and allocated for the life of the job. (Note the device types and numbers of each device type will be all that will be available for any step allocation.) Step S1 has the largest device requirement, with one disk and one tape request.

If this job (EXAM5) was submitted without the SETUP parameter, all of the setupable devices would have been set up and allocated for the life of the job; that is, DD2, DD3, DD5 and PS1.DD1. This method would have two tapes and two disks assigned to this job for its life.

Invalid control card usage will cause the job to be flushed from the system by the Reader/Interpreter DSP after job interpretation.

The following are examples of invalid or redundant control usage:

Example 6:

```
//EXAM6     JOB
//*MAIN     SETUP=(S1.DD1,S1.DD2,S1.DD3),           X
//*SYSTEM=(/MAIN1,MAIN2,SY1),TYPE=MFT
//S1        EXEC
//DD1       DD      DSN=POOREXAM,VOL=SER=123,UNIT=2314,DISP=OLD
//DD2       DD      DSN=SYS1.SVCLIB,DISP=SHR
//DD3       DD      UNIT=2400,DISP=(NEW,KEEP),DSN=NEWOUT
//DD4       DD      SYSOUT=N
//DD5       DD      *
            data
/*
//S2        EXEC
//DD6       DD      DSN=ANY,VOL=SER=456,UNIT=2314,DISP=OLD
//DD7       DD      DSN=TEMP,VOL=SER=33,UNIT=2314,DISP=OLD
//DD8       DD      SYSOUT=9
//DD9       DD      *
/*
```

DD2 references the SYS1.SVCLIB data set which is on a permanently resident device and will not require setup. The device types assigned this job will be one 2314, because of DD1 and one 2400 because of DD3. Step S2 has a requirement for two 2314's; only one 2314 was reserved for the job. This step will enter Allocation Recovery to obtain a

second 2314 and this will cause the job to be canceled. This job could have executed if the SETUP parameter was omitted.

The SETUP parameter is only one of the errors on the //*MAIN card. The SYSTEM parameter is attempting to exclude MAIN1 and allows only MAIN2 and SY1 for scheduling choices, this type of intermixed reference is invalid.

TYPE=MFT is invalid.MFT is not supported.

NONSTANDARD JOB

A nonstandard ASP job is defined as a job with //*PROCESS cards and is a combination of job segments (scheduler elements) that deviates from the standard combination of job segments (that is, Reader/Interpreter, Main Service, Print Service, Punch Service, and Purge). Some examples of nonstandard Scheduler Element combinations are:

- A preprocessing job segment such as a JCL generator plus the five standard job segments
- Fewer than the five standard job segments (for example, Reader/Interpreter, Main Service, Print Service, and Purge, without Punch Service)
- One or more postprocessing job segments plus the five standard job segments
- One or more job segments, of which only Purge is a standard job segment

A nonstandard job is specified to the system by a JOB card, followed by a //*PROCESS card for each job segment, followed by the normal OS deck. In a nonstandard job, there must be a //*PROCESS card for each job segment except Purge.

The following are examples and explanations for nonstandard job execution:

Example 1:

```
//EXAM1     JOB
//*PROCESS  RICONTL
//*PROCESS  MAIN
//*PROCESS  PRINT
//*PROCESS  PUNCH
//*ENDPROCESS
//S1        EXEC
        remaining JCL
```

Example 1 is a simple explanation of submitting a job via //*PROCESS cards. The given example would execute the same as a standard job without ASP control cards. Five scheduler elements would be created for this job. The RICONTL DSP will create the OS control blocks for the Main Processor, the next scheduler element. The Print and Punch DSP's can be scheduled to execute in parallel. PURGE is the last function in any job and has a scheduler element created automatically.

CAUTION: Anytime a //*PROCESS MAIN is used a //*PROCESS RICONTL card must precede it and the two must execute in the same system; that is, a //*PROCESS NJPIO card may not be placed between the MAIN and RICONTL statements.

Example 2:

```
//EXAM2     JOB
//*PROCESS  NJPIO
//*FORMAT   NJP,FROM=LAX,DEST=AKRON
//*MAIN     CLASS=REM,ACMAIN=TSOS
//*PROCESS  RICONTL
//*PROCESS  MAIN
//*PROCESS  ACDS
//*FORMAT   AC,DDNAME=TERMOUT,USER=TERM10
//*ENDPROCESS
//S1        EXEC
        remaining JCL
```

EXAM2 will be sent via NJP to another ASP Support Processor at DEST=AKRON. The processing at that location can be done on any Main Processor in the system. The RI DSP must be executed on the support system to which the Main Processor is connected. The ACDS DSP (//*PROCESS ACDS) will send to the TSO terminal user, TERM10, the data set described by DD name TERMOUT. The terminal user is connected to the Main Processor named TSOS (ACMAIN=). Output to the terminal user is the only output used from this job.

Example 3:

```
//EXAM3     JOB
//*PROCESS  RICONTL
//*PROCESS  MAIN
//*PROCESS  PRINT
//*FORMAT   PR,DDNAME=OUTPUT,CONTROL=DOUBLE
//*PROCESS  PLOT
//*PROCESS  TT
IN=(TA7),MDI=T,ID=DEP836,FILES=2
//*ENDPROCESS
//S1        EXEC
//OUTPUT    DD      SYSOUT=G
//DD1       DD      UNIT=24007,DISP=(NEW,KEEP)
        remaining JCL
```

Here is an example of using user-written DSPs and ASP utilities. PLOT is a user-written DSP and is to be executed after Print Service has completed. The DSP TT is the Tape-to-Tape utility and is followed by the parameters needed to function. The ASP utilities and their usage are covered in Chapter 8 of the ASP Operator's Manual.

The following examples have invalid or redundant entries:

Example 4:

```
//EXAM4     JOB
//*PROCESS  MAIN
//*PROCESS  PRINT
//*PROCESS  PUNCH
//*FORMAT   PR,DDNAME=NONO,DEST=MACHROOM
//*PROCESS  PURGE
//*ENDPROCESS
//S1        EXEC
        remaining JCL
```

The //*PROCESS card for RICONTL is missing, //*FORMAT PR is not following the proper PROCESS card (PROCESS PRINT).

The //*PROCESS Purge card was supplied, but will be ignored by ASP.

Example 5:

```
//EXAM5     JOB
//*PROCESS  RICONTL
//*PROCESS  MAIN
//*PROCESS  PRINT
//*ENDPROCESS
//*FORMAT   PR,DDNAME=NONO2,DEST=3211
//*PROCESS  PUNCH
//*FORMAT   PU,DDNAME=PUNCH,COPIES=2
//S1        EXEC
```

The placement of the //*ENDPROCESS card will cause the remaining ASP control cards to be ignored.

ASPNEWS Facility

The ASPNEWS facility of Print Service provides a DSP that will create an output data set that is printed after every job. This facility can be used to broadcast general information to the ASP system users. Print Service prints the ASPNEWS before the final burst page.

ASPNEWS is updated by a normal OS job with the following JCL:

```
//ASPNEWS   JOB...
//*PROCESS  ASPNEWS
//*DATASET  DDNAME=ASPNEWS
        DATA CARDS
    (each data card produces one line of print)
//*ENDATASET
```

The ASPNEWS output data set is terminated by resubmitting the ASPNEWS job without data cards.

CHAPTER 4. USING DEPENDENT JOB CONTROL (DJC)

Dependent Job Control (DJC) is used when a system or collection of jobs must be executed in a specific order due to job dependencies. It is invoked via the //*NET control card submitted with each job in a defined network of jobs. Its function is to make a given job in a net eligible for scheduling based upon predecessor job completion. Jobs that depend on one or more predecessor jobs to complete are called successor jobs.

DJC SCHEDULING CRITERIA

Predecessor completion (NORMAL or ABNORMAL) is the event in DJC processing which invokes successor job updating. The NHOLD parameter is operated upon based on the NORMAL/ABNORMAL parameter specification of a successor job. A successor job is made eligible for scheduling when its NHOLD count becomes zero. This applies to the NETREL facilities where two distinct networks may have jobs with dependent relationships.

For a standard ASP job, a predecessor is considered completed when it has finished Main Processing. Successful completion is defined as NORMAL job completion. Successors to the completing job are updated, based on their normal parameter specifications. If a predecessor is terminated because of a JCL error at Reader/Interpreter time, it must be corrected and resubmitted. No action is taken to update successors in this case. If the predecessor had abnormally completed on Main Processor, then the successor update function would be based on ABNORMAL parameter specifications. Abnormal completion is defined when an OS message is received for a job with the prefix "IEF45".

Note: Resubmittal of a previously failed DJC job (that is, failed in execution) will be allowed if there is at least one uncompleted job remaining in the network.

An additional job completion option is available for the standard ASP DJC job. A problem program may issue a WTO (write-to-operator) message which will invoke DJC updating at the point in time when the message is received. For example, the WTO could be issued in step one of a ten-step job. Thus, a DJC job may be updated before job completion, causing successor jobs to be operated upon. The WTO format to invoke this option is as follows:

```
*1       *9       *18
ASPDJCx  jobname net-id  where;

x=1      Represents normal job completion
x=2      Represents abnormal job completion

* (field locations in message - fixed format)
```

Nonstandard ASP DJC jobs require a //*PROCESS DJC control card. The position of this card, within the set of //*PROCESS cards for a nonstandard ASP job, indicates at what point this job is considered complete to DJC. That is, the time its successors should be considered for scheduling. The //*NET control card must precede the //*PROCESS DJC card in the input stream. Nonstandard jobs are always considered to complete normally.

Note: If a nonstandard DJC job contains a Main-Service scheduler element, then the //*PROCESS DJC control card may be removed from the job, allowing job execution to determine DJC completion criteria.

The RELSCHCT parameter is a facility which allows early scheduling of a successor job. This function is intended to allow DJC ASP setup jobs to be scheduled through the ASP SETUP function and placed in a HOLD status until its predecessors complete processing. The facility is invoked when the NHOLD count becomes less than or equal to the RELSCHCT count. Care must be exercised in defining the RELSCHCT count for a

THIS PAGE INTENTIONALLY LEFT BLANK

job. If a job is allowed into SETUP too early, then resources may be tied up for an extended period of time. The RELSCHCT parameter is not applicable to nonstandard jobs.

Under the NETREL function, a job of a given net is a predecessor to a job of a different net allowing inter-net dependence. The parameter specifications of the successor jobs in one net are dependent upon the completion of the predecessor job of the given net as well as any other predecessor defined. This function applies to both standard and nonstandard jobs.

Note: For further information on parameters used in Dependent Job Control (DJC) see ASP control card write-up on //*NET Card in Chapter 3.

DJC JOB-NET DEFINITION

Once a collection of dependent jobs has been identified, then it is necessary to translate the intra-net dependencies to //*NET control cards. The following is a suggested guideline for defining DJC networks:

1. Construct a node-diagram of the network, connecting dependent jobs with lines indicating the flow of job execution through the network. Write the jobname of each job inside its respective node. Assign a NETID.

2. Note next to each node the number of predecessors to this job, including predecessors of other job-nets, if applicable. In addition, if early SETUP scheduling is desired, specify here as RS=count.

3. Jobnames of the applicable successor jobs for each node can be observed directly from the diagram. If a node has a successor in a different net, then list the successor jobname and successor net-id in parentheses.

4. Note next to each node of the network the disposition of this job based on predecessor completion.

5. If a node represents a nonstandard job, then write NS inside the node.

6. Based on the node-diagram, construct the //*NET control cards and //*PROCESS DJC control cards, where applicable.

7. NET verification: It may be desirable to verify the //*NET definition of a new network. One way this could be done is to use a network of jobs that execute the OS-job, IEFBR14. The general format for each job of the net would be:

```
//jobname  JOB  ETC,...
//*NET  control card
//STEP1  EXEC PGM=IEFBR14
/*
```

   In this way, all DJC net functions and definitions can be tested without using actual jobs.

The diagram to control card translation is:

```
ITEM 1 =    Jobname and NETID
ITEM 2 =    NHOLD count and RELSCHCT count
```

ITEM 3 = RELEASE jobnames and/or NETREL net-id and jobname
ITEM 4 = ABNORMAL and NORMAL parameters
ITEM 5 = Requires a //*PROCESS DJC control card

The following examples demonstrate the application of the DJC network definition guidelines. The examples progress from simple to more complex networks. Example 1 shows the step-by-step application of the DJC network guidelines.

Example 1: A single string network

Given:

(a) Four jobs, A, B, C, and D

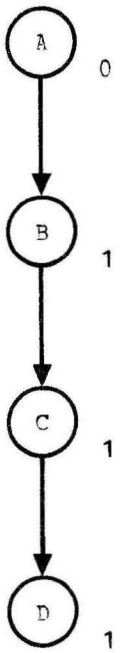
(b) NETID is EXAM1

(c) A, B, C, and D are standard ASP jobs

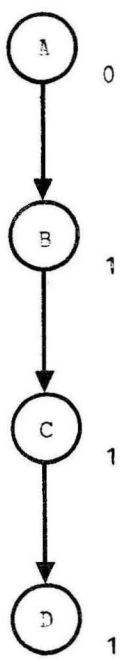
1. Network diagram: EXAM1

From this diagram, the predecessor/successor relationship can be observed.

2. Number of Predecessors:



3. Successor jobnames:

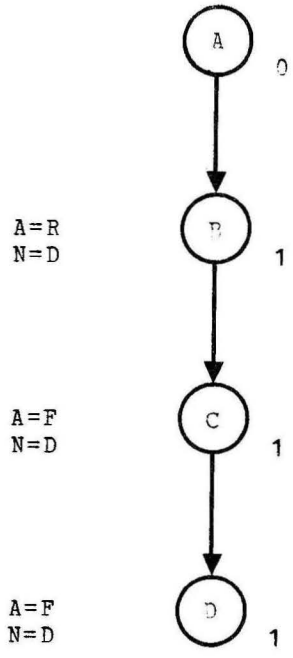


Job B is a successor to A

Job C is a successor to B

Job D is a successor to C

4. Job disposition based on predecessor completion, A=ABNORMAL, N=NORMAL



Note: Defaults: NORMAL=D (decrement) and ABNORMAL=R (retain)

5. All standard jobs

6. //*NET control card definition

JOBNAME	Control Card
A	//*NET NETID=EXAM1,RELEASE=(B)
B	//*NET NETID=EXAM1,RELEASE=(C),NHOLD=1,ABNORMAL=R
C	//*NET NETID=EXAM1,RELEASE=(D),NHOLD=1,ABNORMAL=F
D	//*NET NETID=EXAM1,NHOLD=1

Example 1 comments:

- For Job B, ABNORMAL=R is default and need not be specified.
- If Job B abnormally terminates, then jobs C and D will be flushed from the system by specification of A=F on the //*NET cards. That is, the jobs will be canceled with print.

Example 2: Multiple Successor Jobs

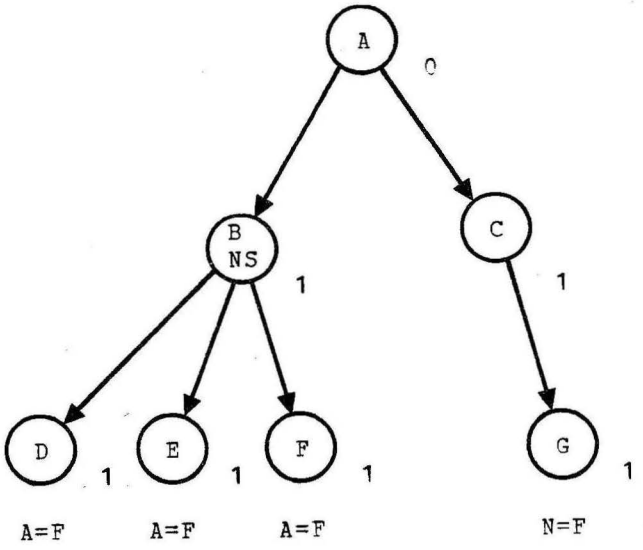
Given:

a) Seven jobs, A through G

b) NETID is EXAM2

c) Job B is a nonstandard job

Network Diagram: EXAM2



//*NET control card definitions:

For Job	Control Card
A	//*NET NETID=EXAM2,RELEASE=(B,C)
B	//*NET NETID=EXAM2,RELEASE=(D,E,F),NHOLD=1
B	//*PROCESS DJC
C	//*NET NETID=EXAM2,RELEASE=(G),NHOLD=1
D	//*NET NETID=EXAM2,NHOLD=1,ABNORMAL=F
E	//*NET NETID=EXAM2,NHOLD=1,ABNORMAL=F
F	//*NET NETID=EXAM2,NHOLD=1,ABNORMAL=F
G	//*NET NETID=EXAM2,NHOLD=1,NORMAL=F

Example 2 comments:

- The ABNORMAL parameter specification for successors to Job B would not be effective since nonstandard jobs are always considered to complete normally.
- Job G will be retained if Job C abnormally terminates.
- Jobs D, E, and F will become eligible for scheduling at the same time.

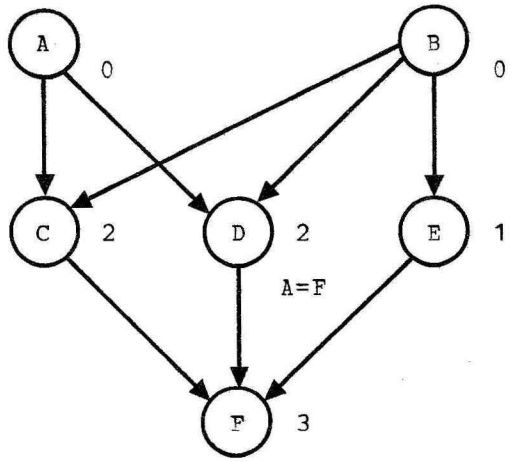
Example 3: Multiple predecessor jobs

Given:

a) Given 6 jobs, A through F

b) NETID is EXAM3

Network Diagram: EXAM3



//*NET control card definitions:

For Job	Control Card
A	//*NET NETID=EXAM3,RELEASE=(C,D)
B	//*NET NETID=EXAM3,RELEASE=(C,D,E)
C	//*NET NETID=EXAM3,RELEASE=(F),NHOLD=2
D	//*NET NETID=EXAM3,RELEASE=(F),NHOLD=2,ABNORMAL=F
E	//*NET NETID=EXAM3,RELEASE=(F),NHOLD=1
F	//*NET NETID=EXAM3,NHOLD=3

Example 3 comments:

- Job F will be made eligible for execution when Jobs C, D, and E have completed Main Processor execution.
- If Job A or B abnormally terminates, then Job D will be flushed from the system which will cause Job F to be flushed. Because Jobs C and E assume default disposition on predecessor completion they would remain in the system if one predecessors completed abnormally. In this situation, a predecessor should be corrected and resubmitted to the system. Once it completes normally, it's successors will be made eligible for scheduling.

Example 4: Complex Network:

Given:

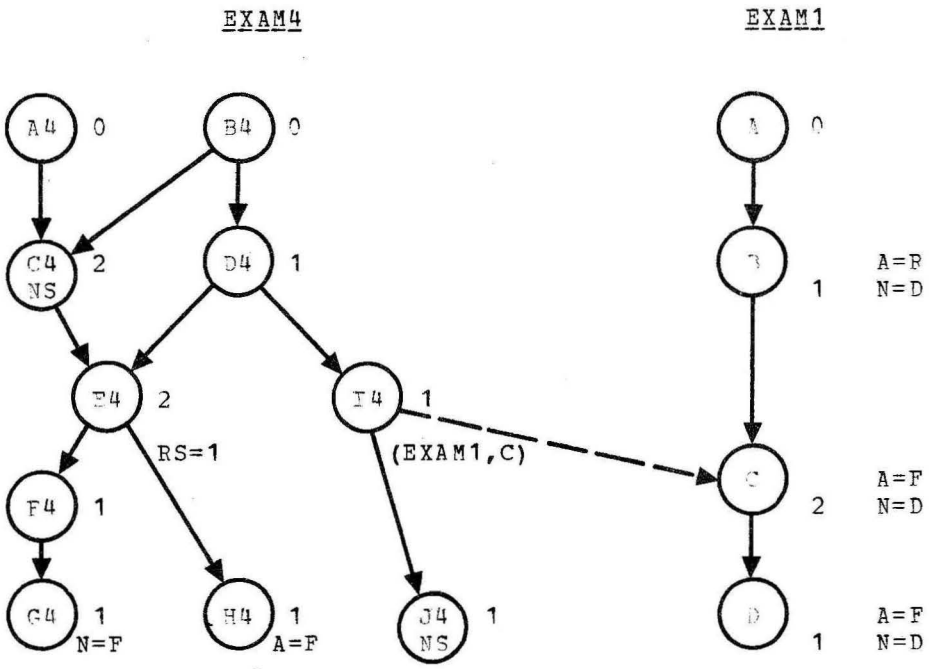
a) Ten jobs, A4 through J4

b) NETID=EXAM4

c) NETREL for JOB I4 is EXAM1 from Example 1, Release jobname is C

d) C4 and J4 are nonstandard jobs

Network Diagram: EXAM4



//*NET control card definition (shorthand notation)

	JOB	Control Card
	A4	//*NET ID=EXAM4,RL=(C4)
	B4	//*NET ID=EXAM4,RL=(C4,D4)
*	C4	//*NET ID=EXAM4,RL=(E4),HC=2
	D4	//*NET ID=EXAM4,RL=(E4,I4),HC=1
	E4	//*NET ID=EXAM4,RL=(F4,H4),HC=2,RS=1
	F4	//*NET ID=EXAM4,RL=(G4),HC=1
	G4	//*NET ID=EXAM4,HC=1,NC=F
	H4	//*NET ID=EXAM4,HC=1,AB=F
	I4	//*NET ID=EXAM4,HC=1,RL=(J4),NR=(EXAM1,C)
*	J4	//*NET ID=EXAM4,HC=1

A	//*NET ID=EXAM1,RL=(B)
B	//*NET ID=EXAM1,RL=(C),HC=1
C	//*NET ID=EXAM1,RL=(D),HC=2,AB=F
D	//*NET ID=EXAM1,HC=1,AB=F

* This job requires a //*PROCESS DJC control card.

Example 4 comments:

- Job E4 has a release schedule count of 1 specified which causes E4 to enter scheduling when its NHOLD count becomes 1. E4 must have no catalog dependencies.
- Job G4 can be assumed to be a cleanup job. That is, if F4 completes normally, then G4 will be flushed from the system. H4 is the converse of G4.

- Job C of EXAM1 has an NHOLD count of 2. It requires completion of I4 and B before it can be scheduled.

CHAPTER 5. USING INTERNAL JOB PROCESSING (IJP)

The ASP system Internal Job Processing (IJP) routines provide an interface to the ASP system during program execution. This generalized interface is composed of:

- An interface subtask called IJPWTR, which is attached by the program that is required to transmit data to the ASP system for processing
- The ASP system IJP Dynamic Support Program that directs the transmitted data to the requested DSP. The program or system requiring IJP may reside on a real or a local ASP Main Processor and may submit data from tape or direct access device for processing.

Some examples of possible IJP usage are:

- A conversational terminal system required to submit job input to ASP for scheduling and processing, such as CALL/OS or CRJE
- A job required to print or punch prior to termination of execution, such as IMS
- A job that builds a JCL input stream and must submit it for processing, such as a user written JCL compiler
- A job that will schedule another selected job upon termination

In essence, IJP can be used to provide a logical extension of the card reader and ASP control card capabilities to the using task.

There are no machine requirements unique to IJP other than those required by ASP and the IJP user task. The OS ATTACH macro which should be used to invoke IJPWTR creates an independent subtask that can wait for I/O, storage, etc., without interfering with the operation of the task that attached it.

Data is input to an ASP DSP during program execution by attaching the IJPWTR module. This module performs the necessary steps to interface to the ASP system. The programming requirements for this use of IJP are:

1. The OS assembler macro ATTACH is used to create a subtask which may execute independently of the originating or attaching task. A parameter list is passed via the ATTACH macro; this list contains:
    - The address (first word in the parameter list) of the six-character volume serial of the disk or tape containing the data set to be processed
    - The address (second word of the parameter list) of the 44-character name of the data set to be processed
    - The address (third word of the parameter list) of the eight-character member name if the data set is partitioned organization; otherwise, the address of an area containing eight blanks

- The address (fourth word of the parameter list) of the eight-character name of the ASP DSP that is to process the data set

- The address (fifth word of the parameter list) of a four-character processing options indicator area - the first of these four characters indicates that the parameters are to be added to a checkpoint data set (C), after which the attaching task may resume execution while the subtask processes the parameters from the checkpoint data set; or (S) indicates serial processing, where the attaching task waits for all subtask services to complete before continuing. The second character indicates that the input is on direct access (D) or on tape (T). The third character, if (S), indicates that a direct access data set to be sent to ASP is to be scratched after being sent to ASP. The fourth character is presently unused. Example options are: C, D, blank, blank (checkpoint, direct access, no-scratch, null).

- The address (sixth word in the parameter list) of a four-character ECB, which is posted by the subtask at the end of serial processing or checkpointing of parameters (initially hexadecimal zero).

- The address (seventh word of the parameter list) of an eight-byte word aligned work area (initially hexadecimal zero) used by the IJPWTR subtask to determine whether to continue processing or to cease processing and return. The attaching task indicates cease processing to the IJPWTR subtask by moving hexadecimal '0F' to the first byte of the doubleword. The attaching task may then determine when the IJPWTR subtask has completed by examining the TCBLTC field in the attaching task's TCB. When this field is zero, no subtasks are outstanding.

2. The additional Job Control Language

requirements for the job that attaches IJPWTR are:

- A DD card of the format //IJPWTR DD SYSOUT=(class), where class is a class defined as TYPE=DSISO on an ASP SYSOUT card in the ASP initialization deck (see ASP System Programmer's Manual, SYSOUT card).

- If the input is on tape, a DD card with the ddname IJPTAPE, describing the standard labeled tape containing the data set to be processed

- If the input is on direct access, one or more DD cards with ddname of the format IJPDAxxx, where xxx is of the user's choosing. This DD card must reference the volume on which the information to be submitted is contained.

Note: No other DD names of this format should be specified. Also, if a cataloged procedure is used to execute the program, the DD cards must be included in the procedure, not appended to it.

- If the checkpoint (C) option is desired, a checkpoint data set is created initially with null (all EBCDIC zero) 80-character entries. This data set is used by the IJPWTR program to save input parameters. A DD entry describing this checkpoint data set must be included with DCB subparameters LRECL and BLKSIZE equal to 80 (card image) with ddname CKPTDD and RECFM=F.

3. A sample usage of ATTACH is:

```
                LA      1,IJPWPLST      LD R1 ADDR PARM LIST
                ATTACH  MF=(E,(1)),EP=IJPWTR
                WAIT    ECB=IJPECB      WAIT FOR SUBTASK POST
                etc.

IJPVOL          DC      CL6'VOLSER'
IJPDSN          DC      CL44'FULLY.QUALIFID.DATA.SET.NAME'
IJPMEM          DC      CL8'MEMBNAME'
IJPDSP          DC      CL8'ASPDSPNM'
IJPOPT          DC      CL4'CD'
IJPECB          DC      F'0'
IJPWORK         DC      2F'0'
IJPWPLST        DS      0F
                DC      A(IJPVOL)
                DC      A(IJPDSN)
                DC      A(IJPMEM)
                DC      A(IJPDSP)
                DC      A(IJPOPT)
                DC      A(IJPECB)
                DC      A(IJPWORK)
```

Note: When coding the ATTACH macro the parameters must appear in the order given.

In addition to the previously mentioned considerations, the ASP IJP DSP must be active for the logical Main Processor from which the input will be sent. Figure 4 shows an overview of the IJP/ASP interface.

The following restrictions apply in the use of IJPWTR:

1. Input data to IJPWTR to be sent to ASP must not have RECFM=U or LRECL greater than ASP buffer size minus 24 bytes.

2. Input data to IJPWTR to be sent to ASP for processing by the ISDRVR or PUNCH DSPs must have record lengths of either 80 or 81 characters. Input for other DSPs must have record lengths less than 252.

3. A given job must wait for an existing IJPWTR subtask to complete prior to attaching a subsequent IJPWTR.

4. Multiple jobs with the same jobname that attach IJPWTR may not be run concurrently.

ECB completion codes returned by IJPWTR subtask:

- x'04' - No more room exists in checkpoint data set for further entries.
  Action: Re-Attach using serial (S) option.

- x'08' - Permanent errors processing checkpoint dataset.
  Action: Contact system programmer to correct/save checkpoint dataset or re-Attach using serial (S) option.

- x'0C' - I/O errors while processing on the channel-to-channel adapter.
  Action: Contact system programmer or try again.

- x'10' - I/O errors while processing Input dataset specified by Attaching task.

Action: Contact system programmer to correct or recreate the dataset.

- x'14' - Processing option of TAPE (T) was specified but no DD name of IJPTAPE was found.
  Action: Correct JCL and resubmit JOB.

THIS PAGE INTENTIONALLY LEFT BLANK

- x'18' - Processing option of DISK (D) was specified and either no DD entry of IJPDAxxx was found or the requested VOLSER was not mounted. Action: Correct JCL and resubmit.
- x'20' - The DISK dataset specified was not found. Action: Create dataset or correct JCL and resubmit.

Information regarding messages issued by IJPWTR (format IJPWXX) is contained in the ASP Version 3 Messages and Codes Manual, GH20-1290.

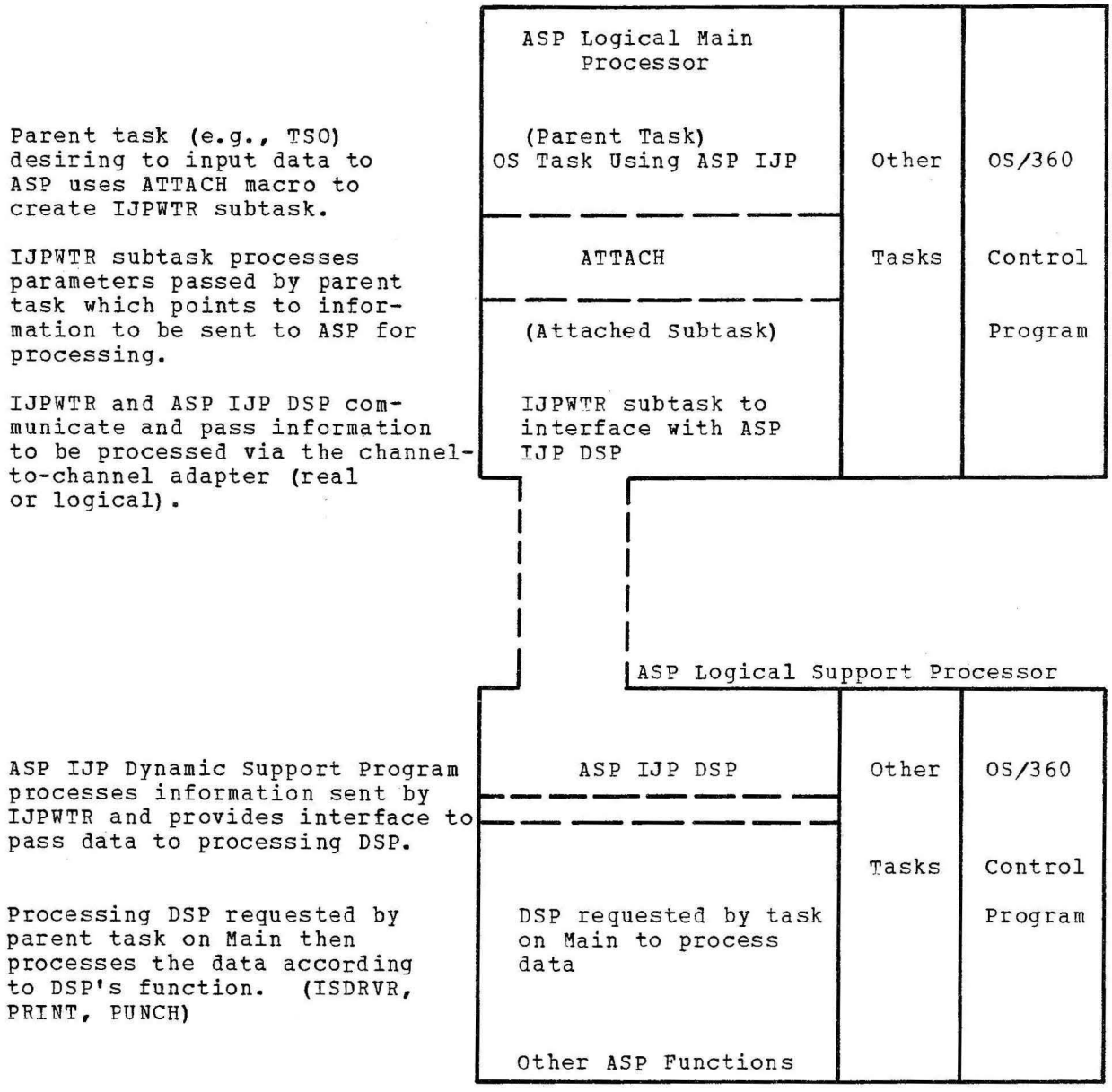


Figure 4. IJP ASP Overview.

Submitting Created Jobs via ISDRVR

If a requirement exists merely to submit a created job stream to the ASP system after job execution is completed, scheduling the ISDRVR DSP through use of the //*PROCESS control card will accomplish this. For example:

```
  //AJOB        JOB 836,JONES,PRTY=2,REGION=50K
  //*PROCESS    RICONTL
  //*PROCESS    MAIN
  //*PROCESS    ISDRVR

MYJOBIN

  //*PROCESS    PRINT
  //*PROCESS    PUNCH
  //*STEP1      EXEC PGM=MYPGM
  //MYJOBIN     DD UNIT=(CTC,,DEFER)
  etc.
```

The above example illustrates a job which will create one or more other jobs to be sumitted to the ASP Input Service routines for processing after termination of execution on the Main Processor. The name on the parameter card following the //*PROCESS card reflects the name of the Job Data Set (JDS) entry that contains this job or jobs to be processed by ISDRVR. ISDRVR is the same module that normally processes jobs submitted from card readers, tape readers, or disk readers.

CHAPTER 6. USING NETWORK JOB PROCESSING (NJP)

Network Job Processing permits two or more ASP Support Processors to schedule and route ASP jobs from one Support Processor to another via communication lines. The programmer may schedule jobs to be executed at a remote ASP installation by using nonstandard job processing with //*PROCESS cards. A job with standard processing (one without //* PROCESS cards) may be scheduled for remote processing via NJP by the ASP operator. NJP may not be used on Dependent Job Control (DJC) jobs.

Scheduling a job to be run remotely via NJP requires the following cards:

```
//*PROCESS  NJPIO

//*FORMAT   NJP,DEST=destination-terminal,FROM=source-terminal
```

The destination and source terminal names can be derived from the NJPTERM initialization control cards.

Jobs that are scheduled for NJP by the //*PROCESS card may execute and complete all functions on the destined system, or specified functions may be selected. NJP scheduling is requested by specifying //*PROCESS NJPIO. If the job is to be transmitted by the ASP operator, all functions of that job will be completed on the destined system.

The following is an example of a job being sent from NJP local terminal LAX to a remote ASP system terminal called NYORK. The job is to execute on the Main Processor at NYORK and then return to LAX for printing and punching:

```
//AJOB      JOB 836,ASP,PRTY=14
//*PROCESS  NJPIO
//*FORMAT   NJP,DEST=NYORK,FROM=LAX
//*PROCESS  RICONTL
//*PROCESS  MAIN
//*PROCESS  NJPIO
//*FORMAT   NJP,DEST=LAX,FROM=NYORK
//*PROCESS  PRINT
//*PROCESS  PUNCH
    .
    .
    .
OS job stream
/*
```

Whatever job steps are required at the remote location should be included between the //*PROCESS NJPIO card sending and the //*PROCESS NJPIO card returning the job. However, it is not necessary to provide a //*PROCESS NJPIO card to return a job if no functions are required locally. A job may perform the same functions at each location or in any combination of locations. This is accomplished by means of the //*PROCESS card.

The following is an example of a job that is to run and execute on a Main Processor, print and punch locally at LAX, and print and punch at NYORK.

```
//AJOB      JOB 836,ASP,PRTY=10
//*PROCESS  RICONTL
//*PROCESS  MAIN
//*PROCESS  PRINT
//*PROCESS  PUNCH
//*PROCESS  NJPIO
//*FORMAT   NJP,DEST=NYORK,FROM=LAX
//*PROCESS  PRINT
//*PROCESS  PUNCH
    •
    •
    •
OS job stream
/*
```

This job needs no NJP scheduling to return it to LAX since no further processing is required there.

With NJP, a group or class of jobs may be transmitted by using the NJPCLASS parameter on the //*MAIN control card. This is a convenient method of identifying a group of jobs eligible to run on a given remote ASP system.

CHAPTER 7. USING PRE-EXECUTION SETUP OF TAPE AND DISK VOLUMES

ASP provides the installation with many tools to help maximize system throughput. One such tool is pre-execution setup. This feature allows pre-mounting of none or all non-resident volumes a job may require before CPU resources have been allocated to the job. In doing this the job cost is reduced by lowering the job's execution time and system throughput is improved by having all resources available prior to occupying main storage.

Three methods are available to cause ASP to do pre-execution setup, they are:

1. Submit the job unmodified - ASP will examine the job's total volume requirements by examining the cards and will allocate the number of tape and disk units required. At the appropriate time ASP will request mounting of the necessary volumes and will verify that mounting was done correctly. In the case of the UNIT=(,,DEFER) parameter ASP will allocate a device but no mount message will be issued until the data set is opened by OS.

   Using this method, if step 1 of the job requires two tape drives and step 2 requires two different tape volumes, ASP will request four tape units to be allocated to this job for the duration of this job on the Main Processor. In this case no operator intervention would be required (other than possible multi-reel changes) during the entire job execution. However, four units were used when two units would have been sufficient to execute the job. The trade-off must be evaluated between number of units allocated of time required for operator intervention. Some factors to consider are: length of job steps, total units available, critical scheduling nature of the job, etc. If it is desirable to utilize fewer units, with the inherent delay waiting for operator mounts, use one of the two methods below.

2. Utilize the SETUP=(ddname,ddname -----) on the //*MAIN control card - The use of this parameter will limit the ASP determined SETUP requirements. Using the same example as in 1: if only two tape units were wanted throughout, the first two fully qualified DDNAMEs should be used. This method will assign two drives to this job for the duration of the job execution, the initial volume mounting will be requested and verified by ASP. Subsequent volume mounts will be requested directly by OS. When this method is used, care must be taken to specify an adequate number of tapes for the maximum job step. If this is not done the job will enter allocation recovery and will be automatically canceled by ASP. The ability to specify fewer devices than required in total applies only to tape; all disk references must be processed by setup.

3. If your installation has included the implicit 'High Watermark' exit within the Reader/Interpreter interface, you may submit your job unmodified. Implicit high watermark setup refers to a setup algorithm which will allocate the minimum number of tape units for a job. For example if job step 1 requires two tape units and job step 2 requires 2 tape units of the same type but no longer needs the tape units of step 1, only 2 tape units will be allocated to the job. Your installations's exit slhould dynamically provide the effect of method 2 above without requiring any external parameters to be supplied by you. Consult

the system programming staff to determine if the implicit high watermark exit is being used.

Note: Job Scheduling Consideration - Demand allocation (device requests by unit address) must be accompanied by a //*MAIN card directing the job to a specific Main Processor.

CHAPTER 8. USING REMOTE JOB PROCESSING (RJP)

Jobs may be submitted to ASP for processing from remote binary synchronous workstations via RJP. Any job submitted from a remote workstation will by default have its output (print and punch) returned to the originating workstation unless ASP has been instructed to do otherwise via //*FORMAT cards. The remote user has almost all the capabilities of the local ASP user with the restriction that Data Mode 2 input and output may not be used. In addition printer overflow specifications may not be uniquely specified by the application programmer.

CHAPTER 9. USING ASP/TSO SUPPORT

ASP/TSO support extends the advantages of ASP multiprocessing to the TSO user. This support encompasses the uniprocessor environment in addition to the multisystem environment.

SUBMITTING JOBS VIA TSO

Background jobs submitted for execution by a TSO terminal user will be temporarily placed in the OS job queue. The job will then be read by an ASP interface routine and sent to ASP for execution scheduling.

The processing of jobs submitted to TSO and scheduled by ASP can be transparent to the terminal user or the user may take advantage of ASP's unique capabilities.

The only requirement for a terminal user is to include in his JCL an ASP ccntrol card, //*FORMAT AC, which defines the job output to which the user desires access or to specify a DD card with SYSOUT=class, where class is defined in the ASP initialization deck SYSOUT card with TYPE=TSO. The SYSMSG dataset output may be obtained by specifying MSGCLASS=class, as described above in the JOB card JCL statement. Refer to the ASP Control Card section for an explanation and examples of the //*FORMAT AC card.

An option of selecting a specific Main Processor in the ASP complex is available via the ASP control card //*MAIN. Submitting a //*MAIN card gives the user the ability to select a computer for its uniqueness, that is, computing characteristic, size, I/O, etc.

JOB OUTPUT AVAILABILITY

The desired output data sets are cataloged on the TSO system and the terminal user is notified of this when he logs on the system. The data set is then accessible via the TSO EDIT command. To define the data set printed at the terminal, the //*FORMAT AC,DDNAME=.... control card is used.

A terminal user may elect to have his data set printed on an ASP defined printer located at his site rather than on his terminal. This feature is available by using the card:

//*FORMAT PR,DDNAME=ddname,DEST=printer-name.

When using the //*FORMAT PR card the output will go directly to the printer, the data set will not be cataloged, nor will the terminal user be notified by ASP as to its availability.

The user may elect to have some data sets sent to a printer at his location and still have data sets printed on the terminal by using both //*FORMAT PR and //*FORMAT AC control cards for the same data set.

SUBMITTING JOBS DESTINED FOR TSO TERMINAL USERS VIA LOCAL OR REMOTE ASP READERS

Jobs may be submitted through a local or remote ASP reader, with selected output sent to a TSO terminal user. Using this facility requires the use of the //*MAIN control card and the USER parameter of

the //*FORMAT AC control card. This type of job is a nonstandard job and must use //*PROCESS cards.

The //*MAIN ACMAIN=system-name must be used to define the Main Processor where TSO resides. To identify the specific terminal user to receive the output, //*FORMAT AC,DDNAME=ddname,USER=userid must be specified.

Sending job output to an ASP defined RJP printer at the terminal user's site will require the following parameters:

```
//*FORMAT PR,DDNAME=ddname,DEST=remote-printer-name
```

The printer-name must be known to ASP.

```
Example:   //EXAM      JOB MSGLEVEL=(1,1)
           //*MAIN     ACMAIN=SY1,ACHOLD=YES
           //*PROCESS  RICONTL
           //*PROCESS  MAIN
           //*PROCESS  ACDS
           //*FORMAT   AC,DDNAME=SYSMSG,USER=TERM1
           //*PROCESS  PRINT
           //*FORMAT   PR,DDNAME=SYSPRINT,DEST=RM001PR1
           //STEP      EXEC PGM=X
```

When program X completes execution on any Main Processor in the ASP system and ACDS has acted upon the //*FORMAT AC card for the system message data set the job will be placed in a terminal HOLD status. The terminal user whose ID is TERM1 on system named SY1 will be notified that the SYSMSG data set has been cataloged on his system.

The terminal user can access the SYSMSG data set via the TSO EDIT command. After examining the data set he may cancel or release the job. If released, ASP Print Service will print the SYSPRINT data set on the printer named RM001PR1.

TSO TERMINAL USER COMMANDS AND MESSAGES

CANCEL {jobname|(jobname,...)}

CANCEL will cause the job to be terminated.

EDIT jobname.ddname DATA

Edit will cause the specified data set (ddname) to be printed on the terminal. The user is to extract only the jobname and ddname from the data set name given in the ADS19 message, TSO will create the fully qualified data set name; that is, userid.jobname.ddname.DATA.

OUTPUT

This command cannot be used. TSO has been modified to reject the command. The job output will be processed as defined by the user or by the installation defined default values for output.

STATUS {jobname|(jobname,jobname)}

STATUS is used to determine the status of a job and its output. Status may reference one or more jobs per submission. The ASP response will be identical to the response to a standard ASP inquiry response command.

SUBMIT {data-set-name|(data-set-name,data-set-name)}

SUBMIT is used to present a background for job or jobs to TSO according to standard TSO methods.

HOLD= {jobname|(jobname,jobname)}

HOLD will place a job or jobs in TSO hold status. The ACDS DSP will process the job but it will not allow Print to execute until the RELEASE command is issued.

RELEASE= {jobname|(jobname,jobname)}

RELEASE allows continuation of processing for one or more jobs which have been placed in hold status.

CHAPTER 10. USING TAPE-TO-PRINT PROCESSING (TP)

The ASP Tape-to-Print (TP) DSP may be used to print output tape volumes under control of the operator or the //*PROCESS card. When a //*PROCESS card is used, parameter cards must follow the //*PROCESS TP card to indicate what is to be printed and how it is to be handled.

In addition, special control records may be included in the print data file which requests forms and carriage tape changes to be made by the operator. The location of this record is the application programmer's responsibility. Under normal circumstances the record should be first on the data file.

The format of this special control record is:

Col.	Data	Explanation
1	.(period)	Identifies this as a control card
2-33	report-name	This names the report that follows and is printed in the operator message (TPR015).
35-42	forms-name	The name of the forms to be mounted.
44-51	carriage tape name	The name of the carriage tape to be mounted.

All parameters which can be specified on the operator call of TP may be specified on the parameter cards for TP (see Chapter 8 in the Operator's Manual). The operator will be notified of printer and tape mounting requirements and TP will remain in control until all data sets are printed.

The following is an example of a job which creates one standard label print tape to be processed after the main program's execution.

```
//A  JOB    836,ASP,PRTY=8
//*PROCESS  RICONTL
//*PROCESS  MAIN
//*PROCESS  PRINT
//*PROCESS  PUNCH
//*PROCESS  TP
LB=S,V=ASP001,NAV=R
//STEP1  EXEC  ANYPROG
//DDOUT      DD  UNIT=TAPE9,DSN=REPORTA,VOL=SER=ASP001,LABEL=(,SL),
                DCB=(RECFM=VBA,LRECL=137,BLKSIZE=8000),DISP=(NEW,KEEP)
----------other JCL----------
/*
```

In the above example, standard labels have been specified (LB=S), volume name of ASP001 (VOL=ASP001) is specified. If devices are unavailable, the job is to be retained (NAV=R). TP has defaulted to a 9-track tape drive (TA9) because this was not explicitly specified. Therefore, after the job has completed its main processing, TP will be scheduled asynchronously with PRINT and PUNCH; and, if either a 9-track tape drive or a print device is unavailable, TP will be rescheduled to await device availability. When the devices become available a message is issued to the operator requesting that tape volume ASP001 be mounted. This is followed by a request for the proper forms to be mounted. When the forms are set up, the operator issues *S TP and the tape begins to print.

CHAPTER 11. ASP OUTPUT

The OS peripheral output, which is processed via Punch Service and Print Service, appears in either card image or line image form as specified by the programmer. To assist both the programmer and the operator to identify the various components of output, the ASP system adds appropriate identification information to the data supplied by the program.

For punch data, each data set is identified by header and trailer cards. These cards contain the job number in columns 21 through 24, the job name in columns 35 through 42, and the ddname in columns 53 through 60. The remaining columns contain punches in the three rows at the top and bottom of the card for ready identification of the separator card. This card also separates multiple copies of a given data set. The format of the card is illustrated in Figure 5.

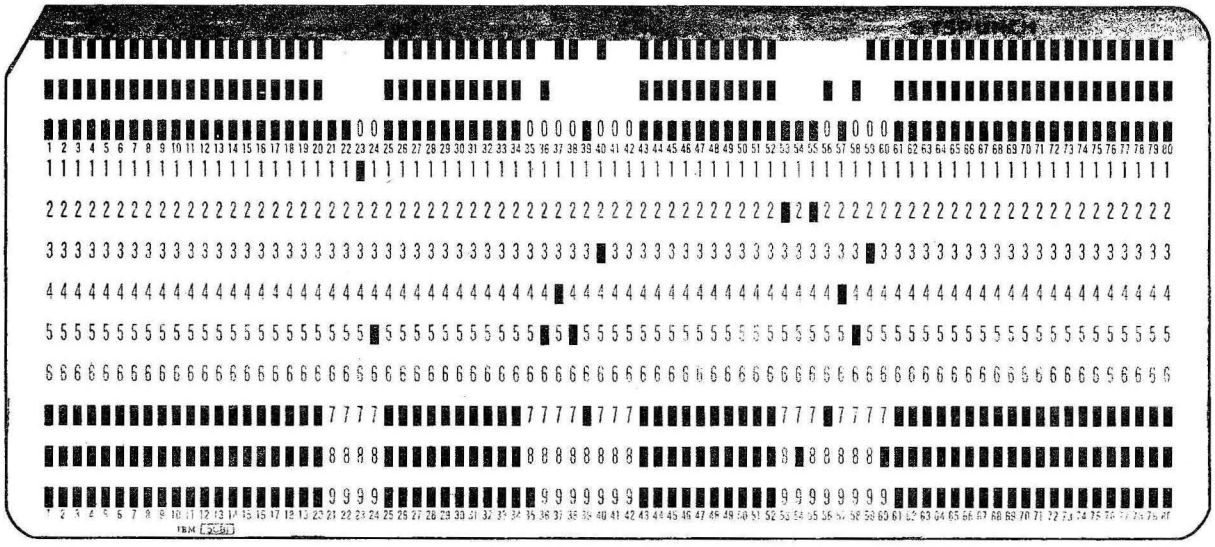


Figure 5. Separator Card.

Printed output is identified by a header page for each data set and a separator page after the last data set on each printer used by the job. The header page contains the job name, job number, and ddname in blocked letters. The trailer page is identified by date and job number, and a copy of the JOB card is printed. The last output on the last printer used is followed by a trailer page; this page has a tabulation of all data sets printed and which contains the job statistics. To assist the ASP operator in collating printed output with punched output for a job, the trailer page indicates how much punched output is produced for this job. The trailer page is illustrated in Figure 6.

Note: An option is provided at ASP Initialization time to suppress burst pages and/or header pages in which case no headers or burst pages will be produced.

```
**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************


       ASP JOB NO. = 0002                                Date = 72.032


//JOBA     JOB     836,SMITH,PRTY=07,MSGLEVEL=1


       ELAPSED TIME ON MAIN PROCESSOR = 000.05, START TIME = 16.35.22

       DDNAME = SYSMSG     PRINTED ON PR1  , LINES = 000024
       DDNAME = SYSOUT     PRINTED ON PR1  , LINES = 001102
       DDNAME = DATASET1   PRINTED ON PR1  , LINES = 000005
       DDNAME = DATASET2   PRINTED ON PR1  , LINES = 000005
       DDNAME = DATASET3   PRINTED ON PR1  , LINES = 000005
       DDNAME = DATASET4   PRINTED ON PR1  , LINES = 000005
       DDNAME = DATASET5   PRINTED ON PR1  , LINES = 000005
       DDNAME = DATASET6   PRINTED ON PR1  , LINES = 000005
       LINES OUTPUT FOR THIS JOB = 001156

       CARD OUTPUT FOR THIS JOB=000100



**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
```

Figure 6. Trailer Page.

Write-to-Operator (WTO/WTOR) messages are placed in the SYSMSG data set of each related job. The messages are listed following the submitted JCL image and before the OS system messages. They are listed in the order they occur, including any replies. Certain OS Control Program messages related to a job are also included. The job name is inserted as a prefix for each WTO/WTOR; if message is not job related, the eight-byte prefix is blank.

Multiple Line Write-to-Operator (MLWTO) messages are routed in the system without modification; that is, the job name prefix will not be appended and the messages will not appear in SYSMSG.

APPENDIX A: DYNAMIC SUPPORT PROGRAM NAMES

The list below contains those Dynamic Support Program (DSP) names for the distributed ASP system that the application programmer may wish to place in //*PROCESS cards. Other DSPs in the ASP system may be called by //*PROCESS cards; however, since they interact with the operator and operations, it would not be normal to do so.

DSP NAME	DESCRIPTION
ACDS	ASP/TSO Support
CC	*Card-to-Card
CP	*Card-to-Printer
CT	*Card-to-Tape
DJC	Dependent Job Control
ISDRVR	Input Service
MAIN	Main Service
NJPIO	Network Job Processing
PRINT	Print Service
PUNCH	Punch Service
RICONTL	Reader/Interpreter
TC	*Tape-to-Card
TD	*Tape Dump
TP	*Tape-to-Printer
TT	*Tape-to-Tape

* Callable DSP's requiring operator action. For more information on callable DSP's refer to the ASP Operator's Manual.

GLOSSARY

Application Programs: Programs that are run on the Main Processor under control of OS. These include such applications as compilations, assemblies, production jobs, and so forth.

ASP. Asymmetric Multiprocessing System - the ASP acronym is based on the now obsolete "Attached Support Processor". These initials have been retained in documentation for continuity purposes.

ASP Control Card. These are introduced into the system along with OS JCL. They communicate to ASP what processes are to be performed with the jobs. The cards are transparent to the operating system.

ASP Input/Output Routines. A resident section of the ASP system on the Support Processor that manages the flow of data between the Support Processor and the ASP queue devices.

Asymmetric Loosely Coupled Multiprocessing. Processing of data shared by two or more computing systems interconnected by an I/O channel-to-channel adapter. The CPUs may be of different types and have their own unique configurations.

Background Job. A job, usually of a utility type, run by ASP on the Support Processor in the ASP region independent of the local or real Main Processor.

Card Image Form. Column binary or data-mode 2 format.

Chained Scheduling. A technique whereby the system bypasses normal I/O routines and dynamically chains several input/output operations together. For further information see OS Data Management Guide. (GC 26-3746).

Console Service. A resident section of the ASP system on the Support Processor which accepts and transmits messages. This support is not dependent on OS MCS.

DATASET Card. ASP control card which enables the introduction of additional input data anytime during a job's life. It is placed at the head of the intended input data and must be terminated by an ENDDATASET card.

Dynamic Support Program. A program, residing on the Support Processor system residence device, that is known to the ASP system by an entry in the Dynamic Support Program Dictionary. The program is executed when a job segment pointing to the Dynamic Support Program is scheduled by the Job Segment Scheduler.

Failsoft. A facility whereby ASP System or Dynamic Support Program failures do not generally halt unaffected jobs. An attempt is made to continue running in a degraded mode.

Format Card. ASP control card which specifies special instructions to the system concerning the print and punch data set processing.

Hot Job. A job designated by the //*MAIN control card as one that should continue executing in the event of an ASP system abend.

Input Service. A set of Dynamic Support Programs that read the input data and build the system input data set and control table entries for each job.

Interleaving. The alternating of transmission input and output on a single communications facility. This facility enables a terminal to read input and to print and punch output simultaneously.

Internal Job Processing (IJP). A facility which allows jobs or output to be submitted to ASP directly from an application program such as CRJE executing under control of ASP.

Job. One or more job segments defined to the ASP system by control cards. (The first card of a job is the JOB card, and the job definition is terminated by another JOB card or by an end-of-file indicator.) The standard ASP job is composed of job segments to:

- Interpret the job's job control information
- Transmit the input data to the Main Processor for execution (Main Service)
- Print the output (Print Service)
- Punch the output (Punch Service)
- Purge the job from the system

Job Control Table (JCT). A table which ASP utilizes in controlling the job.

Job Segment. A logical portion of a job. When the input data is read by Input Service, a Scheduler Element is created for each job segment. The Scheduler Element points to a Dynamic Support Program, which performs the required work.

Job Segment Scheduler. A section of the ASP nucleus that initiates the processing of job segments.

Logical Region. The amount of real storage that is required by a job or job step to execute efficiently. This real storage requirement, which is normally less than the OS region request on a JOB or EXEC card, is referred to as the job's working set.

MAIN Card. ASP control card which communicates main processor requirements to the system.

Main Device Scheduler (MDS). A section of the ASP system that schedules the allocation of devices prior to scheduling problem program execution.

Main Processor. A component of an ASP system comprising a central processing unit and input/output devices. This processor is devoted to the execution of application programs.

Main Service. A Dynamic Support Program that schedules the problem program for execution and manages the flow of data (system input, print, and punch) across the channel-to-channel adapter to and from the Main Processor.

Multifunction Monitor. A section of the ASP nucleus that passes control between the functions within the Support Processor.

NET Card. ASP control card which communicates dependent job control relationships to the system.

Network Job Processing (NJP). A facility that permits the input, processing, and output of jobs to and from compatible remotely located ASP installations.

Nonsetup Padding. The facility to schedule jobs that require no pre-execution setup for execution on Main Processors while processing the setup jobs on the Support Processor.

Nonstandard Job. A job with a different number of steps or a different order of steps from a standard ASP job. These steps must be specified via ASP control cards as opposed to the standard job which does not require the submission of any special cards.

Postprocessing. A phase of ASP processing comprised by the job segments which punch and print job output and a segment which purges the job from the system.

Preprocessing. A phase of processing in ASP where the job is read and interpreted for the system.

Print Service. A Dynamic Support Program that prints the data sets created by the Main Processor.

Process Card. An ASP control card which invokes a specified job segment to be performed. They are necessary only in nonstandard jobs.

Processing. This is the phase of ASP where actual job execution takes place.

Punch Service. A Dynamic Support Program that punches the data sets created by the Main Processor.

Purge. A Dynamic Support Program that deletes the control table entries and data for each job in the system when the job is completed. It may contain the system accounting routine.

Reader/Interpreter. The job segment of ASP which reads and interprets OS JCL for the system.

Remote Job Processing (RJP). A facility that permits the input, processing, and output of jobs to and from terminals remote from the ASP installation.

Resident Programs. Service programs that are common to ASP support functions. These programs reside in storage throughout ASP execution.

Standard Job. An ASP job which is composed of the following processes:

1. Reader/Interpreter
2. Main processing
3. Print
4. Punch
5. Purge

This is standard to the ASP system and no ASP control cards need be used to specify the functions performed.

Scheduler Elements. A synonym for the job segments which make up ASP processing.

Support Processor. The dominant processor of the ASP system in terms of processing control. Its purpose is to control job flow and system input, printing, and punching, to service remote terminals and other ASP installations, and to perform such background services as tape-to-printer or card-to-tape.

INDEX

THIS PAGE INTENTIONALLY LEFT BLANK

IBM Technical Newsletter

This Newsletter No. GN20-9095
Date September 1, 1975

Base Publication No. GH20-1291-1
File No.

Previous Newsletters None

IBM System/360 and System/370
ASP Version 3 Asymmetric Multiprocessing System
Applications Programmers Manual

This Technical Newsletter, a part of Version 3, Modification Level 2, of ASP Asymmetric Multiprocessing System (360A-CX-15X), provides replacement pages for the subject manual. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are listed below:

15-16, 16.1
21-24
27-28, 28.1
29-30
33-36
41-42, 42.1
53-54, 54.1
59-60
65-66
73-75

A vertical rule in the left margin indicates a change.

Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Dept. J74, 2625 Townsgate Road, Westlake Village, CA 91361

Printed in U.S.A.

READER'S COMMENT FORM

GH20-1291-1

IBM System/360 and System/370

Asymmetric Multiprocessing System (ASP)

Version 3, Application Programmer's Manual

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

fold fold

fold fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A. FOLD ON TWO LINES, STAPLE AND MAIL.

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail

No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department J74
1930 Century Park West
Los Angeles, California 90067

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

IBM S/360 and S/370 (ASP) Version 3 APM Printed in U. S. A. GH20-1291-1