GCOS-8 SOFTWARE DESIGN SPECIFICATION

GCOS 8 ARCHITECTURE

Issued by:_____

OS Senior Staff

Issue Date: March 31, 1980
Revision Date: March 31, 1980

Steve Russell
Bruce Carlson

CONTENTS

Page

Section 5
USAGE SCENARIOS

Section 6
ACCOMMODATION AND MIGRATION

SECTION 1

INTRODUCTION

The primary purpose of this specification is to provide an architectural model which is the focal point for the evolution of GCOS 8. It is not a design which can be achieved in a single release, but rather is a common goal which provides direction for the development and evolution of GCOS 8 products such that they operate in a harmonious and consistent manner.

The model presented supports the Distributed System Architecture for a GCOS 8 System operating as a node in a distributed network, as well as the multicomputer capabilities for local networks.

This architecture includes a high-level description of the system views or personalities presented to the end-user (e.g., timesharing, batch, transaction processing, administration), as well as the interactions between them. The accommodation of GCOS 3 users and the migration of these users to the native GCOS 8 environment is also addressed.

SECTION 2

GOALS AND IMPLICATIONS

In the process of arriving at an architectural model for Gcos 8, it became apparent that certain goals have a major impact on design decisions. The following list includes those goals that most significantly shape the architectural model.

1.0 Support of the Distributed Systems Architecture (DSA)

Of all the goals, the support of DSA is the most pervasive. The DSA concepts of workstations, mailboxes, and processes are used not only to define a distributed network, but are also the basic building blocks on which the GCOS 8 architecture rests.

Two DSA concepts which forced major design decisions were:

- Mailboxes are associated with workstations.
- Mailboxes of a workstation are accessible to any process at that workstation.

These two DSA concepts, together with a desire to protect one user from another tend to dictate a model for both timesharing and batch. Refer to section 5.2 and 5.3 for further explanation.

One capability not addressed by DSA was found to be absolutely essential to all of the architectural models considered. As commands arrive at a workstation, it becomes necessary to dynamically create and delete mailboxes and also to associate mailboxes with that existing, executing workstation. This is true because the commands cause the execution of programs that utilize Session Control and, therefore, require the use of specific mailboxes.

2.0 Support_Multicomputer_Capability

Multicomputer capability supports a collection of independent, but tightly coupled computers that give the appearance of a single computer. The multiple computers form a local network that uses DSA interfaces for communication across a high speed link connecting the computers.

The effect of this goal on the design is that certain system functions may be distributed in the local network and a Session Control interface is required for communication rather than the normal CALL interface.

3.0 Accommodate GCOS III Batch, ISS, IDS, and DM-IV Users

GCOS 8 must provide the capabilities to execute GCOS III programs in accommodation mode, as well as to support GCOS III JCL. The migration of users from these GCOS III environments to the native GCOS 8 environment must be as "painless" as possible.

4.0 Support a Uniform Environment

Although GCOS 8 does provide many different personalities or user views (e.g., Timesharing, Batch, TP, Database Query, etc.), the great preponderance of GCOS 8 software is independent of the particular view being presented to a user. A minimal amount of software, usually only the command executive, should be dependent on a particular view.

One implication of this goal is that the batch and timesharing environments become essentially identical, with the exception of scheduling and priority. For example, native batch and timesharing should use the same JCL.

Another implication of this goal is that shared software should be totally independent of the location of the input and output streams for a given user. A common interface will be provided that will accept input commands from either a terminal, via Session Control, or a file. In the same way a common interface is used for output regardless of the destination.

Every effort will be made to provide a uniform environment across the various views of the system presented to the user.

SECTION 3

DSA CONCEPTS

(1) The fundamental architecture of distributed computerized workstations provides the programmer and end user the view of a logical network which is independent of the physical topology of the network of computers.

The cornerstone of the architecture is the workstation which is defined to be a location where some well defined set of functions is executed. A workstation is wholly contained within one computer. Workstations have associated with them processes, tenants, mailboxes and other resources.

The topology of the set of interconnected workstations is the logical view of the network; the physical view is the manner in which the actual computers are interconnected.

The programmer and end user see a network of workstations interconnected by logical connections which are created and destroyed dynamically as needed. (See figure 3-2.a.) The end user in the form of a terminal operator may request some work to be done which requires the cooperation of several workstations (for instance the update of several databases which are in different locations). To do this work the end user is represented in each one of these cooperating workstations by a datastructure called a tenant. The tenant structures act as agents for the end user; they contain the minimum amount of information which needs to be kept at all time for this end user.

The tenants must exchange information among themselves in order to execute the end user work. This is done by exchanging message records via sessions which are logical paths connecting two tenants. (See figure 3-2.b.).

Workstations have one or more associated mailboxes. The mailbox is the mechanism by which a workstation is

---

(1) This section is included verbatim from:
GCOS_8:_Distributed_Database_Access; Georges Colliat

addressed. Logical connections actually connect two mailboxes and may contain several sessions between different tenants of the two workstations.

workstation

logical
connection

network of workstations
interconnected by logical connections

figure 3-2.a

tenant

session

tenant

terminal

network of tenants
located in workstations

figure 3-2.b

sessions which use identical
protocols may share the same
logical connection

figure 3-2.c

tenant

mailbox

workstations are addres
via their mailboxes. L
connections are betwee
mailboxes.

figure 3-2.d

In order for a tenant to do some work it must be associated with a process, which is the basic entity allocated to a processor by the operating system. The structures which compose a process are quite large , 16,000 bytes or more. If a very large number of end users are connected to the system, the association of a process to each tenant representing the end user would require a prohibitive amount of memory resources. But if one notices that in many environments, the amount of processing done by a tenant is negligible compared to the idle time of that tenant and if in addition we notice that during this idle time very little information needs to be retained for the tenant, then it becomes natural to associate a process to a tenant only during processing time or when a large amount of information needs to be retained during idle time. This phenomenon is called MAPPING and UNMAPPING a tenant to a process. A workstation can be created with a certain number of identical processes, which are mapped successively to tenants in a much larger number (ratio in the order of 1 to 1000). The UNMAPPING requires that the useful size of a tenant has shrunk to a small minimum. This is not possible for all type of applications and as a result this facility is not necessarily useful for all types of workstations. Only certain type of workstations have the unmapping option.

Different workstations exist to perform different types of work, i.e. each workstation may be tailored according to the type of work it performs. This tailoring is done by the main procedure which is shared by all the processes at the workstation; it is called the Command Executive. The command executive is the first procedure in control when a process starts and the last procedure in control when the process terminates; in case of faults or exceptions it is also the procedure which gets control. As a result the command executive has full control over the life of the processes of the workstation thus giving its personality to the workstation.

SECTION 4

A CONCEPTUAL MODEL OF GCOS 8

A conceptual model of GCOS 8 must include two dimensions:

- a topological view of the organization of system services, and
- a functional view of the system organization

1.0 GCOS_8_Topological_Model

The software organization of GCOS 8 is represented in figure 4-1. Although more layers can be identified, for simplicity the system can be viewed as composed of three layers or concentric circles.

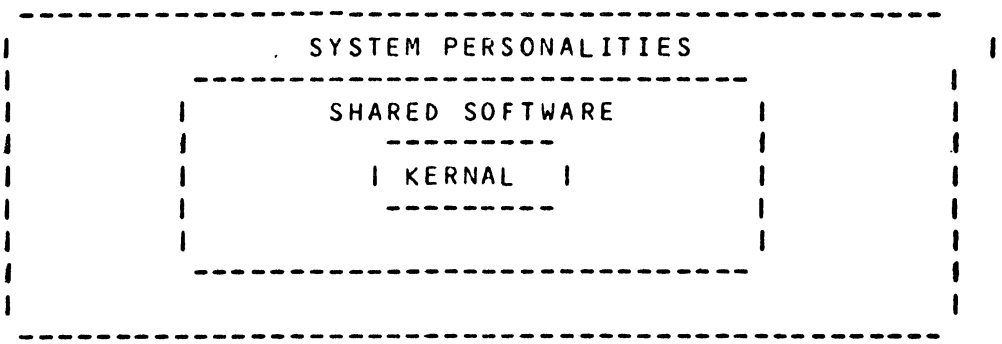

Figure 4-1 Organization of GCOS 8 Software

1.1 Kernal

The innermost layer is known as the Kernal and is that group of operating system procedures that provide the most primitive services. These procedures are extremely dependent on the idiosynchrocies of the hardware and typically execute in privileged master mode. Examples of Kernal procedures are:

- I/O drivers and channel modules
- interrupt handling
- process dispatching
- memory paging

1.2 System_Shared_Software

The next layer is the System Shared Software. This layer provides a common base on which the outermost shell rests. It is the Shared Software that provides the functions common to all users regardless of the view or personality of the system that is being utilized. The System Shared procedures execute in slave mode. Examples of System Shared procedures include:

- data base management
- buffer management
- session control
- file system
- concurrency control
- workstation management

1.3 System_Personality

The outermost layer is composed of a number of views available to the end-user. Examples include:

- batch
- timesharing
- transaction processing
- database query
- administration aids

- distributed maintenance

One user may be running batch programs at the same time another user is interactively issuing queries to a database. The "system" as seen by these two users is radically different and yet these various views or personalities of the system all rest upon the common services provided by the System Shared Software.

2.0 Functional Model

The most basic model of GCOS 8 is shown in Figure 4-2. It consists of three kinds of workstations, a Scheduler-Resource Manager Workstation, one or more Initializer Workstations, and a potentially large number of Application Workstations.

The GCOS 8 Operating System executes on a Level 66/DPS-8/ADP Information Processor that is an element of a DSA network. The GCOS 8 System may control a single computer, as in today's machines, or a multicomputer complex. The multicomputer complex is a collection of independent, but tightly coupled processors, connected by a high speed inter-computer link, that form a single computing resource.

Whether a single computer or a multicomputer complex, the GCOS 8 System contains a single workstation which is the control point for all resources and all processing activities. This Scheduler-Resource Manager (SRM) Workstation schedules batch jobs and activities, reserves resources on behalf of processes and workstations, and performs load leveling functions across the computers of the multicomputer complex. The SRM Workstation is also the one to which end-users initially log-on. The SRM Workstation is then responsible for determining which Application Workstation at the computer complex will be used to process the user's work. If a suitable Application Workstation is not available, the SRM Workstation may create a new Application Workstation for this user. Since the workstation being created may not be in the same computer as the SRM Workstation, SRM communicates with an Initializer Workstation in the proper computer which will create the Application Workstation.

what about recovery/restart

The Initializer Workstations, one per computer, create and terminate Application Workstations on the command of the SRM Workstation.

Application Workstations come in many different forms and provide a variety of facilities: batch input, system output, file transfer, timesharing, transaction processing, etc. A common characteristic of all Application Workstations is that they are created and terminated dynamically by the Initializer Workstation on command of the SRM Workstation.

Application Workstations may have one or many occurrences. For some workstations such as ITP, many users are handled by a single occurence of the workstation. The ITP command executive maps or associates the tenants with one of the processes at the ITP Workstation only when a tenant has work to perform such as when a transaction arrives. Thus, many end-users are handled by a single ITP Workstation, though

several ITP Workstations may be executing concurrently for different transaction processing applications.

Other Application Workstations, such as batch and timesharing, have many occurrences. Each batch activity and each timesharing user is allocated a single tenant, single process workstation to perform his work. This process is dedicated to the tenant for the life of the activity or timesharing session. A single workstation per user permits programs executed at the workstation to use Session Control to correspond with other workstations without interfering with sessions for other users. See section 2.1.

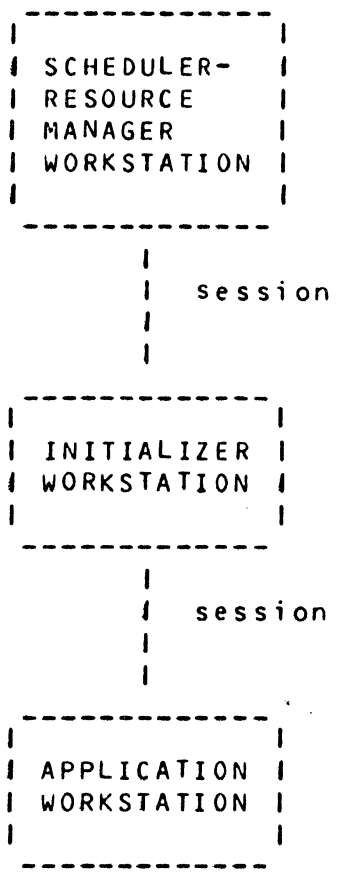

Figure 4-2 Overview of GCOS 8 System

SECTION 5

USAGE SCENARIOS

1.0 Workstation Definition, Creation, and Termination

Workstaions are defined by a System Administrator and the definitions are recorded in a Workstation Definition Database. To accomplish this, GCOS 8 provides an Administrator Workstation which accepts commands which both describe and control the state of workstations. This is shown in figure 5-1.

The System Administrator, at a terminal, logs on by establishing a session with the SRM Workstation. The SRM Workstation validates his use of an Administration Workstation and then selects a particular Administration Workstation if more than one are active. If no Administration Workstations are active, SRM may enable one by calling on the Initializer Workstation. The mailbox of the Administration Workstation is returned to the user and the log-on session is terminated.

The user (or terminal controller) then establishes a session with the Administration Workstation through which he has complete access to the Workstation Definition Database. Commands at the Administrator Workstation fall into two broad categories:

- those that create, modify, and delete entities on the Workstation Definition Database (e.g., workstations, mailboxes, etc.), and

- those that control the state of workstations

Thus, the administrator can define a workstation and then make it operational by entering an ENABLE workstation command. This command causes the Administrator Workstation to establish a session with the Scheduler-Resource Manager (SRM) Workstation to request the enabling of the input Application Workstation.

The SRM Workstation selects a computer (if a multicomputer complex), reserves the resources required, and issues a command to the Initializer Workstation to activate or enable the Application Workstation. The Initializer Workstation retrieves the workstation definition from the database and creates the desired workstation along with the processes and mailboxes defined for the workstation.

Commands are also available to the administrator to terminate and abort workstations.

As users log-on to the system, the SRM Workstation may also automatically cause workstations to be created as required. However before a workstation can be enabled in this fashion, the administrator must set the "automatic enable" attribute for that workstation.

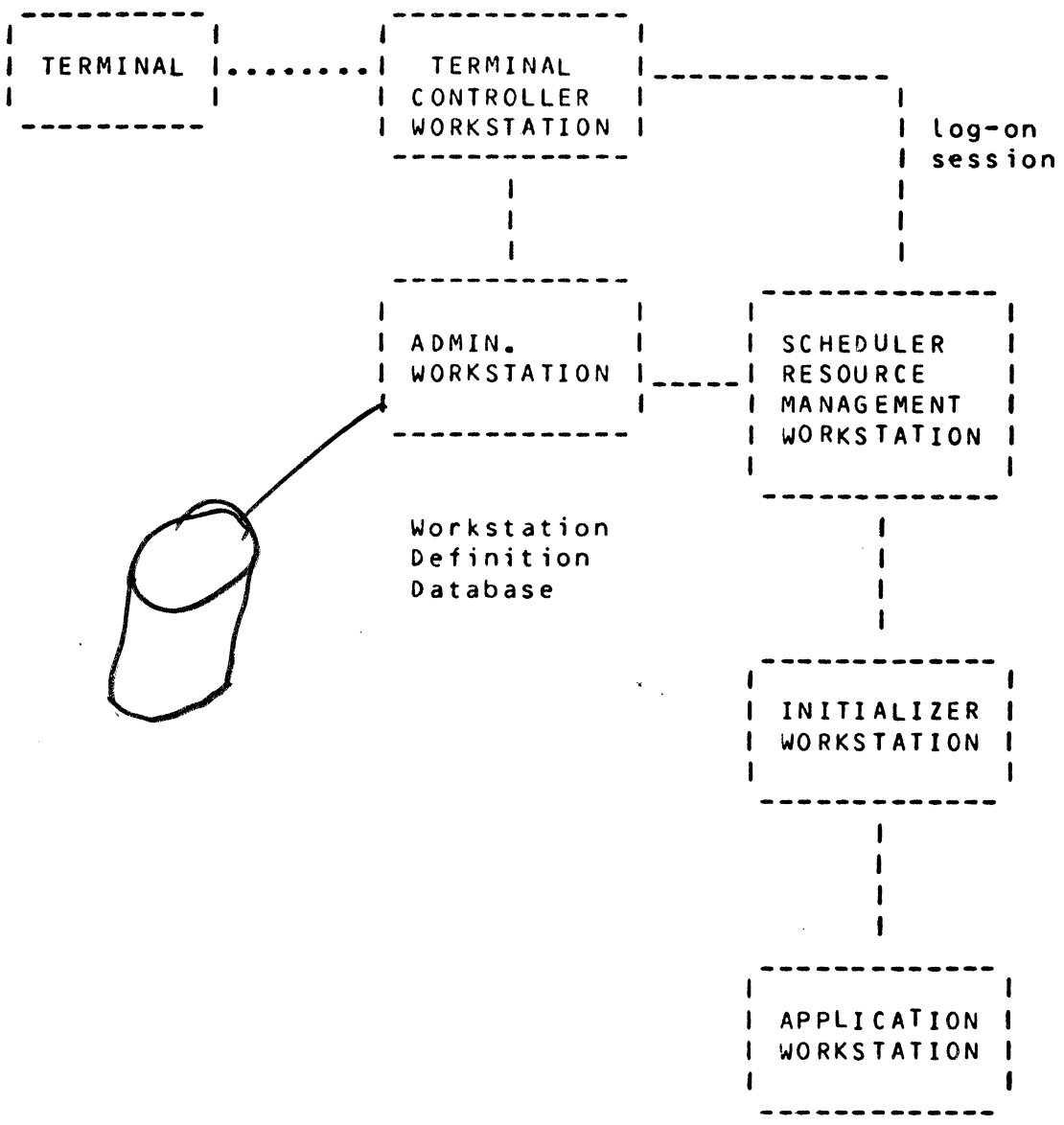

Figure 5-1 Workstation Administration

2.0 TIMESHARING Scenario

Each timesharing user is represented in a GCOS 8 System by a single tenant, single process workstation. This design is required to support the execution of programs which use Session Control under the purview of the Timesharing Workstation. One of the implications of this approach is that mapping/unmapping of tenants onto processes is not applicable to timesharing. Multiplexing of tenants onto processes is only viable when the structure required to describe a tenant is small when compared with the size of the process structure. Since this is not true in the timesharing environment, a process for each tenant, or timesharing user, is required.

When a terminal user wishes to use the native timesharing facility of GCOS 8, a session is initiated from the terminal controller to the SRM Workstation. See figure 5-2. The SRM Workstation performs the initial log-on function, validating the use of the timesharing facility by this user at this complex.

Upon successful completion of the log-on sequence, the SRM Workstation sends a message to the Initializer Workstation which creates a new occurrence of the single process, Timesharing Workstation. Each Timesharing user does not require a separate workstation definition; rather a TSS Workstation definition prototype is used to create all the TSS Workstations.

The user may have specified an initial mailbox name as part of the log-on sequence. If so, that mailbox is associated with the created Timesharing Workstation. If no mailbox name was specified, a mailbox is dynamically assigned to the Timesharing Workstation.

The SRM Workstation sends the name of the mailbox for the new Timesharing Workstation to the user and terminates the session with the Terminal Controller Workstation.

The user (or the terminal controller) establishes a new session with the newly created Timesharing Workstation via the created mailbox. The Timesharing Workstation validates the user and control is passed to the native timesharing command executive. The command executive is implemented as system shared procedure, but is executed under each Timesharing Workstation.

The user may wish to execute programs that use Session Control to communicate with other workstations. The mailboxes referenced by these programs need to be dynamically assigned to the user's Timesharing Workstation. The user should have full control over the use of mailboxes, just as he does for

files. The TSS Command Executive should implement commands that manipulate mailboxes: create mailbox, delete mailbox, assign mailbox to workstation, etc.

Most timesharing commands will be executed by the process at the user's Timesharing Workstation. However, some commands may require a different computer of the multicomputer complex. For example, suppose one computer was designated as the "COBOL machine". All COBOL compilations are executed on this one computer. When the timesharing user enters a COBOL compilation command, the Timesharing Workstation establishes a session with a COBOL Workstation that is located in the COBOL computer and the command is sent to the COBOL Workstation to execute. The COBOL Workstation is able to communicate with the terminal user via the Timesharing Workstation. When the compilation is complete, the session between the Timesharing and COBOL Workstations is terminated.

What about interactive compilers? This sounds like JRN

When the session between the user and the Timesharing Workstation is terminated via a log-off command or a line disconnect, the Timesharing Workstation notifies the Initializer Workstation and the Timesharing Workstation is terminated.

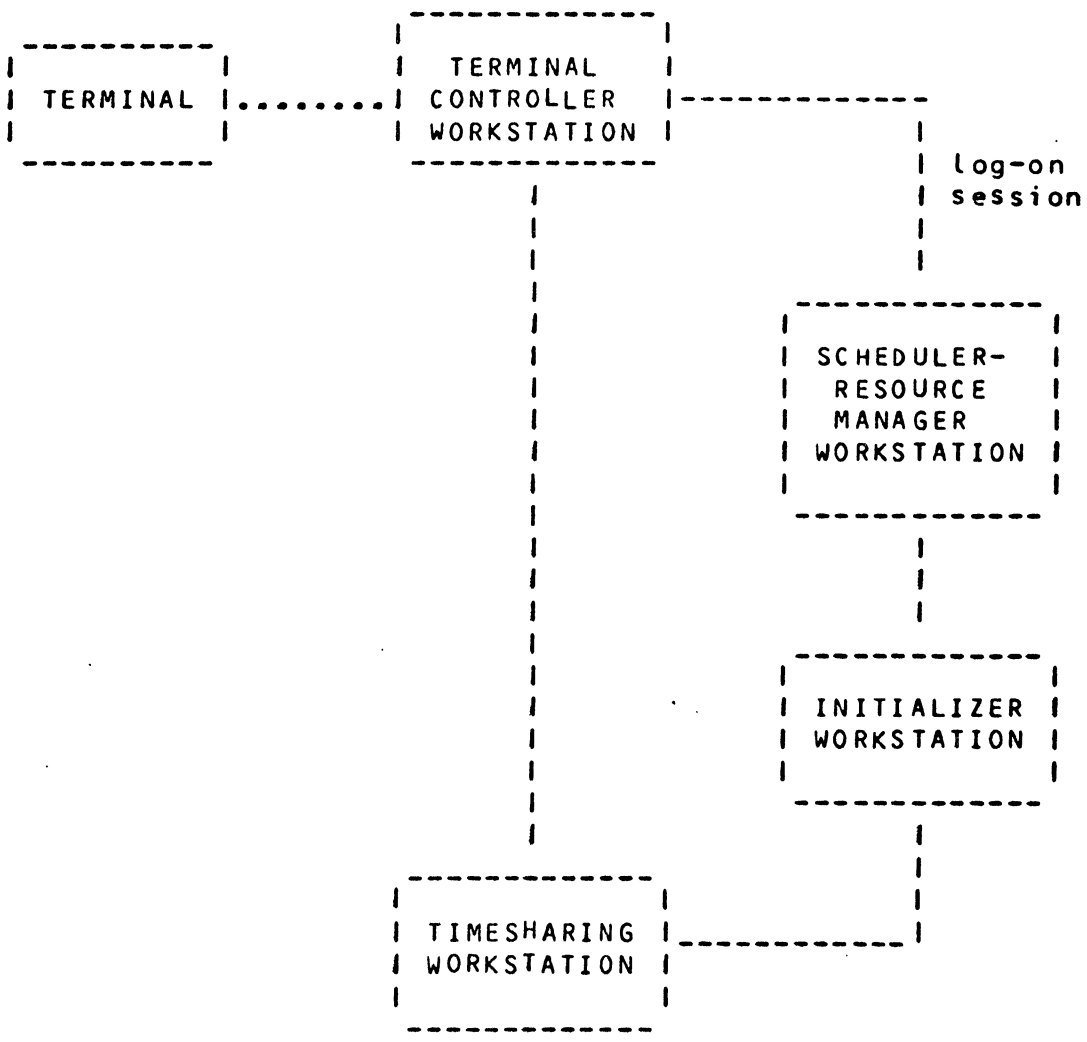

Figure 5-2 Timesharing

3.0 BATCH Scenario

The scenario for native batch execution is precisely the same as that for timesharing. The same JCL is used for each and, probably, the same command executive is used for both environments.

The only significant difference between native batch and timesharing is that batch jobs are entered into the system as a complete unit and are sceduled for execution, whereas the timesharing user enters commands interactively from a terminal.

Each ~~activity of a~~ batch job is executed as a single tenant, single process workstation. This design supports the activity's use of Session Control for the purpose of communication either with other workstations or with the terminal user.

The batch job may originate from a local device such as a card reader, from a remote location in the network, or as a result of a spawn job command in another workstation, such as Timesharing. See figure 5-3.

If the job originates at a remote terminal, the Terminal Controller Workstation establishes a session with the Batch Input Workstation in the destination information processor (e.g., GCOS 8 host). One Batch Input Workstation exists per computer complex. The input job is then transmitted as a series of messages.

The Input Device Control Workstations are types of terminal controllers which accept a local input job stream and convert the data from the external media (cards, tape, etc.) to a series of messages that are relayed to the Batch Input Workstation. Conceptually, there is no difference between local and remote job entry. An Input Device Control Workstation is required only when the local input device is in a different computer than the Batch Input Workstation. The Batch Input Workstation uses a common access method interface to read the input jobs. Depending on the device type and location, the access method will use Session Control for a remote device or will use IOS for a local device.

Jobs spawned at application workstations, such as TSS, are sent to the Batch Input Workstation just as if they were input via a terminal controller.

As it receives a stream of jobs, the Batch Input Workstation performs an operator validation function and then creates a command file for each job of the job stream. Then for each job, the Batch Input Workstation scans the JCL and determines the resource requirements for the job as a whole, as

well as for each activity. A message is then sent to the SRM Workstation signalling the presence of the job, its command file name, user-id, and resource requirements. An alternative design is one in which the SRM Workstation performs all the functions of the Batch Input Workstation, thereby eliminating it. The relative merits of each design have not been closely examined.

The SRM Workstation validates the user-id and enters the job into one of the job queues. Based on resource availability, the SRM Workstation reserves resources for jobs and activities, then sends a message to an Initializer Workstation to start each activity of the job. See figure 5-4.

The Initializer Workstation retrieves information about the job from the message received from the SRM Workstation and/or from the command file. It then allocates the resources which were reserved by the SRM Workstation. This resource allocation involves the creation of system tables (e.g., the PAT Segment), whereas the resource reservation performed by the SRM Workstation involves only entries in concurrency control tables. The Initializer Workstation then creates a single tenant, single process workstation for the activity. This Activity Workstation is created from a batch prototype workstation definition. If a mailbox name was specified for the activity in the JCL, that mailbox is associated with the created workstation. If none was specified, a mailbox is dynamically assigned to the workstation.

The Activity Workstation is then started and the batch activity is executed. The Batch Activity can call on any of the shared system services, including Session Control. This permits the job to communicate with other workstations such as timesharing. At the termination of the activity, the tenant, process, and workstation for that activity are deleted (alternatively, the process and workstation could be reused for a different activity).

Subsequent activites of the job follow the same procedure (i.e., scheduled by the SRM Workstation, tenant, process, and Workstation created and started by the Initializer Workstation). Any information that must be carried across activities of the job is recorded in the job's command file.

At the conclusion of the job, the SRM Workstation signals the System Output Workstation to process the system output files for the job. See figure 5-5. The System Output Workstation, one per computer complex, converts the output from the file into a stream of messages that are relayed to a Terminal Controller Workstation for remote output or to an Output Device Control Workstation for local output. The

Output Device Control Workstation operates as a terminal controller, but handles on-line devices such as line and page printers. As with the Input Device Control Workstation, the Output Device Control Workstation is only required if the local devices are in a different computer from the System Output Workstation.

An alternative design is one in which the SRM Workstation assumes most of the functions of the System Output Workstation and interfaces directly with the Output Device Control Workstations. More design is needed before a proper evaluation of these two choices is possible.

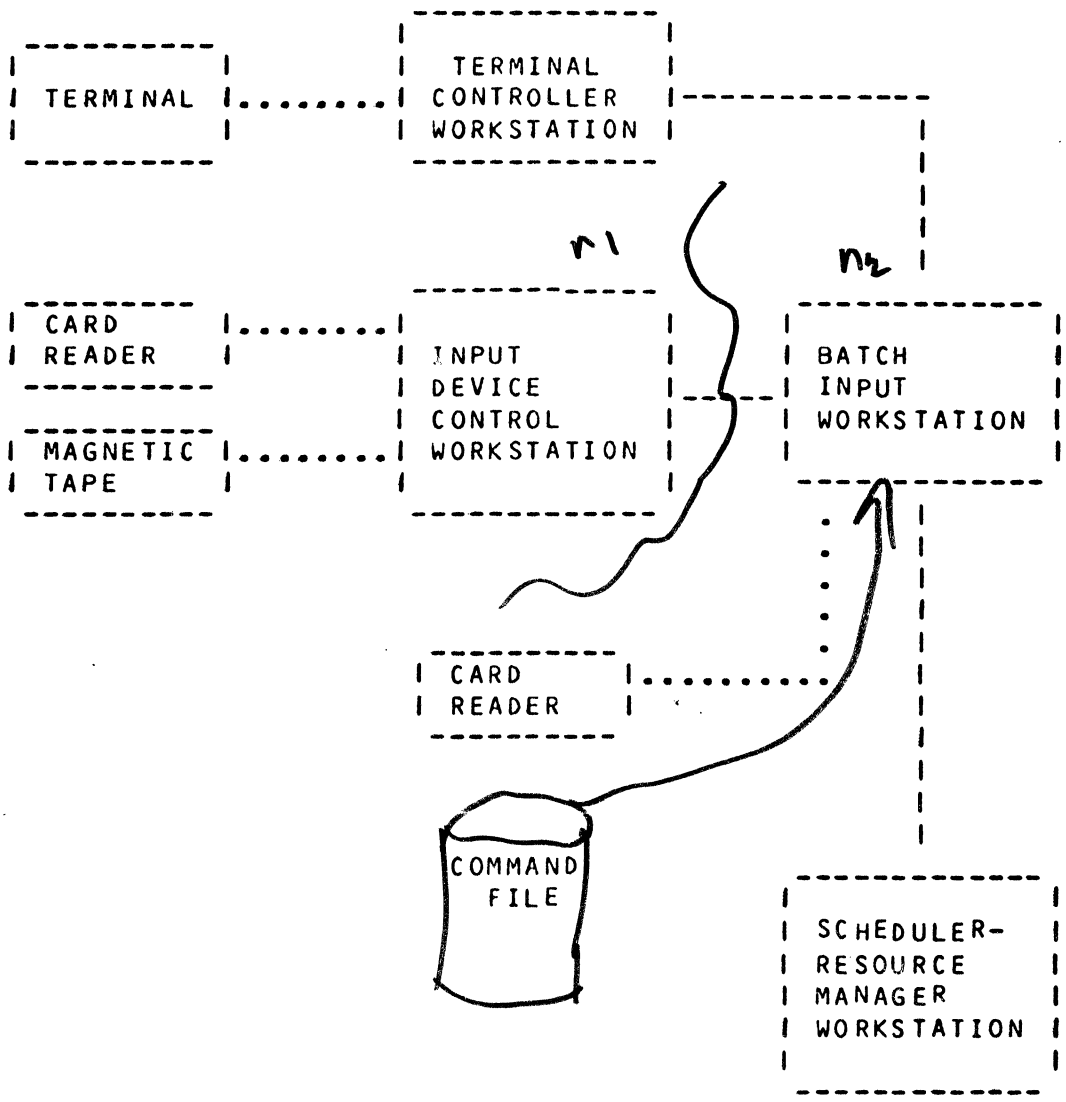

Figure 5-3 Batch Input

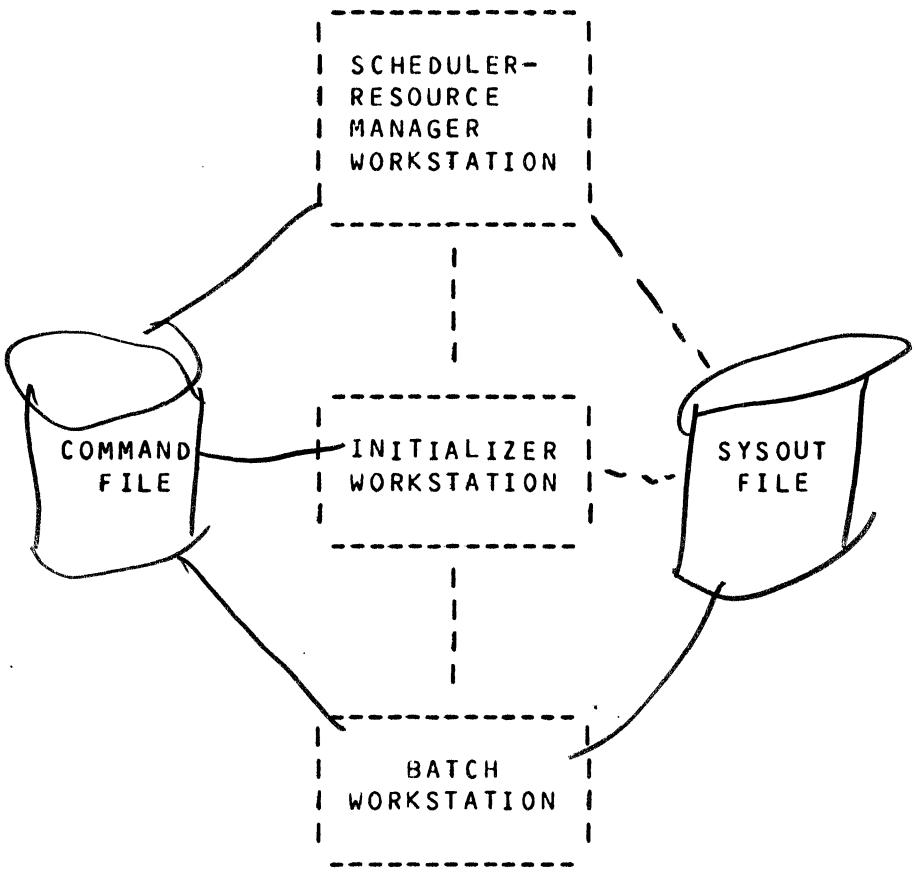

Figure 5-4 Batch Execution

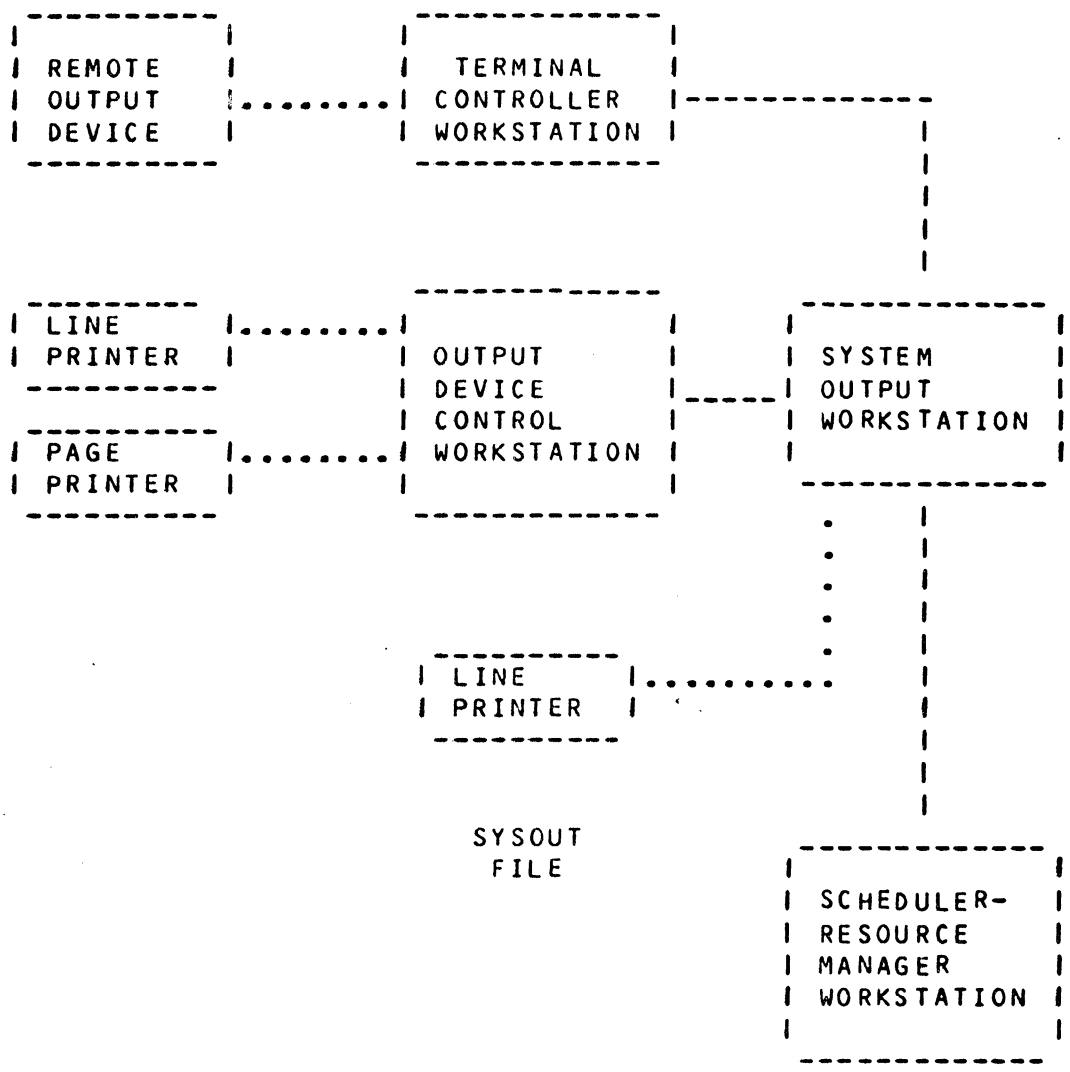

Figure 5-5 Batch Output

4.0 INTEGRATED TRANSACTION PROCESSING Scenario

An ITP Workstation is capable of supporting multiple processes and a large number of tenants. The ITP command executive maps a tenant to a process only when that tenant has work to do, such as when a transaction arrives.

As with timesharing, the ITP user first logs on by establishing a session with the Scheduler-Resource Manager (SRM) Workstation. See figure 5-6. The SRM Workstation validates the user and sends the mailbox name of the ITP Workstation to the user. The user (or the terminal controller) terminates the session with the SRM Workstation and establishes a session with the ITP Workstation.

If the SRM Workstation discovers that the desired ITP Workstation is not present, it may direct the Initializer Workstation to start the ITP Workstation if the "automatic enable" attribute is set on the workstation definition. See section 5.1.

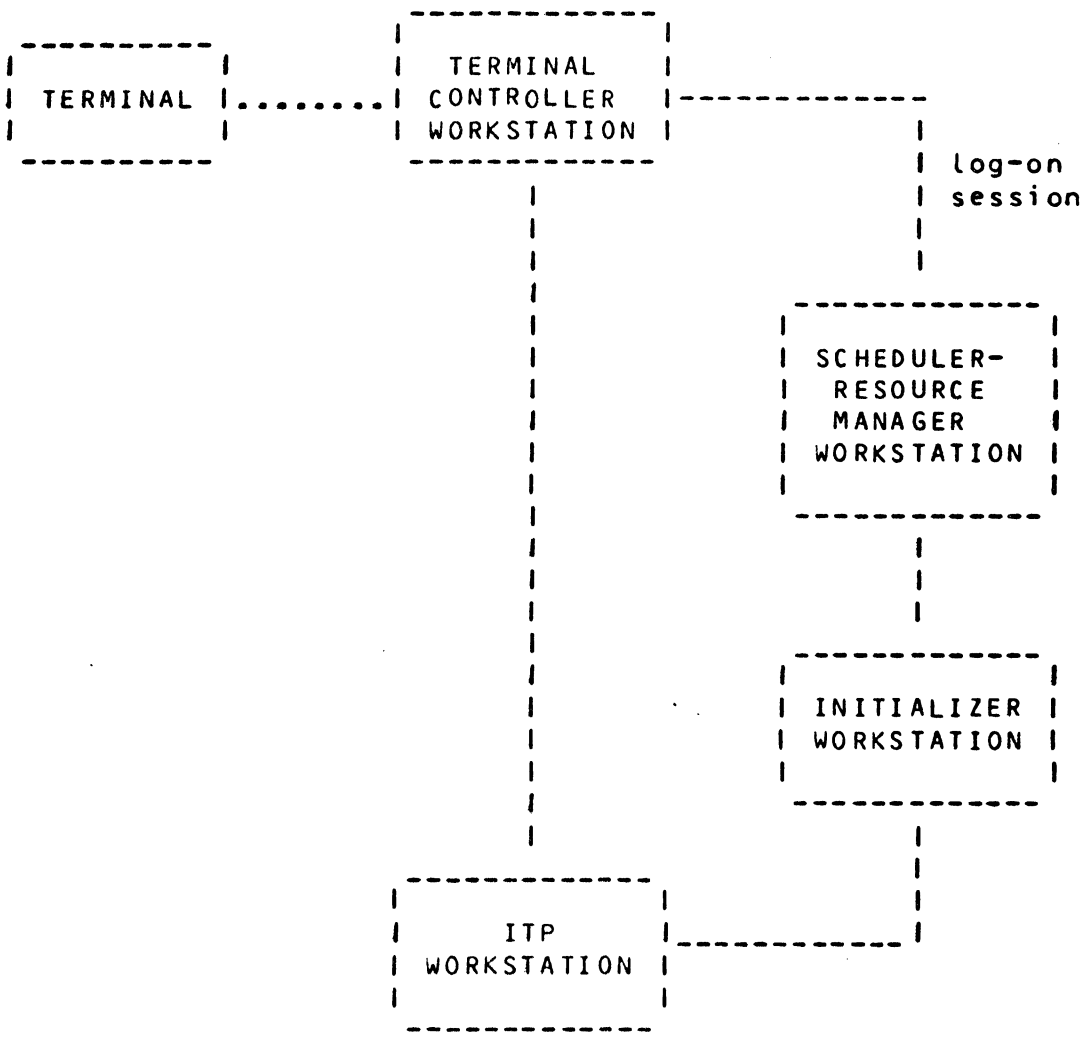

Figure 5-6 Integrated Transaction Processing

SECTION 6

ACCOMMODATION AND MIGRATION

By virtue of the consistent architecture, the shared system software that supports functions of native workstations also supports accommodation workstations. The great majority of the shared software is totally independent of the type of workstation from which it is called, including whether native or accommodation.

GCOS III programs are accommodated on GCOS 8 through two different mechanisms: MME accommodation and run-time support routines.

Programs in system loadable format, such as H*'s, and GMAP programs are accommodated by trapping the system calls (MME's) and converting them from the MME interface into the native system interface. Most, though not all, MME's are accommodated in this fashion.

The run-time support routines are a layer of code that exists between the compiled program and the native system software. A run-time package will be provided for each Higher Order Language compiler, both native and accommodation, to convert the system calls generated by the compiler into native system calls.

Thus a user who is willing to relink his programs will be bound with a GCOS 8 run-time environment which calls on shared software (e.g., UFAS and IDS); whereas, a user who executes an H* is already bound with the GCOS III UFAS or IDS and, therefore, can be accommodated only at the MME level. This of course implies that any new functionality provided by the GCOS 8 shared UFAS and IDS will only be available to users who relink their programs.

1.0 Accommodation Batch

Existing batch jobs are executed in a similar manner to native batch programs. That is, the flow ob batch input, execution, and output is the same as that described in section

5.2. The primary difference between native batch and accommodation batch is the JCL. This implies a different command executive for each and, therefore, a different Application Workstation. As with native batch, each activity of an accommodation job will execute in a separate single tenant, single process Accommodation Batch Workstation.

Accommodation jobs that use communication features (GEROUT/DNET) must be bound with a new run-time environment that uses Session Control interfaces. GMAP programs must be converted to use Session Control before execution. All other system calls (MME's) are trapped and converted to calls to the native shared procedures.

2.0 Accommodation_Timesharing

Accommodation timesharing users are serviced by an Accommodation Timesharing Workstation which is a multi-tenant, single process workstation that functions similarly to its GCOS 3 counterpart.

A native timesharing user is able to spawn batch jobs which are executed at an Activity Workstation for native mode prgrams or an Accommodation Batch Workstation for accommodation mode programs. However, this user will also wish to execute GCOS III programs (C* and H*) from his native timesharing workstation, rather than spawn a batch job. Many accommodation programs should run quite easily at a native workstation. Other programs, however, may have to execute at an accommodation workstation due to dependencies on the GCOS III environment, such as fault processing facilities. This quiestion requires a great deal more study.

3.0 Accommodation Transaction Processing

An Accommodation DM-IV TP Workstation is provided for servicing DM-IV TP users. Like the Accommodation Timesharing, this workstation is also a multi-tenant, single process workstation.