

**UNIVAC<sup>®</sup>**  
**490**



**REAL-TIME SYSTEM**

**TECHNICAL BULLETIN**

***REX***

***Real-Time Executive Routine***

***Programmer's Reference***

***First Edition***

July, 1962

## PREFACE

A knowledge of basic programming related to the UNIVAC 490 System is assumed in the preparation of this document. A prerequisite document is the SPURT Assembly System Programmer's Reference Manual.

# CONTENTS

<b>1. INTRODUCTION</b>	<b>1- 1</b>
<b>A. GENERAL DESCRIPTION</b>	<b>1- 1</b>
<b>B. FUNCTIONAL DESCRIPTION</b>	<b>1- 3</b>
<b>2. OPERATIONAL CONTROL</b>	<b>2- 1</b>
<b>Program Sequencing and Loading</b>	<b>2- 1</b>
<b>A. THE MASTER INSTRUCTION TAPE</b>	<b>2- 1</b>
<b>B. INTERNAL LOAD REQUESTS</b>	<b>2- 6</b>
1. Real Time Extensions	2- 6
2. Batch Load Requests	2- 7
3. Segment Call	2- 7
4. Facility Release	2- 8
<b>C. OPERATOR FUNCTIONS</b>	<b>2-10</b>
1. Start Schedule Format	2-10
2. Hold Schedule Format	2-10
3. Terminate Schedule Format	2-10
4. Lockout Format	2-10
5. Unlock Format	2-10
6. Load Format	2-10
7. Facility Update Format	2-11
<b>D. ALLOCATION OF FACILITIES</b>	<b>2-11</b>
1. Core Memory	2-11
<b>E. ALLOCATION OF PERIPHERALS</b>	<b>2-15</b>

<b>F. LOADING</b>	2-16
1. Absolute	2-16
2. Simple Relative	2-16
3. Complex Relative	2-16
<b>G. CONSOLE INPUT/OUTPUT OPERATIONS</b>	2-17
1. Program-To-Operator Communication	2-17
2. Operator-To-Program Communication	2-20
<b>3. EXECUTIVE CONTROL</b>	3- 1
<b>A. TABLE AND STORAGE AREAS</b>	3- 1
1. The Executive Entry Table	3- 1
2. Standard Information Maintained By REX	3- 2
3. The Executive Addendum	3- 3
4. Utility Request Table	3- 3
5. Delayed Response Table	3- 6
6. Interrupt Entrances	3- 6
7. Communication Interrupt Table	3- 8
8. Time-Table	3- 8
<b>B. STANDARD PERIPHERAL INPUT/OUTPUT</b>	3- 8
1. The Input/Output Request	3- 8
2. Standard Parameters	3-10
3. Register References	3-11
4. Input/Output Supervision	3-11
<b>C. STANDARD PERIPHERAL INPUT/OUTPUT STATUS CHECKING</b>	3-15
1. Purpose	3-15
2. Activation of CKSTAT Routine	3-15
3. Examples of CKSTAT Use	3-16
4. Logical Considerations	3-17
5. Programming Considerations	3-17
<b>D. COMMUNICATION INPUT/OUTPUT</b>	3-19
1. Communication Interrupts	3-19
2. Submission of Interrupts	3-20
3. Acquisition of Stored Interrupts	3-20
4. Interrupt Analysis by the Real-Time Program	3-21
5. The General Purpose Search	3-21
<b>E. INITIATION OF INTERVAL-TIMER INTERRUPTS</b>	3-24
<b>F. WORKER PROGRAM VOLUNTARY RELEASE OF CONTROL</b>	3-24
1. Suspension	3-24
2. Termination	3-24
3. Temporary Release	3-25
4. Exchange	3-25
<b>G. THE SWITCHER</b>	3-25
1. Priority Considerations	3-25
2. Operation	3-26

<b>4. CONTINGENCY CONTROL</b>	4- 1
<b>A. CONTINGENCY INTERRUPTS</b>	4- 1
1. Fault Interrupt	4- 1
2. Interval-Timer Interrupt	4- 1
<b>B. CONTINGENCY DIVERSION OF PROGRAM FLOW</b>	4- 2
1. Addendum Overflow	4- 2
2. An Excessive Accumulation of Input/Output Requests Without Associated Status Checking	4- 2
3. Communication Interrupt Table Overflow	4- 2
<b>C. OPERATOR CONTINGENCY INTERVENTIONS</b>	4- 3
1. Program Start	4- 3
2. Suspend	4- 3
3. Termination	4- 3
4. Interlock Response	4- 3
<b>5. UTILITY SERVICES</b>	5- 1
<b>A. OPERATOR REQUESTS</b>	5- 1
1. Inspect Drum	5- 1
2. Inspect Core	5- 1
3. Change Drum	5- 2
4. Change Core	5- 2
5. Print Drum	5- 2
6. Print Core	5- 3
7. Site Utility	5- 3
8. Additional Operator Entries	5- 4
<b>B. PROGRAM REQUESTS</b>	5- 4
1. Print Drum	5- 4
2. Print Core	5- 5
3. Assistance in Establishing Rerun Dump	5- 5
4. Utilization of Rerun Dump	5- 7
<b>6. PROGRAM PREPARATION</b>	6- 1
<b>A. SOURCE LANGUAGE</b>	6- 1
<b>B. USE OF JUMP KEYS</b>	6- 1
<b>C. USE OF CONDITIONAL AND UNCONDITIONAL STOPS</b>	6- 1
<b>D. STANDARD LOCATIONS</b>	6- 2
<b>E. SEGMENTATION</b>	6- 3
<b>F. FACILITY REQUIREMENTS</b>	6- 3
<b>APPENDIX A</b>	A- 1
<b>BASIC PROGRAM FORMATS</b>	A- 1
<b>Modification Codes</b>	A- 2
1. Complex Relative Format	A- 2
2. Simple Relative Format	A- 3

<b>A. ABSOLUTE</b> .....	A- 3
1. Identification Record .....	A- 3
2. Instruction Record .....	A- 4
3. End-of-Program Sentinel .....	A- 4
4. Storage Format .....	A- 5
<b>B. SIMPLE RELATIVE</b> .....	A- 6
1. Identification Record .....	A- 6
2. Instruction Record .....	A- 6
3. End-of-Program Sentinel .....	A- 7
4. Storage Format .....	A- 7
<b>C. COMPLEX RELATIVE</b> .....	A- 8
1. Identification Record .....	A- 8
2. Facility Record .....	A- 8
3. Segment Description Record .....	A-10
4. File Description Record .....	A-10
5. Control Segment Record .....	A-11
6. Secondary Segment Record .....	A-12
7. End-of-Program Sentinel .....	A-12
8. Storage Format .....	A-13
9. Segment Description Record When Modified .....	A-14
<b>APPENDIX B</b> .....	B- 1
<b>EXAMPLE 1</b> .....	B- 1
<b>EXAMPLE 2</b> .....	B- 2
<b>EXAMPLE 3</b> .....	B- 3
<b>EXAMPLE 4</b> .....	B- 4
<b>EXAMPLE 5</b> .....	B- 6
<b>EXAMPLE 6</b> .....	B- 7
<b>APPENDIX C</b> .....	C- 1
<b>GENERAL COMMENTS ON OPERATOR ENTRIES</b> .....	C- 1

# 1. INTRODUCTION

## A. General Description

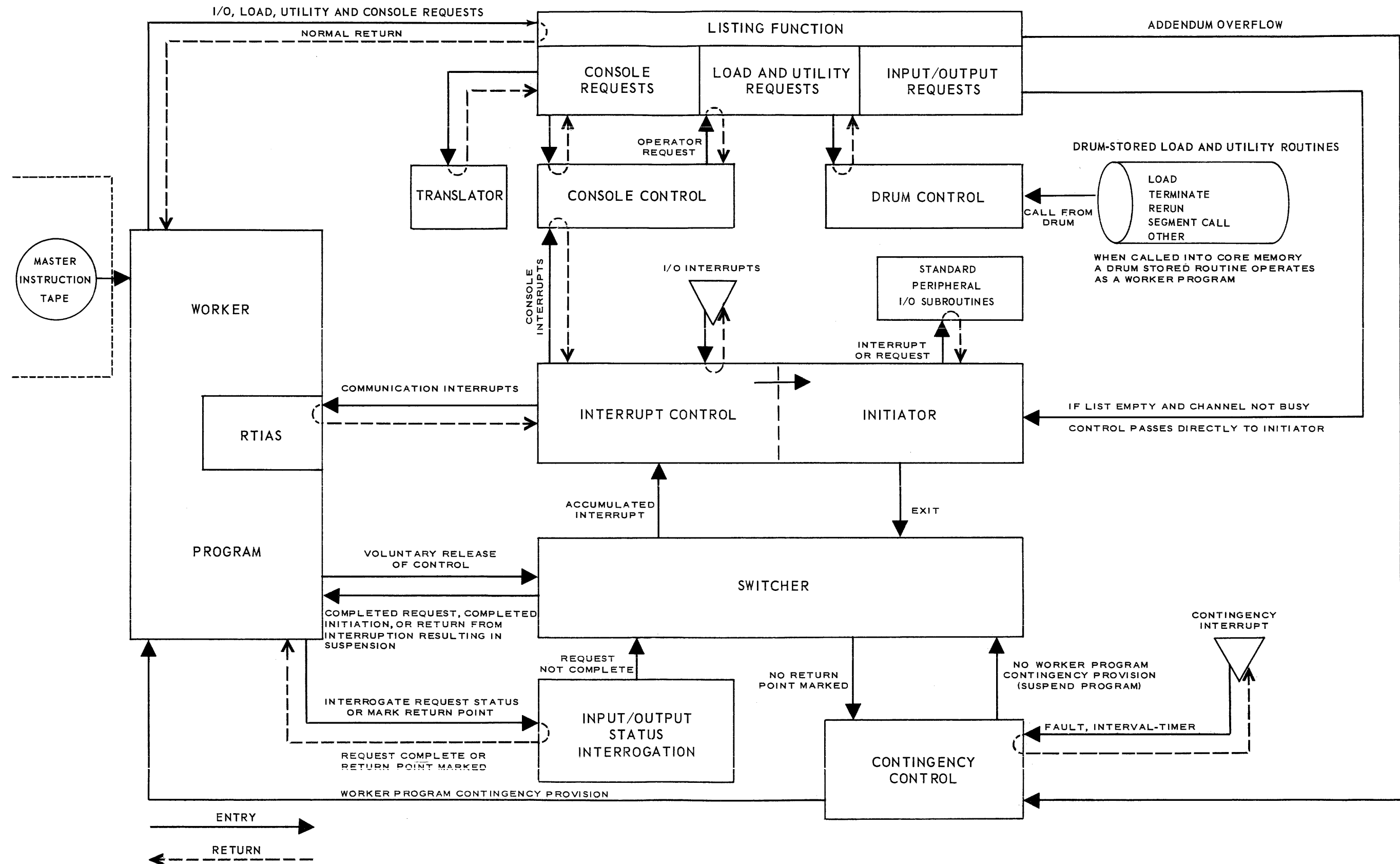
The Real-Time Executive Routine (REX) controls, sequences, and provides for the most efficient use of facilities for user programs operating in the UNIVAC® 490 Real-Time System. REX is related to the other routines which comprise the complete software package for the system. The SPURT Assembly System will translate a program written in a symbolic language designed to simplify program coding. SPURT output is a program suitable as input to the computer. Symbolic notation in SPURT language will produce coding which will provide linkage to REX routines.

A group of Utility Routines are designed to operate under REX control. These routines may be stored in the high-speed memory of the computer or on peripheral drum units. They provide the user with sophisticated and tested routines for many common tasks such as core and drum memory printouts and routines to change core memory. Utility Routines may be initiated by coding within a program or by entries on the console typewriter.

Special consideration must be given to the demands of the real-time program in any real-time system. The real-time program is characterized by random requests made upon the facilities of the system. It is possible to make general estimates of the facility requirements of the real-time program. The actual demands for facilities at any particular time cannot be estimated, however, as this will be dependent upon variable circumstances. This is in contrast to a program which will receive a batch of input information, where both the volume of input and the facilities required may be determined prior to the initiation of the program. A distinction is therefore made between the real-time program and batch programs.

A major consideration in the design of REX was to provide a priority structure that would release facilities to the real-time program upon demand. There are times when the demands of the real-time program will require full use of facilities; while at other times only an occasional request will be presented for processing. One or several batch programs may be run concurrently with the real-time program at these times. REX will provide for the interruption of batch programs when the real-time program is processing a request. REX will also attempt to overlap the input/output time associated with the real-time request.

FIGURE 1  
UNIVAC 490  
REAL-TIME EXECUTIVE PROGRAM





The high-speed (core) memory of the computer will normally contain the REX Routine, a real-time program, and one or more batch programs.

## **B. Functional Description**

The flow paths to and from the routines that comprise the major functions of REX are shown in figure 1. A brief description of each function will follow to provide an overall description. A detailed description of each function may be found in the remaining sections of the manual.

### **1. SELECTION AND LOADING**

Programs are presented for selection on a Master Instruction Tape (MIT). A routine is provided which will extract a program from a library provided by the REX user. This will be accompanied by card input which will describe when and how the program will be loaded. Options are provided so that a program may be placed on the MIT and called only on demand by an operator console entry. A program may also be selected to run and then inhibited by console entry. The facility requirements of the program will be defined by information on the MIT.

### **2. LISTING**

The environment that is controlled by REX is a complex one in which the real-time program and one or several batch programs will make requests for the use of facilities. The listing function provides a means through which REX may give order to these requests for program loading, input/output facilities, and various utility functions. If the necessary facilities are available, REX will provide them and keep a central record to indicate that they are in use. When facilities are no longer required they are released and requests are removed from the list.

### **3. CONSOLE CONTROL**

Provision is made for a running program to inform the operator of conditions that may exist during operation, such as the completion of a load, or the malfunction of a peripheral unit. Provision is also made for the operator to effect the operation of a program by input messages. These requests are made through the console typewriter.

Console Control will supervise the use of the console. Buffer areas contained in an addendum created at load time with hold console data prior to its submission to the console typewriter. A subsidiary function is the translation routine which will convert information from machine notation to a form that may be printed on the console typewriter. An example of this would be a request to print a location in high speed memory. The binary contents of the location interpreted as octal digits are translated to the notation required for output to the typewriter.

### **4. DRUM CONTROL**

A number of utility functions are provided to operate under REX control, such as a printout of drum or core locations. Utility functions may also be created by the user. The routines that perform these functions are usually kept in drum storage and called into core memory when needed. The Drum Control Routine will load utility routines, and once loaded they will operate as a worker program under REX control.

### **5. INITIATION**

The Initiator Routine will maintain a priority supervision over standard peripheral input/output requests. It will give priority to real-time program requests. Batch processor requests will be initiated in the order of submission.

## 6. SWITCHING

The Switcher Routine is the means by which REX relinquishes control and establishes inter-program priorities. Programs will be arranged so that those with relatively little input/output time will be operated within the input/output time of a higher ranking program. The Switcher Routine will also provide for the processing of interrupts that have occurred when a non-suspendible routine was operating.

## 7. REAL-TIME INTERRUPT ANALYSIS

The Real-Time Interrupt Analysis Subroutines (RTIAS) are provided by the user. REX makes provision for entry to and exit from these routines which are designed in accordance with the communication system configuration to assure a continuous and orderly flow of input/output information. REX will provide assistance in controlling communications interrupts and an optional routine to search for terminated input/output buffers. Standard peripheral input/output requests are controlled by subroutines assigned to each channel.

## 8. INPUT/OUTPUT INTERROGATION

A worker program may determine the condition of a standard input/output request by entering the Input/Output Interrogation Routine. An option is provided whereby the program may wait for completion of an input/output request or it may continue after marking a return point to which REX will return control upon completion of the request. Provision is made for a program to release control to REX when it cannot logically make further requests for the use of facilities. Requests that have not been completed will be supervised by REX and control will be returned to the worker program at the specified return points. A program that has reached its logical termination point will release all its facilities by a separate utility request.

## 9. CONTINGENCY CONTROL

Contingency Control provides for situations where the normal flow of a program is interrupted. The interruption may be the result of a logical inconsistency within a program, such as the overflow of table areas associated with program requests or communications requests, or it may be one of the standard machine generated interruptions associated with fault recovery, or interval-timer update. The worker program may contain coding to provide for these interrupts. REX will provide for the orderly acceptance of interruptions of this type, and if no user coding is provided the program will in most cases be suspended.

## 2. OPERATIONAL CONTROL

### PROGRAM SEQUENCING AND LOADING

It is the function of the routines described in this section to sequence, load, and initiate programs in order to make the most efficient use of facilities at any point in time.

Programs are presented for selection on a Master Instruction Tape (MIT). A dynamic selection process is initiated when the schedule is started and upon the termination of a running program. The set of programs examined at this time is governed by inter-program priority and a relation in which one program or group of programs may be dependent upon the output of some other program. These inter-program relationships are determined when a MIT is created.

The family of routines associated with program sequencing and loading will provide for:

- the computer operator to make deletions and additions to the set of programs defined by the MIT.
- loading programs in a simple format more suitable for debugging operations.
- loading a real-time program and extensions. These extensions will consist of routines within the real time program that are not used frequently. They are placed in peripheral storage and called on demand.
- loading and restarting run dumps to recreate to environment existing before a fault or error.

#### A. The Master Instruction Tape

The MIT may be thought of as a schedule of programs to be run. The tape will contain the instruction coding for each program in complex relative form, which means that peripheral facilities and core memory are assigned at load time. (See complex relative format Appendix A.) Information required by the load routine is supplied by Index Records and Program Facility summaries which precede the instruction coding records on the MIT.

Each program is executed with a minimum of operator intervention. The loading of a program will be accompanied by a console type-out describing the facilities required for the run. After the peripherals have been set up, the operator starts the program by console type-in. Loading will continue until all scheduled programs have been run or until inhibited by operator intervention.

a. Preparation of the Master Instruction Tape.

The REX user is provided with a Master Instruction Tape Assembly Routine which will accept scheduling requests from cards. Card format information and operating instructions for this routine are in the Utility Routine Manual. The routine will extract requested programs from a reference library, provide loading information records, and produce a MIT.

b. Definition of Terms.

The following terms are used in the description of MIT records which follow:

(1) Priority Group. A group of programs all of which must be initiated before the next priority group becomes eligible for initiation.

(2) String. A subset of a priority group comprised of programs which must be executed serially. A one-program string is permissible.

(3) String Leader. The first program of a string.

(4) Computer Estimate (CE). An optional scheduling parameter representing the ratio of central processor time within a processing cycle to total processing cycle time. The ratio is expressed in tenths. A value of 1 would mean the central processor is used during only 1/10 of the basic processing cycle; a value of 10, the program is capable of keeping the central processor continuously busy.

(5) Running Time. An optional scheduling parameter representing the estimated running time of a program in minutes were it to run independently.

(6) Program Lock. An indicator associated with each program. If set it means the program will be bypassed when executing the schedule.

(7) String Lock. An indicator meaningful in string context. If set it means this and subsequent programs of the string will be bypassed when executing the schedule.

(8) MIT Number. A sequential number assigned by the Master Instruction Tape Assembly Routine to each program as it is placed on the MIT. It is used for internal identification and tape position control.

The program lock and string lock options described above are provided to allow deletion of programs or strings after a schedule has been started. Programs may be included on a MIT which may or may not be run depending upon operator decision.

c. Description of Records.

A MIT may contain a maximum of 64 individual programs. It contains contiguous groups of records in the order listed below.

(1) A label record.

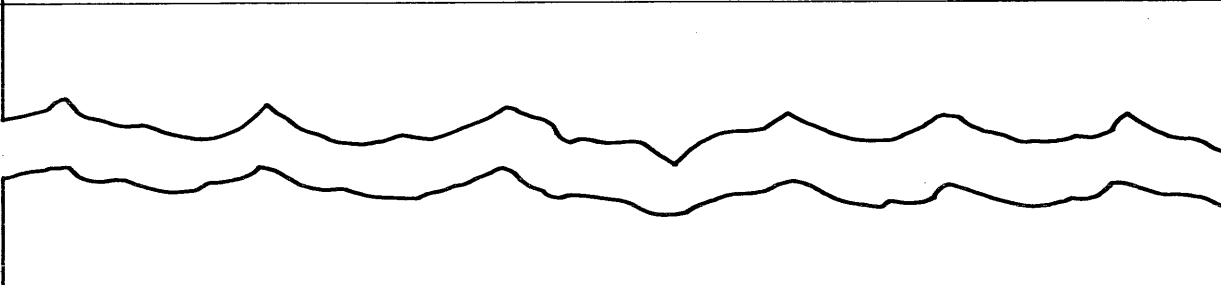
(2) One or a group of index records.

(3) One or a group of Program Facility Summaries.

(4) One or more programs in complex relative format.

(5) Two Standard End of File Sentinels.

## Label Record

0	7	2	7	2	7	2	7	2	7	2
1	7	2	7	2	7	2	7	2	7	2
2	Δ		M		I		T		Δ	
3	Δ		Δ		Δ		Δ		Δ	
4	Δ		Δ		Δ		Δ		n	n
5	y1		y2		d1		d2		d3	
Not used										
21	Number of programs					Number of priority groups				
22	7	2	7	2	7	2	7	2	7	2
23	7	2	7	2	7	2	7	2	7	2

nn is an octal identifier assigned at time of creation. yyddd is year and day. ddd is from 001 to 366.

## Index Record

There is one index record for each priority group. A maximum of 8 priority groups may be contained on the MIT, and a priority group may contain any number of programs. The MIT, however, can contain a maximum of 64 individual programs.

0	7 7 7 7 7					priority group					} string index
1	c		MIT number			string running time					
2	One-word index for each string within the priority group.										
3											
.											
.											
.											
.											
66 (max)	7 7 7 7 7					priority group					

c is lock condition indicator in bit positions 27-29.

- 1 string lock
- 2 program lock
- 6 no lock

MIT no. is that of the string leader.

#### Program Facility Summary Record

0	library number			MIT number		
1	successors library number			successors MIT number		
2	c	y	compute estimate	program running time		
3	minimum core			maximum core		
4	min $S_3$	min $S_2$	min $S_1$	max $S_3$	max $S_2$	max $S_1$
5	minimum card reader			minimum card punch		
6	minimum paper tape reader			minimum paper tape punch		
7				minimum high-speed printer		
8		minimum relocatable drum area				
9		maximum relocatable drum area				
10	drum base of program or zero					
11	Additional summaries, maximum of eight per block					
12						
...						
...						
...						
...						
88 Max.						

Summary 1

Summary 1

c is lock condition indicator in bit positions 27-29 as in index record.

y is an indicator to show the presence or absence of operational parameters. (See Operational Parameters).

S3 indicates IBM Compatible servos used by program.

S2 indicates UNIVAC III servos used by program.

S1 indicates UNIVAC IIA servos used by program.

Drum base when it appears in the last word includes drum channel normalized right in the high-order 6 bits.

Minimum and maximum requirements are used by REX in allocating facilities. The use of these entries will be more fully explained in later sections.

#### Program Format

Program format is complex relative - (See Appendix A). Block descriptions are modified to include MIT number as the most significant part of the descriptor word during MIT creation.

MIT number	0	0	0	0	0
------------	---	---	---	---	---

(Identification Record)

MIT number	0	0	0	0	1
------------	---	---	---	---	---

(Facility Record)

MIT number	0	0	0	0	2
------------	---	---	---	---	---

(Segment Description Record)

MIT number	0	0	0	0	3
------------	---	---	---	---	---

(File Description Record)

MIT number	0	0	0	0	4
------------	---	---	---	---	---

(Control Segment Record)

MIT number	segment number				5
------------	----------------	--	--	--	---

(Secondary Segment Record)

Only the first block of each record bears descriptors as shown. Subsequent blocks bear descriptors of binary zeros. This also applies to parameter records.

#### Operational Parameters

Operational parameters may be inserted within a program at the time a MIT is created. The parameter record generated will be the final record of the program.

0	MIT number	0	0	0	0	6
1	Additional block indicator	No. of parameter words				
2	parameter word 1					
3	parameter word 2					
4						
.						
.						
.						
	parameter word 48					
49	check sum					
50	MIT number	0	0	0	0	6

(51-wd block)

The upper half of word 1 is set non-zero if another block follows. The lower half represents the number of parameter words in this block.

#### Sentinel

A MIT will be terminated with two standard end-of-file sentinels.

### B. Internal Load Requests

Operating programs may request various load and load-related functions. They are initiated by entry to REX routines accompanied by a packet of information. This is inserted as coding within the requesting program.

#### 1. REAL TIME EXTENSIONS

Tasks within the real-time program may be arranged in an order of priority. Control routines and frequently used task routines are kept in core memory at all times if possible. Routines that are used less frequently may be drum stored in absolute form with overlapped core area assigned to those routines not likely to be concurrently executed. Routines that are least frequently required may be peripherally stored in relative address form and allocated to available core storage when needed.

REX will provide assistance for loading routines of the latter category. These extensions must be in simple relative program format (See Appendix A). They may be stored on the drum or on magnetic tapes. They will be loaded into any available core area with preference being given to that nearest the permanently allocated real-time program area. If necessary one or more batch-processors will be suspended and drum-stored in order to accomodate a requested extension. Suspended batch processors will be reloaded when a suitable environment is again established. One extension request at a time may be pending. Extensions cannot acquire peripheral facilities.

Format:

0	6	4	1	1	0	0	0	1	4	4
1	EAS								5 2	0 0
2	i				d				7 chan	4 3 unit 0
3	drum address or search word									
4	starting location					No. of words loaded				

DONE → (control here when load completed)

#### WORD

- 0 This word contains an indirect jump instruction to the executive entry table to initiate the extension load.
- 1 The upper half of this word will contain an executive action specifier (EAS). This allows control to be returned within the program. EAS use is explained in detail as



## WORD

part of the description of the CKSTAT operator. Positions 0-5 of this word contain the function code which is replaced by 77 if the requested routine cannot be loaded before control is returned at DONE. Possible causes are that the requested routine was not found or peripheral error.

- 2 A two bit indicator (i) specifies the storage medium.

00 specifies magnetic tape.

10 specifies drum.

Another indicator (d) may be used to imply the tape identified in the remaining portions of this word contains more than one program and that programs are in ascending sequence by identification record block descriptor. REX will record the position of the tape after performance of an operation so that tape movement for the next operation will be in the proper direction. Only one tape may be used with this option. The indicator is 1.

A zero indicator implies a forward search with a rewind following each operation.

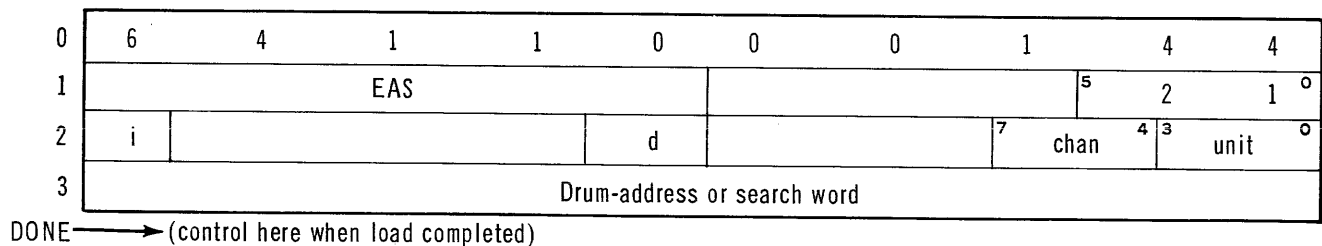
- 3 If a tape load – identification record block descriptor of routine to be loaded. If a drum load – drum address and channel.
- 4 This information is supplied by REX load before control is returned at DONE address.

## 2. BATCH LOAD REQUESTS

- a. At infrequent intervals the real-time program may desire that a task be performed which would require additional facilities and be essentially of a batch nature. An example of such a task would be preparation of a report based upon accumulation of a predetermined quantity of data or passage of a given time increment.

REX will service requests submitted by the real-time program to load batch processors. Programs loaded in response to this type of request will be treated as individual programs and may acquire peripheral facilities.

- b. Format:



All notes describing the extension request apply to batch requests.

## 3. SEGMENT CALL

- a. This function provides for loading segments. The segmentation function and the related symbolic coding are described in the SPURT Programmer Reference Manual. If a segment cannot be loaded because of peripheral difficulties, the requesting program will be suspended pending operator intervention.

b. Format:

0	6	4	1	1	0	0	0	1	4	4
1	0	0	0	0	0			5	2	3 <sup>0</sup>
2	Segment No.									

DONE → (control returned here when segment has been entered)

Segment number is used to find the locator pair within the executive addendum of the program which specifies the location and the form of storage for the segment. Locator pairs are incorporated into the executive addendum of the segmented program by the load routine. The format of the locator pair is described under EXECUTIVE ADDENDUM.

4. FACILITY RELEASE

The functions described below are designed to permit the release of facilities so that they may be used by other programs.

a. Unit release format:

0	6	4	1	1	0	0	0	1	4	4
1								5	2	4 <sup>0</sup>
2	1	2	0	0	0			7	chan	4 <sup>3</sup> unit <sup>0</sup>

DONE → (control returned here as soon as request has been listed)

The unit designated is made available for use by other programs.

b. Core release format:

0	6	4	1	1	0	0	0	1	4	4
1								5	2	5 <sup>0</sup>
2	starting location					number of words				

DONE → (control returned here as soon as request has been listed)

The core area designated is released. The area may be at the end of a running program, or it will serve to terminate a real-time extension. Extensions must be terminated in this way.

c. Drum release format:

0	6	4	1	1	0	0	0	1	4	4
1								5	2	6 <sup>0</sup>
2	starting address									
3	number of words									

DONE → (control returned here as soon as request has been listed)

## INTERNAL REQUEST USAGE

TYPE	WHEN SUBMITTED	PERMISSABLE FORMAT AND SOURCE	COMMENTS
EXTENSION	When no other request of like kind is pending	Simple Relative { Other tape Drum	Extension and Batch Load requests may be submitted by the real-time program only.
BATCH LOAD		Complex Relative { Other tape Drum	
SEGMENT CALL	Anytime	Complex Relative { Other tape MIT Drum	Complex Relative is the only format in which segmented programs will be put out by SPURT.
FACILITY RELEASE		Any	

*Figure 2. Internal Request Usage*

The relocatable drum area specified will be released. Release must be from the high-addressed end of the assigned area.

#### Summary

A summary of requirements and limitations for internal load requests and release functions appears in figure 2.

### C. Operator Functions

The operator has ultimate control over run sequencing and initiation. He may wish to operate with or without a MIT. If operating with a MIT he may wish to delete scheduled programs, remove locks imposed at time of creation or temporarily suspend execution of the schedule. All operator prerogatives are exercised by console type-in. A description of type-in formats and their functions are listed below. See GENERAL COMMENTS ON OPERATOR ENTRIES, Appendix C.

#### 1. START SCHEDULE FORMAT:

SS ☐ ch ☐ sv ☐

The function initiated by this type-in provides for initiating the MIT located on the designated channel and servo. This function may also be used to release a hold imposed upon a schedule subsequent to its initiation. The designation of a channel and servo is not required for release.

#### 2. HOLD SCHEDULE FORMAT:

HS ☐

This function will prevent further automatic loading from the MIT.

#### 3. TERMINATE SCHEDULE FORMAT:

TS ☐

This function will terminate a MIT. It may be used regardless of whether the MIT is active, in a hold condition, or exhausted. (Exhausted means that all scheduled programs have been executed, a condition communicated to the operator by console type-out).

Once this function has been entered all internal records pertaining to this MIT are purged. No MIT references are meaningful until a new schedule is started.

#### 4. LOCKOUT FORMAT:

LO ☐ MIT no. ☐ S(tring)  
or ☐  
P(rogram) ☐

This function may be used to impose a program or string lock upon any program on the MIT.

#### 5. UNLOCK FORMAT:

UL ☐ MIT no. ☐

This function will abrogate any lock condition existing for the designated program.

#### 6. LOAD FORMAT:

LD ☐ ps ☐ ch ☐ sv ☐ pl ☐ bs ☐ ty ☐

This function provides for all loading other than that performed automatically during execution of a MIT.

**ps**      Program source  
          T    magnetic tape  
          D    drum  
          C    cards (errata only)

**ch**      Channel

**sv**      servo designation if from magnetic tape, otherwise omitted.

**pl**      defines program location. If program is to be loaded from magnetic tape, this will be the block descriptor of the program identification record. (7474747474 whenever a rerun dump is to be loaded). If load is from drum, this will be the drum address of the program identification record.

**bs**      Base address if load format is simple relative. Dump identification if loading a rerun dump.

**ty**      Type of loading operation. See figure 3.  
          R    load real-time program.  
          E    load errata.  
          B    batch

#### 7. FACILITY UPDATE FORMAT:

FU ☐ ch ☐ un ☐ or ☒ <sup>U</sup><sub>D</sub>

This function provides for maintaining the central facility table from which REX makes facility assignments. The channel (ch) and unit (un) are each defined by two digits. Omit unit when channel specifies drum or disc subsystem. When channel specifies card or paper tape subsystem, unit 1 means reader, unit 2 means punch.

**U**      Up  
**D**      Down

#### D. Allocation of Facilities

##### 1. CORE MEMORY

The division of core between REX, a real-time program and batch programs is shown below. The Batch Lower Boundary represents the lowest address to which batch programs can extend. An area for real-time extensions may be allocated. This would tend to reduce interference with batch programs. If the Batch Lower Boundary is not specified, it will be automatically defined as the upper boundary of the real-time program. Or, if the real-time program is not operating, it will be defined as the upper boundary of REX.

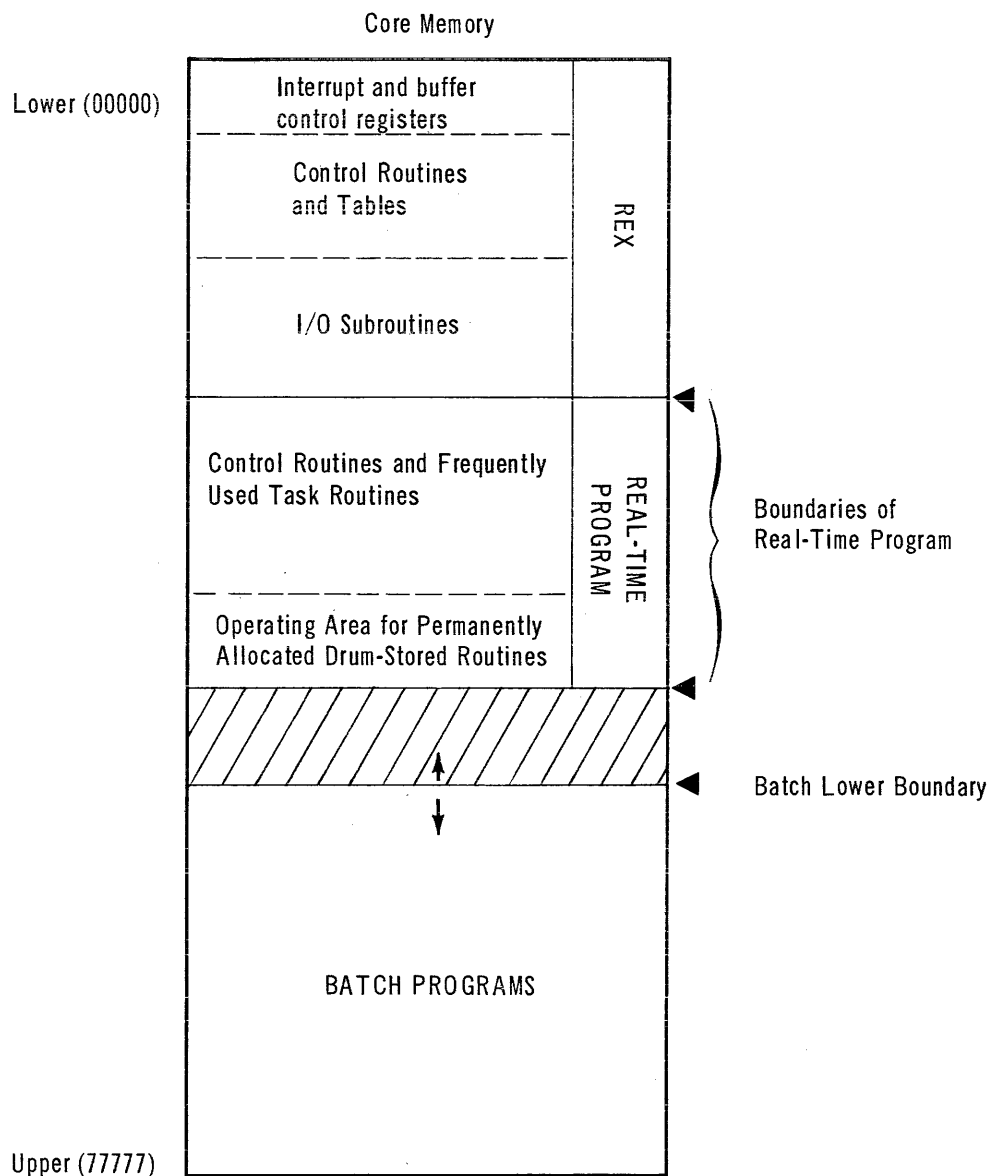
##### a. The Real-Time Program

The real-time program should be allocated as shown. If it is loaded from complex relative form, it will be so placed. Conflict at load time with batch programs already operating will

## OPERATOR LOAD REQUEST USAGE

TYPE OF LOADING OPER.	WHEN ENTERED	PERMISSIBLE FORMAT AND SOURCE		EFFECT	COMMENTS
R	Other than when Real Time Program already running	Absolute	{ Other tape Drum	Will hold in abeyance loading of scheduled programs from MIT until requested program has been loaded.  Once a simple relative or absolute program (Other than the RTP) has been loaded, no further loading will occur until it terminates.	Only one operator request may be pending. Entry of a second will cause obliteration of the first.  “Other tape” means some magnetic tape other than that designated as MIT by the SS function. It may contain any number or mixture of programs so long as the descriptor entered is unique to the identification record of the program to be loaded.  If a segmented program is running from an “other tape” no other program can be loaded from that tape.
B	Anytime	Complex Relative	{ Other tape Drum		
		Simple Relative	{ Other tape Drum		
		Absolute	{ Other tape Drum		
E	Anytime	Absolute	{ Other tape Cards	No placement checking is performed by REX. Consequently, the load will be carried out regardless of the condition of memory and will not effect the schedule or subsequent initiation.	
Simple Relative	{ Other tape				

Figure 3. Operator Load Request Usage



be resolved by holding in abeyance further batch loading and waiting until the competing batch program(s) terminates. A form of operator termination which will cause the terminated program to be rescheduled is explained under Contingency Control.

**b. Real-Time Extensions**

Allocation of real-time extensions will start at the upper boundary of the real-time program and continue to the upper end of core. Conflict with batch programs will be resolved by temporarily suspending and drum-storing them.

The allocation routine will always seek running area starting at the upper boundary of the real-time program. This will have the effect of filling in holes caused by termination of previous extensions.

**c. Allocation from MIT**

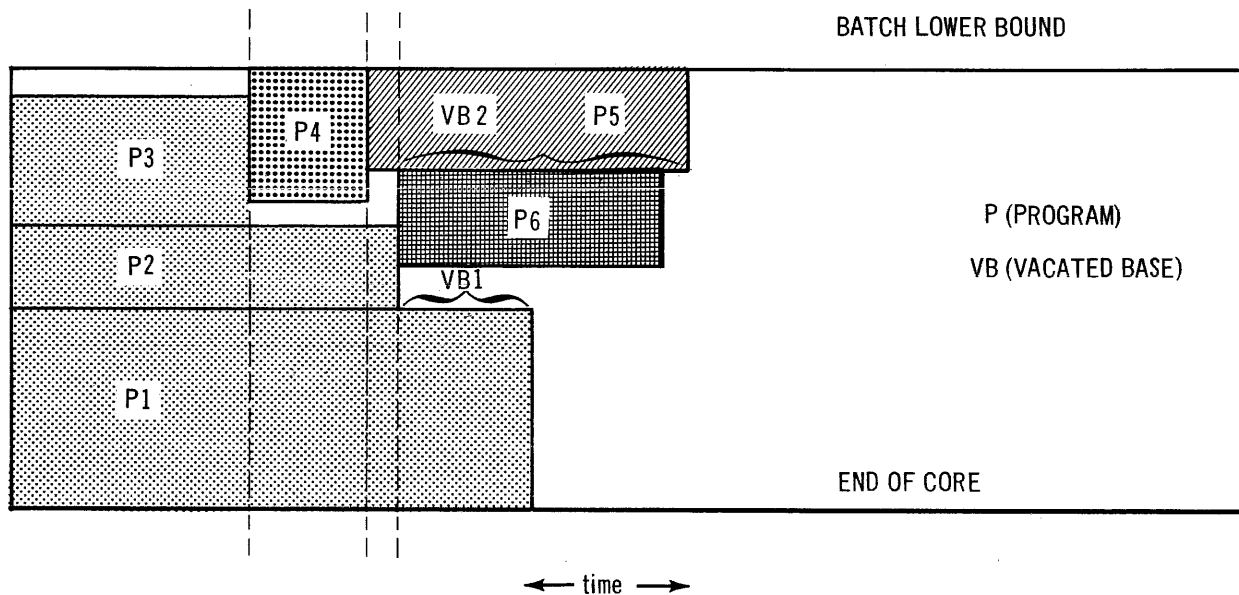
Allocation algorithms are based upon a concept of forming and maintaining program pyramids.

#### Pyramid Formation Involves:

- (1) Selecting from string leaders of the priority group currently being executed a set of programs which can be accommodated by existing facilities. String leaders are considered according to string running time, that is, the strings having the longest remaining time are always inspected first.
- (2) Forming the chosen set into a pyramid based on individual program running time.

Pyramid maintenance involves adding to an existing pyramid. The first string leader that will fit upon a vacant base will be loaded. String leaders are considered, in the order described above, according to string running time. At each termination allocation is re-evaluated and, if possible, one or more new programs introduced. A pyramid is formed whenever a boundary upon which it may be placed is free. If the real-time program is not operating placement may be on the Batch Lower Boundary or End-of-Core.

If the real-time program is operating only End-of-Core will be used. This will tend to minimize use of core nearest the real-time program, which is the area most likely to be invaded by real-time extensions.



The illustration above shows the area provided for batch programs. The real-time program is not operating. The allocation of programs within the batch area is dependent upon available area (shown as a vertical function) and running time (shown as a horizontal function). A new pyramid is formed at the termination of programs 3 and 4. At termination of program 2, two vacated bases exist upon which additions may be made. The base belonging to the pyramid placed on End-of-Core is first considered. Program 6 would not fit on the base and as a second choice was loaded on the other vacated base area because of time considerations. REX maintains a record of the remaining running time of programs based upon information supplied on the MIT.

#### d. Operator Load Requests

Programs requested by the Operator will be loaded as soon as sufficient area exists. If the program is in simple relative or absolute format, it is not loaded until all batch processors have terminated. Core usage is investigated and recorded to prevent interference with REX or the real-time program and its extensions.



#### e. Internal Batch Load

Internal load requests are loaded in the same manner as operator requests.

### E. Allocation of Peripherals

REX will maintain a central facility registry which will reflect the status and availability of all peripherals within a system. In addition, REX will maintain a relocatable drum area map. Programs loaded from complex relative format will have relocatable facility requirements satisfied by assigning, from the registry, those currently available. The registry will be updated.

#### a. Relocatable Drum Memory

A program may operate with varying quantities of relocatable drum area assigned. For example, a typical segmented program such as a Sort/Merge program might have the following drum requirement:

##### Declared Area

minimum = 6K

maximum = 600K

##### Segment Storage

minimum = 0

maximum = 8K (total length of all segments)

##### Allocation algorithm:

##### Declared Area

- (1) Assign the maximum requirement within the smallest area which will accommodate the maximum.
- (2) If 1 fails, assign the largest area that will accommodate the minimum.
- (3) If 2 fails, the program cannot be loaded.

#### b. Segment Area

Allocation of segment storage area is not related to allocation of declared area. Consequently, the two areas may be discontinuous.

- 1)  $n + 1$  storage requirements of decreasing length are calculated.

$n$  is the number of segments.

$$\text{requirement } 1 = 1_1 + 1_2 + 1_3 + \dots + 1_n$$

$$2 = 1_1 + 1_2 + 1_3 + \dots + 1_{n-1}$$

⋮

$$n = 1_1$$

$$n + 1 = 0$$

- 2) An iterative process which attempts to allocate each requirement (starting with the largest) to the smallest satisfactory area is initiated.

Failure to allocate storage for any segment means that segments will be loaded from tape and modified to running form each time called.

#### c. Relocatable Units

A relocatable unit requirement is defined in terms of channel groups. A channel group con-

sists of all units assigned a particular channel. (See ASSIGN operator in SPURT Manual). It may be expressed in terms of a maximum (desired) and a minimum (required) number of units. An attempt will be made to first allocate the maximum, then the maximum less one, etc. until sufficient units exist or until the minimum cannot be satisfied and the program must be rejected.

A channel group may be split across two actual channels or two channel groups may be consolidated on a single actual channel. Consolidation will not be performed unless absolutely necessary.

a. Fixed Requirements

(1) A fixed facility requirement reflecting use of mass storage channels or units may be expressed. Such a requirement is not relocatable at load time except by operator direction.

In the case of mass storage channels the only action performed by REX is a check to see if the peripheral type sought is actually on the channel expressed and is operable. If it is not on the channel, the operator will be given the opportunity to designate the proper channel so that substitution may be effected during loading.

In the case of units, a check will be performed to see if the peripheral type is on the channel expressed, if the unit expressed is present and operable, and if the unit is not currently busy. A negative result for any of these tests will cause REX to seek operator substitution.

(2) The Real-Time Program

The real-time program may be loaded from complex relative format in which case allocation of facilities would be as for any other program with the exception that core memory area would commence immediately above REX.

It would seem likely, however, for an intricate program subject to change that an absolute format would be easier to maintain, faster to load and simpler for an operator to service. Consequently, a provision will be made for marking units within the central facility registry as belonging to the real-time program. Whenever the real-time program is loaded from absolute format these units will be set busy.

F. Loading

1. ABSOLUTE

Program length is extracted from the identification record. This, in conjunction with the initial program address expressed in the first instruction record, is used to calculate core occupancy. If no conflict exists the program is transferred into core memory. No modification is made during the loading process. (Facilities are set busy if this is the real-time program). If the program is loaded from magnetic tape, the load channel and servo will be defined by the low-order eight bits of B<sup>1</sup> at activation.

2. SIMPLE RELATIVE

The operator-supplied base address in conjunction with the program length is used to calculate core occupancy. If no conflict exists the program is transferred into core memory. Address modification is performed during loading. If loaded from magnetic tape, load servo designation will be available at activation as described for absolute format.

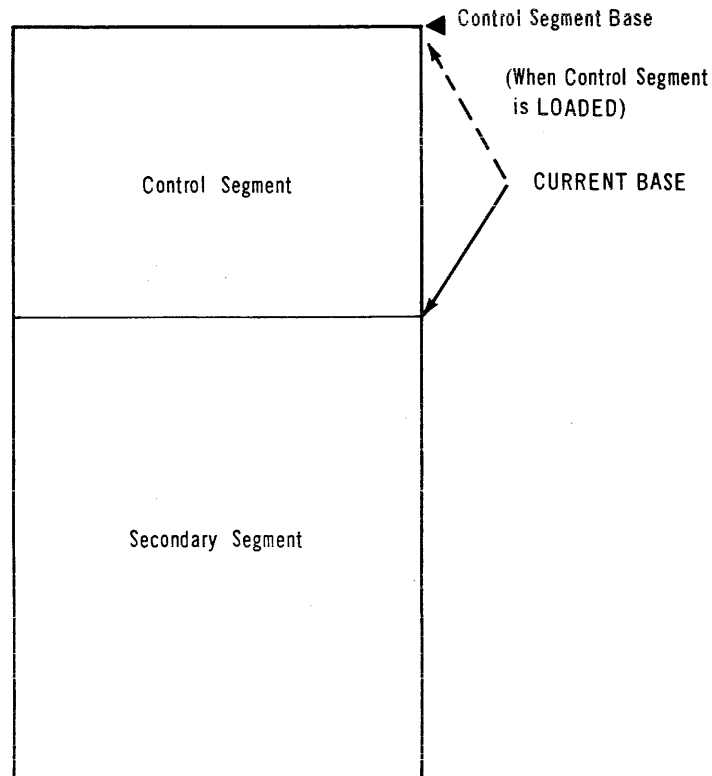
3. COMPLEX RELATIVE

a. Process

Once it has been ascertained that the program can be loaded and facilities have been allocated, a type-out is performed listing assigned facilities. The control segment is modified to running form and loaded into running location. Each secondary segment to be drum-stored is

modified to running form and transferred to the drum. Finally, any operational parameters to be loaded are placed in core memory. It should be noted that the secondary segment in core memory at the time the program is started will be that which was last stored to the drum and is unpredictable.

**b. Arrangement of Segmented Programs**



Two addresses are important in modifying a program to running form. They are, Control Segment Base and Current Base.

The Current Base is that address at which the segment being loaded starts.

Control Segment Base is the address at which the control segment starts. This address is remembered for use during loading of secondary segments.

When the control segment is being loaded current base is identical to control segment base.

**G. Console Input/Output Operations**

A running program sometimes finds it necessary to type out short messages advising the operator of conditions that exist during the execution of the program or upon termination. In addition, it is sometimes necessary to communicate with a running program. This communication may be a response to a message that has been typed out, or it may be an unsolicited entry that will in some way affect the execution of the program.

**1. PROGRAM TO OPERATOR COMMUNICATION**

Information is typed out by submitting packets to the appropriate subroutine controlled by REX. The information to be typed out is transferred by REX into the executive addendum of the program, where it is held until the console printer is available. The program may continue while waiting for the printer and the buffer may be modified if desired.

REX will absorb only one request per program. Therefore, a program submitting several consecutive requests will be held up at each submission until the prior one is complete.

#### a. CONSOLE OUTPUT REQUESTS

The most convenient method to specify the information to be typed out, with an entry to the appropriate subroutine, is by the use of the SPURT operators TYPET and TYPEC. These are described in detail under CONSOLE TYPEWRITER FUNCTIONAL SUBROUTINES in the SPURT manual.

##### (1) TYPET (Type Fieldata Text)

SILRJP.U(142)					
0	0	0	0	0	No. of characters
0	0	0	0	0	1st address of buffer

The packet above is generated by the TYPET operator and is usually accompanied by a Fieldata text. It is necessary to observe this format when typing out internally generated information. The maximum of 70 characters, left-justified in the buffer area, must also be observed.

##### (2) TYPEC (Type Contents)

The packet generated by a TYPEC operator is described in the SPURT manual. Entrance to the REX Encode Routine will cause the specified registers or memory locations to be translated to Fieldata code which will be typed out in TYPET format.

REX Encode operates at the priority level of the submitting program and can accommodate several programs concurrently. REX encode uses the buffer area within the executive addendum of the program to encode a maximum of 70 characters.

#### b. CONSOLE FORMAT REQUESTS

The operation which follows will restrict the use of the console to the program submitting the request. This will insure a contiguous format in situations where a number of TYPET and TYPEC operations are required to complete a given format. The packet format for this operation will be generated by the CONSOLE.HOLD operator (See SPURT Manual) and will appear as follows:

SILRJP.U(142)					
0	3	0	0	0	

The individual TYPET and TYPEC operations that will produce the contiguous message should immediately follow the request for a console hold. REX will employ checks to prevent undue monopolization of the console.

The normal method of terminating a hold mode is by submitting the following packet to REX:

SILRJP.U(142)			
0	4	0	0 0

The above packet will be generated by the CONSOLE RELEASE operator (see SPURT Manual.)

A console hold may also be terminated by the submission of an accept request.

### c. CONSOLE TYPEWRITER FORMAT

For each independent console submission REX attaches a triple line feed, a program number and 6 spaces to precede the supplied data. Thus the successive mnemonics.

TYPET: START Δ OF Δ JOB | CR | NR. 10576

TYPET: JOB Δ 10575 Δ COMPLETE

would result in console output

6 spaces  
Pxx START OF JOB  
NR. 10576

Pxx JOB 10575 COMPLETE

Since the line feed and program number are automatically supplied for each independent request, a | CR | as a first character of an independent request is ignored. All other | CR | characters are honored (as shown) with a carriage return, line feed and 9-space indentation.

A string of requests within a HOLD mode is considered one independent request. Thus the sequence

CONSOLE . HOLD  
 TYPET: START Δ OF Δ JOB | CR | NR. 10576  
 TYPET: JOB Δ 10575 Δ COMPLETE  
 CONSOLE . RELEASE

would appear as

Pxx START OF JOB  
NR. 10576  
JOB 10575 COMPLETE

## 2. OPERATOR TO PROGRAM COMMUNICATION

Operator input to a program may be either solicited or unsolicited. Solicited input would require a previous program to operator transmission describing the entry to be made and also specifying the buffer to accept the input.

### a. SOLICITED REQUESTS

#### (1) ACCEPT (Accept Characters)

SILRJP.U(142)		
0 2	Number of characters	1st address of buffer
		EAS

Return Point

The ACCEPT packet conditions Console Control to expect and accept an operator entry of a specified maximum number of characters to be stored in the buffer indicated.

Normally, an ACCEPT request would be preceded by a CONSOLE HOLD and one or more TYPET and/or TYPEC requests describing the entry to be made. The ACCEPT request, like CONSOLE RELEASE, terminates a CONSOLE HOLD mode.

REX assigns a delay number (Dxx) by which the operator will respond to the solicitation. The identification of this delay number is made by printout ACCEPT: DXX. Depending on whether or not the ACCEPT operator terminates a CONSOLE HOLD mode, the ACCEPT type-out will be either an independent request with program number typed out or a dependent part of a HOLD series.

When the entry has been completed, the line following the packet will be eligible for control at DONE in the same manner as the completion of a standard input/output request. The parameter "EAS" may be used to retain control without waiting for the response. If EAS is left zero the program remains suspended until the entry is made and the DONE address is made eligible for control. If EAS is non-zero, the address specified by EAS will regain control immediately. Control will not appear at DONE until operator response has been made and use of REX TAKEOVER has resulted in use of this return point.

The DONE address, REX TAKEOVER, and EAS are explained in detail under STANDARD PERIPHERAL INPUT/OUTPUT CONTROL. The accept packet is generated by an ACCEPT operator in SPURT code.

### b. UNSOLICITED REQUESTS

#### (1) Indicator and Entry Format

This entry is made solely on the initiative of the operator. Console control will determine if the addressee program has provided for the acceptance of such unsolicited entries. The indicator optionally specified in the Executive Information Region will be compared to zero. If no indicator is specified, or if the indicator is non-zero the entry will be rejected. Otherwise characters will be accepted and stored into the consecutive locations following the indicator word, so long as the maximum number of acceptable characters specified in the lower half of the indicator word is not exceeded.

When all characters have been entered, the upper half of the indicator word will be set non-zero by storing the number of characters entered. If the maximum number of characters acceptable is violated the operator will be so informed and the indicator left as zero.

The program accepting unsolicited entries interrogates the indicator whenever it is interested in accommodating an unsolicited entry and resets the indicator to zero to permit additional unsolicited entries.

Indicator			Maximum number of characters acceptable			Indicator word characters
C1	C2	C3	C4	C5		
C6	C7	..	..	..		

## (2) Input Format

Unsolicited entries are begun by entering Pxx ☐ to describe the program addressed (where xx is the appropriate octal coded program number). If an entry is unacceptable to the addressee, a printout will so inform the operator. Otherwise the characters comprising the entry may be entered.

Response to input requests made by ACCEPT is addressed by entering Dxx ☐ to describe the entry in the delay table (where Dxx is the identifier printed as part of the ACCEPT Dxx output).

## c. CHARACTER ACCEPTANCE

The characters accepted from operator type-in will be stored left-justified, 5 characters per word into the buffer specified. If a number of characters greater than "no. of characters" parameter applicable is entered, the entry will be rejected with explanatory print-out by REX. The operator may then again attempt entry.

All characters typed (excepting carriage return-04-and backspace-77) will be entered into the buffer. Each character will be inspected to see if it is 57, the stop character, or 77, the backspace character. If 57, the character will be counted and stored and the accept mode terminated. If 77, the last preceding character will be erased from the buffer. The 77 will not be stored. Three consecutive 77's will cause the entire entry to be erased.

Unused character positions in the last buffer word into which characters are stored will be set to Fieldata master spaces (binary zeros). The number of characters should therefore be a multiple of 5 to include trailing spaces.

### 3. EXECUTIVE CONTROL

The section which follows will describe the table areas required by REX. Executive functions will be generally described with a detailed explanation of parameter requirements and other information which the user of REX would require.

#### A. Tables and Storage Areas

##### 1. THE EXECUTIVE ENTRY TABLE

The entrance to REX routines will be recorded in the Executive Entry Table occupying octal core memory locations 140-146 inclusive. A worker program may enter by executing an indirect arithmetic return jump (64) to the appropriate half word in the table.

When SPURT input/output and executive-oriented mnemonics are used, entrance linkage will be generated and need not concern the user.

EXECUTIVE ENTRY TABLE		
ADDRESS	UPPER HALF	LOWER HALF
140	A	B
141	C	D
142	E	F
143	G	H
144	I	J
145	K	L
146	M	



ENTRY	USE
A	All standard peripheral input/output requests (not console)
B	Input/output request status interrogation (CKSTAT operator).
C	Temporary release of control to REX. (REX . TAKEOVER operator).
D	REX Translation Routine (TYPEPEC entry for encoding.)
E	Console requests, STOPRUN and TERMINATE operators, internal load requests.
F	Servo lockout release.
G	Fetch communication interrupt.
H	Exchange present position for a marked return point.
I	REX internal use. Entry to error message routine.
J	Program – submitted utility requests and Initialization of Real-Time Program.
K	Enter routine to set Interval-timer.
L	Not assigned.
M	Not assigned.

## 2. STANDARD INFORMATION MAINTAINED BY REX

### a. DATE

The address of today's date will be maintained in the lower half of memory location 00146. The date will consist of year and day. Days will be numbered consecutively starting with 1 on January first. Entries are in Fielddata code, and the format of the whole word addressed by the lower portion of 00146 is:

yy	ddd
year	day

The above is a suggested format. The date will be called for by REX as part of the initialization load and some other five-digit Fielddata format may be substituted.

### b. TIME

REX will answer day clock interrupts and maintain the time in hours, minutes and 30 second indicator in memory location 00147.

The time will be stored in Fielddata code. When the 30 second indicator is 0, the Fielddata space (05 will be stored in the right-most 6 bits. When the 30 second indicator is 1, the Fielddata character for plus (42) will be stored.

ADDRESS	CONTENTS		
00147	hh	mm	i
	hours	minutes	30-second indicator

### 3. THE EXECUTIVE ADDENDUM

Every program to be controlled by REX will be assigned a variable length storage adjacent to and preceding the program. This area will be referred to as an executive addendum. The use of areas within the executive addendum is explained in figure 4. The addendum is used by REX to implement control over a running program. It provides for the storage of operational registers when a program is interrupted. B-registers at time of input/output request, control indicators relative to the operating status of the program, and buffers to accommodate console typewriter input/output requests have storage areas provided in the addendum. Reference will be made to the addendum in describing those routines which utilize addendum areas. Only those entries which are of immediate concern to the user are fully described. The remaining descriptions are included for general information.

The responsibility of the programmer in defining this area is to specify the number of addendum storage elements which will be required. Section VI, PROGRAM PREPARATION will clarify this responsibility.

### 4. UTILITY REQUEST TABLE

- a. The REX Utility Request Table is for internal use by REX and provides for the orderly submission of utility requests. The area assigned consists of one location which describes the space available. This is immediately followed by table entries of variable length. See figure 5. Requests are stored until honored by REX. The user will not ordinarily be concerned with entries to the Utility Request Table. Requests will be stored by REX in response to operator entry or the submission of a packet. The user will be required to access the table only when he makes use of a Site Utility Routine and the necessary procedures will be described in that section.

The table may be searched by scanning successive function codes using count (N) of first function to skip over parameters of that function to the next function. The search may be terminated by testing for the start of the inactive area (the address indicated in lower half of word zero).

#### b. TABLE ENTRY FORMAT

Utility requests may be entered in the table through the console typewriter or by an internal request effected by the submission of a packet.

##### (1) Console Requests

Utility requests via the console are accepted by Console Control which generates an entry for the utility table whenever the function description is valid, maximum permissible character count is not exceeded, and the field format is acceptable. A field is a variable string of characters describing a particular parameter and is terminated by the special character ☐ (Fielddata 76). Characters within a field are right justified. The character ☒ (Fielddata 57) which terminates an entry, also terminates the current field.

An entry may be typed in the following format:

PC ☐ 230 ☐ 777 ☐ 0 ☐ 4 ☐ P1 ☒  
1        2        3        4        5        6

This is a request to print core (field 1), from 00230 to 00777 (field 2 and 3), in octal (field 4), on printer 1 (field 6), which is on channel 4 (field 5). This example is entered in the table represented in figure 5.





# EXECUTIVE ADDENDUM

WORD  
(OCTAL) 29

15 14

0

COMMENTS

0	* Lost Control Indicator (LCI) P-Register		(B7)		Lost Control Re-Entry (LCR)		
1	(B1)		(B2)				
2	(B3)		(B4)				
3	(B5)		(B6)				
4	(A)						
5	(Q)				14 I/O Request can be Accommodated Before use of CKSTAT is Required to Free Slot in Table.		
6			Program I/O Bound (Queue Full at List Time)				
7	Label of Submitted I/O Request		Place to go (PTG) Storage Element Assigned				
14-Word Table for Linking							
							
24	I/O Request with Associated CKSTAT					Since REXENCODE Operates at Program's Priority, LCR Module cannot be used at TYPEC Time.	
25	(Q) at TYPEC Submission						
26	(A) at TYPEC Submission						
27	(B6) at TYPEC Submission		(B7) at TYPEC Submission				
30	Current Address - TYPEC		Save Code - TYPEC				
31	Built-Up TYPEC Code					TYPET Request	
32			Character Count 1-5				
33			* Assigned Program Number				
34							
35			Total Character Count				
36			* Addendum Base + 40 <sub>8</sub>				
37	Temporary Storage		Transdata Address - Exit After Submission				
40							
14-Word Buffer for Accomodation of Generated TYPEC Messages and for Absorption of TYPET Buffers from Submitting Program. A Maximum of 70 Characters can be Accommodated.							
55							
56	Overflow Storage		Overflow Storage				
57							
60	t	Channel A Servo Lock-Out by Select Bit		Channel B Servo Lock-Out, etc.	2	1	0
61	Count of Storage Elements Started			Count of Storage Elements Completed			
62	s	Cause of Suspension	27-REX 26-Operator	25-Voluntary Self	Address of PTG being Monitored for 1st return (CIO)		
63	Ordering Count of I/O Requests Submitted						

t - Terminate Program if set. Operator Response is Recorded by Master Bit Scheme.

s - Program in Suspended condition if Set

Figure 4.

# EXECUTIVE ADDENDUM (Cont'd)

WORD (OCTAL)	29	15	14	0	COMMENTS	
64	* Number of I/O Storage Elements Specified			* Address of EIR	Addendum Storage Element	
65	Element Status Indicator			Exit Address or Time		
66	(B1) – at CKSTAT Time			(B2)		
67	(B3)			(B4)		
70	(B5)			(B6)		
71	I/O Routine (or REX) Generated Status Word					
72	Error Address (EA)			Normal Completion (done)		
73	(B1) – at I/O Request Time			(B2)		
74	(B3)			(B4)		
75	(B5)			(B6)		
76	Addendum Base Address			Address of I/O Request	Addition Addendum Storage Elements as Specified in EIR of Program.	
* Designates Entries made by Loader Initialization Routine Loader Places Zeros in Remainder of Addendum.						
Last Addendum Storage Element						
SEG n	29	28	Segment Length		x – Source Indicator 10 – Drum 00 – Magnetic Tape Program From MIT, Segment not Drum Stored.	
	x	Address to be used as Current Base				
	MIT Number			Segment Number	2 5 0	
SEG 2	0	0	Segment Length		Program from other Tape, Segment Not Drum Stored	
29	5	27	0 0 0 0	0 Segment Number		
SEG 1	1	0	Segment Length		Segment is Drum-Stored y = 1 if Modified to Running Form, if not Modified, y = 0.	
	y	Address to be used as Current Base				
	Drum Address					
Executive Information Region						
Program Coding						

Figure 4.

## (2) Program Requests

An entry on the utility table made as a result of the submission of a packet by an operating program will result in octal information rather than Fielddata characters arranged in fields. This will cause the operation header to be entered followed by the address of the packet. The digits comprising the header and the packet address will be added to N as in console entry.

## 5. DELAYED RESPONSE TABLE

The delayed response table is used internally by REX. The table contains 2 word items relative to solicitations directed to the operator. Entries in this table are the result of either program solicitation of operator response by use of ACCEPT, or a REX solicitation in connection with an interlock error from an input/output initiation.

These solicitations are automatically assigned a slot in the table in order to avoid excessive processing delays within Console Control. As explained in the Console Control description, an identifier (Dxx) is typed out with solicitations and eventual response is to this identifier. Dxx, in effect, becomes the function code of the computer operator response.

The delay table will be monitored so that entries in the table which have not been responded to within a fixed time period will again be brought to the operator's attention. If the entry remains unused at the end of a second period, the delay entry will be automatically answered. Entries related to interlock errors will be answered to cause reinitiation; entries related to accept requests will be returned with one arbitrary character to the requesting program. That character will be ⑤ Fielddata 57.

## 6. INTERRUPT ENTRANCES

Core memory locations 00020-00035, 00040-00055, and 00060-00075 are the interrupt entrance registers. When an interrupt occurs the next instruction to be executed is that contained in the interrupt entrance register associated with the channel and type of interrupt. The use of these registers is restricted to REX.

WORD	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0	number of words available for new entries																next entry starting address															
1	Used by UTLCNTRL																				n		function code									
2	parameters comprising entry																															
Termination code for first entry																								▶		5		7				
Sample Entry																																
Parameter for UTLCNTRL																				5		1		0								
0 0				6 2				6 3				6 0				<input type="checkbox"/>		(76)														
0 0				6 7				6 7				6 7				<input type="checkbox"/>																
0 0				0 0				0 0				2 4				<input type="checkbox"/>																
0 0				0 0				0 0				6 4				<input type="checkbox"/>																
0 0				0 0				2 5				6 1				<input checked="" type="checkbox"/>		(57)														

P1 - printer 1

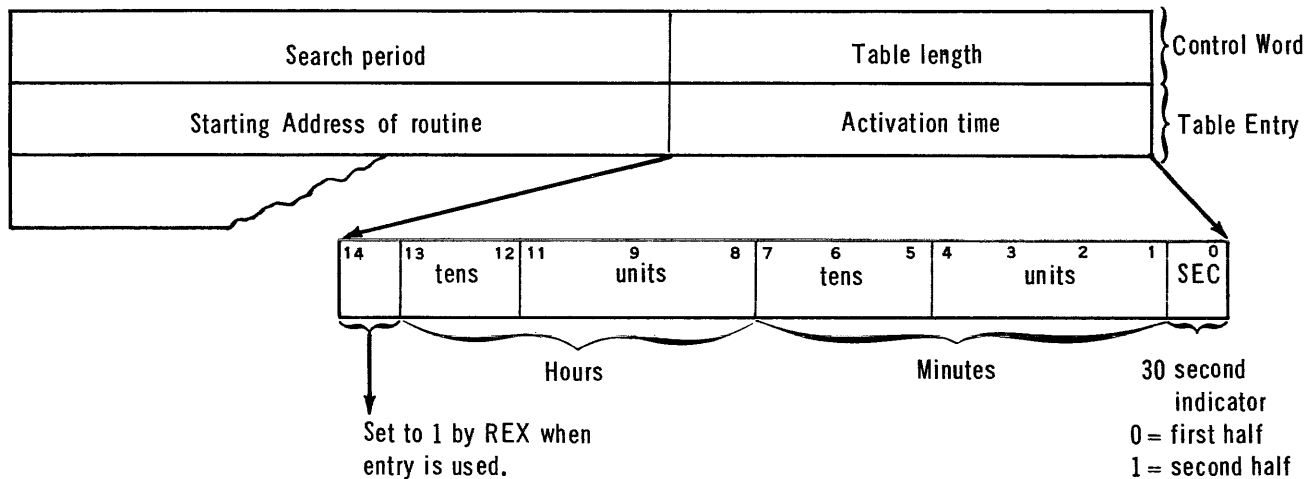
## 7. COMMUNICATION INTERRUPT TABLE

The real-time program must specify the memory set aside for storing communication interrupts which occur with REX or the real-time interrupt analysis subroutine in control. This specification is part of the initialization packet by which the real-time program describes to REX characteristics of its particular organization.

Interrupts are stored in communication interrupt tables as one word entries. The control of the specified table is left to REX which passes stored interrupts upon discovery in the switcher or upon request by the real-time program. See figure 6.

## 8. TIME-TABLE

REX will service a time-table to activate routines at specified times if this is assigned by the real-time program. It consists of a control word and one or more entries.



Search period is the desired interval between REX inspections. It is specified in units of 30 seconds. For example, a value of 4 would cause the table to be inspected every 2 minutes. Table length represents the number of locations to be inspected.

Each time the table is inspected REX will look at all entries seeking times equal to or less than present time. The starting address associated with each qualifying time will be marked as a return point. If an addendum storage element in which to mark the return point is not available, REX will pass over the entry during this search. Attempts to mark a return point will continue at each subsequent day-clock interrupt (regardless of search period) until a return point is marked. Failure to find a storage element will not cause addendum overflow.

Processed entries will be identified by setting bit position 14 to binary 1.

## B. Standard Peripheral Input/Output

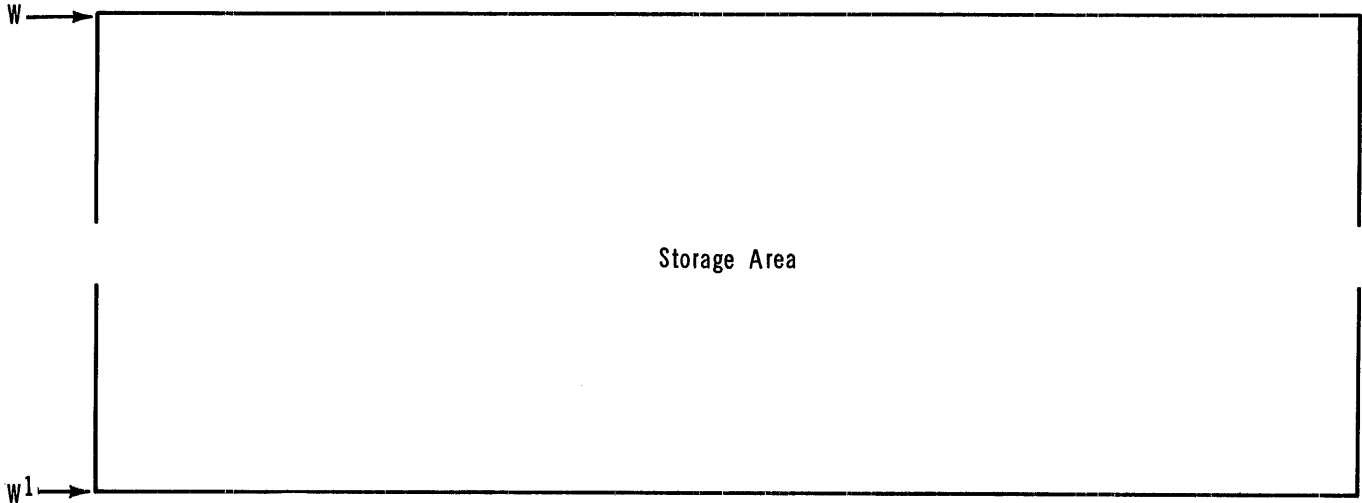
### 1. THE INPUT/OUTPUT REQUEST

A program requests input/output functions by an indirect arithmetic return jump to the address contained in the upper half of word 140. The jump instruction will be followed by a variable number of adjacent parameter words.

COMMUNICATION INTERRUPT TABLE

29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Address of next interrupt out																Address to store next interrupt in													
Start of table (W)																End of table (W <sup>1</sup> )													
No. of interrupts currently stored																Address of RTIAS entry applicable													

Within REX



Within Real  
Time Program

*One such table would exist for each category of interrupt (A common storage area might be defined for different kinds of interrupts. This would have the effect of eliminating the REX distinction between categories).*

Figure 6.



## 2. STANDARD PARAMETERS

A standard format has been established for input/output request parameters. It is applicable, with minor variations, to all peripherals except the console. The entry to REX and the proper parameters may be most conveniently inserted within program coding by the use of SPURT operators, their function, and their relation to REX are described in the SPURT manual under SPURT INPUT-OUTPUT OPERATIONS ASSOCIATED WITH THE REAL TIME EXECUTIVE ROUTINE.

LINE						Entry
1	SILRJP · U(140)					
2	Function Word					Parameter Words
3	N <sup>24</sup>	C <sup>20</sup>	P <sup>15</sup>	14		
4	<sup>29</sup> 1 1	<sup>23</sup> 0 <sup>21</sup>	<sup>18</sup> 3	<sup>15</sup> B <sup>n</sup>	14 XXXXX	
5	<sup>29</sup> 1 1	<sup>23</sup> 0 <sup>21</sup>	<sup>18</sup> K	<sup>15</sup> B <sup>n</sup>	14 YYYYYY	

- LINE 1** A Set Interrupt Lockout Return Jump to the Executive Input/Output Routine entry. (See Executive Entry Table).
- LINE 2** Specifies the function to be performed by peripheral equipment. A beginning drum address, UNISERVO number, or special information peculiar to the equipment type is specified. (SPURT output inserts channel number into bit positions 4-7 where applicable for use by the loader).
- LINE 3** N, C, and P specify information necessary to executive input/output routines. N is the number of parameter words excluding the return jump. C is the channel on which the function is to be performed. P will contain a code defining the type of peripheral addressed prior to program load time.

P	PERIPHERAL ADDRESSED
1	Drum
2	Disc
3	UNISERVO IIa Magnetic Tape.
4	Paper Tape Reader
5	High-Speed Printer
6	Card Reader
7	Card Punch
10	IBM Compatible Magnetic Tape.
11	Paper Tape Punch
12	UNISERVO III Magnetic Tape

At load time (when loading from complex relative format only) an identification assigned to the program in which the parameters appear is substituted for the peripheral type.

- LINE 4** This line consists of a fixed portion, which is an instruction, that will be executed by the REX input/output routine. The buffer control word contained in the memory location represented by  $XXXXX + B_n$  will be brought to the A register by the execution of this instruction.
- LINE 5** Similar to LINE 4. The search identifier will be brought to the A register from the memory location represented by  $YYYYY + B$  (with variable K designator) by the execution of this instruction.

Lines 1 through 3 are always present. Lines 4 and 5 are optional since one or both may be unnecessary for some functions such as a servo rewind. If words are present they must be in the form and order shown.

### 3. REGISTER REFERENCES

Only B1 through B6 may be referenced in a parameter packet. REX will use registers A, Q, and B7 with no provision for storage of values prior to submission of the input/output request.

Once a request has been submitted the parameter words cannot be altered in memory. The appropriate REX routines will access these words relative to the address supplied by the return jump instruction. They may, therefore, be changed only when request completion has been ascertained. One or several additional requests may be submitted by changing the values of registers B1 through B6, as this will not affect the parameter word values. The determination of request completion is explained under STATUS CHECKING.

### 4. INPUT/OUTPUT SUPERVISION

#### a. LISTING

When a request is submitted by a worker program the packet address will be placed on a request queue associated with the channel. Index-registers 1 through 6 will be saved within the submitting program's executive addendum in a storage element assigned to this particular request. Two queues will be maintained for each channel; one for real-time requests, the other for batch processor requests. Whenever a real-time request is queued a channel-critical indicator will be set. The capacity of the queues will depend on the type of peripheral equipment on the channel. When the queue for a channel is full, any program attempting to submit a request on that channel will be momentarily suspended until its request can be queued. Queues will be sufficiently large to make this condition infrequent.

For each program a count will be maintained and assigned to input/output requests as submission sequence. This order controls the use of return points among completed requests and also controls the order of initiation within a priority class. Priority among input/output requests will apply only to the real-time program and is optional. When a request is submitted the B register specified in the REAL-TIME INITIALIZATION TABLE may contain a priority number from 0 to 7. Zero is the highest priority. Higher priorities will be inserted in the queue above lesser priority requests and will consequently be initiated first. All batch program requests will be assigned the same priority without option. The priority code will be superimposed as the most significant bits of the chronological code.

## b. INITIATION

After a request is queued, an attempt will be made to initiate the request. If the channel is busy, control will be returned to the submitting program unless an interrupt has occurred during the processing of the request requiring action by the Switcher. If the channel is not busy, a request will be immediately initiated and control follows the same path.

In order to process input/output requests as quickly as possible, a queued request will normally be initiated as soon as a channel becomes not busy. An attempt will first be made to initiate a real-time request on that channel. If, however, the real-time queue is empty, a request from the batch processor queue will be initiated. Whenever a batch processor request is initiated, the channel-critical indicator will be cleared. Real-time requests will be initiated in order of submission within the class of priorities. Batch requests will be initiated on a strict first in-first out basis. When initiation results in successful completion, the status word associated with the completion will be recorded in the executive addendum of the submitting program within the storage element assigned the completed request.

## c. INTERRUPTS

(1) ACTION AT TIME OF OCCURRENCE. Whenever an interrupt condition exists, control will go to REX by virtue of a return jump instruction loaded in the interrupt entrance register. REX will determine the channel on which the interrupt occurred and the type of interrupt (internal or external). This information will be recorded in the interrupt record word.

The program interrupted will be examined next. If REX or the interrupt analysis subroutine of the real-time program (RTIAS) was interrupted, control will be returned to the point of interruption. This procedure makes REX and the interrupt analysis subroutine non-suspendible, and is necessary to prevent a second entry to the interrupted routine.

If the real-time program proper is interrupted, and the channel-critical indicator is set, the interrupt will be analyzed. If, however, the channel-critical indicator is clear, control will be returned to the point of interruption without analysis. Interrupts which are not analyzed at this time will be considered when the Switcher is next entered.

If a batch processor was interrupted, the interrupt will be analyzed. An exception is made to accommodate an input/output subroutine written to operate in a continuous mode from a single input/output request if any program other than REX or RTIAS is interrupted. For example, the card reader input/output subroutine accepts requests to read more than one card. An input/output operation is continuous whenever an external function is to be initiated upon receipt of an interrupt signifying successful completion of the previous function without the necessity of another request. The input/output subroutine will employ a special interrupt entrance which it will open when in this mode. This action by REX in conjunction with the input/output subroutine will enable initiation of the next external function and immediate return of control to the program interrupted.

The time otherwise spent going through the Switcher following analysis of the interrupt will be saved. These very brief excursions will be with interrupt inhibited.

(2) ANALYSIS. When an interrupt is to be analyzed, the captured value of P, along with interrupt values of A, Q and B1-B7 will be stored in the executive addendum of the interrupted

program, as a lost-control re-entry. A code word defining the type of interrupt will be placed in the A register and control given to the appropriate input/output subroutine for analysis of the interrupt.

The input/output subroutine will analyze the interrupt. It may initiate another external function because of the nature of the request, or because of the occurrence of an error from which it is attempting to recover. It may find the request satisfactorily completed, or it may find that it cannot recover from an error which has occurred. In any event, before exiting, it will place a status word in the accumulator to define the status of the request. REX will interpret this status word. If satisfactory completion has occurred, control will be given to Input/Output Initiation for initiation of the next request on the channel. If the request is still in progress REX will exit to the Switcher. REX takes special action in case of a interlock error or magnetic tape errors. For other errors the status word is simply stored in the executive addendum at this time within the storage element assigned this request.

#### d. ERROR PROCEDURES

When an error condition occurs all practical mechanical recovery measures are taken automatically by REX input/output subroutines without worker program attention. If the error condition persists it is ultimately reported to the responsible worker program via the status word. The worker program may wish to engage in further recovery attempts founded in personal knowledge of file structure or alternate sources of information (logical recovery measures).

The following information pertinent to error recovery will be available in transitory registers:

REGISTER	CONTENTS		
A	A status word indicative of the type of error.		
Q (upper)	The address of the input/output request.		
Q (lower)	The address following the CKSTAT packet (DONE address).		
B 7	An address aaaaa which refers to a three word area containing B-register values at the time the request was submitted. This area is in the following format:		
		U	L
	aaaaa	B1	B2
	aaaaa + 1	B3	B4
	aaaaa + 2	B5	B6

In light of mechanical recovery measures already taken, it is extremely unlikely that resubmission of the request which precipitated the error will result in recovery and, therefore, resubmission is not recommended as a worker program action.

(1) INTERLOCK ERRORS. Interlock errors, such as card jams, occurring on any subsystem during the processing of an input/output request may often be corrected by operator action. The executive routine will sense these potentially correctable errors, and inform the operator,

giving him the cause to the extent that it can be determined, and also the program and channel. It will then suspend operation of the source queue until operator action is taken. The error-producing request will remain in its queue positioned for reinitiation. The operator will have the choice of instructing REX to suspend the program, ignore the error, or reinitiate the request. By the reply I (ignore) the operator will cause the interlock error to be passed to the program which submitted the packet in the normal manner via the appropriate error address. At such time the program knows that operator intervention has been futile or waived, and any recovery procedure is left to the program. Following the "I" response, the source queue is automatically activated and initiation from it is resumed. If the interlock was on a magnetic tape subsystem, a logical lock-out is set for the interlocked servo.

(2) **NON-INTERLOCK ERRORS** *Magnetic Tape Subsystem*. In event of error a logical servo lock-out will be set. The request which caused the error will have assigned to it a status word specifying the error which has occurred. Requests subsequently presented for initiation (including any in the queue at time of error) will be assigned a special status word and will not be initiated. The special status word assigned will have the following form:

29		23	22	19	18	15	14		6	5		0	
	0	0		Channel		Unit		0	0	0		7	7

where channel and unit uniquely identify the error servo.

Rejection will continue until logical servo lock-out is released. Release is accomplished by entering the A register with an 8-bit channel-unit designation, setting Q to indicate release option and executing a SILRJP to L(142), Executive Entry F.

29							8	7	4	3	0
									Channel		Unit

Two release options exist. They are similar in that either will cause the lock-out condition to be terminated. They differ in treatment of accumulated, rejected requests.

**OPTION 1.** Jettison accumulated, rejected requests. CKSTAT need not be used for jettisoned requests. If a jettisoned request has already been interrogated by use of CKSTAT, the return point marked will be erased. In either case, the resultant condition is as if such requests had never been submitted.

To exercise this option set Q negative.

**OPTION 2.** Do not jettison accumulated rejected requests. If this option is exercised rejected requests may, in effect, be recovered. This is because CKSTAT returns will be via the error address with all information necessary to recreate the original at-submission environment.

To exercise this option set Q positive.

REX return will be to the instruction following the SILRJP. Normal operations may be resumed.

*Other Subsystems.* An error detected will be reported to the responsible program by descriptive status word. Initiation of queued requests will continue. Logical lock-out will not be used. See SPURT manual for Status Words and their meanings.

## C. Standard Peripheral Input/Output Status Checking

### 1. PURPOSE

The REX CKSTAT routine can be called on by a worker program to determine the status of a submitted input/output request.

#### a. Return Point Marking

An obvious purpose of the REX CKSTAT routine is to enable the worker program to determine whether or not a particular input/output operation has been successfully completed. The worker program specifies alternative locations to which control may be returned upon completion of the input/output request, an error address to be honored if the request was not completed successfully, and a normal address to be honored if the request has been completed successfully. The normal address (DONE), is always implied and is the location immediately following the packet generated by the CKSTAT mnemonic operator. The error address is specifiable.

#### b. Return Time Marking.

In addition, the REX CKSTAT routine allows the worker program to specify when control is next to be returned within itself. Three possibilities are allowed: 1) control will be next returned to this program only when the particular input/output request being checked has been completed, 2) control will be next returned when any previously initiated and CKSTAT interrogated request has been completed, or 3) control will next be returned as soon as possible regardless of the status of any input/output request. In the last instance, control will be transferred to a particular location indicated in the CKSTAT operator. This alternative allows the user to transfer control to a location not associated with a previously initiated input/output request which is specified by the EAS described below.

### 2. ACTIVATION OF CKSTAT ROUTINE

#### a. The CKSTAT Operator

The CKSTAT mnemonic operator is to be employed by the worker program to check the status of a previously requested operation. In response to the CKSTAT operator a return jump to the REX CKSTAT routine is generated along with the necessary parameters.

The CKSTAT operator with the various parameter entries and their uses are described in the SPURT Manual under CKSTAT. The actual packet generated by the SPURT operator takes the following form:

LINE

1	29	15	14	0	} Entry Param eter Words
	SILJP			L(140)	
2	29	15	14	0	
	RL + 1			EAS	}
3	29	15	14	0	
	00000			EA	
4	DONE				

LINE 1 Set interrupt lockout return jump to REX.

LINE 2 RL is the address of the request being CKSTAT interrogated.

EAS indicates the Executive Action Specifier. If this operand is omitted, EAS will be set to the code 00000. If the operand is TAKEOVER, EAS will be set to the code 00001. If a worker program label, it will be set to the allocated address.

LINE 3 EA indicates the error address. Use of STOPRUN or omitting the EA operand will cause this location to be set to the code 00001.

LINE 4 The address following the packet referred to as DONE is the eventual return address to be honored only if the request being checked was completed successfully. It must contain a legitimate instruction.

### 3. EXAMPLES OF CKSTAT USE

An example of CKSTAT use is included in the SPURT Manual related to the SPURT coding. The three options inferred in the use of the CKSTAT operator are explained below.

These uses are illustrated with diagrammatic examples.

(1) When a program needs the result of a particular input/output request and does not want control until that request has been completed, the programmer should CKSTAT that request and leave EAS blank. Control will not be given to his program again until that input/output request is completed, and then it will be returned at DONE or EA. (See Example 1, Appendix B).

(2) If a program has reached a point in its processing cycle where it cannot proceed until some outstanding input/output request has been completed, it may CKSTAT a submitted input/output request (for which CKSTAT has not been used) using TAKEOVER as EAS. (See Example 2, Appendix B). As a result, control is relinquished until any one of the previously submitted input/output requests, which have been interregated by a CKSTAT, are completed.

(3) A major use foreseen in designing CKSTAT is to accommodate a well organized set of logically independent "job" subroutines controlled by a master routine. This master routine would start a particular job subroutine, the subroutine might submit an input/output request and need to wait for its completion; it would CKSTAT specifying its master routine as EAS.

After the input/output request had been initiated, control would be returned to a label specified by EAS where a new job and input/output request may be submitted. This sequence of jobs could continue until there were no more jobs to start, or the worker program's memory was full, etc. At this point the master program would exit to REX-TAKEOVER (not part of CKSTAT; an independent operator which acts similarly to TAKEOVER in CKSTAT). This operator enables REX to return control to any of the job subroutines whose input/output request is completed. No bookkeeping on the part of the program is necessary for automatic returns from TAKEOVER. (See Example 3, Appendix B).

#### 4. LOGICAL CONSIDERATIONS

(1) *Order of Completion of Requests.* Among requests on several channels, the relationship between submission and completion cannot be predicted. This is because a queue may exist on one channel at time of submission and not on another, and also because peripherals differ in speed. Requests on a single channel, however, are initiated and completed in order of submission (except when the real-time program exercises its priority option, in that case, initiation is in order of submission within priority class).

(2) *Order of Return of Control.* Once a program has voluntarily given up control by using TAKEOVER, completed, CKSTAT interrogated, input/output requests for that program will be sought. If more than one request qualifies, control will be returned to that first submitted within the highest priority class.

(3) *Checking the Status of an Unsubmitted Request.* If CKSTAT is used referring to a label for which no input/output packet has been submitted, control will be returned to EA with a special status word of zero. If an input/output request is interrogated by a CKSTAT more than once, second and subsequent returns will be of this type.

(4) *No Return Points Marked.* If control is relinquished by a batch processor with no marked return points established, the program will be suspended pending operator intervention.

#### 5. PROGRAMMING CONSIDERATIONS

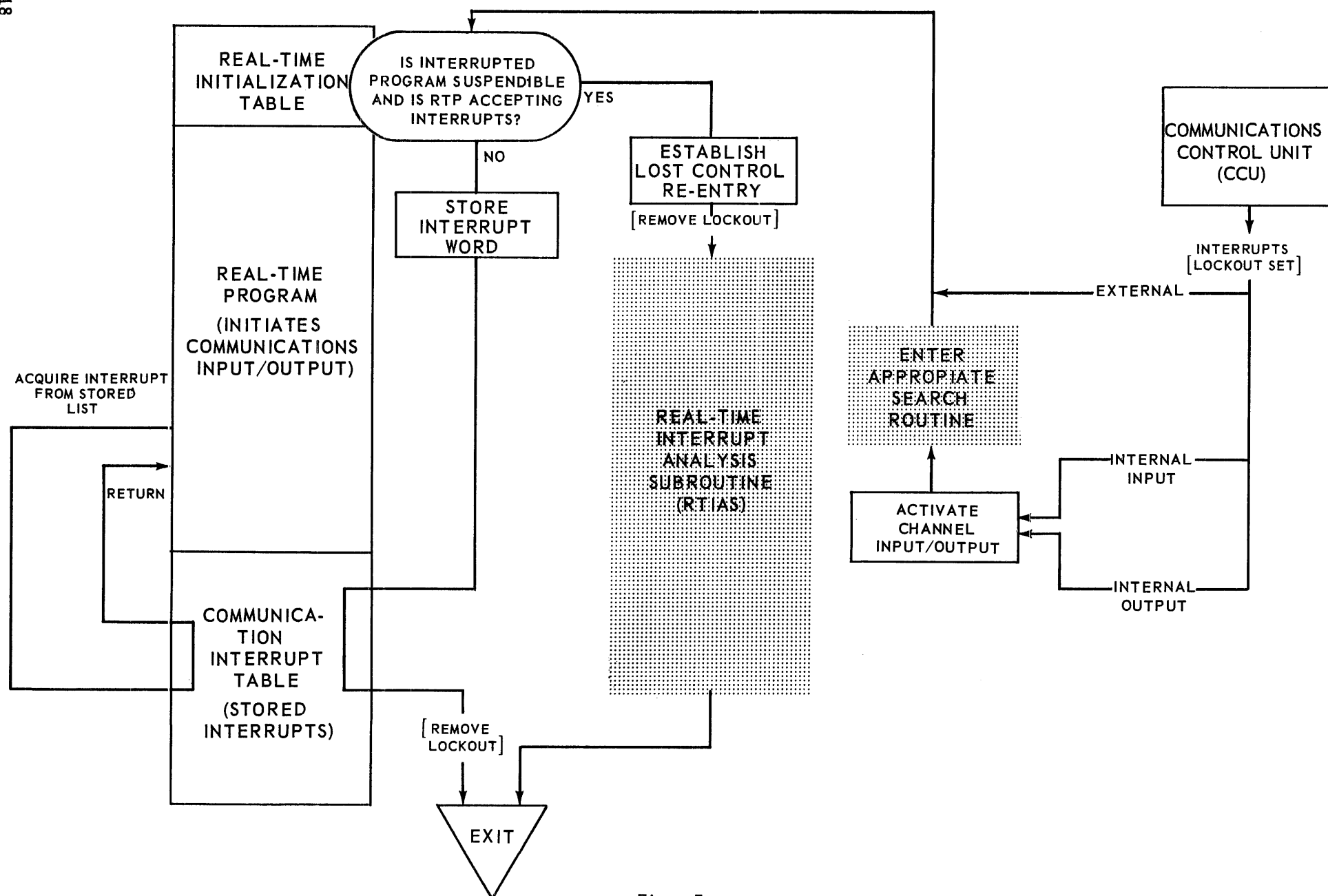
(1) An input/output request must be labeled to provide linkage for a subsequent CKSTAT.

(2) When a program has no other function to perform, it must relinquish control by using the operator REX . TAKEOVER.

(3) All input/output requests must be interrogated by a CKSTAT operator. Requests to the console printer input/output subroutine are not considered input/output requests, therefore it is not necessary to CKSTAT console requests.



## COMMUNICATIONS INPUT/OUTPUT



**Figure 7.**

## D. Communication Input/Output

Programming requirements for communications input/output will depend upon the type and configuration of the equipment for each installation. REX will provide optional programming in the form of generalized routines where applicable.

Figure 7 is a flow diagram of the major paths involving REX communications input/output. Input/output operations are initiated by the Real-Time Program. The operations following initiation are, in most cases, controlled by values contained within the Real-Time Initialization Table which is described in the section on Program Preparation.

### 1. COMMUNICATION INTERRUPTS

REX action at the time of interrupt will depend upon the type of interrupt.

#### a. INTERNAL INTERRUPT - INPUT

It is recommended that the use of this type of interrupt be limited to signalling the presence of segments of character trains of indeterminate length, where the entire character train is too long to be accommodated in a single buffer. Following this type of interrupt, REX will activate channel input logic, save operational registers A, B6, and B7, set B7 to channel number and transfer control to a search routine at the address specified in the Real-Time Initialization Table. Operational registers other than those saved by REX must be restored to at-entry values before exit if they are needed. Two options exist at this point.

(1) The user may code his own search routine. This will permit an evaluation of the importance of the interrupts. Some interrupts may be discarded at this point; this will obviate the need for presentation of interrupts to a RTIAS.

(2) The user may utilize the General Purpose Search Routine provided as a generalized routine. (See General Purpose Search Routine for description and entry requirements.) This routine will perform the following functions:

- (a) Inspect Buffer Control Registers (BCR) on a specified channel in quest of a terminated Buffer Control Word (BCW).
- (b) Access the upper half of the word immediately following the terminated buffer. This half word will be considered a "linking" address at which a substitute BCW defining an alternate buffer will be found. The real-time program is responsible for maintaining "linking" addresses and/or substitute Buffer Control Words.
- (c) Extract the substitute BCW and store it to the BCR thereby overlaying the terminated BCW.

When an interrupt word is to be synthesized upon exit from the first option, or whenever exit is made from the second option, REX will expect B7 to contain a value to be placed in the lower portion of a synthesized interrupt word. REX will synthesize an interrupt word comprised of time of occurrence in the upper half and the contents of B7 in the lower half. B7 could be an increment which would access the terminated BCR when added to a base address.

(3) If the interrupt is disposed of by the search routine, REX will expect B7 to contain 00000. REX exit will be to the point of interrupt.

#### b. INTERNAL INTERRUPT – OUTPUT

An interrupt of this type may be used to signal the termination of either poll or message output buffers. Following this type of interrupt, REX will activate channel output logic, save operational registers A, B6, and B7, set B7 to channel number, and transfer control to the search routine at the address specified in the Real-Time Initialization Table. Operational registers other than those saved by REX must be restored to at-entry values before exit if they are needed. Two options will exist at this point as described above for Internal Interrupt-Input.

#### c. EXTERNAL INTERRUPT

It is recommended that whenever possible external interrupt be used to signal input completion, poll completion or message output completion. External Interrupt is accompanied by a hardware-generated interrupt word which contains the address of the BCR in use. No search is required.

At time of interrupt REX will remove the hardware-generated interrupt word from the data lines.

### 2. SUBMISSION OF INTERRUPTS.

Once an interrupt word is available REX will determine if the interrupted program is suspendible. Two routines are permanently non-suspendible: they are REX and the Interrupt Analysis Subroutine of the Real-Time Program (RTIAS). The Real-Time Program itself is selectively non-suspendible depending upon values contained in the indicators in the Real-Time Initialization Table. These indicators may determine that a particular category of communication interrupt, or all communications interrupts are not wanted at this time.

If the program is suspendible, REX will establish a lost-control re-entry for the program and transfer control to the appropriate Real-Time Interrupt Analysis Subroutine. (See Real-Time Initialization Table.) The A register will contain the interrupt word and B1 will contain a count of interrupts of a similar kind. Before transferring control to the RTIAS, REX will remove the interrupt lock-out which has been set since the time of the interrupt.

If the interrupted program is non-suspendible, REX will record the interrupt word in the Communication Interrupt Table. The lock-out established at the time of interrupt will be removed and control will be returned to the point of interrupt.

### 3. ACQUISITION OF STORED INTERRUPTS.

The Real-Time Program may obtain a stored communication interrupt from the Communication Interrupt Table. The Acquisition Routine is entered by return jump (SILRJP) to the upper portion of word 143 (entry G of the Executive Entry Table). The type of interrupt may be specified as follows:

B1 set to:	Interrupt Requested:
0	Internal – Input
1	Internal – Output
2	External

Control will be returned to the instruction following the return jump. The register values that will exist at this time are:

- A        will contain the interrupt; or if register A is zero there were no interrupts of this type. The oldest interrupt will be presented.
- B1       will contain a count of the remaining stored interrupts of the type requested.

#### 4. INTERRUPT ANALYSIS BY THE REAL-TIME PROGRAM.

The real-time program must provide a closed subroutine (RTIAS) to analyze interrupts occurring on communication channels. The presentation of an interrupt to this routine may occur at time of interrupt or upon detection of an interrupt during an excursion through the Switcher.

##### a. Required Action of RTIAS

This subroutine must assume responsibility for interrupts presented to it. The functions performed by RTIAS may be as simple as setting a bit to show a particular CCU ready for output, or as complex as entering an input buffer, analyzing the importance of the input message and, as a result of this analysis, making an entry on one of a set of priority-oriented task queues.

If the interrupt signalled the presence of a segment of an indeterminate character train (Internal Interrupt - Input), this subroutine will be responsible for taking action necessary to assure that alternate buffering will exist at next interrupt. See Examples 4 and 5 of APPENDIX B for suggested schemes employing one buffer and two buffers for a central CCU.

##### b. Optional Action by RTIAS

The RTIAS may submit input/output requests. It may also check its addendum to see if a lost control re-entry exists (saved P register of Executive Addendum not equal to zero).

If a lost control re-entry does not exist, the subroutine may create one by storing desired register values to the appropriate addendum fields. Some routine within the real-time program may be activated in this way without waiting for an input/output completion to regain control. Whether or not a lost control re-entry is created, exit must be to the address defined by REX entry.

#### 5. THE GENERAL PURPOSE SEARCH.

The REX user will be provided with a general purpose search subroutine for optional inclusion within the real-time program. This subroutine will be capable of locating terminated buffer control words resulting in either input or output internal interrupts. It will search any channel using either of two search options.

Option 1 - Seek lower half of BCW greater than upper half.

Option 2 - Perform repeated masked comparison on the specified number of low-order bits of each BCW seeking equality with search key.

Option 1 is self explanatory. An illustration of how option 2 is related to buffer length and location is presented as Example 6, APPENDIX B.

a. Operation.

For each channel to be searched GPS will be modified to include 4 parameters: number of Communication Control Units, base address of CCU channel group, search key and search mask. Absence of the last two will imply L > U search.

Input Interrupt – GPS will inspect the input BCR of each CCU comprising the channel group. The search option dictated by channel parameters will be used. Inspection will start with the BCR having the lowest address and proceed through that having the highest. One comparison for each CCU is required.

Output Interrupt – GPS will treat all units on a channel as if they were a 5-level TFD CCU (the reason for this will become apparent when arrangement of BCRs is discussed). The search may require, in the worst case, three comparisons for each CCU. The search option dictated by channel parameters will be used to inspect buffer control registers according to the following plan:

1. Inspect all Output Buffer Control Registers
2. Inspect all Poll Buffer Control Registers
3. Inspect all "Special" Buffer Control Registers

In each case inspection will start with that BCR having the lowest address and continue through that having the highest. A find at any time, of course, terminates the search.

b. Choice of Search Option.

If the mean number of comparisons per find is greater than five, option two should be used. If less than or equal to 5, option one should be used.

c. Special Search

Personalized search subroutines tailored to peculiarities of a particular system may be substituted for GPS as long as REX interface rules are observed.

d. Arrangement of Buffer Control Registers.

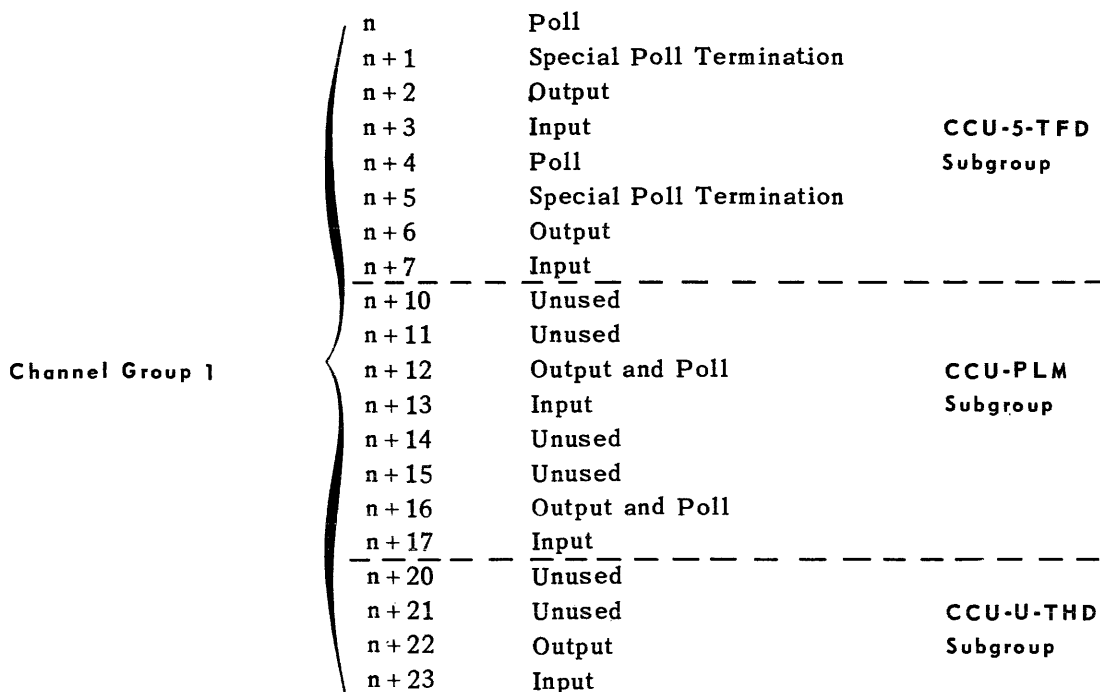
Each CCU will require four contiguous locations.

CCU TYPE	ADDRESS	USE
CCU-5-TFD	*00	Poll
(telegraphic 5-level	*01	Special Poll Termination
full duplex Subsystem)	*10	Output
	*11	Input

CCU TYPE	ADDRESS	USE
CCU-PLM	00	Not used
(Telephone party line	01	Not used
Subsystem)	*10	Output and Poll
	*11	Input
CCU-7-THD	00	Not used
(Telegraphic 7-level	01	Not used
half duplex Subsystem)	*10	Output
	*11	Input

“Address” represents bit positions  $2^1$  and  $2^0$  of the externally specified index generated by the CCU. Combinations marked with an asterisk are the only combinations which the associated CCU can generate. This implies certain address restrictions. For each of the units listed above, the Output BCR must be located at an address ending in 2 or 6; the Input BCR must be located at an address ending in 3 or 7; etc.

To use GPS the following arrangement should be observed. If some other search routine is incorporated it may dictate the arrangement.



n represents channel group base address and must be XXXX0 or XXXX4.

Channel group 1 would have as its base address  $00150_8$ .

This arrangement of subgroups is predicated on the assumption that “Special” buffers need not terminate and, therefore, need not cause interrupt. This being the case, no “Unused”

locations would be inspected during the output search. The reason "Special" buffers need not be used so as to cause interrupt is because either of the conditions forcing their use will, in itself, cause interrupt. These two conditions are a "Business" reply to a poll transmission (which means an input message is on the way) and an error condition which causes poll sequence termination and results in external interrupt.

#### E. Initiation of Interval – Timer Interrupts

REX will provide an interval – timer which may be used by the real-time program to obtain interrupts as a function of time. The real-time program defines the desired period by expressing it in milliseconds in B7. For example, if B7 contained 1001 this would define an interval of 9 milliseconds. When B7 contains the desired value, the interval – timer is set by a return jump (SILRJP) to the address specified in the upper portion of word 145 (entry K of the Executive Entry Table). This will activate a REX routine which will negate any previous timer definition, clear the timer word specified in the Real-Time Initialization Table, commence timing, and return control following the initiating return jump. When the timer runs out REX will increment the timer word by 1 and reset the timer.

If the indicator associated with the interval – timer entry to the RTIAS is set to accept interrupts, control will appear at the specified entry as soon as possible. Priority considerations for the interval – timer are discussed in the Switcher.

Once initiated the above process is continuous. The period may be redefined at any time. Defining a period of zero in B7 will terminate REX action.

#### F. Worker Program Voluntary Release Of Control

Several options are provided for conditional, voluntary release of control. These options supplement the CKSTAT operation in determining the logical flow of a program and accomodate contingencies concerned with this flow.

##### 1. SUSPENSION

SILRJP	U(142)
0 5 0 0 0	0 0 0 0 0

The submission of this packet causes REX to remove the requesting program from the switcher list. The program remains suspended until operator action either restarts or terminates the program. The fact of suspension will be typed out along with the P-Register and B registers 1 thru 6. This packet is generated by SPURT in response to the mnemonic operator REX . STOPRUN.

The effect of this packet is also achieved by an error address of STOPRUN with a CKSTAT operator.

##### 2. TERMINATION

SILRJP	U(142)
0 5 0 0 0	0 0 0 0 1

The submission of this packet causes REX to remove the requesting program from all tables and queues and releases the facilities assigned to the program. This operator is the normal means of indicating a complete run. This packet is generated by SPURT in response to the mnemonic operator REX . TERMRUN.

### 3. TEMPORARY RELEASE.

SILRJP	U(141)
--------	--------

This instruction is generated by SPURT in response to the mnemonic operator REX . TAKEOVER. The operator is used to release control until some previously submitted request is completed and the associated return point is eligible for control.

The effect of this instruction is also achieved by an EAS parameter of TAKEOVER with a CKSTAT operator. The order of returning control to completed return points is described under Standard Peripheral Input/Output.

### 4. EXCHANGE

SILRJP	L(143)
--------	--------

This operator allows a program which can proceed on the current path to trade its current position for a marked return point which is eligible for control. If no such return points exist when the proposal to trade is made the requesting program retains control one line beyond this instruction.

The real-time program may assign a priority to each input/output request; this priority is reflected in the order in which marked return points are used. The real-time program may desire to exchange the current position for a return point associated with a higher priority job. In this case the real-time program must, if exercising the priority option, set the priority B-register specified in the Real-Time Initialization Table to a number 0 thru 7 before an exchange, so that only marked return points associated with a higher priority task (lower number) will be considered.

The value set in the B-register is the priority of an acceptable trade. Thus, if the B-register is set to 2, the proposed exchange is for a marked return point of priority 0, 1 or 2. Only operational registers B1 thru B6 are preserved if an exchange is made. The priority assigned the new exchange-created return point will be the original B-register value assigned.

## G. The Switcher

### 1. PRIORITY CONSIDERATIONS.

The switcher routine provides for sequencing the operation of programs constituting the current memory mix. Consistent with the principle of expediting the real-time application, the switcher will settle any competition for priority in favor of the real-time program. Batch processors in memory will have an order of priority among themselves by virtue of the order in which they are considered for control. The programs will be arranged so that those with relatively little input/output time will be operated within input/output time of a higher ranking program. Among a given set of programs the order in which they are considered for control (scanned) will be determined by the load program, based on a programmer estimate of the percentage of time spent waiting for



input/output within a basic processing cycle. A new program to be initiated will be assigned a rank within the consideration order based on the above estimate (the higher the percentage, the higher the rank).

## 2. OPERATION.

The switcher provides the means by which REX gives up control. However, before control is given to a worker program interrupts which occurred during the time non-suspendible routines were operating must be considered.

Critical interrupts are considered first. As explained previously, critical interrupts result from a real-time program standard peripheral request in execution, occurrence of an interrupt on a standard peripheral channel that has a real-time request waiting to be initiated, occurrence of an interrupt on a standard peripheral channel that has a real-time request waiting to be initiated, occurrence of an interrupt on a communication channel, or interval-timer interrupt. Critical interrupts are considered in the following order:

- (1) Interval-timer
- (2) Communication Internal Input
- (3) Communication Internal Output
- (4) Standard Peripheral
- (5) Communication External

Presentation of interval-timer and communication interrupts may be selectively inhibited by the real-time program. (See Real-Time Initialization Table.)

After all wanted critical interrupts have been processed, the real-time program will be considered for control. If a lost control re-entry exists operational registers will be restored and control returned at the stored P value. If the real-time program has not lost control, completed standard peripheral input/output requests for the real-time program will be examined with respect to the CKSTAT operator. If a completed request exists, control will be returned to the real-time program at a point determined by its associated CKSTAT.

If control cannot be returned to the real-time program in any of the above situations, all non-critical standard peripheral interrupts will be analyzed at this time. Next the batch processors will be considered according to the priority scheme described above. For a particular program a lost control re-entry point will be sought first, and then, if one is not established, completed input/output requests. Control will be given to the program in these two cases in the same manner as for the real-time program.

If none of the programs currently in memory can operate, the switcher will be re-entered and the above procedure repeated until an exit to one of the programs can be achieved.

## 4. CONTINGENCY CONTROL

Contingency interruptions may occur as the result of machine controlled interrupts, such as fault and interval-timer interrupts. The overflow of storage areas, the unavailability of peripheral units, or logical faults within a program are another type of interruption. REX provides routines that will provide for these interruptions and permits the use of various options.

### A. Contingency Interrupts

#### 1. FAULT INTERRUPT

The result of execution of an illegal operation (00 or 77). The address of a fault routine may be specified in the lower half of word zero of a program's Executive Information Region. If a fault occurs while the program is operating, control will be transferred to this address. Operational registers will be the same as at the time of fault. The value of the P-register when the fault occurred will be stored in the lower half of word 4 of the program's Executive Information Region.

If no address is supplied, a faulting program will be suspended. A printout noting the suspension and specifying values in operational registers at the time of fault will be made.

#### 2. INTERVAL-TIMER INTERRUPT

Some of the more obvious uses of interval-timer interrupts are associated with communication polling in the real-time program. Access to a routine which will maintain this type of interrupt is therefore provided in the Real-Time Initialization Table.

The procedures for initiating and maintaining interval-timer interrupts are discussed under Executive Control (see INITIATION OF INTERVAL-TIMER INTERRUPTS).

## **B. Contingency Diversion of Program Flow**

### **1. ADDENDUM OVERFLOW.**

Overflow occurs when a function is requested of REX that requires use of an Addendum Storage Element and none is available. The entrance to a routine to recover from this contingency may be specified in word three of the Executive Information Region. Parameters upon entry are:

REGISTER	CONTENTS
A	Zero
B1-B6	Values existing at submission of the request precipitating overflow.
B7	Address of the request precipitating overflow.

The request has not been listed or initiated. A possible recovery would be to save the address of the request causing overflow and to use the Exchange Operation to establish a return point. When control appears at this return point the request may be resubmitted during a later pass through the switcher.

### **2. AN EXCESSIVE ACCUMULATION OF INPUT/OUTPUT REQUESTS WITHOUT ASSOCIATED STATUS CHECKING.**

A program may accumulate a maximum of 14 submitted input/output requests which have not as yet been interrogated by a CKSTAT. Submission of a fifteenth request will precipitate REX action as described for addendum overflow. To distinguish between this condition and addendum overflow, A will be set non-zero. Other registers will be set as previously described.

### **3. COMMUNICATION INTERRUPT TABLE OVERFLOW.**

This contingency applies only to the real-time program. It arises when a communication interrupt is to be stored in the table specified at real-time initialization and the table is full.

The recovery routine specified by the Real-Time Initialization Table will be entered by SILRJP with the following register values:

A	— interrupt word to be stored
	0 — internal input
B1	1 — internal output
	2 — external

The recovery routine must operate as a closed subroutine, and must prevent premature reentry by either operating with interrupt lockout set or employing recursive logic.

### C. Operator Contingency Interventions.

Operator entry is required to terminate or restart a suspended program. It is also required in interlock error situations and may be used to suspend a running program at any time.

#### 1. PROGRAM START

The program specified by the operand XX is to be started (or restarted) at the starting address or at any specified address with operational registers specified in order P, A, Q, B1 thru B7. Not all operational registers need be specified, but when any are specified the preceding registers in this sequence must be described.

	P	A	Q	B1	B2	
format:	PS	XX	ppppp	aaaaa	qqq	yyy zzz . . . . ⑤
	XX is program number					

#### 2. SUSPEND

The program specified is suspended from further operation by this operator entry. It remains inactive in memory until terminated or restarted by the operator. The status of the program and operational register contents will be typed out in response to the suspension. This data can be used to restart the suspended program.

format: SP XX ⑤

XX is program number

#### 3. TERMINATION

This function will terminate the designated program whether it is in a suspended or active mode. If a second operand (R) is included in the type-in, the program will be terminated and then repeated.

format for normal termination:

TP XX ⑤

format for termination and repeat:

TP XX R ⑤

XX is program number

#### 4. INTERLOCK RESPONSE

Interlock errors occurring during the processing of an input/output request may often be corrected by operator action. After a type-out by the input/output functional subroutine REX will type a request for operator response in the form:

REX ADVISE, Pxx, CHyy, Dxx

This requests operator action on the interlock on channel yy, for the request from program xx. The delay table entry xx has been set up to receive the response and in the interim will free the console printer. Operator response is by Dxx and may be one of three types.

- a) Dxx ☐ F ⑤ This entry informs REX the cause of interlock has been remedied and the request should be reinitiated.
- b) Dxx ☐ I ⑤ This entry instructs REX to pass the error back to the program in the same manner as other errors. The program knows that operator intervention has been futile or waived when its error address is reached.
- c) Dxx ☐ S ⑤ This entry imposes an error address of STOPRUN for this input/output request, frees the channel, and returns control to the program. The program will be automatically suspended when the error address is reached in the normal flow.

## 5. UTILITY SERVICES

The operator or a program may request that REX perform utility functions. Operator request is via the keyboard while programs submit special parameter packets; the requests will be listed in the Utility Request Table until executed by the appropriate drum-stored routine. Drum-stored routines are called by Utility Control, a drum control program that loads and initiates all REX drum-stored routines.

When the table holding utility requests is full, other requests will be rejected either by informing the operator when entry is attempted or by temporarily suspending a submitting program.

### A. Operator Requests

A function code identifies the action requested. The operands necessary to the function may be entered after the function code is acknowledged (typed back without rejection). Format of operands will be investigated by the activated utility routine after the entire message has been extracted from the utility request table. If the format is violated the utility routine may either solicit a correct entry or reject the message requiring a new operator request.

The format for operator requests is described with each function. (See APPENDIX C for general comments concerning console input/output).

#### 1. INSPECT DRUM

This function provides console output for small blocks of drum storage. Operands specify channel, initial drum address and the number of consecutive locations to be printed. The contents of drum locations are interpreted as octal numbers.

format: ID    xx    aaaaaaaa    nn S

channel number    starting address    locations to be typed  
(maximum 77)

#### 2. INSPECT CORE

Identical to ID except printout is from core memory.

format: IC    aaaaa    nn S

starting address    locations to be typed  
(maximum 77)

### 3. CHANGE DRUM

This function provides the ability to set the contents of specified drum locations. Output on the console printer shows the contents of the changed locations both before and after. Positive or negative, decimal or octal, constants may be entered. Decimal constants are of the form:

(-) xxxxxxxxD

Old contents are printed decimally, if the input was decimal. If input was negative and old contents are negative, they are printed as negative. Octal input results in straight octal output.

A maximum of three consecutive locations may be set with one CD request. The request is terminated with ⑤ immediately after the last constant.

format: CD ☐ XX ☐ AAAAAAA ☐ C<sub>1</sub> ☐ C<sub>2</sub> ☐ C<sub>3</sub> ⑤

channel for      first address      first      second      third con-  
magnetic drum      to be changed      constant      constant      stant

### 4. CHANGE CORE

Identical to CD except that storage is to core memory.

format: CC ☐ AAAAA ☐ C<sub>1</sub> ☐ C<sub>2</sub> ⑤

starting address      1st      2nd      (maximum 3  
constant      constant      constants)

### 5. PRINT DRUM

This request activates a routine to print the contents of blocks of drum storage on the high-speed printer, or alternately to dump these print images on a magnetic tape one line per record.

The output will be either as Fieldata characters (5 characters per location) or as octal coded information. The output format is eight words per printer line. The format is fixed, regardless of the requested starting address; that is, the left-most location is always an address ending in 0 and the right-most word is the contents of an address ending in 7.

Octal Printer format:

HEADER:	LOCATION	0	1	7
1st line	YYYYYYY0	XXXXX XXXXX	XXXXX XXXXX	XXXXX XXXXX

Fieldata Printer format:

HEADER:	LOCATION	0	1	7
1st line	YYYYYYY0	XXXXX XXXXX	XXXXX	

In the fielddata format each character is checked for a 77 code which terminates a print line. Whenever this code is found the former print line is repeated with spaces filled in up to the 77 code. By this process the 77 code appears as a "break" and all characters are printed.

format:

PD	<input type="checkbox"/>	XX	<input type="checkbox"/>	bbbbbb	<input type="checkbox"/>	eeeeee	<input type="checkbox"/>	y	<input type="checkbox"/>	zz	<input type="checkbox"/>	v	<input checked="" type="checkbox"/>
		channel		beginning		ending		O-octal		Output		Sxx-servo	
		for drum		drum address		drum		format		channel		unit	
				up to 8		address		C-field				Px-printer	
				characters		up to 8		data				Unit	
						characters		code					

## 6. PRINT CORE

Identical to PD except printout is from core memory.

PC	<input type="checkbox"/>	bbbb	<input type="checkbox"/>	eeee	<input type="checkbox"/>	y	<input type="checkbox"/>	zz	<input type="checkbox"/>	v	<input checked="" type="checkbox"/>
		beginning		ending		O-octal		output		Servo Unit-Sxx	
		core address		core address		C-field		channel		Printer Unit-Px	
						data					
						code					

## 7. SITE UTILITY


Provision is made for the conveyance of parameters to a utility routine to be activated by Utility Control. Once activated it will bear the same relationship to REX as do REX utility and load routines. The entry to any utility routine is by return jump to the first instruction. The address of the Utility Table will be contained in B1. The first word of the entry which resulted in activation of the Site Utility routine will be located at (B1) + 1.

The routine loaded by Utility Control may itself be a control program capable of calling other utility programs as dictated by the parameters conveyed. A site utility hierarchy may be created in this way to provide personal utility functions.

The following limitations apply to site utility routines:

- (1) While a site utility routine is being executed no REX routine in the utility or load family can operate.
- (2) The size of the site utility conglomerate in core memory cannot exceed 400D consecutive locations, the first of which is that to which entry was initially made.
- (3) Site utility may use a maximum of two addendum storage elements.
- (4) Site utility is prohibited from making a program utility request of REX.
- (5) Before a site utility exits, the function code of the parameter entry must be set to 77. Exit will be to the address provided by the return jump entry.

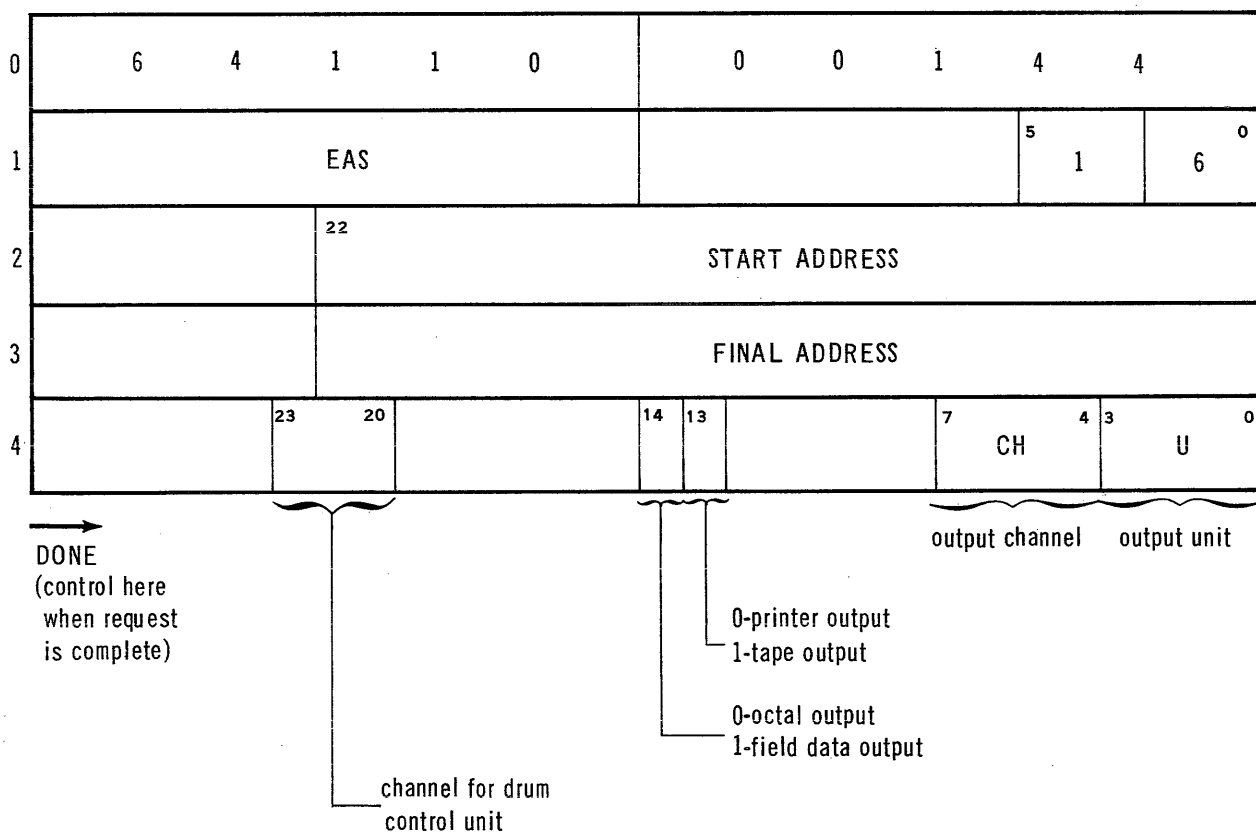


SU ☐  5

Section C of Contingency Control and Sections A and B of Console Control treat of other operator entries.

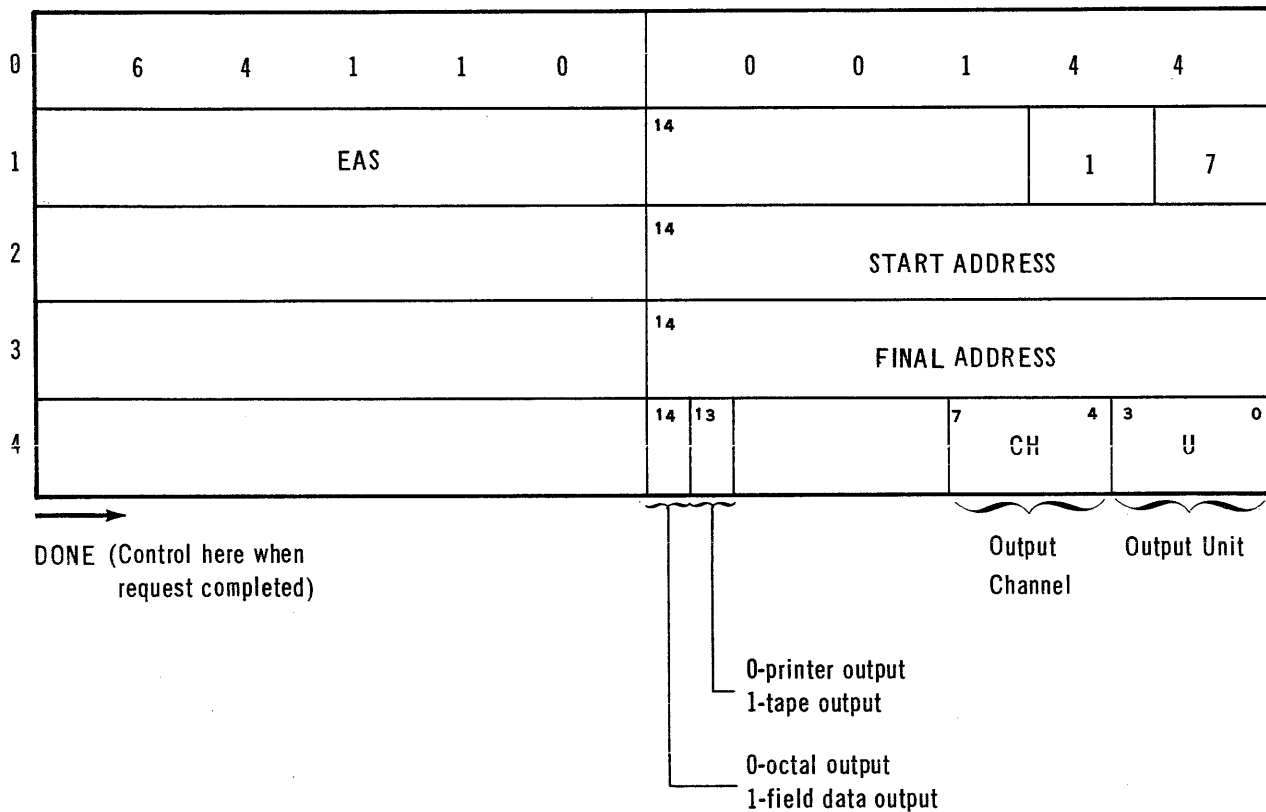
## 1. PRINT DRUM

This internal request achieves the printout of blocks of drum storage in the format described under the operator request Print Drum. If EAS is zero the program will remain suspended until the request is complete. If EAS is an address (not zero), control will be transferred to the specified address immediately.



## 2. PRINT CORE

This internal request achieves the printout of blocks of core storage in the format described under operator request Print Core. If EAS is zero the program will remain suspended until the request is complete. If EAS is an address (not zero), control will be transferred to the specified address immediately.



## 3. ASSISTANCE IN ESTABLISHING RERUN DUMP

The purpose of this request is to provide REX with information necessary to establish a rerun dump.


REX will perform the following actions:

- (1) Write onto the drum an image of the core memory assigned to the requesting program.
- (2) Compose and write onto the dump servo a bypass sentinel followed by REX facility and control information.

- (3) Write specified peripheral areas onto the dump servo using the core memory of the requesting program area as a transfer buffer.
- (4) Reload the drum-stored program image, write it onto the dump servo and follow it with another bypass sentinel.
- (5) Perform an identifying console type-out.

When control is returned to the requesting program the A register will contain a count of the number of blocks written onto the dump tape (including bypass sentinels). The Q register will contain a count of the number of words written (including inter-record gaps). If REX could not perform the requested dump because of a non-static program condition or because of tape error, A will be set to zero.

a. Format

0	6      4      1      1      0					14      0      0      1      4      4					
1						14      1      5					
2	octal count for dump identifications					14      dump servo		7      chan		3      unit      0	
3	restart address if dump is used					14      nnnnn					
4	chan		23      starting address of area one								
5	P		23      ending address of area one								
											
chan		23      starting address of area n									
P		23      ending address of area n									
						14      number of tape file designs					
address of tape file design 1						14      address of tape file design 2					
address of tape file design n						14					

DONE→ control returned here when request completed

nnnnn represents the number of peripheral storage areas to be saved. Zero would imply save none. 77777 would imply save all relocatable area.

P represents peripheral type code: 1 drum, 2 disc

File design is the first address of the standard (COBOL) file design.

b. Type-out

<u>REX</u>	<u>RERUN DUMP ccc OF PROG.</u>	<u>XXXXXXXXXX ON CHyy, Svv</u>
Standard Utility Routine Identifier	Dump Identification	Program Permanent identification

4. UTILIZATION OF A RERUN DUMP.

A variation of the load type-in (LD), which was explained under Operational Control, will be used to reload a particular rerun dump. REX will locate the specified dump, reload peripheral areas, reload core memory, type instructions for mounting tapes (including the program tape if applicable), position each tape according to the block count specified within its file design and return control to the restart address. A and Q will be set as at dump time. Format for rerun is as follows:

LD ☐ T ☐ ch ☐ serve ☐ 7474747474 ☐ ccc ☐ B ☒

ch                      channel number

ccc                      dump identification specified in printout

REX will produce a type-out in the following format:

REX	<u>MOUNT REEL nn OF X . . . . . X ON CHyy, Svv</u> <div style="display: flex; justify-content: center; gap: 50px; margin-top: -10px;"> <div style="text-align: center;"> reel number </div> <div style="text-align: center;"> 15 character file identification </div> </div>
ENTER M WHEN ALL TAPES MOUNTED. ACCEPT Dxx	

A console entry of the character M will initiate the rerun.

## **6. PROGRAM PREPARATION**

### **A. Source Language**

Worker programs to be run under control of REX may be written in either SPURT language or COBOL, which is translated to SPURT language during compilation.

Use of SPURT input/output and REX-oriented macros will cause REX-required parameters and entry jumps to be generated during assembly. Generated parameters are assigned modification codes that will allow the load program to recognize and suitably alter them when modifying the worker program to running form.

A programmer may, of course, code parameters and entry jumps in lieu of using the aforementioned macros. If he does this, he must assume responsibility for modifying these parameters to running form himself.

### **B. USE OF JUMP KEYS**

Worker programs should not be dependent upon jump key settings. A substitute, unsolicited operator type-in, is provided.

### **C. Use of Conditional and Unconditional Stops**

Worker programs should not stop the central processor. The following substitutes are provided:

- (1) A worker program may suspend itself pending operator intervention by using either of the SPURT mnemonics REX · STOPRUN or ACCEPT.
- (2) A worker program may terminate itself by using the SPURT mnemonic REX · TERMRUN.

## D. Standard Locations

### EXECUTIVE INFORMATION REGION

In order to facilitate exchange of information between worker programs and REX, the first five locations relative to the initial address of a worker program, or the control segment thereof, are assigned specific uses. These locations will be referred to as the Executive Information Region. Special additional information is required of the real-time program. This information is made available to REX via an Initialization Table defined by the real-time program as part of its initialization procedure.

#### EXECUTIVE INFORMATION REGION

WORD	UPPER HALF	LOWER HALF
0	Worker program starting address.	Entrance address of fault recovery routine. (note 1)
1	Address of relocatable area bounding addresses. (note 2)	No. of Addendum Storage Elements to be provided. (note 4)
2	not assigned	
3	Address of unsolicited operator entry indicator word.	Entrance address of addendum overflow recovery routine. (note 1)
4	Starting Address of parameter storage area. (note 3)	P-register value at time of fault.

#### NOTES:

- Optional. If zero, REX will automatically suspend the program pending operator intervention.
- Inserted by REX. Bounding core and bounding relocatable drum area addresses are preserved by the Loader within a program's addendum. They will remain there until the program performs a console type-out of more than 50 characters.

This is word number  $54_8$  of the executive addendum (see Executive Addendum). This address -  $54_8$  will access the first word (word 0) of the addendum. This is associated with possible RTIAS establishment of a lost control re-entry.

Ending Core		Beginning Core	
29	Chan	24	Beginning Drum
		Ending Drum	

- Optional. It is required only if operational parameters are to be conveyed to the program at load time.
- An Addendum Storage Element is a 10-word temporary storage within a program's executive addendum. One element is placed into use each time a program performs one of the following operations:
  - (1) Submits an input/output request.
  - (2) Uses ACCEPT
  - (3) Requests a Real-Time Extension

- (4) Requests a Batch Load
- (5) Calls a segment
- (6) Requests a core or drum memory dump
- (7) Has a time table routine started.

An element is released for reuse each time the result of the operation which originally caused its use is reported to the program. That is, whenever control appears at the DONE (or, if applicable, ERROR) address associated with the operation.

Consider, for example, a typical input/output sequence. When the request is submitted a vacant storage element is located. B-register values and bookkeeping information are saved therein. When the request is processed the routine-generated status word is saved within the same element. When the status of the request is interrogated, the status word is reported to the program and the element is freed. (Note, however, that if CKSTAT is used to mark a return point, the element is not freed but will remain in use until request status is subsequently reported in response to program use of TAKEOVER).

It can readily be seen that if each time a program requested an operation it chose to wait until that operation was complete before requesting another, only one storage element would be required. If the program was more complex, additional elements would be required.

- 5. Unused fields should be set to zero.

#### *REAL-TIME INITIALIZATION TABLE*

The purpose of this table is to provide the real-time program with a means by which it may specify certain options. (See Figure 8.) The table may be thought of as a packet of information which is conveyed to REX and is preceded by a SILRJP to entry J (lower portion of word 144). When REX has absorbed the parameters contained within the table, control will be returned following the table.

#### **E. Segmentation**

Due to limited core space, it may be necessary for a programmer to break a program into segments. There must be one control segment and there may be any number of secondary segments. The control segment will remain in core memory during the entire time a program is running. At any given time one, and only one, secondary segment may share core memory with the control segment. It will be adjacent to and immediately following the control segment.

REX will load secondary segments as requested by the program. The SPURT mnemonic "LOAD" may be used to make requests. Segments will always be called from initial form.

Segment delineation and inter-segment communication are described in the SPURT Programmer's Reference Manual.

#### **F. Facility Requirements**

The facility requirement of a program must be declared by the programmer. For declarative operators and usage rules see the SPURT Programmer's Reference Manual.

## REAL-TIME PROGRAM INITIALIZATION TABLE

WORD

0

1

2

3

4

5

6

7

8

9

		5	13
Entrance to RTIAS for external interrupt*	Start address of time table		
Entrance to RTIAS for internal input interrupt*	Entrance to closed subroutine to effect search for internal input interrupt		
Entrance to RTIAS for internal output interrupt*	Entrance to closed subroutine to effect search for internal output interrupt		
Starting address for communication external interrupt table	Final address for table		
Starting address for communication internal interrupt input table	Final address for table		
Starting address for communication internal interrupt output table	Final address for table		
Address of overflow contingency	B register used to indicate I/O request priority. 0 implies no priority used.		
ENTRANCE TO RTIAS for Interval-timer Interrupt*			
Not assigned			

Figure 8. Real-Time Program Initialization Table

\* The upper half word of each entry is an indicator. If a particular indicator is zero, it means the real-time program will permit itself to be suspended for interrupts of that type. If a particular indicator is non-zero, the real-time program is currently not permitting itself to be suspended for interrupts of that type. Communication interrupts will be stored by REX until the indicator is cleared or until a specific request via executive entry G is made. Interval-timer interrupts will be recorded by incrementing the location immediately preceding the interval-timer entry defined in the upper portion of word 8. This timer word will therefore contain a count of timer interrupts.



## APPENDIX A

### BASIC PROGRAM FORMATS

The following conventions are equally applicable to all formats:

- (1) All unused words, fields and portions of fields must be filled with binary zero.
- (2) Check sums are formed by adding together all words in a block. Full word (30-bit) adds are performed.
- (3) Magnetic tape is written at high density.
- (4) Identification information, such as "PROGRAM NAME", "PROGRAMMER" and "DATE", is in Fieldata code with left justification.
- (5) SPURT places the output number in the library number field.

## Modification Codes

### 1. COMPLEX RELATIVE FORMAT

A six-bit modification code is associated with each word of a program in complex relative format. Each instruction word modified will be stored in a location determined by the "storage counter." This counter will initially be set to the current base and incremented (or reset in accordance with modification codes) as loading progresses.

#### (a) Miscellaneous Modifications

- 00 No modification
- 01 End of Segment
- 02 End of Program

#### (b) Modification by the Current Base

- 03 Add current base to lower half
- 04 Add current base to upper half
- 05 Add current base to upper and lower half

#### (c) Modification by Control Segment Base

- 06 Add control segment base to lower half
- 07 Add control segment base to upper half
- 10 Add control segment base to upper and lower half

#### (d) Modification by the Control Segment Base and the Current Base

- 11 Add current base to lower half and control segment base to upper half
- 12 Add current base to upper half and control segment to lower half

#### (e) Changes to the Storage Counter

- 13 Add the lower half to the counter. Do not store any word. This is for "Reserve" instruction. No. of locations skipped will be equal to the number in the lower half of the word.
- 14 Add the control segment base to the lower half. Set the counter to the new value. Do not store any word. The words following this code will be stored in the control segment after modification. This creates the necessary entry tables in the control segment for intersegment communication.

#### (f) Modification of Peripheral Equipment References

- 15 Channel in bits 4-7, unit in bits 0-3. Substitute assigned channel and unit. Relocatable system. If an assignment has not been made, substitute a word of 7's.
- 16 Same as 15, fixed system.
- 17 Channel in bits 20-23, peripheral type in bits 15-19. Substitute assigned channel and change peripheral type to program number. Relocatable system.\*
- 20 Same as 17, fixed system.
- 21 Channel in bits 20-23. Substitute assigned channel. Add control segment base to lower half. Relocatable system.\*
- 22 Channel in bits 20-23. Substitute assigned channel. Add control segment base to lower half. Relocatable system.\*

- 23 Channel in bits 20-23. Substitute assigned channel. Add current base to lower half. Relocatable system.\*
- 24 Same as 21, fixed system.
- 25 Same as 22, fixed system.
- 26 Same as 23, fixed system.
- 27 Channel in bits 24-29, drum address in bits 0-23. Relocatable system.

\* If channel has not been assigned, substitute channel 17.

## 2. SIMPLE RELATIVE FORMAT

A 3-bit modification code is associated with each word of a program in simple relative format. Each instruction word will be modified and stored in a location determined by the "storage counter." This counter will initially be set to the assigned base and incremented (or reset in accordance with modification codes) as loading progresses.

- 0 No modification
- 1 Add the base address to the lower half.
- 2 Add the base address to the upper half.
- 3 Add the base address to both halves.
- 4 Execute the instruction (used to reset the storage counter).
- 5 End of program.

### A. Absolute

A program in absolute format is comprised of an Identification Record, one or more Instruction Records, and an End-of-Program Sentinel.

#### 1. IDENTIFICATION RECORD

7	4	7	4	7	Library Number		
PROGRAM NAME (10 characters)							
PROGRAMMER (10 characters)							
DATE (10 characters)							
Memory Requirement of Program					0		0
7	4	7	4	7	Library Number		

(9-wd block)

## 2. INSTRUCTION RECORD

Ending core address (U)	Beginning core address (L)
Check Sum for following block	

Instruction words to be read into the core  
memory area delimited by U and L.

$$\text{Block length} = (U - L) + 1$$

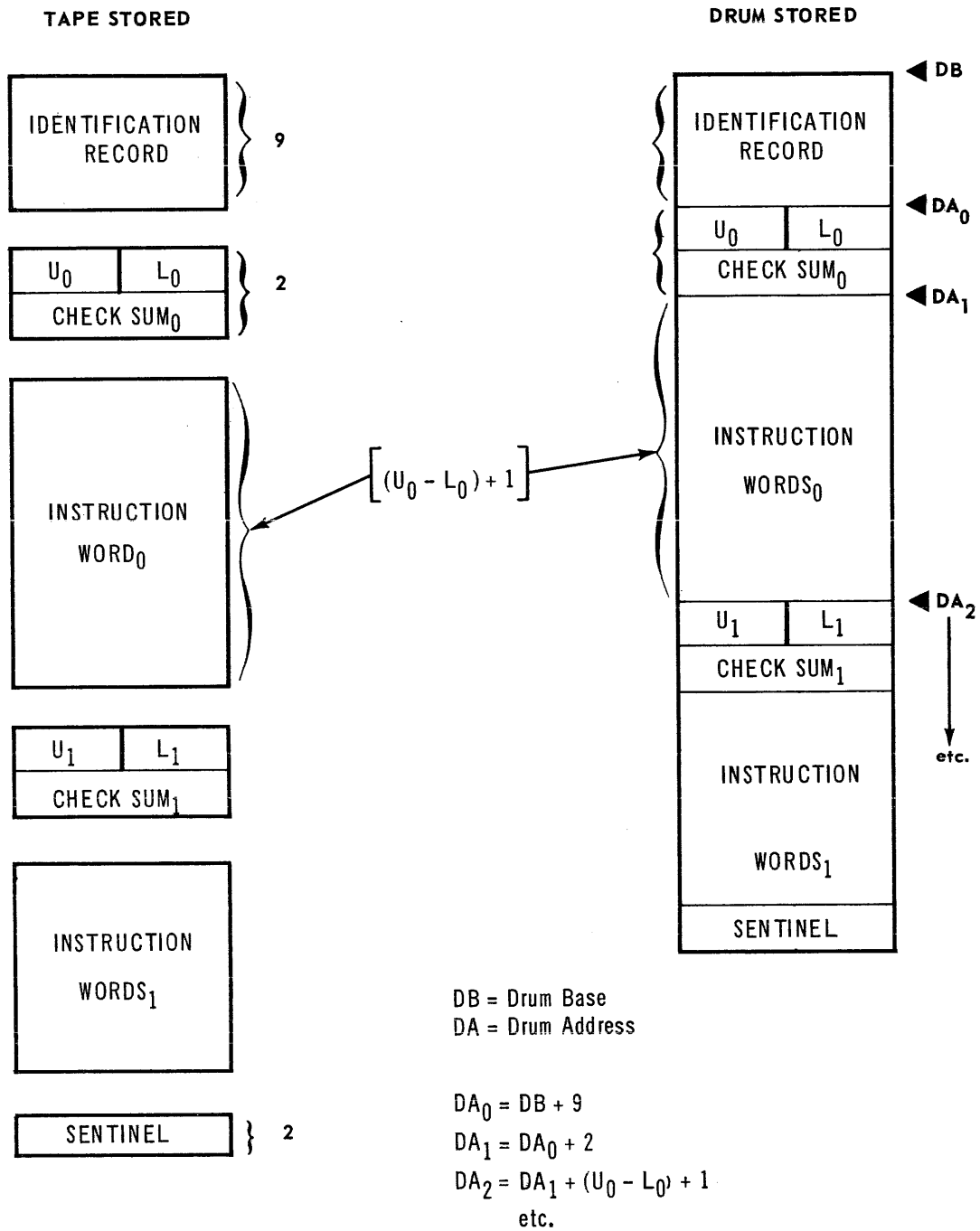
- (1) This record is comprised of two blocks; the first of 2 words, the second of any length.
- (2) When outputed by SPURT the length of the second block is less than or equal to 256 words.

## 3. END-OF-PROGRAM SENTINEL

1	2	2	3	1	1	2	4	1	3
1	1	3	2	2	2	2	5	0	5

(2-wd block)

#### 4. STORAGE FORMAT



A drum-stored program occupies a continuous drum area. There are no inter-record gaps or inserts.

### B. Simple Relative

A program in simple relative format is comprised of an Identification Record, one or more Instruction Records and an End-of-Program Sentinel.

## 1. IDENTIFICATION RECORD

7	4	7	4	7	Library number
PROGRAM NAME (10 characters)					
PROGRAMMER (10 characters)					
DATE (10 characters)					
Memory Requirement of Program					2
7	4	7	4	7	Library number

(9-wd block)

## 2. INSTRUCTION RECORD

mc <sub>1</sub>	mc <sub>2</sub>							mc <sub>9</sub>	mc <sub>10</sub>
	mc <sub>12</sub>	mc <sub>13</sub>							
instruction word 1									
instruction word 2									
								mc <sub>50</sub>	
instruction word 50									
Check Sum									

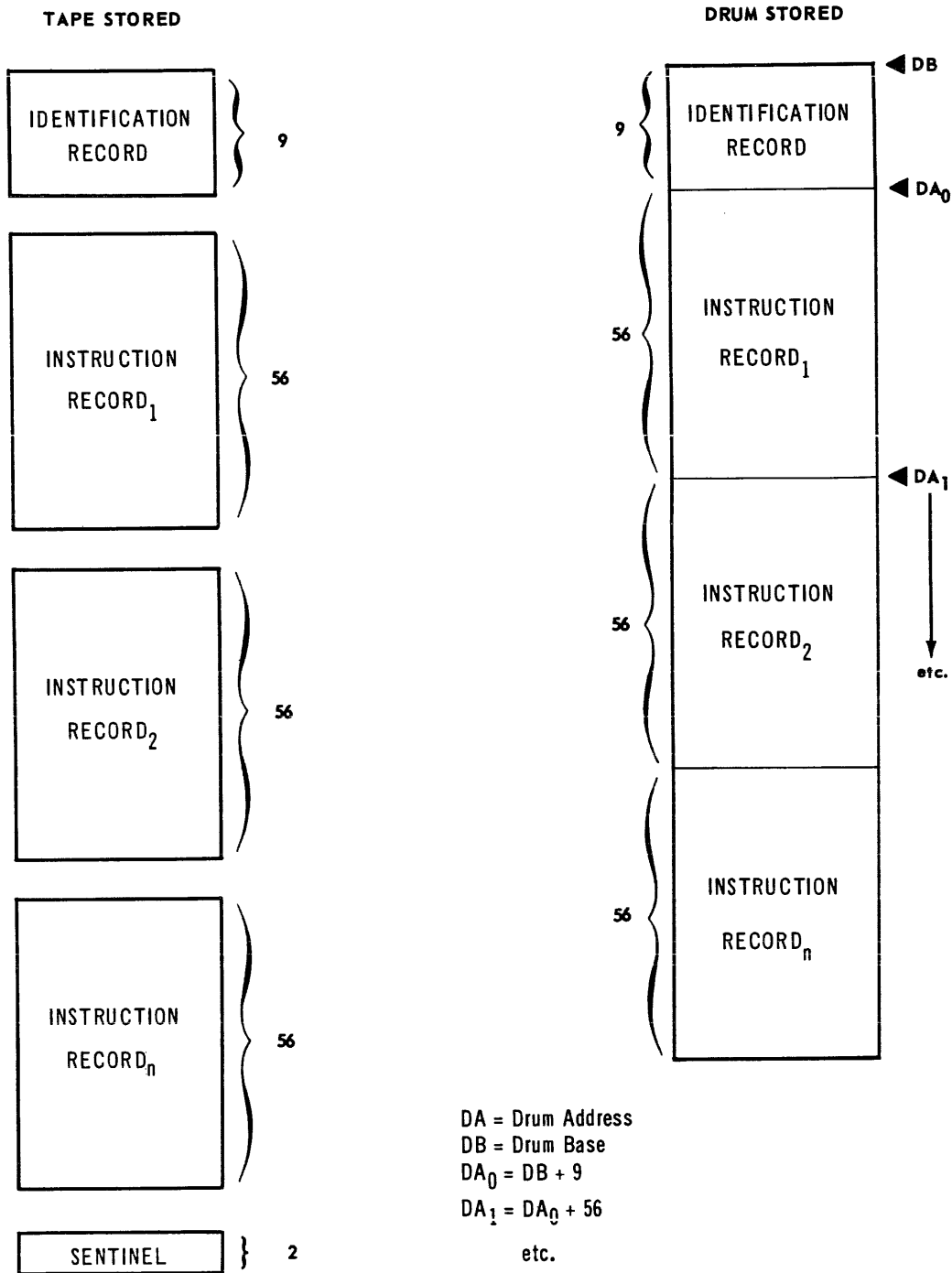
(56-wd block)

### 3. END-OF-PROGRAM SENTINEL

1	2	2	3	1	1	2	4	1	3
1	1	3	2	2	2	2	5	0	5

(2-wd block)

### 4. STORAGE FORMAT



A drum-stored program occupies a continuous area. There are no inter-record gaps or inserts.

### C. Complex Relative

A program in complex relative format is comprised of an Identification Record, a Facility Record, a Segment Description Record (if segmented), a File Description Record (if desired), a Control Segment Record, one or more Secondary Segment Records (if segmented) and an End-of-Program Sentinel.

## 1. IDENTIFICATION RECORD

7	4	7	4	7	Library number
PROGRAM NAME (10 characters)					
PROGRAMMER (10 characters)					
DATE (10 characters)					
					3
7	4	7	4	7	Library number

(9-wd block)

## 2. FACILITY RECORD

25	1	27	26					3	2	1	0
28	No. of return points			24	23	No. of segments		15	14	Desired core memory supplement	
Length of longest segment						Length of control segment					
not assigned											
(Fixed requirements)											
<div> <div>14</div> <div>End sentinel (binary ones)</div> </div>											
(Relocatable requirements)											
<div> <div>14</div> <div>End sentinel (binary ones)</div> </div>											
CHECK SUM											
1										1	0

- (1) The record is comprised of one 51D-word block.
- (2) Program length of a non-segmented program will appear in "Length of control segment" field.



a. Fixed Requirement Entries

FORMAT A

	23	20	19	15	
	c		p		

Format A will apply to standard peripheral subsystems without units and communication subsystems.

“c” represents channel number.

“p” represents peripheral type code.

FORMAT B

	23	20	19	15	14	No. of units required	9		5	No. of units desired	0
	c		p								
								7	channel	4	3
									unit		0
									channel		unit
									channel		unit

Format B will apply to standard peripheral subsystems with units.

These subsystems are:

- (a) Magnetic Tape
- (b) Card
- (c) Paper Tape
- (d) High Speed Printer

“No. of units required” expresses the minimum number of units of this peripheral type necessary to run the program. “No. of units desired” expresses the maximum number which could be utilized if available. Both entries are required.

Following the “summary” line will be one “detail” line for each unit. The number of detail lines must equal “No. of units desired.”

b. Relocatable Requirement Entries

FORMAT C

	23	20	19	15	
	c		p		
No. of words required					
No. of words desired					

A relocatable drum area requirement will be expressed in Format C. “No. of words required” defines the minimum number of contiguous drum locations necessary to run this program. “No. of words desired” defines the maximum number which could be utilized if available. Both entries are required.

Format B will apply to relocatable standard peripheral subsystems with units.

### 3. SEGMENT DESCRIPTION RECORD

29	2	27	26				3	2	2	0
				14	Length of Segment 1					
				Length of Segment 2						
				Length of Segment 3						
				etc.						
CHECK SUM										
2									2	

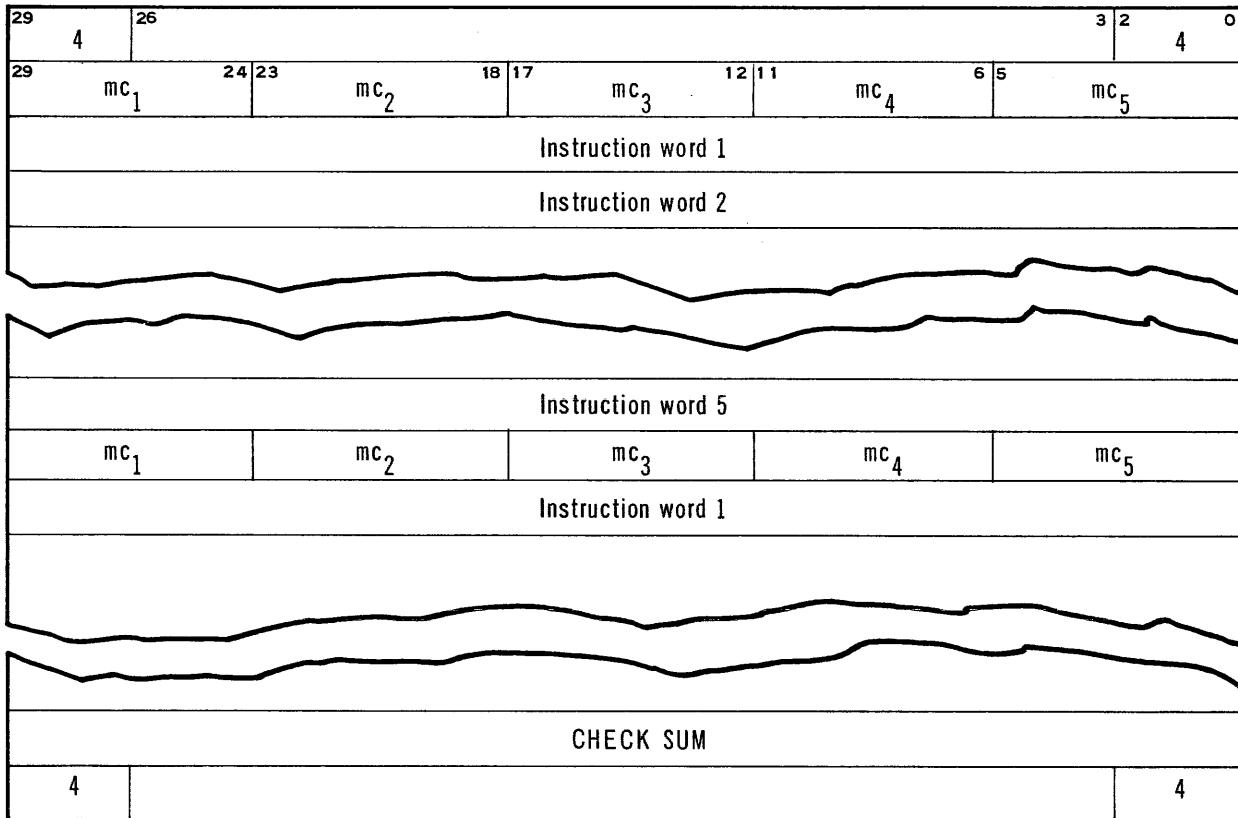
(1) This record is required for segmented programs only. It will consist of one or more 51D-wd blocks.

### 4. FILE DESCRIPTION RECORD

29	3	27	26				3	2	3	0
File Label (15 characters)										
				19	Peripheral Type		15			
							7	Channel	4	3
							Channel		Unit	
End sentinel (binary ones)										
CHECK SUM										
3									3	

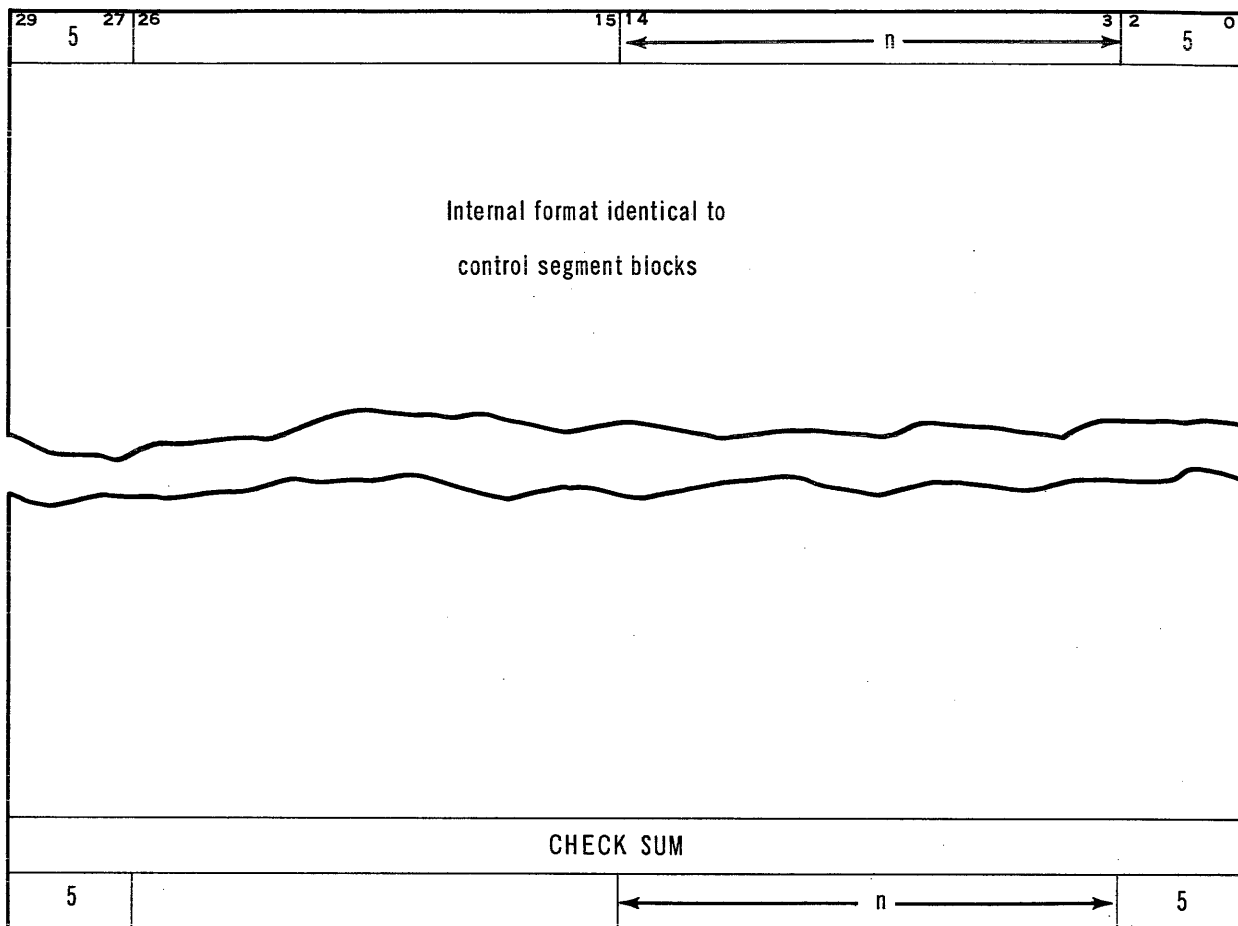
- (1) This record is optional. If present, it will be comprised of one or more 51D-word blocks. Additional blocks will be of identical format (excluding blocks descriptors).
- (2) Each entry will consist of 6 words: 3 text and 3 facility words. It will be assumed that text is in Fielddata code. All 15 characters will be typed without inspection.
- (3) Facility word 1 will always be considered significant. Facility words 2 and 3 will be considered significant only if non-zero. If the number of entries is a multiple of 8, the End sentinel must be omitted.

#### 5. CONTROL SEGMENT RECORD



- (1) This record is comprised of one or more 51D word blocks. Each block contains eight, 6-word instruction modules. An instruction module is comprised of one word of modification codes and 5 instruction words. "mc" represents modification code. "mc<sub>1</sub>" corresponds to instruction word 1 "mc<sub>2</sub>" to instruction word 2, etc.
- (2) Only the first block will bear block descriptors as shown. Subsequent blocks will bear descriptors of binary zeros.
- (3) Non-segmented programs will be marked as control segments.

## 6. SECONDARY SEGMENT RECCRD



(1) There is one record for each secondary segment. Each record is comprised of one or more 51D word blocks. The first block of each record will bear block descriptors as shown. Each subsequent block will bear descriptors of binary zeros.

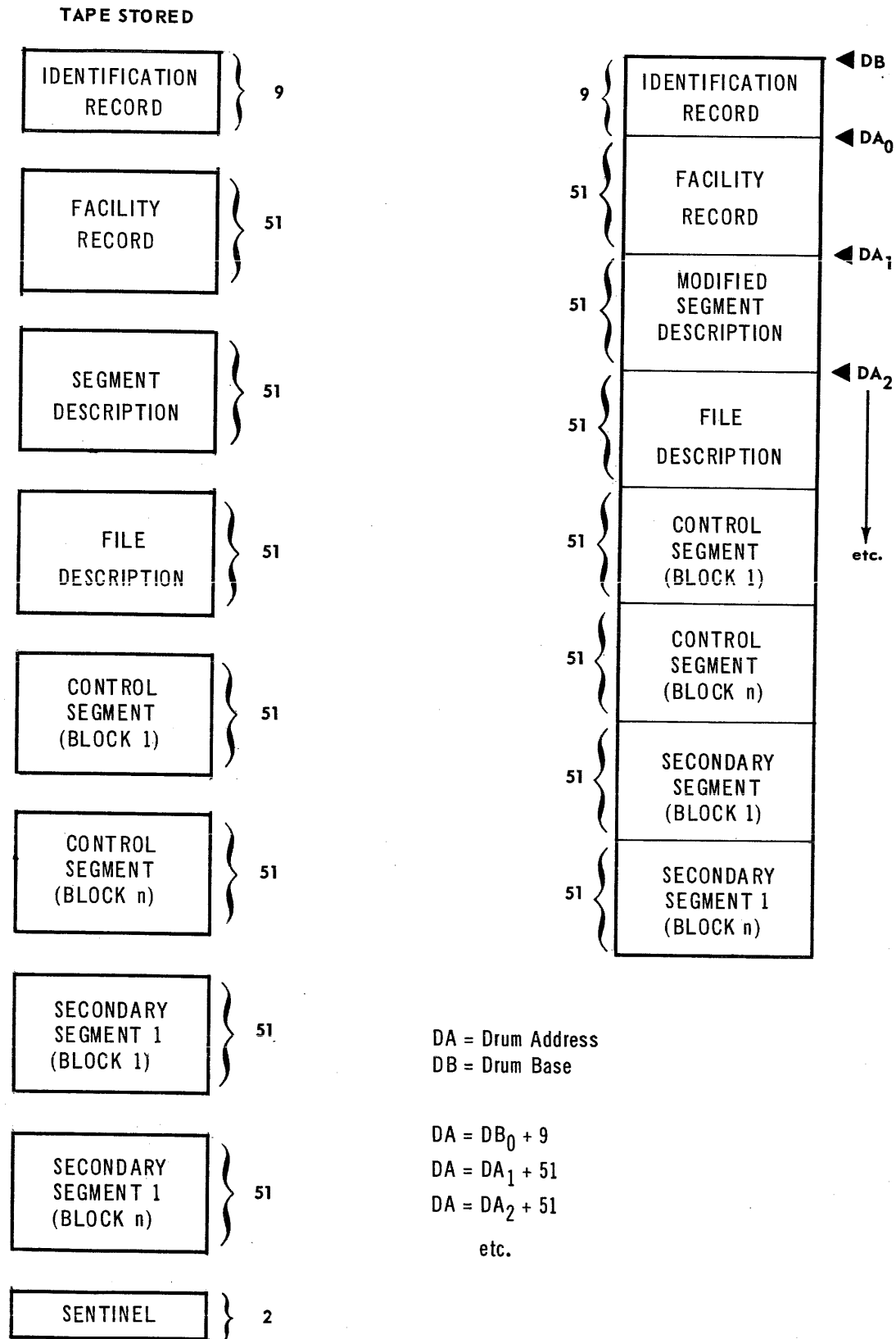
(2) "n" represents segment number. The first segment of output will be 1, the second 2, etc.

## 7. END-OF-PROGRAM SENTINEL

1	2	2	3	1	1	2	4	1	3
1	1	3	2	2	2	2	5	0	5

(2-wd block)

## 8. STORAGE FORMAT



A drum-stored program occupies a continuous area. There are no inter-record gaps or inserts.

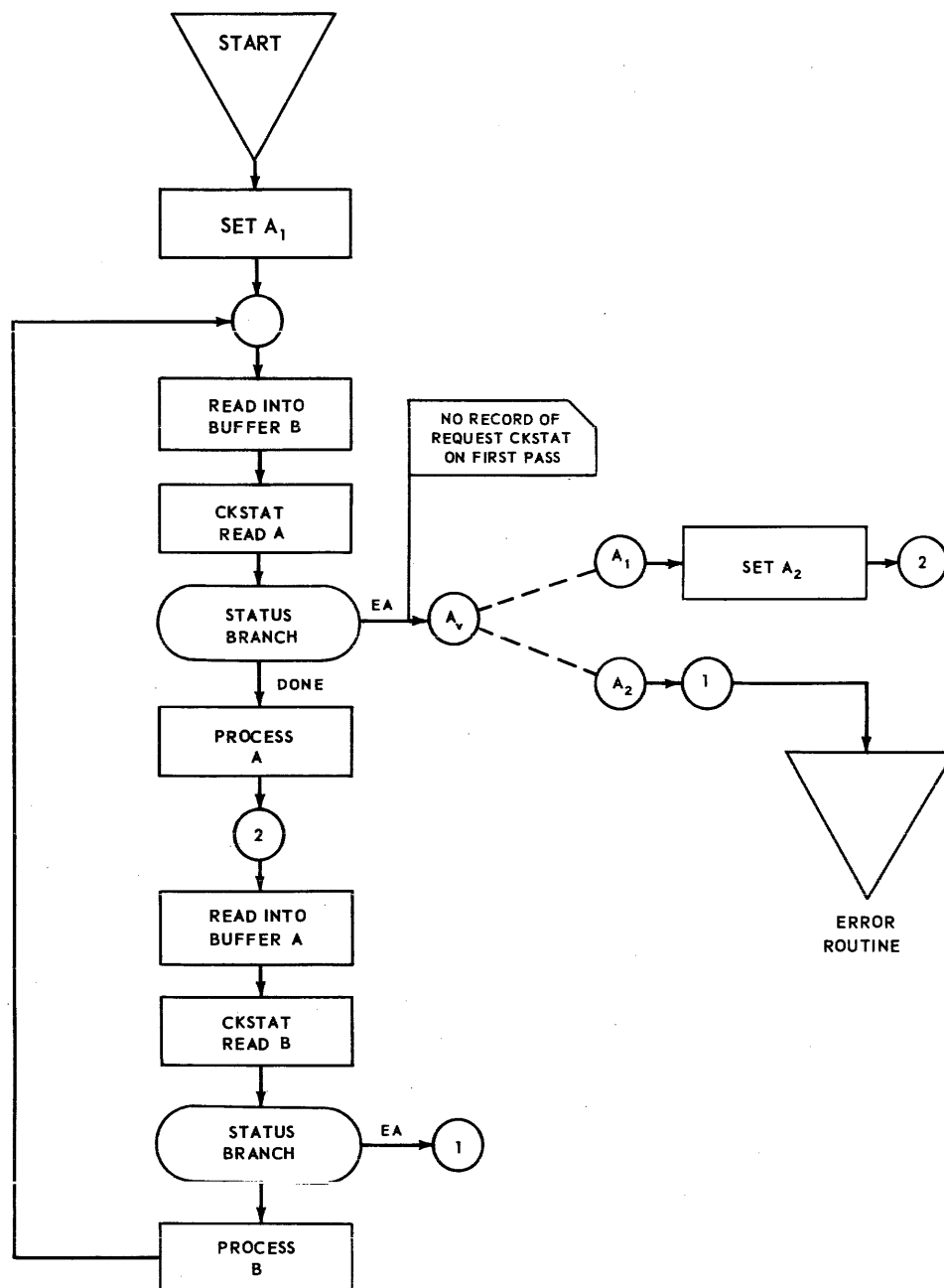
### 9. SEGMENT DESCRIPTION RECORD WHEN MODIFIED

[illegible]

(1) "Address relative to drum base" of any record equals the drum address of that record minus the drum base. It is calculated and inserted into the Segment Description Record at the time a program is transferred to the drum.

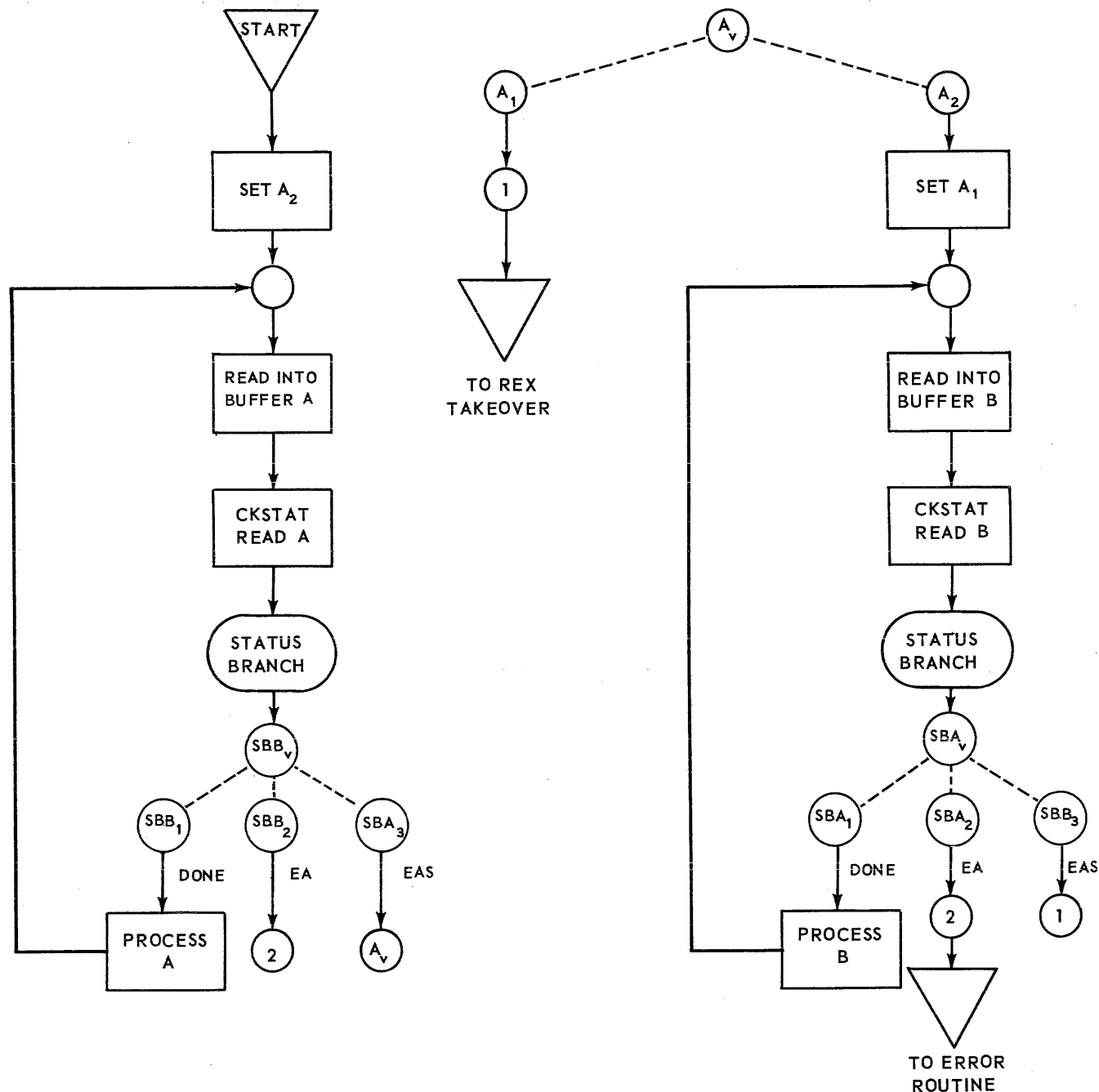
**EXAMPLE 1. Input/Output Logic for a Program to Read and Summarize a Magnetic Tape.**

The purpose of the program is to summarize the information contained on a magnetic tape. The summary is built up in core memory from two input buffers. One buffer acts as a standby input while the other is being processed. The CKSTAT operators illustrate the use of a blank EAS. The first pass will include an error exit from the first CKSTAT, which is not associated with a previous read request. REX will supply a status word of zeros and a return to the error address (EA).



**EXAMPLE 2. Another Method of Performing the Task Illustrated by Example 1 Utilizing the TAKE-OVER Operator.**

The following flow diagram shows the use of CKSTAT employing an EAS to return control to REX upon completion of either of two read operations. The read into Buffer A is interrogated by a CKSTAT. During the first pass, control is immediately given to a read of Buffer B through switch  $A_v$ . Switch  $A_v$  is set so that succeeding passes will go immediately to a TAKEOVER operator. The read of Buffer B is interrogated by a CKSTAT and control is given to REX to await the completion of either of the two reads. Completion of a read will return control following the CKSTAT which enters a routine to process the data in the completed buffer and then initiates a read of new data into the buffer. A dynamic condition is thereafter established so that REX will control reads to assure completion. For both reads, EA will go to a common error routine.





**EXAMPLE 3.** Diagrammatic Example of a Real-Time Control of Associated Subroutines.

The diagrammatic representation on this page shows a real-time program and a group of associated subroutines. An interrupt that is presented to the Real-Time Interrupt Analysis Subroutine through REX will cause the Real-Time Control routine to activate one of the subroutines associated with that type of interrupt. Subroutine A, for example, may be required to access the drum to acquire information for processing. The subroutine initiates a drum read and then returns control to the control program.

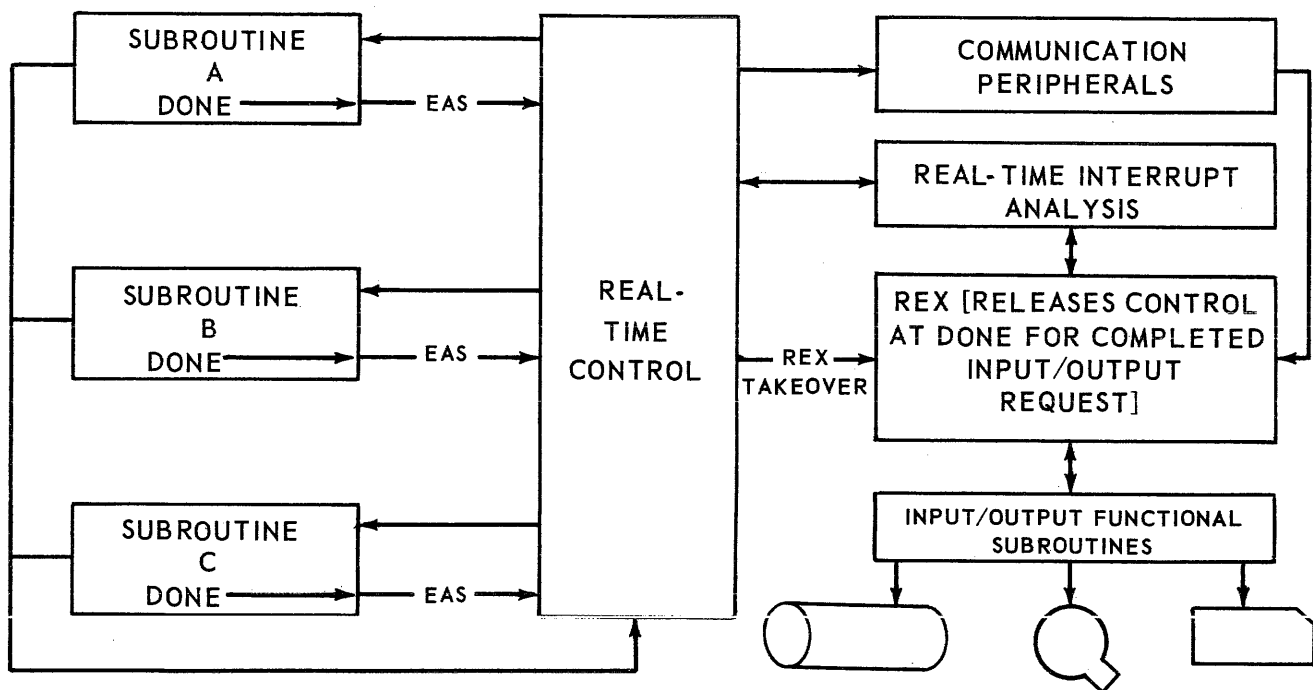
In the meantime, an interrupt presented to Real-Time Control indicates that Subroutine B should be activated. Subroutine B may be required to write a message to a magnetic tape as part of its processing cycle. The tape write is initiated and control is returned to Real-Time Control.

If Real-Time Control has not received an interrupt which would activate some other subroutine, it will give control to REX to await completion of either the drum read or the tape write that was initiated by Subroutine A or Subroutine B respectively.

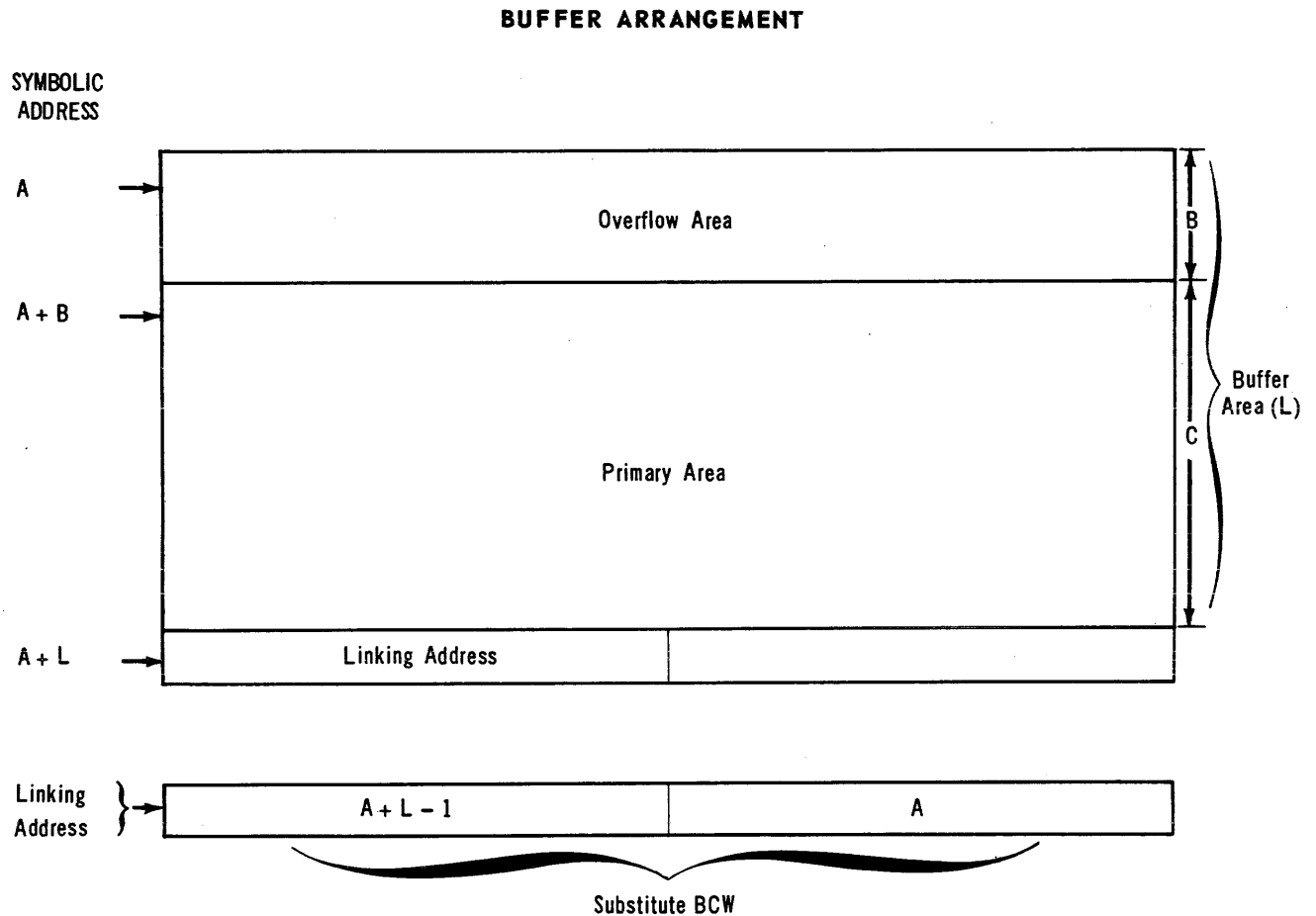
Another interrupt may occur which will cause REX to give control to Real-Time Control. Real-Time Control may activate Subroutine C which will submit a card punch request as part of the processing cycle. While awaiting completion of this request control is given to Real-Time Control which in turn releases control to REX if it has no other subroutine that it may activate at this time.

REX will return control to each subroutine at the DONE address upon completion of the peripheral request. Control will be given to Real-Time Control when the subroutine has completed the processing cycle. Real-Time Control will either initiate another subroutine or again relinquish control to REX by a REX · TAKEOVER operator.

The result of the use of CKSTAT in this example is to make REX responsible for all "bookkeeping" of input/output requests.



**EXAMPLE 4.** A scheme for using one buffer area per central CCU and extending it when overflow occurs.



Let  $A$  be the first address of a buffer area. Let  $C$  be the number of characters determined for a message segment. Let  $B$  be an arbitrary increment chosen to provide desired time delay.  $L$  represents buffer length and is equal to  $B + C$ .

Then set aside an area in core of length  $L + 1$  beginning at  $A$  and extending through address  $A + L$ , where locations  $A$  thru  $A + L - 1$  represent buffer area and location  $A + L$  contains the linking address at which a substitute BCW will be found.

TIME	CCU BUFFER CONTROL REGISTER (INPUT)	
(1) Initially	$A + L - 1$	$A + B$
(2) At Time of Interrupt	$A + L - 1$	$A + L$
(3) As loaded by REX at time of interrupt.	$A + L - 1$	$A$
(4) As adjusted by RTIAS	$A + L - 1$	$A + Y + B$

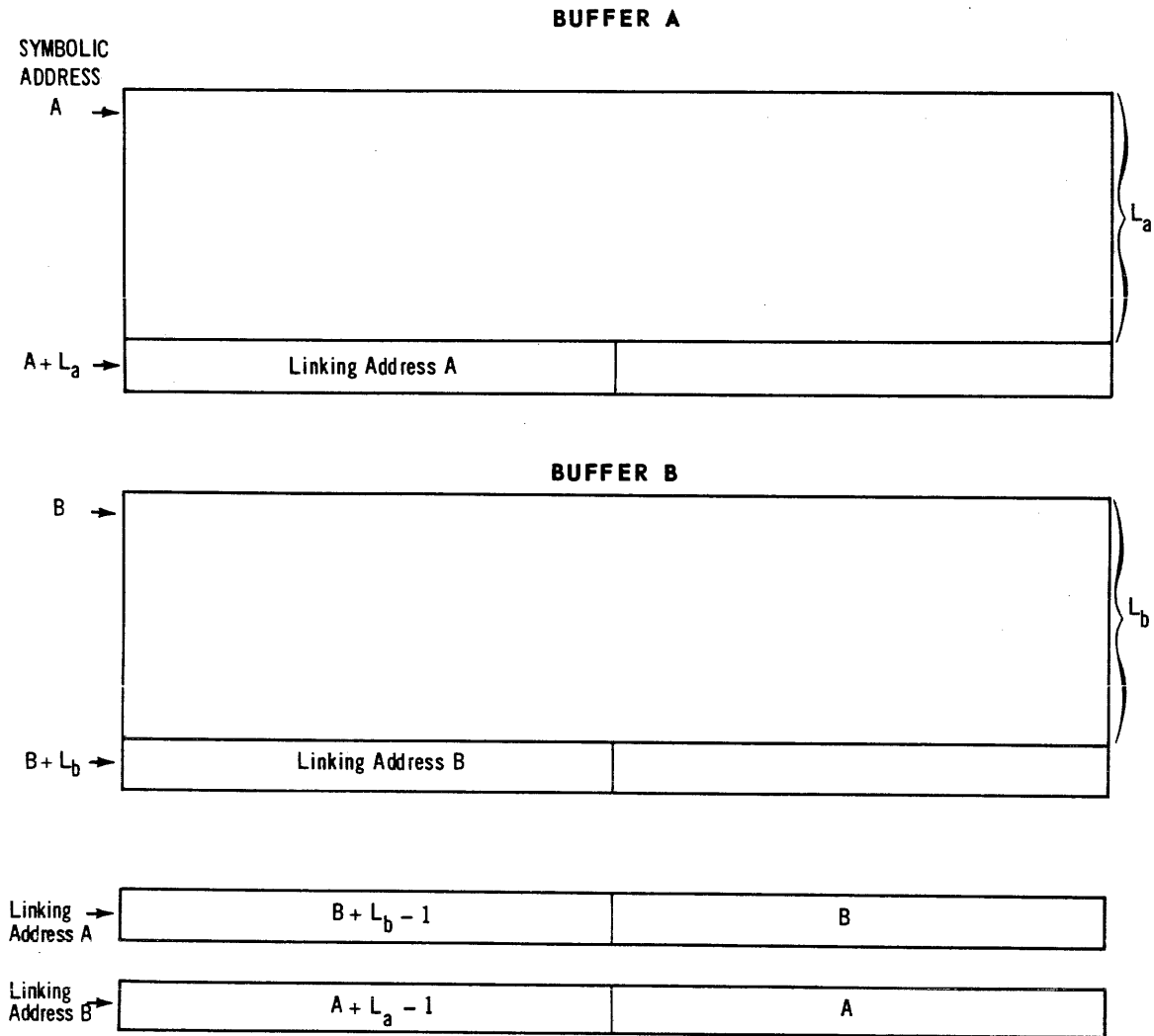
The CCU-BCW is initially set as shown at time (1). When C characters have been entered into core memory the BCW will stand as shown at time (2) and an internal interrupt will occur. REX will answer the interrupt, reactivate input/output logic, ascertain the terminated buffer, pick up the substitute BCW located at the linking address and store this value into the BCR. The BCW now stands as shown at time (3).

REX will transfer control to the RTIAS (if the interrupted program is suspendible) with the synthesized interrupt word in A. The RTIAS could pack the characters located in the primary area into an output buffer. After packing has been completed, the RTIAS could use a RPL  $A + Y$  to increment the current value of the lower half of the BCW by value B.

It could then move characters buffered into the overflow area during the time packing was taking place down B locations into the primary area. RTIAS would then be free to perform whatever house-keeping or processing it might require before exiting to REX.

The net effect of this one-buffer scheme is to extend the time period available for emptying the primary area by the time it takes to receive B characters.

**EXAMPLE 5.** A scheme employing two buffers per central CCU and alternating them when overflow occurs.



TIME	CCU BUFFER CONTROL REGISTER (INPUT)	
(1) Initially	A + L <sub>a</sub> - 1	A
(2) At time of interrupt one.	A + L <sub>a</sub> - 1	A + L <sub>a</sub>
(3) As loaded by REX at time of interrupt one.	B + L <sub>b</sub> - 1	B
(4) At time of interrupt two.	B + L <sub>b</sub> - 1	B + L <sub>b</sub>
(5) As loaded by REX at time of interrupt two.	A + L <sub>a</sub> - 1	A

REX action at time of interrupt is as described for example 4. The above process repeats itself until end-of-character train (signalled by external interrupt) arrives.

**EXAMPLE 6.** An illustration of how search option two relates to buffer length and location.

Buffer location is chosen in such manner that some number of low-order bits (m), of the BCW express upon buffer termination a value which cannot exist while the buffer is active. This unique value is referred to as the Search Key.

ADDRESS		
OCTAL		BINARY
XXXX0 =		0000
XXXX1 =		0001
XXXX2 =		0010
XXXX3 =		0011
XXXX4 =		0100
XXXX5 =		0101
XXXX6 =		0110
XXXX7 =		0111
XXX 10 =		1000
XXX 11 =		1001
XXX 12 =		1010
XXX 13 =		1011
XXX 14 =		1100
XXX 15 =		1101
XXX 16 =		1110
XXX 17 =		1111
XXX 20 =	1	0000
XXX 21 =	1	0001
XXX 22 =	1	0010
XXX 23 =	1	0011
XXX 24 =	1	0100
XXX 25 =	1	0101
XXX 26 =	1	0110
XXX 27 =	1	0111
XXX 30 =	1	1000
XXX 31 =	1	1001
XXX 32 =	1	1010
XXX 33 =	1	1011
XXX 34 =	1	1100
XXX 35 =	1	1101
XXX 36 =	1	1110
XXX 37 =	1	1111

Buffer A

Key

Buffer B

Key

m = 4

Suppose, for example, m was chosen as 4 and the Search Key was defined as 1101. Then, buffer A is defined such that the 4 low-order bits of the BCW will equal the Key at the time of interrupt (termination). Likewise, buffer B.

The starting address of the buffer is immaterial so long as the length of the buffer does not exceed  $2^m - 1$  words.

BUFFER CONTROL WORDS AT INTERRUPT	
Buffer A	X X X 1 4   X X X 1 5
Buffer B	X X X 3 4   X X X 3 5

## APPENDIX C

### GENERAL COMMENTS ON OPERATOR ENTRIES

The acceptance of console input is indicated by a type-out of the information that has been entered on the keyboard. For example, if a print core operation is desired, the first entry will be PC. If REX can accept console input, the PC will be typed on the console. If there is no accompanying type-out the entry will have to be retyped until REX acceptance is indicated.

REX console control will also check to see that the maximum permissible character count is not exceeded, that the field format is acceptable, and that the function description is valid.

The following characters have special meaning for REX Console Control:

KEY SYMBOL	OCTAL FIELDATA VALUE	MEANING
□	76	Indicates the end of a variable field of characters.
Ⓢ (Special)	57	Terminates a console entry.
	77	This will erase the last previous character. Three consecutive special characters (77) will erase the entire entry.

# INDEX

ACQUISITION OF STORED INTERRUPTS (COMMUNICATION) . . . . .	3-20	LISTING (INPUT/OUTPUT REQUESTS) . . . . .	3-11
ADDENDUM OVERFLOW. . . . .	4- 2	LOAD (TYPE-IN) . . . . .	2-10
ALLOCATION OF FACILITIES. . . . .	2-11	LOADING. . . . .	2-16
ALLOCATION OF PERIPHERALS . . . . .	2-15	LOCKOUT (TYPE-IN) . . . . .	2-10
BATCH LOAD REQUESTS . . . . .	2- 7	MASTER INSTRUCTION TAPE . . . . .	2- 1
CHANGE CORE. . . . .	5- 2	MIT NUMBER . . . . .	2- 2
CHANGE DRUM. . . . .	5- 2	OPERATIONAL PARAMETERS . . . . .	2- 5
COMMUNICATION INPUT/OUTPUT . . . . .	3-19	PREPARATION OF MASTER INSTRUCTION TAPE. . . . .	2- 2
COMMUNICATION INTERRUPTS . . . . .	3-19	PRINT CORE . . . . .	5- 2
COMMUNICATION INTERRUPT TABLE . . . . .	3- 8	PRINT DRUM . . . . .	5- 2
COMMUNICATION INTERRUPT TABLE OVERFLOW. . . . .	4- 2	PRIORITY GROUP . . . . .	2- 2
COMPUTER ESTIMATE (CE) . . . . .	2- 2	PROGRAM LOCK. . . . .	2- 2
CONSOLE INPUT/OUTPUT OPERATIONS . . . . .	2-17	PROGRAM START . . . . .	4- 3
DATE (PROVIDED BY REX) . . . . .	3- 2	PROGRAM FACILITY RECORD (MIT). . . . .	2- 4
DELAYED RESPONSE TABLE . . . . .	3- 6	PROGRAM FACILITY SUMMARY RECORD . . . . .	2- 4
ERROR PROCEDURES (STANDARD INPUT/OUTPUT) . . . . .	3-13	PROGRAM SEQUENCING AND LOADING . . . . .	2- 1
EXECUTIVE ADDENDUM . . . . .	3- 3	REAL-TIME EXTENSIONS. . . . .	2- 6
EXECUTIVE ENTRY TABLE. . . . .	3- 1	REAL-TIME INITIALIZATION TABLE . . . . .	6- 3
EXECUTIVE INFORMATION REGION . . . . .	6- 2	REAL-TIME INTERRUPT ANALYSIS (RTIAS) . . . . .	3-21
FACILITY RELEASE . . . . .	2- 8	RERUN DUMP . . . . .	5- 5
FACILITY UPDATE (TYPE-IN) . . . . .	2-11	RUNNING TIME . . . . .	2- 2
FAULT INTERRUPT. . . . .	4- 1	SEGMENT CALL. . . . .	2- 7
GENERAL PURPOSE SEARCH . . . . .	3-21	SITE UTILITY. . . . .	5- 3
HOLD SCHEDULE (TYPE-IN). . . . .	2-10	START SCHEDULE (TYPE-IN). . . . .	2-10
INDEX RECORD. . . . .	2- 3	STATUS CHECKING . . . . .	3-15
INSPECT CORE. . . . .	5- 1	STRING. . . . .	2- 2
INSPECT DRUM. . . . .	5- 1	STRING LEADER . . . . .	2- 2
INTERLOCK RESPONSE . . . . .	4- 3	STRING LOCK. . . . .	2- 2
INTERNAL INTERRUPT-INPUT . . . . .	3-19	SUBMISSION OF INTERRUPTS (COMMUNICATION) . . . . .	3-20
INTERNAL INTERRUPT-OUTPUT . . . . .	3-19	SUSPEND (PROGRAM) . . . . .	4- 3
INTERNAL LOAD REQUESTS . . . . .	2- 6	SWITCHER ROUTINE. . . . .	3-25
INTERRUPT ENTRANCES . . . . .	3- 6	TIME (MAINTAINED BY REX) . . . . .	3- 2
INTERVAL-TIMER INTERRUPTION . . . . .	3-24, 4- 1	TERMINATE (PROGRAM) . . . . .	4- 3
JUMP KEYS (USE OF) . . . . .	6- 1	TERMINATE SCHEDULE (TYPE-IN) . . . . .	2-10
LABEL RECORD (MIT). . . . .	2- 3	TIME-TABLE. . . . .	3- 8
		TYPEC. . . . .	2-18
		TYPET. . . . .	2-18
		UNLOCK (TYPE-IN) . . . . .	2-10
		UNSOLICITED REQUESTS . . . . .	2-20
		UTILITY REQUEST TABLE. . . . .	3- 3

# **UNIVAC**

**DIVISION OF SPERRY RAND CORPORATION**