INITIALIZATION AND PROCESSOR FAULT RECOVERY SOFTWARE SUBSYSTEM DESCRIPTION NO. 3 ELECTRONIC SWITCHING SYSTEM

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

1. GENERAL

INTRODUCTION

1.01 This section describes the processor initialization function as applied to a No. 3 Electronic Switching System (ESS). An initialization can consist of the following functions:

(a) Restoring the 3A Central Control (3A CC) to a known good state

(b) Restoring the periphery access to a known good state

(c) Aborting certain activities

(d) Zeroing or otherwise initializing temporary data

(e) Reloading the resident generic programs and translations from magnetic tape

(f) Saving previous system states for trouble analysis.

Not all of the above functions are performed on every initialization. The severity of the problem encountered determines the action required. In general, the system reaction becomes more drastic on successive recovery attempts. Progressively higher level recovery action is accompanied by more rigorous system response.

1.02 Whenever this section is reissued, the reason(s) for reissue will be listed in this paragraph.

1.03 Table A provides a list of abbreviations and acronyms with applicable terms used in this section.

SCOPE OF SECTION

1.04 This section includes the following information:

(a) Purpose of initialization

(b) Initialization causes

(c) Initialization programs

(d) Hardware initialization

(e) Bootstrapping and memory reload

(f) Initialization levels.

PURPOSE OF INITIALIZATION

1.05 Initialization is a means for providing the system with a known good operating configuration. This consists of confirming one or more of the following:

(a) A valid data base exists in memory for performing system operation.

(b) System control is under one 3A CC and is protected from outside interference during the initialization process.

(c) Power is applied to the peripheral units selected to operate on-line.

(d) A side designated to go on-line is capable of operating in a sane manner.

All of the above are designed to provide a continuity of office operation.

STIMULUS FOR EXECUTION

1.06 An initialization may be performed any time that a signal is received from one of the following sources:

(a) Program timer (PT) first or second time-out

(b) Error register

(c) Switch (of system control) request via the maintenance channel (MCH)

(d) Manual initialization signal from system status panel (SSP).

INITIALIZATION PROGRAMS

1.07 System initialization is accomplished by sequencing through several programs and subprograms. The major portion of initialization is performed by the following programs:

(a) ***CINIT (Common Initialization):*** The primary control program used for initialization

(b) ***CIPL (Common Initial Program Loader):*** Used as a means of bootstrapping the memory

(c) ***INITA (Application Portion of the Initialization Program):*** Used with the common system program CINIT to make CINIT applicable to the No. 3 ESS, which is designed to control peripheral unit power turnon and system clearing

(d) ***MASACS (Main Store Access Routines):*** Used as an inferface between tape format and main memory during memory unload.

REFERENCES

1.08 The following Bell System Practices may aid in understanding this section:

SECTION	TITLE
233-153-140	Peripheral Unit Fault Recovery
254-300-100	3A Processor, Description, No. 3 ESS/Common Systems
254-300-110	3A CC, Description, No. 3 ESS/Common Systems
254-300-120	3A CC, Theory of Operation, No. 3 ESS/Common Systems

2. DESCRIPTION OF INITIALIZATION AND FAULT RECOVERY

CAUSES OF INITIALIZATION AND FAULT RECOVERY

A. Program Timer

2.01 The PT is utilized when the 3A CC is executing instructions to provide a maximum time limit in which any set of instructions must be completed. The PT is reset after each major

program section is completed. The first time-out occurs at the setting of the sixth bit of the PT 8-bit binary counter. If the time-out occurs in the on-line machine, it will stop and send a stop-and-switch message via the MCH to the standby 3A CC, which will then initialize and go on-line. Should the time-out occur in the standby machine, it will initialize, restart, and then a check of the status of the on-line machine will be made. This check verifys the on-line is functioning and a switch is not necessary. The second time-out (bit 7 of the PT) occurs only when an initialization fails on the first attempt, thus providing a second try.

B. Error Register

2.02 Error signals are divided into three active types:

- Those which result in a switch to the off-line 3A CC
- Those which cause a hardware initialization
- Those which cause an interrupt.

The state of these error signals is stored in a 22-bit error register (ER) as shown in Fig. 1. Error signals that result in a switch are in bits 0 through 9. Bit 10 is a passive flag that records the occurrence of a store read parity error without any action being taken. The error signals that result in a hardware initialization are in ER bits 11 through 13. Error signals resulting in an interrupt are in ER bits 14 through P_{11}.

C. Requested Switch

2.03 During normal operation, control of the office will be switched from one 3A CC to the other at preprogrammed intervals. Prior to the programmed switch, the machine to come on-line is initialized. This initialization assures that the standby system is in the proper state to take over system control.

D. Manual Request

2.04 Initialization can be requested manually from the SSP (Fig. 2) via system initialization keys. The highest level of initialization is available only as a manual request through the SSP. This action could entail the termination of stable calls through the office.

TYPES OF INITIALIZATION

2.05 Initialization proceeds in the following stages:

- Hardware
- Microcode
- Main program.

A. Hardware Initialization

2.06 To assure correct machine start-up, the following items are initialized to the state shown by hardware before the first microcycle is entered:

(a) Block hardware check (BHC=1) disables the output of the error register which switches processors.

(b) Stop (STP=0) is cleared by hardware initialization to allow the microcontrol to sequence. With STP not cleared, the microaddress register is jammed to a maintenance address causing all zeros to be read out of microstore.

(c) Microinterpret mode (MINT=0) bit reset assures that the machine will be under microcontrol and that main memory words will not be interpreted as microinstructions.

(d) Maintenance state register (MSR=0) is a 16-bit register used to set up conditions in the 3A CC for testing purposes. The MSR is cleared since most of the conditions set up by the MSR will cause errors.

(e) Clock equals phase 3 to set the clock state. The clock is set to phase 3 to allow desired data to be clocked into the microinstruction register (MIR) on the next clock phase, P0.

(f) Freeze (FRZ=0) is cleared since in the set state it inhibits clearing and changing the memory address register (MAR).

(g) Memory address register (MAR=X) is jammed to a hardwired address which is the starting location of the initialization microcode.

(h) Block timer check (BTC=1) bit blocks the input and output from the program timer.

(i) Initialization of the maintenance channel assures that the other 3A CC cannot interfere with the initializing machine via the maintenance channel.

B. Microcode Initialization

2.07 Further initialization is accomplished by the following microprogram which is also shown in flowchart form in Fig. 3:

(1) SAVE SS in AI, IS in DI, IM in DK, and PA in AK

(2) MAIN MEMORY STATUS-NORMAL READ OF MICROSTORE

(3) ISO=1

(4) UPD=0

(5) RU=1

(6) IS=0

(7) DR=1

(8) IM=ALLOW PANEL AND OTHER 3A CC INTERRUPTS ONLY

(9) IF RESET CIRCUIT=1, THEN GO TO HALT LOOP

(10) BIN=1

(11) CC=1

(12) SEND INITIALIZATION ORDER TO MAIN STORE CONTROLLER

(13) IF ISC1=0, THEN SET ISC1=1 AND GO TO MAIN STORE MEMORY INITIALIZATION PROGRAM

(14) IDLE MAINTENANCE CHANNEL

(15) IF ISC2=0, THEN SET ISC2=1, ISC1=0, AND STOP AND SWITCH TO OFF-LINE 3A CC

(16) RELOAD MAIN MEMORY FROM TAPE

(17) HALT LOOP: RESET CIRCUITS=0

(18) BIN=0

(19) CC=0

(20) LOOP: IF INT=1, THEN SET HLT=0 AND GO TO INTERRUPT ROUTINE

(21) HLT=1

(22) GO TO LOOP.

2.08 This section of microcode was entered via the starting address which was jammed into the MAR by the previous hardware initialization. The following actions occur:

(a) Predetermined system states are stored in the specified registers. These states direct the machine to the return point upon successful completion of initialization.

(b) Isolate bit (ISO=1) prevents the other 3A CC from accessing the memory of the active 3A CC.

(c) Update (UPD=0) prevents writes sent into the on-line memory from being written into the off-line memory.

(d) Return address register update (RU=1) bit is set. The return address register (RAR) is used as a duplicate of the microaddress register to check the gating into it from the microinstruction register and the microstore.

(e) Interrupt set register (IS=0) is cleared to eliminate interrupts which occurred prior to the initialization.

(f) Data ready (DR=1) is set and indicates that the last main memory cycle is complete. This frees the main memory for accessing by the initialization microprogram.

(g) The interrupt mask register (IM) is set to all ones, except for bits 13 and 7, to inhibit the machine from attempting to service interrupts. The panel interrupt, which is bit 13 of the IS, is zero to allow interrupts from the system status panel. This allows manual accessing of the machine during the initialization process.

(h) The 3A CC reset key is checked; and if set, the machine is halted. Manual initialization

should not be attempted via the 3A CC reset key since it is possible to force both machines off-line momentarily if the reset is operated on the on-line initializing machine.

(i) The initialization sequence counter (ISC) is a 2-bit counter consisting of bits ISC1 and ISC2. The counter is incremented as an initialization process is entered to ensure that successive initializations (if required for recovery) are not repetitions of the previous action. The ISC states and associated actions are as follows:

ISC State on Entry	Action
0	CINIT—increment ISC
1	Stop and switch—increment ISC
2	CINIT—increment ISC
3	Bootstrap.

C. Main Memory Initialization Program

2.09 When the ISC2 bit is clear (0), the hardware initialization microprogram transfers directly to a main memory program (CINIT) for further system initialization. Should a stop and switch (of 3A CCs) or a bootstrap memory reload be required, it would be implemented prior to main store program initialization.

2.10 System initialization is functionally divided into the following sections:

(a) 3A CC initialization

(b) Input/output (I/O) and memory contents initialization

(c) Administration.

After each of these sections, an entry is made to the application initialization program (INITA) to allow it to perform work.

3A CENTRAL CONTROL INITIALIZATION AND FAULT RECOVERY

2.11 The 3A CC initialization (CINIT) performs the functions listed in the following paragraphs.

A. System Status (Post-Mortem)

2.12 The 3A CC status is maintained in a 64-word block known as the post-mortem area (Fig. 4). The first half (32 words) is always loaded during an initialization. If the initialization occurs when the system is not in an initialization interval, this data is also copied into the second half (32 words). Thus, the first half always corresponds to system status at the most recent initialization, while the second half denotes 3A CC status at the time of the first initialization of a sequence. This record provides a means of determining why the system trouble occurred (error detection) and how the system recovered (fault recovery) from the problems.

B. Register Initialization

2.13 Special registers and the MCH are initialized to assure that registers to be used later do not contain bad parity or other erroneous data.

C. System Status Panel Requests

2.14 The SSP is interrogated for key requests. If a memory reload request is found, program control is transferred to the bootstrap loader (CIPL) and a complete memory reload is initiated immediately.

D. On-Line/Off-Line Decision

2.15 The decision of whether to go on-line is made using a set of priority rules (Fig. 5). This determination is made by examination of the following actions:

(a) The lock off-line (LOF) bit disables the I/O channels so that the locked off-line 3A CC cannot interfere with the system. The LOF bit is set by a signal from the system status panel. If set, the designated 3A CC goes off-line.

(b) The lock on-line (LON) bit forces all hardware switch messages to initialize the on-line 3A CC in order to keep it on-line. The LON bit is set by a signal from the system status panel. When set, the designated 3A CC goes on-line.

(c) The diagnostic test (DGN TEST) is checked to see if the machine should be held off-line because diagnostics are being run.

(d) Examination is made for active initialization sources in this machine. The legitimate

sources are first or second PT time-out or an MCH initialization message. The 3A CC will go on-line if either is active.

(e) If none of the previous conditions apply, the state of the 3A CC flip-flop is examined. When the processor has been on-line previously (CC=1), it will initialize and remain on-line.

(f) The final factor is to determine on-line/off-line status (if none of the previous applies) if the other 3A CC is stopped. If the other 3A CC is stopped, it is imperative that the on-line 3A CC remain on-line and initialize itself.

E. Sanity Check

2.16 The sanity of the initializing 3A CC is checked to see whether the 3A CC is reasonably capable of functioning. A subroutine of CINIT performs this test by exercising as much of the machine as possible. The CINIT assumes that the self-checking circuits will detect errors. The basic technique is to send ones and zeros through as many gates and registers as possible by using as many to, from, and miscellaneous decoder crosspoints as possible. A maximum number of data manipulation logic (DML) functions are exercised while minimizing the number of words and microinterprets within the shortest possible execution time. A pattern of alternating ones and zeros is accumulated in register R0 throughout the test. The sanity test is executed on the on-line 3A CC during initialization to determine that it is reasonably sane before allowing it to take over. It is executed on the off-line 3A CC to determine whether it should be removed from service.

F. Memory Validation

2.17 If the initialization involves a manual bootstrap request, the complete memory is rewritten from tape. When the bootstrap memory reload is entered via automatic mode, recovery speedup is attempted by using checksums to determine which areas of store need to be copied. Less reload time is required if only mutilated sections of memory are reloaded and the other sections validated by use of checksums.

G. Main Store Initialization

2.18 The main store initialization that occurs at this point is a hardware initialization. This initialization sets the main store hardware to a known state by clearing serial I/O buffers and resetting write-protect areas.

H. Other Central Control Status

2.19 After a switch of 3A CCs occurs, the status of the off-line 3A CC is saved in the post-mortem area. This data is required for later analysis to determine why the system required the initialization action.

I. Other Main Store Initialization

2.20 A hardware initialization is performed on the off-line main store (OMAS) if a switch of 3A CCs occurred. This controls the state of the OMAS in the event that it must be used for a memory update or other initialization functions.

J. Memory Update

2.21 If the main memory in the initializing machine is not up to date, it must be copied from the off-line main store provided the off-line 3A CC is not in manual. It is not known if the contents of the OMAS can be successfully transferred to the on-line MAS; but if it is successfully initiated at this time, the post-mortem will be overwritten. To prevent this, an attempt is made to transfer the post-mortem data from the on-line MAS to the OMAS. The contents of the other MAS are then transferred to the on-line MAS and marked as the up-to-date main store; therefore, the off-line memory is no longer up to date.

K. Level Count Increment

2.22 To determine the severity of initialization action, the level count must be incremented each time initialization is attempted. It was not necessary to increment the level count previously, because up to this point protection was provided by the hardware/microsequence of ISC1 and ISC2. Incrementing the level count is important to ensure reaching higher levels of initialization should successive attempts be required.

L. Software States Saved

2.23 Information which may be required for determining further initialization action is stored in the post-mortem area. This information consists of data such as which 3A CC is on-line,

the level of initialization, the count of the hold-get counter, and related states.

M. Hardware Checks Enabled

2.24 The hardware check circuits which were previously disabled (to prevent error flags) are now enabled to provide hardware error checks during further initialization provided by the early entry into INITA.

N. Early Application Routine

2.25 The common system initializing control program (CINIT) can pass initialization control to the application program (INITA) at the following points during initialization:

(a) After the 3A CC and permanent memory are initialized

(b) After all common system temporary store is initialized

(c) After all common system initialization is complete.

2.26 The first entry of CINIT into INITA is used to analyze the temporary store to determine the kind of initialization to be performed. If a partial clear is required, it is implemented immediately.

INPUT/OUTPUT AND MEMORY INITIALIZATION

A. Input/Output Channels Initialized

2.27 The registers of the I/O channels are cleared and made available for use during the initialization sequence. This ensures access to peripheral controllers as required by initialization programs.

B. Common System Memory Clear

2.28 Depending on the level of initialization, certain areas of temporary memory are cleared. This action is explained in paragraphs 2.34 through 2.44.

C. Memory Parity Check

2.29 Critical areas of memory are checked for bad parity. If bad parity is detected, the associated memory area is zeroed. This action can possibly result in the clearing of a transient call which might have originally caused the system fault, necessitating recovery.

D. Multiscan Functions Stopped

2.30 All multiscan functions are stopped. These are requests for various services which are stored in temporary memory. If the memory was cleared, all records of these requests have already disappeared.

E. Second Application Routine

2.31 The primary task of the second application entry is to clear areas of temporary store. The extent of memory clearing is dependent on the initialization level being executed. After the selected memory is cleared, the peripheral circuit initialization routines are called to initialize peripheral controllers.

2.32 Program INITA initializes peripheral controllers by determining that power is applied to the selected controllers. When a switch occurs, INITA turns on power to the proper controllers and ascertains that power actually comes on. If a transient clear or a stable clear is required, INITA executes the tearing down of the required hardware connections. After completion, control is returned to CINIT.

ADMINISTRATION

2.33 To provide for orderly and nonrepetitive initialization attempts, the following administrative actions must be executed.

(a) The two ISC bits are administered now that CINIT has successfully completed.

(b) The TTY recovery message is formatted.

(c) The SSP is partially initialized.

(d) If a 3A CC switch occurred, the final decision is made on taking the off-line 3A CC out of service. It is taken out of service if it failed the sanity test or suffered excessive errors resulting in two initializations within a prescribed timed interval.

INITIALIZATION LEVELS

2.34 The following levels of initialization are defined for the No. 3 ESS:

Level 1—Partial clear

Level 2—Partial clear

Level 3—Partial clear

Level 4—Transient call clear

Level 5—Stable call clear.

The tasks performed by the initialization programs are dependent upon initialization levels as shown.

A. Partial Clear—Level 1

2.35 ***Partial clear*** (level 1) is the first level of initialization. It is executed when no emergency action keys are operated, translation store has not been altered by bootstrap load, and a good copy of temporary store is available.

2.36 During the early entry (INITAE), registers in the post-mortem area are examined. When the register points to a transient call record (TCR), this TCR is put in the failure state so that it will be removed when call processing resumes.

2.37 The middle entry (INITAM) will clear the following temporary store for a level 1 clear:

AUDCNTL—Audit control

SCRATCH—Base level scratch area

XSLERROR—Translation error buffer

CMP_BUF—Camp-on buffer

DIREQS—Peripheral diagnostic request buffer

PATHDATA—Path hunt parameters

SPNFLD—Interrupt scan data

RTP DATA—Ringing and tone plant data

RTP COUNT—Ringing and tone plant count

QC_TABLE—Error analysis—quick check entries

QC_CTRS—Quick check control

HOPOVCNT—Line origination hopper overflow count

AUDMAP—Network audit map

EA_INBUF—Error analysis input buffer

MISCBITS—Miscellaneous flags

HG_SAVE—Hold-get counter

HGAREA—Hold-get area.

The ***last-look*** and ***ignore*** bits of miscellaneous scan points are initialized to their expected values. This provides for reporting all off-normal indicators (such as alarms) following the initialization. The last task of the middle entry is the initialization of peripheral controllers.

2.38 The ringing and tone plant counter is set to all ones by the initialization action of the final entry (INITAF).

B. Partial Clear—Levels 2 and 3

2.39 A ***partial clear*** occurs on initialization level 2 or 3. Four additional tasks (in addition to those done during partial clear—level 1) are performed. First, the hold-get stack is analyzed, and any TCR found active at a higher level than the level experiencing the error interrupt is marked for processing by failure routines when call processing resumes. Second, all TCRs containing MAINT or MTC_NW progress marks are marked for failure processing, because these are under control of multiscan functions which were aborted by CINIT. Third, the following additional temporary store is cleared.

Levels 2 and 3

STAT_TBL—Maintenance status table

I_OBUF—Cassette tape handling I/O buffer

SCNSTATE—Line scan status

NRM_SCR—Resident-nonresident interface

ADM_COM—Administrative control common areas

DISTSTAT—Recent change reallocation status

AULC—Alarm unit lamp counter

ALMBLK—Alarm block

MISPWR—Miscellaneous power status

Level 3 Only

FLT_LIST—Maintenance known fault list

CLID—Calling line identification table for trace

Fourth, the off-line processor peripheral controllers are removed from service during a level 3 initialization.

C. Transient Call Clear—Level 4

2.40 A ***transient call clear*** can be initiated manually or automatically. A transient clear is initiated by operating the ENABLE and EXECUTE keys on the system status panel. If a level 4 initialization is reached, a transient clear is initiated automatically.

2.41 During transient clear, the additional temporary store is cleared.

EA_TABLE—Error analysis table

TCRs—Transient call records

MTCR—Maintenance TCR

LASTIME—Last base level loop start time

TTY_MFA—Common name for all TTY temporary store

SYSTIME—System timer

TDPRHOP—Trunk dial pulse receiver hopper

NETQ—Network queues

RCVRSTAT—Receiver status area

TMHOPR—Timing hopper

TV_STAT—Test vertical status

TRAFFIC—Traffic data.

2.42 Transient clear must tear down all transient network connections and idle all circuits involved in transient connections while leaving stable calls untouched. The initialization is then set to zero, causing the next initialization, if required, to be a level 1. This ensures that a stable clear is not invoked automatically via initialization level count.

D. Stable Call Clear—Level 5

2.43 A ***stable call clear*** can be initiated by any of the following reasons:

(a) Emergency action keys on the system status panel are operated in the following sequence:

ENABLE STABLE CALLS	MEMORY RELOAD PAST OFFICE DATA BAKDATE OFFICE DATA	INIT EXECUTE

Note: These keys may be active in any pattern.

(b) If a processor switch occurs where the on-line processor has an out-of-date store and no access is available to the other processor store, a stable clear occurs automatically because all the temporary store is lost.

2.44 All of the stable clear is performed during the middle entry (INITAM). The first task is to zero the following temporary store in addition to that which was cleared at lower levels.

OSLIST—Line out-of-service list

TSVLIM—Trunk and service circuit out-of-service counters

LINLIST—List of plugged up lines

NETMAP—Network map

TMRs—Terminal memory records

INPTHOP—Trunk/junctor input hopper

INTHOP—Interrupt input hopper

LORGHOP—Line origination hopper

SCANSTAT—Line and junctor status lists

CKTSTAT—Trunk and service circuit status bits

TV_OSTAT—Test vertical status 0

SYSTEM software clock

TV_1STAT—Test vertical status 1

DATA—ODA assignable area

(a) Stable clear attempts to correct parity in the store on the on-line side. The areas accessed are:

(1) All ATSD

(2) The memory not currently being used by translations but reserved for future growth.

The remainder of the stable clear zeroes all peripheral pulse distributor relays, opens all first-stage network crosspoints, opens all line cutoffs, and then closes all line cutoffs.

E. Execution Times

2.45 Approximate execution times for the initialization levels are as follows:

Partial clear—30 milliseconds

Transient call clear—160 seconds for 15 concentrator groups

Stable call clear—160 seconds for 15 concentrator groups

Bootstrap with stable call clear—May take up to 5 minutes.

3. BOOTSTRAPPING

INTRODUCTION

3.01 Bootstrapping a processor consists of loading writable memory from an external source and using the processor as the controlling device. In a system using a volatile memory, software may not exist to aid in the reload. Therefore, a reload mechanism must be designed into the processor which will enable it to retrieve a program from an outside source and load it into memory. Since the 3A CC is a microprogrammed processor, a small sequence of microcode is dedicated to the reload mechanism.

3.02 The processor reload mechanism can be allowed to completely rewrite memory. However, a loader capable of high-speed, reliable reloading requires more microstore than is economically feasible. This is overcome by microcoding a very basic loader and allowing it to fetch a more sophisticated loader program from tape.

MEMORY BACKUP

3.03 The external device that will supply the 3A CC with data during bootstrap is a 4-track tape cartridge. Each 3A CC has access to its own bootstrap dedicated tape unit with the facility to utilize the other 3A CC tape unit at the appropriate time if required.

3.04 The primary use of the tape medium by the 3A Processor is that of backup for the system program and system data tables, commonly known as translation in ESS applications. Secondary system benefits are derived from an extension of memory such as the creation of historical system performance files. However, the layout of the information on the tape is engineered for the bootstrap operation.

3.05 The system program and system translation are the most important items stored on the tape and must be given the best chance of recovery. The two inner tracks (1 and 2) of the tape cartridge are used to store the program and translation backup. The system program resides on track 1 and is write-protected. This protection will prevent an inadvertent overwrite of the system program which would place the system in a precarious state. Since the translation is subject to frequent change, it will reside on track 2 which will remain writable. Large translation systems may require two tracks of storage. Provisions have been made to store the extra translation on track 3.

3.06 Since the 3A Processor is a duplicated system, both tape units will contain the same program and translation-related data.

BOOTSTRAP SEQUENCE

3.07 After an initialization of the 3A CC occurs, the microstore initialization program determines whether a software mutilation has taken place and whether a reload attempt is required. The tape unit associated with the 3A CC in trouble is initialized and rewound to the beginning of tape (BOT). The microcode starts the tape to read. It unloads the first 128 words that come off the tape and stores them at positions 1 through 128 (Fig. 6). The microprogram then transfers control to location 6 which is the first executable word read off the tape.

3.08 The first segment of the bootstrap sequence consists of 122 executable words. This sequence must be capable of continuing to unload program from tape until the second segment (Fig. 6) is completely loaded into memory. The bootstrap programs are duplicated on tracks 1 and 2 so that an attempt to recover mutilated data can be made without disturbing the other tape device. This is only true for the bootstrap. If the duplicated data is bad also, a time-out occurs, resulting in a reboot. The cyclic check mechanism is used to determine if a block of data has been mutilated.

3.09 Program control is passed to the second segment to complete the loading of the bootstrap program, system initialization programs, and checksum data (Fig. 6). This sequence of bootstrapping utilizes the same recovery process as the first sequence.

3.10 Successful completion of the second segment of the bootstrap sequence ensures that the common system initialization program (CINIT) is properly loaded. Control will be passed to CINIT at this time for execution of hardware initialization and disabling of the other 3A CC before further bootstrap action occurs in this machine.

TAPE FORMAT

3.11 The bootstrap program and the central core of system programs are placed at the beginning of tape on tracks 1 and 2 (Fig. 7). The bootstrap microsequence is programmed to read from the start of the write-protected track and to sequence to track 2 if it encounters difficulty in reading track 1 on a reboot attempt. The sequence continues until CINIT executes correctly.

3.12 Positioned immediately at the end of this file is the file containing all the checksums for the main store information. This file is placed on track 2 for update as changes are made to either the system program or translations.

3.13 Immediately following the checksums, the system program will be stored in logical segments of 4096 words on track 1. Each 4096-word boundary will begin on a new block within the file, and each unit will end with a specially marked block.

3.14 The start of translation data will be on track 2, following the completion of the system program (generic). Any overflow of translations will be accommodated on track 3 of the tape. Following the translation on either track 2 or track 3, the file of system program patches will be stored.

3.15 Finally, the last file associated with the bootstrap function is the application translation backdate files. This is the earliest stored translation that the system may need to recover from an error in the current copy of translation.

3.16 This layout of the individual files requires only a switching of tracks at the appropriate points to gain access to all the files on tracks 1 and 2. If both tape devices are being used simultaneously, the worst-case bootstrap tape read amounts to a single pass across the tape. For tape control and checks, the data is assembled on each track as shown in Fig. 8.

BOOTSTRAP PROGRAM

3.17 The bootstrap program (CIPL) consists of three major loading segments. The first segment is responsible for bringing in enough of a loader to continue loading from transient memory instead of program store. The second segment, in turn, continues to load a more sophisticated system loader and its supporting programs. The third segment is used to selectively reload the system program and translations.

A. First Program Segment

3.18 The first segment of the bootstrap loader program is loaded into memory under control of the microprogram. The first segment (122 executable words) is loaded into low memory since it is more expedient and economical for the microcode to do so. This sequence loads the second segment into the temporary area of store where it will not be overwritten by system programs which reside in low memory.

B. Second Program Segment

3.19 At this point, the entire CIPL program has been read into memory, and any further data brought off the tape is considered support to continue the loading process during the third segment.

3.20 Each block on the tape incorporates a cyclic redundancy check (CRC) character for the data stored within that block (Fig. 8). After reading in each block, a recomputed CRC character will be compared against the stored CRC to validate the data transferred.

3.21 The next step for the second segment of the bootstrap loader is to unload and store the supporting programs requiring initialization. When all the required data and programs are loaded, control is given to the 3A system initialization program (CINIT) to do some basic hardware initialization and checks. The sanity check is performed; and if it should fail, the recovery responsibility is passed to the other 3A CC.

3.22 If the sanity test passes, the other 3A CC is prevented from taking any steps toward controlling the processor environment. This is performed by the CINIT program over the maintenance channel. This off-line control unit is inhibited from executing any main store code, and its microstore control is placed in an idle state (halt loop).

3.23 When this stable state is reached, the second segment of the CIPL program regains control and the main store checksum verification begins. The bootstrap program is aware of which areas of main store are used for program and translation and which areas are writable (call store) from data stored in the checksum file. As the checksum algorithm is applied to each 4096-word block of store, a table is created to retain the pass-fail status of each block.

3.24 The last function performed by the second segment of CIPL is the initialization of the other 3A CC tape unit in preparation for use by the last segment of CIPL. If no communication can be established to this unit, control words are set up to retain this fact so that the third phase of bootstrap can take the appropriate action.

C. Third Program Segment

3.25 The most complex and final segment of the CIPL program is now entered. To reduce the amount of time required to recover the downed system, an attempt will be made to use both tape devices to continue the reload of the processor. One tape device will be used to unload the first two tracks, and the other tape device is used to recover software from the remaining tracks if required.

3.26 Handling both tapes simultaneously requires a time-shared driver to assure that both devices are periodically serviced. Also, some sort of software timing is required to aid in the mechanical handling aspects of tape devices. The third segment adapts to a looping structure of several time-shared modules. Each module is responsible for one or more of the functions performed by the third segment.

3.27 The loop consists of four major states and a number of substates which are shared by most of the major states. This sharing is possible since the substates are basically tape handling software which if chained in the proper sequence will perform any of a number of physical tape manipulations and data transfer operations. The major states analyze the function to be performed and schedule the substates as needed to bring the function to completion. It must keep track of the progress made and what is yet to be done, plus assure that the next major state gets control to perpetuate the loop.

3.28 The major states are referred to as the A-, B-, and C-task schedulers, the program timer reset, and the audio-visual display routine. The program timer reset routine passes control sequentially to the A-, B-, and C-schedulers which perpetuate the loop by passing control to the program timer reset routine. The A-scheduler is

responsible for reloading the system program located on track 1 of the tape and any translation located on track 2. The B-scheduler operates with the other 3A CC tape unit, if available, and is responsible for any translation stored on track 3 of that unit. The C-scheduler also uses the other tape device and is responsible for the patch file and any application-oriented bootstrap function such as loading an earlier version of translation. The program timer reset and audio-visual routine prevent the timer from reinitializing the system by resetting its count, ringing the TTY bell, and providing a light display in the display board and SSP. Also, it is responsible for updating a software timer which will be used by the software to measure elapsed time.

3.29 A system that does not achieve the bell and light display within 30 seconds after a bootstrap is initiated (this time allows for worst-case tape rewind) is not recovering, and help should be sought. Once the audio-visual display begins, it should not be interrupted until the system has recovered. If it is interrupted, assistance is required.

3.30 Each scheduler utilizes its own block of scratch memory to store control and status information. The breakdown of the data maintained in the block is as follows:

(a) The substate in control of the tape unit

(b) The present position of the tape unit controlled by this scheduler

(c) The I/O address of the controlled tape unit

(d) Miscellaneous store word (buffer counter, track number, timing request bit, etc)

(e) Destination block number for search

(f) Time constant storage.

A-Scheduler

3.31 The first operation of the A-scheduler involves testing if it is in a timing mode; ie, waiting for a specific amount of time to elapse before passing control to one of the substates. A previously executed substate is capable of placing a scheduler in the timing mode while waiting for some mechanical tape operation to terminate. If the scheduler is timing and has not yet timed out, control is passed to the B-scheduler.

3.32 After the time-out has occurred, control is passed to a substate (if one is scheduled in the substate control word) for execution. The lack of a substate number in the control word indicates that a new 4096-word mutilated store must be located by the A-scheduler if another such block exists.

3.33 The CHECKTBL routine of the A-scheduler is entered when the substate control word is zero. This routine scans the checksum pass-fail table created in a segment of CIPL. Each entry represents a 4096-word block of memory and is used to generate an index to locate the tape blocks containing the data for that 4096-word block. When an entry signifies a mutilated segment of memory, the A-scheduler converts the entry number into a tape block number and begins the substate sequencing to reload memory. From this data, it can determine just where the program and translation files are located and how much of each resides on tracks 1 and 2.

3.34 The general sequence of events brought about by the substate requests of the A-scheduler is as follows:

(a) The verification, if needed, of the present tape position.

(b) The calculation of the distance to destination of the next mutilated 4K segment on the tape.

(c) The determination of the speed of the search based upon the physical distance to destination.

(d) The tape is searched.

(e) Reverification after search if needed. (Slow search is a block-at-a-time movement; therefore, verification of position is not necessary.)

(f) Recalculation of destination distance or reading of the 4K segment, depending upon the result of the distance calculation.

3.35 The A-scheduler also works as an overseer of the B- and C-schedulers. If the other tape device is inoperative, the B- and C-schedulers are unable to simultaneously carry on the reload

process with the A-scheduler. When the A-scheduler has completed its primary objectives, it scans the control blocks of the B- and C-schedulers to ascertain their statuses. If the B- and C-schedulers have been placed out of service, the A-scheduler will reactivate them by changing their tape unit information to point to the single working device.

B-Scheduler

3.36 The B-scheduler is charged with reloading the mutilated translation from the backup stored on track 3 of the tape. Like the A-scheduler, the B-scheduler has its own checksum pass-fail table search routine.

3.37 The B-scheduler first checks to see if its work is complete or to see if it is idle waiting for the C-scheduler to complete. If not completed, it ascertains that it is operative. If operative, it checks to determine that the tape unit it is using has been to the BOT position. This would satisfy all the necessary conditions to initiate its table search and reload functions. From this point on, the operation of A- and B-schedulers are identical.

C-Scheduler

3.38 The C-scheduler becomes active when the B-scheduler has terminated its work and has gone dormant. The C-scheduler will overwrite the program that the A- and B-schedulers have reloaded with patches located in the patch file just beyond the last block of the system parameters. It will reload every current patch and all patch areas for all the system software.

3.39 The C-scheduler passes control to an application program at this point to check whether any specific application tape manipulation is required. As long as the application code returns with an active status, the C-scheduler will continue the loop. An inactive status will begin the termination of bootstrap.

3.40 The completion of the application segment will be followed by the final task of the C-scheduler. It will scan the control block of the A-scheduler to check if it has terminated its work. When all three schedulers have terminated their work, the C-scheduler will pass control to the initialization program for continued restoration of the control units. This finishes the required work of the bootstrap program, and it will be cleared out of memory when the system initialization programs zero out transient memory as part of the system recovery.

Substates

3.41 Each substate is usually responsible for one tape operation which requires an interval of time to elapse before it is completed or before another can be initiated. The following is an enumeration of each substate and a brief description of its function.

(a) ***Slow Position Substate:*** The tape is moved toward a destination via the use of a backspace or read-a-block command.

(b) ***Forward Time Substate:*** When the slow position substate uses a read-a-block command to position the tape, the tape data controller (TDC) must be serviced since data transfer will take place. No data is read into the memory since the objective is tape motion toward a given tape block.

(c) ***Fast Position Substate:*** This substate will place the tape unit in question into either a fast-forward or fast-reverse speed based upon the sought-after destination. A timing request is made based upon how far the objective is away from the present tape position.

(d) ***Fast Stop Substate:*** This substate will issue a stop command to terminate a fast-forward or reverse speed. A timing request is made to allow the tape to recover from the fast speed.

(e) ***Verify Substate:*** After a fast tape motion is terminated, the position of the tape heads may be over some portion of a data block. No read can be successfully completed from this position. The tape heads must be placed in an interblock gap (IBG) before any reads are attempted.

(f) ***Read Verify Substate:*** A fast search has been stopped, and the new position of the tape must now be obtained. The read verify substate will initiate a read-a-block command to update the present tape position.

(g) ***Read On Substate:*** This substate services the TDC buffers during the reading

of the tape. It will verify the information in each block read off the tape via the use of the CRC character. It will either store the incoming information into memory or update the present tape position based on control left by the previous substate. It will also initiate a block reread if the CRC character check fails.

(h) ***Recovery Substate:*** If a CRC failure occurs, this substate is responsible for backspacing the tape and scheduling a reread of the failing block. Failure of the reread forces another level of recovery which interrupts the other tape to recover the mate block. Failure of this results in termination and a restart of the bootstrap reload. Prior to the termination, a flashing, checkerboard pattern is displayed.

ERROR REGISTER		
BIT	SYMBOL	FUNCTION
0	TODER	TO DECODER ERROR
1	FRMDER	FROM DECODER ERROR
2	IBER	IB X, Y FIELD ERROR
3	BUSER	BUS PARITY ERROR
4	DMLER	DML MATCH ERROR
5	MARER	MAR ERROR
6	CLKER	CLOCK ERROR
7	-	- - -
8	MADER	MAD-RAR ERROR
9	FRER	FUNCTION REG PARITY ERROR
10	SRPER	STORE READ PARITY ERROR
11	MSTRER	MY STORE ERROR
12	MFSTM	MY STORE FAST TIME-OUT
13	BAER	BRANCH ALLOWED ERROR
14	OWRTER	OTHER STORE WRITE PROTECT
15	OSTRER	OTHER STORE ERROR
16	OFSTM	OTHER STORE FAST TIME-OUT
17	IOMLTER	I/O MULTIPLE CHANNEL SELECT
18	PTRER	PT RESET RECEIVED BY ON-LINE 3A CC
19	SWER	SWITCH RECEIVED BY ON-LINE 3A CC
P_L	IOER	I/O CHANNEL ERROR
P_H	IOPARER	I/O BAD PARITY RECEIVED

Fig. 1—Error Register

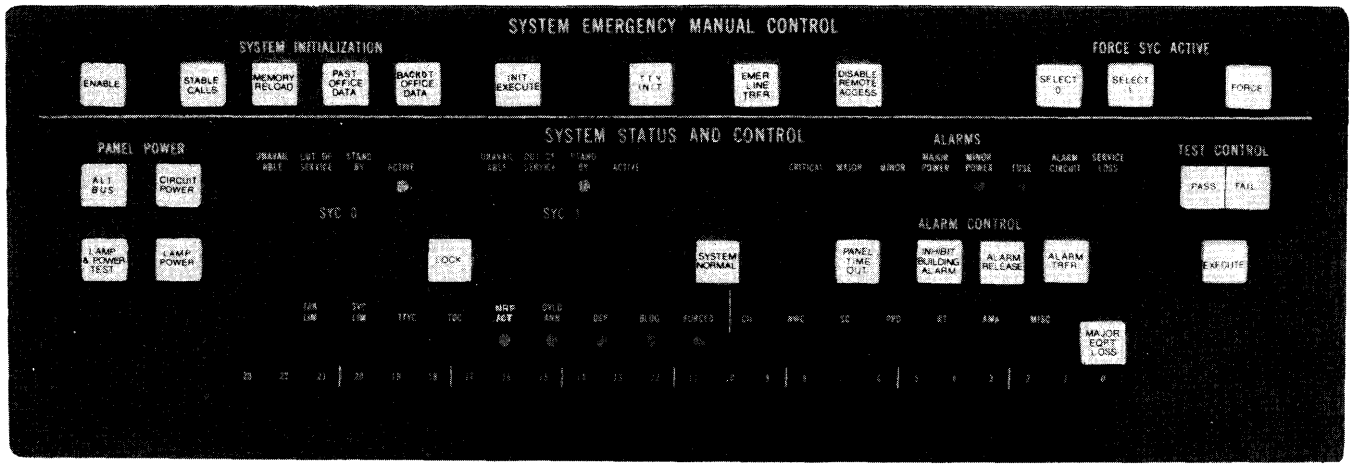


Fig. 2—System Status Panel

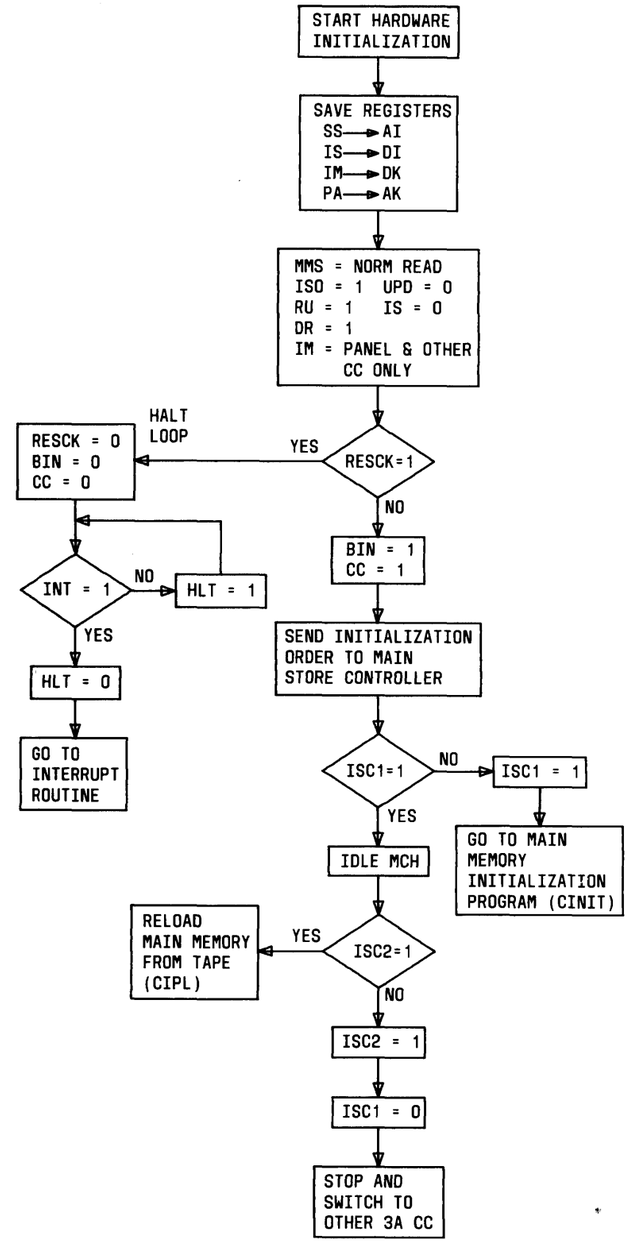


Fig. 3—Initialization Microprogram

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00	MINUTES PAST LAST HOUR						SECONDS PAST LAST MINUTE						RCYCLVL	INITIALIZE LEVEL OLD NUMBER		
01			STABLE CLEAR							PPD 1	NC 1	SC 1		PPD 0	NC 0	SC 0
02	OFF-LINE POWER KEY OPERATED	UNCOND SWITCH IN PROGRESS	RESTORE CU IN PROGRESS	MCH OUT-OF-SERVICE	SSP OUT-OF-SERVICE	OFF-LINE MAS OUT-OF-SERVICE	0 = CU 0 1 = CU 1 ON-LINE	OFF-LINE CU PANEL IN MANUAL	PROGRAM USE OF OFF-LINE CU	UPDATING OFF-LINE MAS		OFF-LINE CU REMOVED AUTOMATICALLY	SYSTEM IN INIT INTERVAL	OFF-LINE CU IS UNAVAILABLE	OFF-LINE CU OUT-OF-SERVICE	OFF-LINE CU READY
03	SSP MAN_INIT REQ	MEMORY RELOAD REQ					MCH FAILED	0=CU 0 1=CU 1 INITIALIZING	NO ACCESS TO OTHER MAS	MAS OUT-OF-DATE	INIT START BY RELOADING MEMORY	CU LOCKED OR FORCED ON-LINE	2ND TIME-OUT	1ST TIME-OUT	MCH MESSAGE CAUSE OF INIT	IF = 1 NO SWITCH OCCURRED
04	2ND_TO	1ST_TO	25 MS COUNTER								TIMING COUNTER					
05	CONTENTS OF IS															
06	CONTENTS OF BITS 19-16 OF PA REGISTER															
07	CONTENTS OF BITS 15-0 OF PA REGISTER															
08	CONTENTS OF R8															
09	CONTENTS OF R9															
10	CONTENTS OF R10															
11	CONTENTS OF R11															
12	SPARE															
13	CONTENTS OF INTERRUPT MASK REGISTER															
14	CONTENTS OF BITS 19-16 OF DB REGISTER															
15	CONTENTS OF BITS 15-0 OF DB REGISTER															
16	CONTENTS OF R12															
17	CONTENTS OF R13															
18	CONTENTS OF R14															
19	CONTENTS OF R15															
20	CONTENTS OF HOLD-GET COUNTER															
21	SPARE															
22 | | 31	CONTENTS OF SYSTEM STATUS REGISTER, WORDS 0 & 1 OF 3 HOLD-GET SLOTS, AND CONTENTS OF ERROR REGISTER															

Fig. 4—Post-Mortem Format

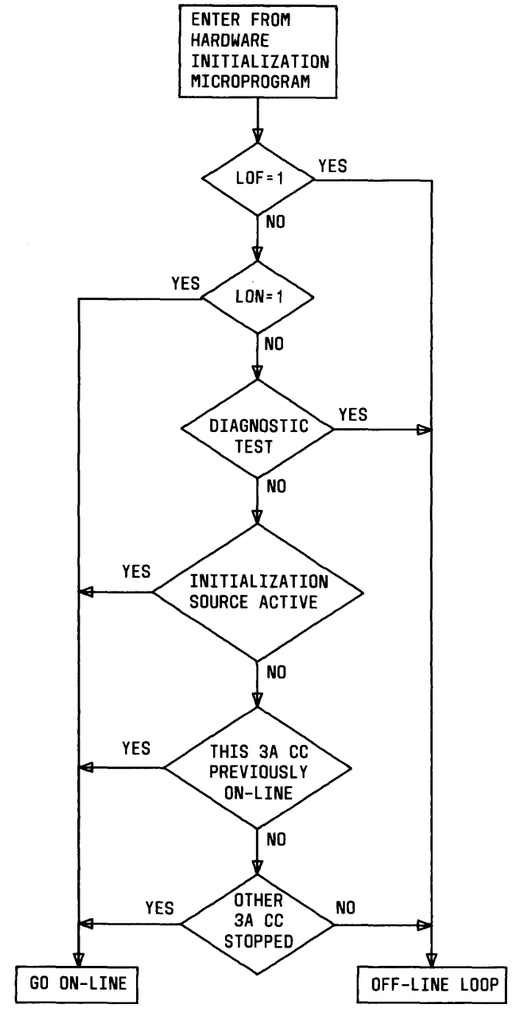


Fig. 5—Initialization Main Program

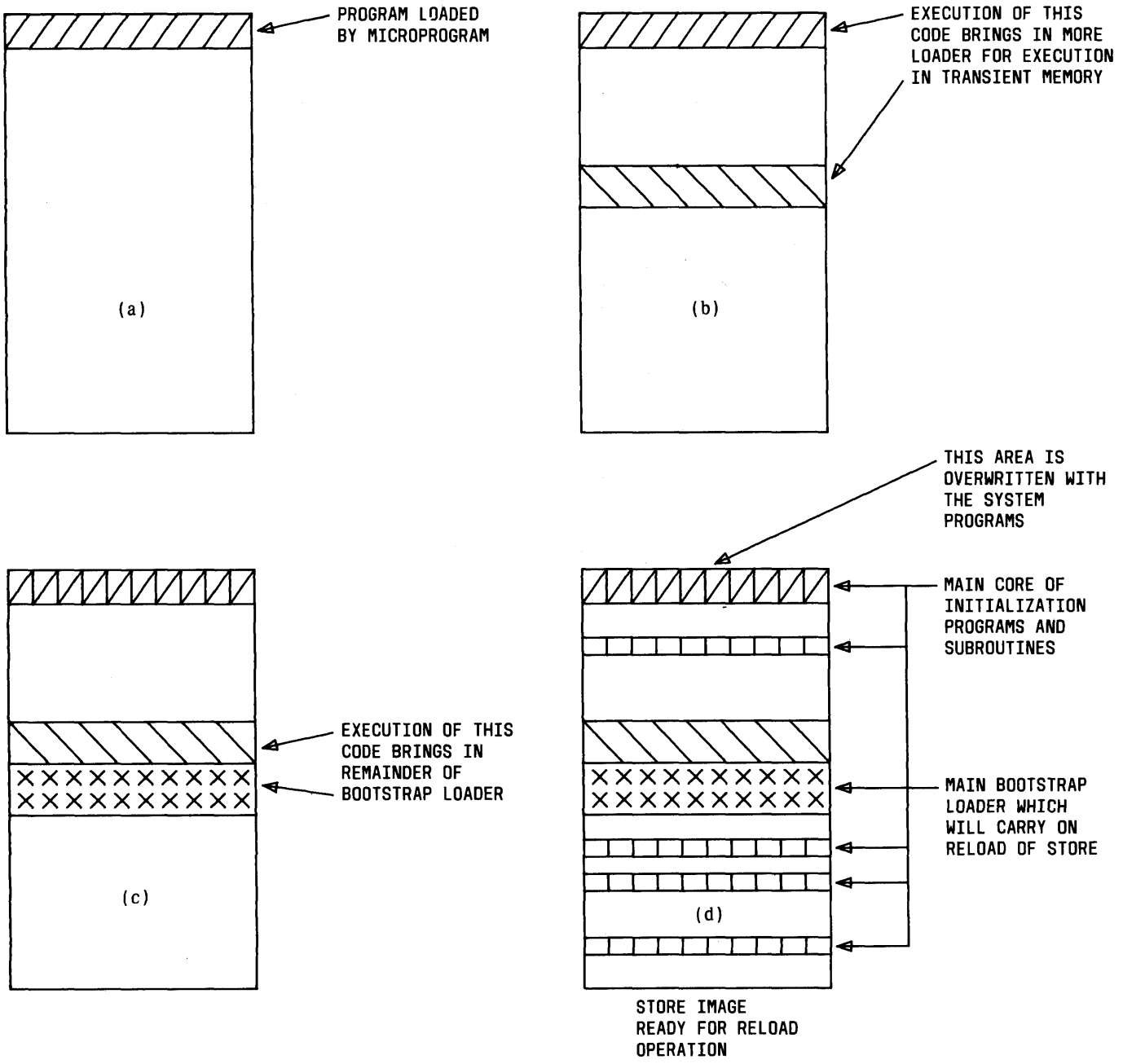
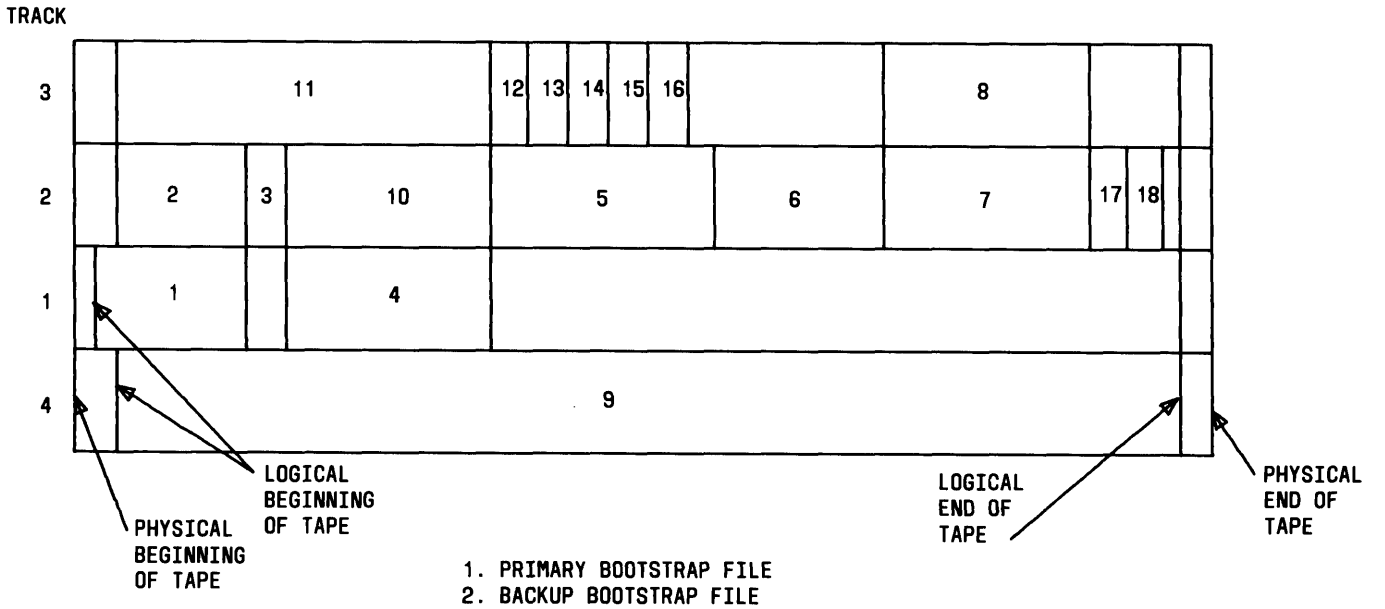


Fig. 6—Initialization Program Memory Register



1. PRIMARY BOOTSTRAP FILE
2. BACKUP BOOTSTRAP FILE
3. PRIMARY CHECKSUM FILE
4. GENERIC FILE
5. TRANSLATION FILE
6. PATCH FILE
7. BACKDATED TRANSLATION FILE (BACKDT1)
8. BACKDATED TRANSLATION FILE (BACKDT2)
9. PAGED PROGRAM FILE
10. OFFICE RECORD FORM FILE
11. OFFICE RECORD FORM FILE
12. MULTISCAN FUNCTION OVERLAY SAVE FILE (ROLL)
13. LINE CLASS CODE TABLE FILES (LCCTBL0)
14. LINE CLASS CODE TABLE FILES (LCCTBL1)
15. LINE CLASS CODE TABLE FILES (LCCTBL2)
16. TRAFFIC FILE
17. DIRECTORY
18. LABEL FILE

Fig. 7—Tape Format

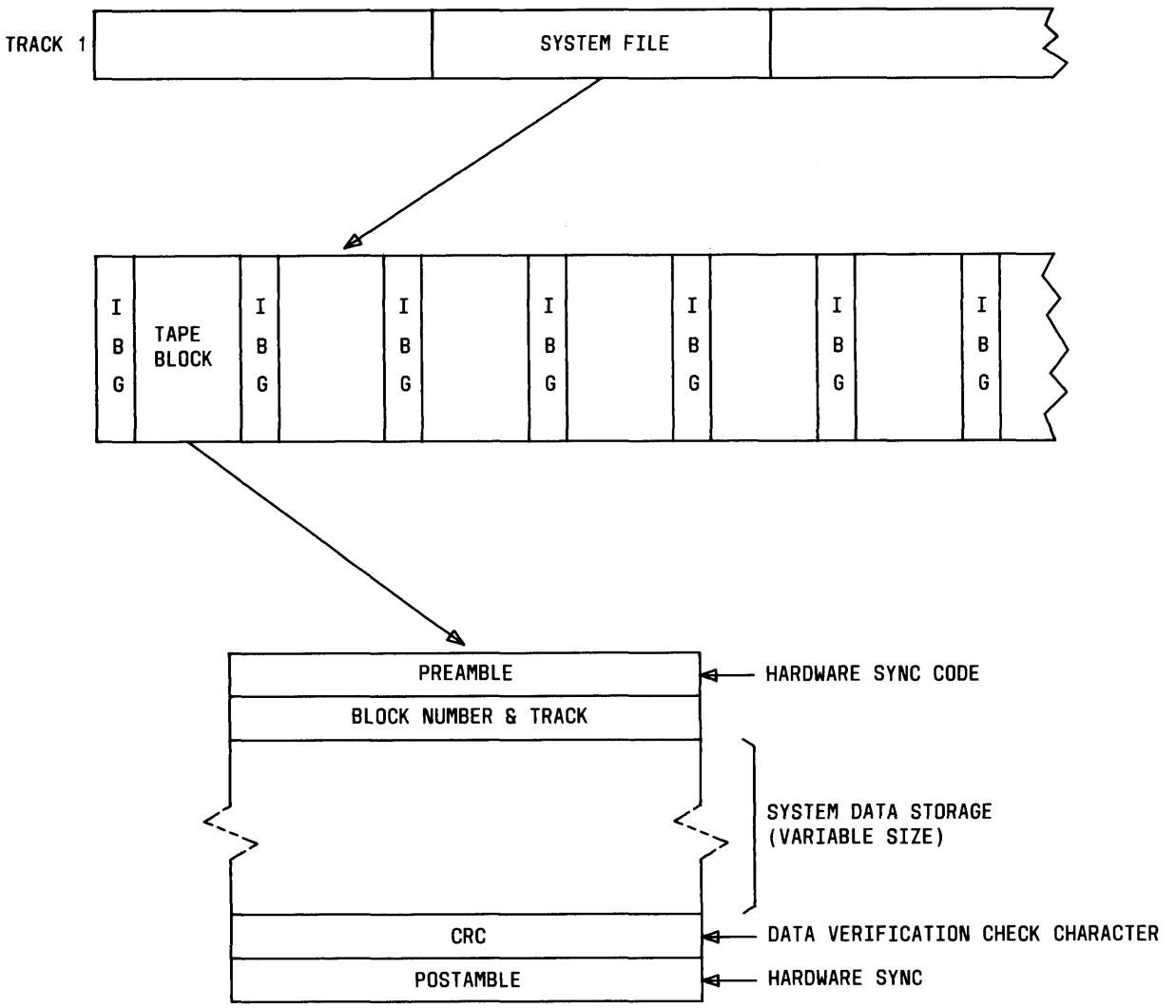


Fig. 8—Tape Track Layout

TABLE A

ABBREVIATIONS AND ACRONYMS

ABBREVIATION	TERM
BHC	Block Hardware Check
BIN	Block Interrupt
BOT	Beginning of Tape
BTC	Block Timer Check
CRC	Cyclic Redundancy Check
DML	Data Manipulation Logic
DR	Data Ready
ER	Error Register
ESS	Electronic Switching System
IBG	Interblock Gap
IM	Interrupt Mask Register
I/O	Input/Output
IS	Interrupt Set Register
ISC	Initialization Sequence Counter
ISO	Isolate Bit
LOF	Lock Off-Line
LON	Lock On-Line
MAR	Memory Address Register
MCH	Maintenance Channel
MINT	Microinterpret Mode
MIR	Microinstruction Register
MSR	Maintenance State Register
OMAS	Off-Line Main Store
P_H	Parity High
P_L	Parity Low
PT	Program Timer
RAR	Return Address Register
RU	Return Address Register Update
SSP	System Status Panel
STP	Stop
TCR	Transient Call Record
TDC	Tape Data Controller
UPD	Update
3A CC	3A Central Control