SYSTEM CONTROL 3E3 GENERIC SOFTWARE SUBSYSTEM DESCRIPTION NO. 3 ELECTRONIC SWITCHING SYSTEM

CONTENTS PAGE

1. GENERAL

1.01 This section provides a functional description of the software required for system control for the No. 3 Electronic Switching System (ESS). The software discussed includes:

- Base level control
- 3A Central Control (3A CC) System clock

- Interrupts
- Common system subroutines.

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

1.03 The information in this section applies only to the 3E3 generic or later.

1.04 The software subsystem description for system control in No. 3 ESS compatible with Issue 4 of the SO-2 generic is in Section 233-152-125.

1.05 Part 3 contains a glossary of terms, abbreviations, and definitions necessary for comprehension of the information contained in this document.

1.06 The following Bell System Practices may be helpful in understanding system control:

SECTION	TITLE
233-150-100	General Description Software Subsystem Description No. 3 ESS
233-151-105	Call Processing Software Subsystem Description No. 3 ESS
233-152-120	Teletypewriter Software Subsystem Description No. 3 ESS
233-152-130	Tape Operations Software Subsystem Description No. 3 ESS
233-153-125	Alarms and Status Reporting Software Subsystem Description No. 3 ESS
233-153-130	Initialization and Processor Fault Recovery Software Subsystem Description No. 3 ESS
254-300-110	3A Central Control Description Common System
254-300-120	3A Central Control (3A CC) Theory of Operation Common Systems
254-300-180	System Status Panel, System Status Panel Controller, and System Status Panel Relay Unit Description and Theory of Operation Common Systems 3A Processor.

1.07 Information contained in this section will aid in accessing the software listings which contain detailed program functions and coded software instructions used for system control. Table A contains the acronyms, names, and program listing numbers for each program referenced in this document.

2. SOFTWARE SYSTEM CONTROL STRUCTURE

GENERAL

2.01 The basic program structure of the No. 3 ESS is a closed loop in which a set of major programs executes sequentially. This loop is the base level loop. Most call processing programs and those maintenance tasks which can be deferred are performed during base level. Time critical functions are performed during interrupts of the base level loop; interrupts are described in paragraphs 2.49 through 2.73. The common base level monitor (CBLM) controls the base level loop since it determines the sequencing of programs in the loop. Refer to Fig. 1 for a general diagram of the base level loop.

BASE LEVEL LOOP

2.02 The base level transient call record (TCR) scanning routine (TCRSCN) is normally considered to be the first routine in the base level loop and the dispatcher routines are the last routines as shown in Fig. 1.

2.03 The TCRSCN program is normally considered first in the base level loop. Control is passed to TCRSCN by CBLM. The TCRSCN program examines all of the TCRs sequentially. When the TCR is active, its timer word is decremented by the time elapsed since the last scan. When the TCR timer is zero or when the TCR BACTION bit (which indicates base level action is needed) is set to one, TCRSCN invokes the routine associated with base level progress mark in the TCR.

2.04 The TCRSCN program uses the value of the base level progress mark for indexing into the base level progress mark branch table to transfer control to the proper routine. The base level progress mark routine performs the call processing function(s) such as coin functions, terminating call functions, digit interpretation, outgoing call functions, disconnect, error handling, etc, needed for the call at that time and returns to TCRSCN. The next TCR is then processed by TCRSCN.

2.05 The base level progress mark go to return address (GOTORA) is a special progress mark which causes a branch to the address stored in the TCR words RAD0 and RAD1. It is normally used after a real-time break for transferring control to the next function to be performed.

2.06 After TCR scanning, control passes to the input monitor program (INPUT), which performs input processing functions. Input processing consists of:

- Distribution of service requests and supervisory state changes to the appropriate processing routines for service of the inputs
- Timing functions for hits and disconnects
- Dynamic service protection
- Base level scan control (scanning of lines and scanning of trunks, junctors, and service circuits except immediate start and operator trunks is done in SCANS, which is called by INPUT)
- ~~Processor overload control.~~

Depending on the time already spent in the base level loop, INPUT may bypass the processing of some trunk, junctor, and service circuit inputs, the processing of some or all line requests, and the scanning of lines for originations.

2.07 The INPUT program gives control to CBLM after all input processing. Next, program BLMMA determines which application base level monitor functions are performed in this cycle of the loop.

2.08 The BLMMA program continues by initiating the performance of such tasks as timing for audible alarms and alarm key interface. Base level monitor functions and traffic functions are performed in alternate cycles of the base level loop, the base level monitor first determines which functions are to be performed during the present base level loop. Traffic functions are performed during the first time through the loop after initialization so BLMMA immediately transfers control to the traffic and plant measurements program, TRAFIC. TRAFIC performs traffic and plant measurement functions which are described in Section 233-152-135.

2.09 Control is then given to CBLM, the common base level monitor. Administrative functions such as the system state detector, system status panel controller, and time monitor software are contained in CBLM. After the monitors contained in CBLM are executed, CBLM initiates the execution of three other monitors, which are independent common system programs. They are the tape handler, the TTY handler, and the common system utilities handler. See paragraphs 2.19 through 2.48 for functions performed by CBLM.

2.10 When all of the above-described nondeferrable work has been completed in the base level loop, control is passed to the dispatcher in program CMMON. The function of the dispatcher is to allocate the remaining time of each base level loop to the deferrable jobs. In No. 3 ESS there are three types of deferrable jobs defined. These are:

(a) MSFC—Multiscan function controller

(b) AUDITS—Call processing audits

(c) MAS AUDIT—Main Store audit.

2.11 The dispatcher passes control between these three types of jobs according to a predefined priority schedule in program MMONA ensuring that no job falls below a minimum execution rate. In overload conditions, when an insufficient amount of time is available to maintain the above-mentioned minimum execution rate, the dispatcher will eventually enter a force mode to guarantee that the jobs will be executed.

2.12 A timer is used to ensure proper progression through the program system. The purpose of the program timer (bits 8 through 13 of the timing register) is to provide an initialization function if the timer is not periodically reset by software. Therefore, a reset of the program timer in various strategic points of code provides a means of checking software sanity. For example, CBLM resets the program timer (and increments the memory word SYSTIM) before passing control to each application monitor. When the software incorrectly branches around a reset of the timer or loops, the program timer will time out and an initialization will occur. Paragraphs 2.41 through 2.44 contain more detailed information related to program timer.

A. Application Portion of Base Level Monitor

2.13 The application portion of the base level monitor program BLMMA (1) contains tables used by CBLM, (2) contains several routines that could have been placed in other listings but for convenience are collected in BLMMA, and (3) invokes routines which are executed each time BLMMA gets control for base level monitor functions.

2.14 Included in the tables contained in BLMMA are the time monitor tables (SECTBL, MINTBL, and HRTBL) which contain the addresses of routines to be performed on a specific schedule. The time monitor in CBLM schedules and controls the execution of the routines and is described in paragraphs 2.41 through 2.44. The application monitor table (MONTBL) contains base level loop application monitors executed by CBLM. There are two entries to this table. The first entry in the table is ENTRY, the entry point to the TCR scanning program (TCRSCN). The second entry point is BMMAST, the entry point to the application base level monitor functions in program BLMMA. The multiscan function table (MSFTBL) contains an entry for each multiscan function (a function requiring several base level loops to complete). Paragraphs 2.31 through 2.40 provide more details regarding MSFTBL and multiscan function processing. BLMMA also contains the main store controller input/output (I/O) subchannel table (MASCIOSC).

2.15 Miscellaneous routines and entry points which reside in the BLMMA program listing but could have been placed elsewhere are as follows:

DSP CHK—Checks for dynamic service protection and controls the associated lamps.

PHWALK—Entered once each day to change the path hunt preference pointer to exercise the complete network when making connections.

CLKCHGD—Called by common system routines when the system clock is changed. When Local Automatic Message Accounting (LAMA) is provided, this routing enters the time change entry into the recording system and sets the TC bit in each LAMA call terminal memory record (TMR).

INITQUEND—Called to turn off the service-protected lamp when an initialization ends.

APPUTIL—Called during a utility interrupt to provide the opportunity for display of a register or another unique function provided by the application.

MSFAPCTL—Used by routines invoking multiscan functions to detect the termination of the multiscan function.

ASWCHK; ASWBEGIN; ASWCOMPL—These entry points provide for periphery initialization during a processor switch.

TTYEIRTN—Even TTY interrupt return point which provides a branch to the even tape interrupt entry point for tape interrupt processing in the common tape handling program, CTAPH.

TTYOIRTN—Odd TTY interrupt return point which provides a branch to the odd tape interrupt entry point for tape interrupt processing in CTAPH.

TAPEIRTN and TAPOIRTN—Tape interrupt return points. A branch is made to routine INTEND in the common systems subroutines program (CSYSUB) for ending the TTY/tape interrupts.

AUDOVER—Provides the base level loop monitor return from audit routines. Control is then transferred to TCRSCN (the first program in the base level loop).

CDGTST—The 3A CC diagnostic test vector table.

OA_CHK—Overload announcement system status panel indicator update routine.

2.16 Program CBLM passes control to BLMMA at entry point BLMMAST. BLMMA clears the network queue bits to prevent blocking of the queue. Control is then passed to EA_ENTRY in the trunk, line, service circuit, and network link error analysis program, TSVEA, which analyzes peripheral test failure information in the error analysis buffer. Control is then given to program TLTPC if the trunk and line test panel is in the active state.

2.17 Since miscellaneous base level monitor functions and traffic functions are performed in alternate cycles of the base level loop, BLMMA determines which functions are to be performed. If traffic functions are to be performed, first control is given to the traffic program (TRAFIC).

2.18 Otherwise, miscellaneous base level monitor functions are to be performed. BLMMA calls MCSSPS in the maintenance subroutines program, MCSUB, for system status panel administration functions. Then the HT_CHECK subroutine in the test vertical status administration program, TVADM, is called to check for heavy traffic conditions. Once BLMMA functions are complete, control is transferred to MONRTN in CBLM for common systems base level monitor functions.

B. Common Base Level Monitor Functions

2.19 The CBLM program determines the sequencing of base level programs. Three of the common system monitors in the base level loop are contained in CBLM. These monitors are: system state detector, system status panel controller, and the time monitor. There are three other common system monitors which are independent programs not contained in CBLM but given control by CBLM. These monitors are: the tape handler (CTAPH), teletypewriter handler (CTTYH), and common utilities (CUTIL). Tape handler functions are described in Section 233-152-130, TTY functions in Section 233-152-120, and common utility functions in Section 254-340-080. The multiscan function controller is controlled by the dispatcher in program CMMON.

Monitor Sequencer

2.20 The base level sequencer (MONSEQ) executes each common system monitor once and can execute a series of major application monitors and/or routines. The order and identity of the application monitors are contained in a table named MONTBL which is defined in program BLMMA. MONTBL contains a 2-word entry for each execution of a monitor during a single base level loop. The entry consists of the entry point address and the order of the entries is in the order of execution. The CBLM resets the program timer and increments memory word SYSTIM between each MONTBL entry.

System State Detector

2.21 The system state detector in CBLM performs three types of functions: (1) administrative, (2) critical system audits, and (3) system state determination.

2.22 Administrative functions include the initialization of constants and checking for the initiation or termination of certain system conditions. The system state detector initializes the interrupt counter in memory to minus the maximum number of external interrupts permitted in a base level loop. The maximum number is retrieved from the common temporary store definition (CTSD) area of memory. Each time an external interrupt occurs, the interrupt counter is incremented by the interrupt begin subroutine (INTBGNX) in program CSYSUB and is used to detect stuck interrupts. When the maximum number of allowed interrupts is exceeded in a base level loop, the begin interrupt routine blocks the problem interrupt by setting the corresponding bit in the interrupt mask. Stuck internal interrupts cause a switch to the other central control (CC) because of a time-out of the program timer. Administrative functions performed by the system state detector also include incrementing the word in the CTSD reserved for the on-line CC to accumulate the length of time the CC remains on-line. Some initialization processing is performed by the system state detector and is described in Section 233-153-130 on system initialization and fault recovery.

2.23 If several control registers are left in some abnormal state for even a short time, processing capability is seriously affected. These registers are therefore audited each base level loop. The hold-get register is usually set to the highest hold-get slot upon entry to the system state detector; therefore, it can be audited against this known value to protect against drifting in value. The audit is not done, however, when it is known that the register contains a nonstandard value.

2.24 The system status register of the on-line CC is audited also to detect illegal combinations or states of bits. Errors in critical bits result in a stop and switch of the processors. Checks are made to ensure that certain important hardware functions (eg, interrupts, hardware checks, etc) are not inhibited. When these bits are in the wrong state, they are corrected and an error TTY printout is made. After the on-line system status register is audited, the off-line system status register is accessed and the CC bit is checked to verify it is 0 indicating that CC is off-line.

2.25 After the system status register audits, the system state detector performs an on-line manual switch audit. The power key, manual key, and test mode reversal switch should normally not be operated (after cutover) in the on-line CC. Likewise, neither the test mode reversal switch nor the power switch should be operated in the off-line CC. An audit is therefore made of the status of the keys and an error message is printed when an error is detected for the first time.

2.26 After the key audit, an audit is performed on the maintenance state register and the interrupt mask register with errors being reported by a TTY message. The interrupt mask register is normally zero at base level with the only exception being the recovery mode for external stuck interrupts. To prevent accidental blocking of an interrupt, a record of blocked interrupts is maintained in a word in main store; therefore, the audit compares the word with the interrupt mask register. Interrupts corresponding to bits which do not match are unblocked and an error message is printed (except in the case of another CC interrupt).

2.27 The third function of the system state detector is to update the system state word (SYSTATE). Key elements of the system (both hardware and software) are analyzed and combined to produce an overall system state. There are four major system states:

- Off-line standby—the absence of the other three states
- Off-line unavailable—the system is locked or forced from the system status panel
- Initialization in progress—initialization occurred less than 1024 base level loops ago
- Off-line out-of-service—the presence of one of the following substates.

The off-line out-of-service substates are:

- The off-line was removed from service automatically because a sanity test failed or excessive errors were encountered while the CC was on-line
- The off-line was removed from service by a TTY request
- The off-line store is not in date with the on-line store
- The off-line CC is currently being used by a program
- The off-line CC is currently under manual control.

2.28 Update of the system state is normally performed by the routine causing a change; therefore, the use of system state detector routine is primarily to ensure that the software state remains compatible with the actual state of the system.

System Status Panel Controller

2.29 The system status panel (SSP) is a common system circuit which is used to provide key functions for manual requests and to display status information on a set of lamps. Use of the keys and lamps is split between the common system and the application. The SSP is divided into modules which consist of 8 key functions and 16 status lamps. Modules are divided into three buffers which form the basic logical unit of the SSP. Two key function buffers and two status buffers are reserved for common system use. The buffers are used to display CC status and provide control of system initialization, system overrides, and the step and repeat mode of multiscan function.

2.30 The system status panel controller which resides in CBLM provides the software to control the SSP and administer the SSP common system portion. The SSP controller is only executed when the SSP is not out of service. The system status panel controller consists of two areas: (1) the synchronizing of the SSP and the memory map of SSP status bits, and (2) subroutines for performing other SSP functions. The subroutines include: the SSP key buffer update, the update of the SSP buffers from the memory map, the change or read system status panel buffer subroutine, and the remove and restore system status panel subroutines. Refer to Section 233-153-125 for more details on alarm and status reporting and Section 254-300-180 on the system status panel and system status panel controller.

Multiscan Function Controller (MSFC)

2.31 The MSFC provides centralized control for miscellaneous routines which must be controlled in order to avoid interference between functions. A multiscan function (MSF) is by definition any function which requires real-time breaks and is not regularly scheduled. A real-time break is taken when a function cannot be completed in the time allotted during a particular period. In addition, all nonresident programs and any function that can operate under step or repeat control must be multiscan functions.

2.32 Control for multiscan functions is maintained in a block of temporary store named the MSF matrix (MSFMTX). The multiscan functions correspond to columns and multiscan function states correspond to rows in the matrix. The four state rows are request buffer (RQB), abort request (ABB), in progress (IP), and abort (AB). Therefore, there are 16 possible states for a multiscan function.

2.33 In addition to the four state variables, there are three "allow" variables for each MSF.

NA—MSF is not allowed by the system state (automatic)

FORCA—Forced allowed by TTY (override of NA)

FORCNA—Forced not allowed by TTY (override of NA).

When a multiscan function is "not allowed," future requests for that multiscan function are denied. If the multiscan function is presently requested or in progress, it is terminated. The subroutine which processes the allow and inhibit multiscan function TTY input messages is a part of the MSFC in CBLM. The subroutine updates the appropriate "allow" bits for that multiscan function.

2.34 There are four possible relationships between two multiscan functions (MSF1 and MSF2):

DENY—When MSF1 is requested while MSF2 is requested or in progress, the request is denied. The craft can abort MSF2 and then rerequest MSF1.

HOLD—Requests for MSF1 are accepted but do not go in progress while MSF2 is in progress or being aborted.

GO—Requests for MSF1 are accepted and go in progress independent of MSF2.

PREEMPT—Requests for MSF1 are accepted and MSF2 is aborted. MSF1 does not go in progress while MSF2 is in progress or aborting.

The relationsips are unidirectional (that is, the relationship of MSF1 to MSF2 is not necessarily the same relationship as MSF2 to MSF1). The four relationships are encoded by two variables (MSFRELA and MSFRELB).

2.35 The two relationship variables (MSFRELA and MSFRELB) are a part of the table MSFTBL. In the table, a 4-word entry is dedicated to each multiscan function. The first two words contain the starting address of the corresponding multiscan function, while the remaining two words contain MSFRELA and MSFRELB. Bit N of the two words gives the relationship of that multiscan function to the Nth multiscan function. The table MSFTBL resides in BLMMA.

2.36 The normal method for requesting a multiscan function to be executed is via the subroutine MSFREQ in CBLM. The type of request (normal, step mode, repeat mode, and repeat mode with no print option) and the number of the multiscan function being requested are identified to the subroutine. The subroutine analyzes the type of multiscan function and MSF data to determine whether a request can be accepted or must be denied. A request is denied when either the "not allowed" or "forced not allowed" bits are set. A request is also denied when a conflict exists with other multiscan functions determined from analysis of the MSFRELA and MSFRELB variables and the MSF matrix. When a request is accepted, the MSF matrix is updated to show that multiscan function requested, the mask of lower priority multiscan functions to be aborted is updated and the MSF abort subroutine is called, and the program requesting the multiscan function is signaled that the request is accepted.

2.37 A request to terminate a requested or in-progress multiscan function is processed by the MSF abort subroutine (MSFABT). A mask of multiscan functions to be aborted is passed to the subroutine. The subroutine updates the matrix to request the abort and provides the appropriate return code to the calling program. Either all multiscan functions for which aborts were requested were inactive or already aborting, or at least one abort was actually performed. When the multiscan function is in step or repeat mode, the repeat and step lamps are also turned off. The abort subroutine only requests the termination of a multiscan function; the multiscan function terminates itself.

2.38 Upon entry to the multiscan function controller, a check is made for active repeat and step modes and the appropriate action is taken when the EXECUTE button on the system status panel has been pushed. For the step function, the multiscan function is requested again. For the repeat function, either the repetitive testing is stopped or reinitiated because of operation of the EXECUTE switch.

2.39 Next, the main analysis of the MSF matrix and multiscan function interactions is performed by the MSFANLZ subroutine. Any requests or in-progress multiscan functions which are inhibited (associated "not allowed" bits set) are marked to be aborted. Requested or in-progress multiscan functions are aborted when preempted by a request. Requests which preempt aborting multiscan functions are held until the abort is complete. Requests to be held for other active multiscan functions are held. Otherwise, the state of the multiscan function is updated to an in-progress state. After all requests are processed, a check is made for aborting or in-progress multiscan functions. At most, one segment of one multiscan function is executed on a given entry to the controller. Aborting multiscan functions have priority. If there are no aborts, a search for an in-progress multiscan function is performed beginning with the first multiscan function after the multiscan function executed last by the MSF controller. The MSFANLZ subroutine branches directly to the multiscan function to be executed or aborted using the address given in the MSFTBL entry. An entry code is passed to the multiscan function to indicate the type of entry:

Bit 0—0 = in-progress entry, 1 = MSF abort entry

Bit 1—0 = per scan entry, 1 = first entry

Bit 2—0 = aborting an in-progress MSF, 1 = aborting a request.

The multiscan function then analyzes the entry condition, executes the proper segment, and returns control to the MSF controller along with specification of return conditions in a return code:

0—Reserved for internal control

1—Reserved for internal control

2—MSF will stay in process or continue to abort

3—Final return, no printing to be done

4—Final return, fail, print TTY message

5—Final return, pass, print TTY message

6—Final return, fail, print TTY message + supplement

7—Final return, pass, print TTY message + supplement

The MSF controller analyzes the return code and takes the appropriate action. Also, when a completing multiscan function is in the repeat or step mode, the lamps on the system status panel are updated to show pass or fail status and a message is printed on the TTY if it is the first execution or if the result is not the same as that of the last execution.

2.40 On occasion, execution of a multiscan function must be suspended at a particular point and a real-time break taken. A multiscan function can take a real-time break by calling the WAIT subroutine. WAIT saves system status and returns control to the MSF controller for one base level loop. During the next base level loop when the MSF controller is being executed, the WAIT subroutine is given control and thereby returns control to the multiscan function which took the real-time break. In addition to the WAIT subroutine, there are several other routines contained in the multiscan function controller including:

- A subroutine to determine the state of a MSF
- A subroutine which handles allow or inhibit TTY messages
- Clear step and repeat MSF TTY input message subroutine
- A subroutine which identifies all in-progress and aborting multiscan functions.

There are several audit routines which reside in the multiscan function controller as well as a multiscan function for the update of off-line store.

Time Monitor

2.41 The time monitor maintains a software clock giving seconds, minutes, hours, days, months, and years and initiates time periodic functions relative to the clock. The temporary store words SYSTIM and COUNT are used to determine when one second has elapsed (1 second = forty 25-ms intervals). COUNT is initially set equal to the contents of SYSTIM + 40. Then the two words are compared upon each entry to the time monitor. When SYSTIM becomes equal to or greater than COUNT, at least one second has passed. When a second passes, the time monitor updates the software clock—that is, the appropriate units of time are incremented.

2.42 The time monitor uses the software clock to determine which, if any, time periodic functions should be performed. These periodic functions are listed in three tables (SECTBL, MINTBL, and HRTBL) in BLMMA. Each entry in the tables consists of two words containing the entry point address and a 12-bit constant. The constant specifies which second past the minute (SECTBL), minute past the hour (MINTBL), or hour of the day (HRTBL) the function is to be performed. The functions may be located in BLMMA or other programs. When a unit of time has recycled, the appropriate table(s) is examined to determine whether a function(s) must be performed. A branch is made directly from the time monitor to the function address with control being returned to the monitor when the function is completed.

2.43 The time monitor also contains several miscellaneous subroutines. There is a subroutine to process the **SET:CLK** (set clock) input message used to initialize the software clock. One subroutine processes the **OP:CLK** (output clock) input message used to request a printout of the software clock. One subroutine (TIMSYNC) synchronizes the software clock to an external source.

Other Common System Monitors

2.44 The remaining monitors controlled (called) by CBLM are independent common system programs and do not reside in CBLM. They are the common tape handler (CTAPH), common TTY handler (CTTYH), and common utilities (CUTIL). Tape handling is described in Section 233-152-130, the TTY handler in Section 233-152-120, and utilities in Section 254-340-080.

3A CENTRAL CONTROL SYSTEM CLOCK

2.45 The system clock is a 4-phase clock composed of a crystal oscillator, phase-splitting circuit, and a set of counter circuits. The basic function is to provide timing signals used for controlling various system functions such as data timing, gate control, synchronization of events, timed interval interrupts, etc. Another function performed includes initiation of a control unit switchover if software execution breaks down.

2.46 Clock phase 1 drives a binary counter to generate timing intervals of 1.2 microseconds, 19.2 microseconds, 38.4 microseconds, 76.8 microseconds, and 153.6 microseconds. The timing register is driven by the 19.2-microsecond interval from the counter to provide counts for timed interrupts at intervals of 1.25, 5.0, 10.0, and 25.0 milliseconds. This is done in the timing counter portion of the register (bits 0 through 7).

2.47 In addition, the timing register provides a basic system sanity check via the program timer (bits 8 through 15), which derives its timing inputs from the timing counter. In order to prevent the program timer from timing out and initializing the 3A CC, software must routinely reset the program timer (paragraph 2.12). Because the input to the program timer is the output of the 25-ms count, the program timer does a straight binary count of the 25-ms counts. When the bit 6 of the timer in the on-line 3A CC is set, a stop-and-switch message is generated to initialize the off-line 3A CC. The program timer continues to count and when the off-line 3A CC does not come on-line and reset the program timer, bit 7 will be set and another initialization message generated. In this case, an initialization to the 3A CC in which the timer is contained will result and it will attempt to initialize itself in order to achieve an operating state.

2.48 When the time-out occurs in the timer of the off-line machine and bit 6 is set, an initialization of the off-line 3A CC will occur. (The on-line machine is expected to send a periodic maintenance channel order to the timer in the off-line machine to prevent it timing out.)

INTERRUPTS

A. Interrupt Structure

2.49 The interrupt facility provides the means of breaking into the program flow so that a timed or more urgent task may be performed. Interrupts may be caused by such inputs as control panel operations, timing counter signals, and certain error conditions.

2.50 The interrupt structure of the 3A CC consists of 16 separate interrupt levels specified by an interrupt set (IS) register and a corresponding 16-bit interrupt mask (IM) register which is used to inhibit interrupts. The IS register buffers incoming interrupts until each is acted on by the 3A CC. When an interrupt occurs, the corresponding bit in the IS register is set. Each of the interrupt bits in the IS register has a corresponding masking bit in the IM register which, if set, will block recognition of the interrupt. This mechanism permits the relative priority of the interrupts to be effectively controlled during the execution of an interrupt routine.

2.51 Interrupts are serviced in order of priority which is established by the structure of the IS register where bit 0 represents the highest priority and bit 15 represents the lowest. However, relative priority of the interrupts may be controlled by setting an interrupt mask during the execution of interrupt routines. In this case, a routine may set an interrupt mask in which a 0 will designate a priority for a specific interrupt which will give it higher priority than the interrupt being serviced by the routine. Interrupts of equal or lower priority are identified by 1s in the IM register.

2.52 During certain processing (eg, system initialization), it is desirable to block all interrupts. This is accomplished by setting the block interrupts (BIN) bit in the system status (SS) register. When the BIN bit is set, all interrupts are blocked, regardless of the contents of the IM register. When the BIN bit is reset, interrupts are enabled according to the IM register.

2.53 The current 3A CC interrupt assignment is:

Level 0—Unassigned

Level 1—Unassiged

Level 2—Unassigned

Level 3—Panel matcher interrupt

Level 4—Unassigned

Level 5—Error interrupt

Level 6—Unassigned

Level 7—Other 3A CC interrupt

Level 8—Unassigned

Level 9—Timer interrupt (10 ms)

Level 10—Even TTY and even tape interrupt input

Level 11—Odd TTY and odd tape interrupt input

Level 12—High-speed TTY interrupt

Level 13—Manual panel execute interrupt

Level 14—Even tape interrupt

Level 15—Odd tape interrupt.

B. Interrupt Functioning

2.54 When an interrupt occurs and the corresponding bit in the IS register is set, all unmasked bits of the IS register are ORed together to generate an interrupt present (INTPRS) signal.

When an interrupt is accepted (BIN bit = 0), an interrupt signal (INTRP) is generated and the address for the interrupt servicing microcode sequence is forced into the microaddress register (MAR). Interrupts are serviced at the end of microinstruction sequences. The microsequence tests the IS register for the highest level of interrupt and translates the bit position into a data constant which points to a main memory location containing a pointer to the appropriate interrupt routine. This location is defined as the interrupt transfer vector and is contained in a transfer vector table defined in the transfer vector table program, TVTAB.

2.55 The interrupt facility in the 3A CC is used not only as part of the maintenance and error-reporting function but also as a part of normal data transfer functions. When a peripheral device which uses interrupts needs to communicate with the 3A CC, it sets the interrupt (INTP) lead, which sets the associated bit in the IS register. When the 3A CC accepts an interrupt for servicing, ACKI is generated and returned to the interrupting device to indicate that the request is being processed and that the device should identify itself. (The bit set in the IS register identifies only the channel unit.) Each peripheral device is assigned a specific INF lead to be used for identification purposes. When ACKI is received, the interrupting device sets its INF lead to logic 1, while all other devices set their INF leads to logic 0.

2.56 Interrupt begin and end subroutines in the common system subroutine program, CSYSUB, are used by interrupt routines to save and restore the state of the system for interrupts. INTBEGIN and INTBGNX begin the interrupt by storing system status and updating the IM register to block lower priority interrupts. INTBGNX is used for interrupts from external devices, while INTBEGIN is used for all other interrupts. Only a limited number of external interrupts are allowed each base level loop; therefore, an excessive number of calls to INTBGNX may result from a stuck interrupt. INTBGNX increments a counter each time it is executed and compares the counter with the maximum number of interrupts allowed. When the counter exceeds the maximum, the interrupt is blocked by setting the appropriate bit in the IM register, printing a TTY message, and returning to base level. Stuck internal interrupts result in a switch to the other 3A CC because the program timer times out. The INTEND subroutine is called by an interrupt routine to end an interrupt. INTEND restores system status and transfers control back to the location in the base level loop which was interrupted.

C. Interrupt Software Descriptions

Panel Matcher Interrupt (3)

2.57 The conditions for the panel matcher interrupt are established by a craft person entering a TTY message enabling the panel matchers to cause an interrupt when the match of an address or data word is detected. When the match is detected, interrupt 3 will occur. Entry from the transfer vector is to the MATCHINT subroutine in the common utilities program (CUTIL). MATCHINT increments an interrupt counter and tests to ensure that the number of interrupts which have occurred since the counter was cleared is within a prescribed limit. When the limit is exceeded, the interrupt is disallowed. When the number is within limits, MATCHINT will cause the register of the program being interrupted to be saved and will then call the UTILPROC subroutine in CUTIL to process the request. The function may be to monitor or load certain store locations or registers, as requested by the original messages.

Error Interrupt (5)

2.58 An error interrupt is initiated when bit 14 or higher is set in the error register (ER). This group of bits is ORed together and the resulting signal sets bit 5 in the IS. The type of errors that can cause interrupt 5 are:

(a) Attempted on-line store write in write-protected area (bit 11)

(b) Attempted off-line store write in write-protected area (bit 14)

(c) Off-line store parity error (bit 15)

(d) Off-line store fast time-out on read or write function (bit 16)

(e) Error in I/O main channel selection or error in 3-out-of-6 code check circuit (bit 17)

(f) A program timer reset received by the on-line 3A CC (bit 18)

(g) Switch message received by on-line 3A CC telling it to go on-line (bit 19)

(h) An I/O channel sequence error or I/O subchannel selection error (bit PL)

(i) An I/O bad parity received (bit PH).

2.59 Error interrupts are processed by the ERR_INT routine in the common initialization program (CINIT). ERR_INT causes the registers of the interrupted program to be saved and clears the IS. All information is collected from the ER and assuming only one error, the error causing the interrupt is identified. Once this is complete, the ER is cleared. Based on the error, a branch is made to the appropriate error correction routine.

Other 3A CC Interrupt (7)

2.60 Interrupt number 7 from the other 3A CC (off-line) can initiate one of three different requests as a backup to the maintenance channel (MCH). The three are: a stop request (SWINIT) resulting from a stop and switch activity, a switch completed notification (SWCOMPL) given when the other processor has started execution, or a zero program timer request (ZEROPT).

2.61 The interrupt routine which processes interrupt 7 is MCH_INT in CINIT. Initial processing is the same in each case and consists of saving the registers of the interrupted program and determining which of the three functions was requested. When the function has been identified, the proper routine is given control. The SWINIT function is performed by a microsequence and consists entirely of stopping the processor. SWCOMPL is entered after the other processor has taken over execution and is operating properly. The routine does general restoral activity on registers and prepares the processor to respond when another stop and switch is requested.

10-ms Timed Interrupt (9)

2.62 Base level is interrupted every 10 ms for the performance of frequently required tasks. See Fig. 2 for a flow diagram of 10-ms interrupts. The interrupt is initiated by the system clock (described in paragraphs 2.45 through 2.48). Control is given to TEN_MS_I in program DIGPRO (10-ms interrupt program—digit receiving and sending).

2.63 The first task is to store the system status and the registers. DIGPRO next scans the receivers and stores the results for subsequent digit-receiving functions. All dial pulse receiving trunks in the process of receiving digits are scanned and digit-processing functions are performed.

2.64 Control is transferred by DIGPRO to the network controller queue monitor (QMON) routine in the network queue and timing hopper monitor (QTMON). QMON processes the network controller queues, which contain TCR numbers that need orders issued to the network controllers. One order (if needed) is issued to each controller each interrupt and is verified the next interrupt.

2.65 Upon completion of network controller queue processing, control is returned to DIGPRO which then performs digit-receiving tasks for the receivers already scanned. Digit-receiving tasks include such functions as counting dial pulses, translating tones into digits, storing digits in the TCR, and signaling base level when base level action is needed. DIGPRO looks for interdigital periods and abandonments periodically (when a specified number of interrupts have occurred since the last check).

2.66 The program DIGPRO then passes control to the fast trunk scanning program, FASTTK. FASTTK does the scanning for stop dial and start signals and scanning for inputs from immediate start and operator trunks because of the frequency of scanning necessary for these trunks. FASTTK also performs timing and operation functions for the ringing and tone plant relays.

2.67 The FASTTK program passes control to the test vertical administration program (TVADM) which does a test vertical status block audit when required (every 640 ms). TVADM passes control to the timing hopper monitor (TMON) in the program QTMON. The timing hopper is used to store the numbers of TCRs which have sending functions to be performed and to provide real-time breaks in peripheral order processing. The TMON routine decrements the timers of all active slots in the hopper. It sends TCR numbers in any slots that time out to POINT (the peripheral order interpreter) for further peripheral processing.

2.68 The POINT program examines the interrupt progress mark in the TCR to determine whether sending or peripheral work is required.

When peripheral work is needed, POINT uses the peripheral progress mark in the TCR to address into the catalog of peripheral control sequences (PCAT) to determine the next peripheral functions that must be performed. It continues peripheral control functions for that TCR until another real-time break is required or peripheral work for the TCR is complete and then returns control to TMON, which processes the next hopper entry.

2.69 When the interrupt level progress mark in the TCR indicates sending functions are needed, POINT gives control to DIGPRO. The DIGPRO program determines from the peripheral progress mark in the TCR the sending functions to be performed and then performs the required functions. DIGPRO returns control to TMON for further timing hopper processing.

2.70 After processing the timing hopper, TMON returns control to FASTTK which adds one to the system timer. FASTTK then calls INTEND in CSYSUB to restore the system status to complete interrupt processing. Control is returned to base level at the point it was interrupted.

TTY and Tape Interrupts (10, 11, and 12)

2.71 Interrupt 10 results in control being given to TTYE_INT in the common TTY handler program, CTTYH, for even numbered TTY interrupts. (Each character entered or output on a TTY initiates an interrupt). After completion of TTY interrupt functions, control is returned to TTYEIRTN in BLMMA, which then sets interrupt bit 14 in the IS register for tape interrupt read and write functions. Interrupt 14 results in control being given to TAPEINT in program BLMMA, which gives control to TAPINTE in program CTAPH. Likewise, interrupt 11 results in control being given to TTYO_INT in CTTYH for odd numbered TTY interrupts. Return is made to TTYOIRTN in BLMMA which then sets bit 15 in the IS register for tape interrupt functions. Interrupt 15 results in control being given to TAPOINT in program BLMMA which gives control to TAPINTO in program CTAPH. After setting the interrupt bit in both cases, BLMMA ends the interrupt normally by calling subroutine INTEND in program CSYSUB. Level 12 is used for interrupts from high-speed TTYs and control is given to HSP_INT in program CTTYH for processing. For more details on TTY interrupt processing, refer to Section 233-152-120. For more information on tape interrupt processing, refer to Section 233-152-130.

Panel Manual Execute (13)

2.72 The panel functions are executed by a panel-halt loop microcode sequence. The loop is initiated when the 3A CC is off-line and the MANUAL switch on the panel is operated. Under these conditions, operating the EXECUTE switch on the 3A CC control panel sets interrupt bit 13.

2.73 When interrupt bit 13 is set, the states of the panel switches are interrogated by the microcode panel-halt loop sequence and the functions designated by the switches are performed. After the requested function is executed, the panel HALTED lamp is again lit. The panel-halt loop sequence is then restarted by operating the HALT switch. Interrupt 13 is never individually blocked since there is no 3A CC code that the interrupt causes to be executed and under normal processing conditions, the panel can be disabled by the MANUAL switch.

COMMON SYSTEM SUBROUTINES

2.74 The program CSYSUB contains a collection of common system subroutines. The subroutines are used by more than one program and most perform system control functions. See Table B for the subroutines in CSYSUB and corresponding functions performed. This list should be used as examples of the types of functions performed. Subroutines may be added or subtracted for different generic issues; therefore, for a complete up-to-date listing of the subroutines, refer to the CSYSUB program listing.

3. GLOSSARY

3.01 Terms, abbreviations, and definitions used frequently in this document follow.

BACTION Bit—Bit in TCR which is set to indicate base level action is needed.

Base Level—Major software loop including all functions not done during interrupt level.

Bit—The binary unit of information which is represented by one of two possible conditions, such as the digits 0 and 1, high potential or low potential, on or off.

CDPR—Customer dial pulse receiver.

Clear—To restore a storage device to the "Zero" state.

Dynamic Service Protection (DSP)—An automatic way of protecting the service of class A lines during a traffic overload.

Hoppers—Dedicated areas of writable memory into which entries with a fixed format are made.

Immediate Start Trunk—A trunk which does not wait for a signal before beginning to send dial pulses (usually from a step-by-step office).

10-ms Interrupt—A hardware-initiated interrupt which interrupts the base level loop every 10 ms for a period of time necessary to perform frequently required functions.

LAMA—Local Automatic Message Accounting.

Macro—A sequence of operations called by an abbreviated notation.

Operator Trunk—One of five types of trunks (TSP, TSPS, toll switching, recording completing, operator office trunk).

Outpulsing—Generation of pulses to match the stored digit information and of the proper type to be used by the distant switching office.

Progress Marks—Areas in TCR which indicate next software routines to be executed for the call.

Signal Digit—Area in the TCR to indicate the location of the digit to be received before base level is alerted for more base level action.

Subroutine—A sequence of instruction which performs a well-defined function and is called by another section of instructions.

TCR—(Transient Call Record)—A 16-word block of writable storage assigned to a call in a transient state.

TMR—(Terminal Memory Record)—A 4-word block of writable memory assigned to each junctor and used for storing call connection information.

TSPS—Traffic Service Position System.

Word—A set of characters which occupies one location in storage and is treated by the system as a unit.

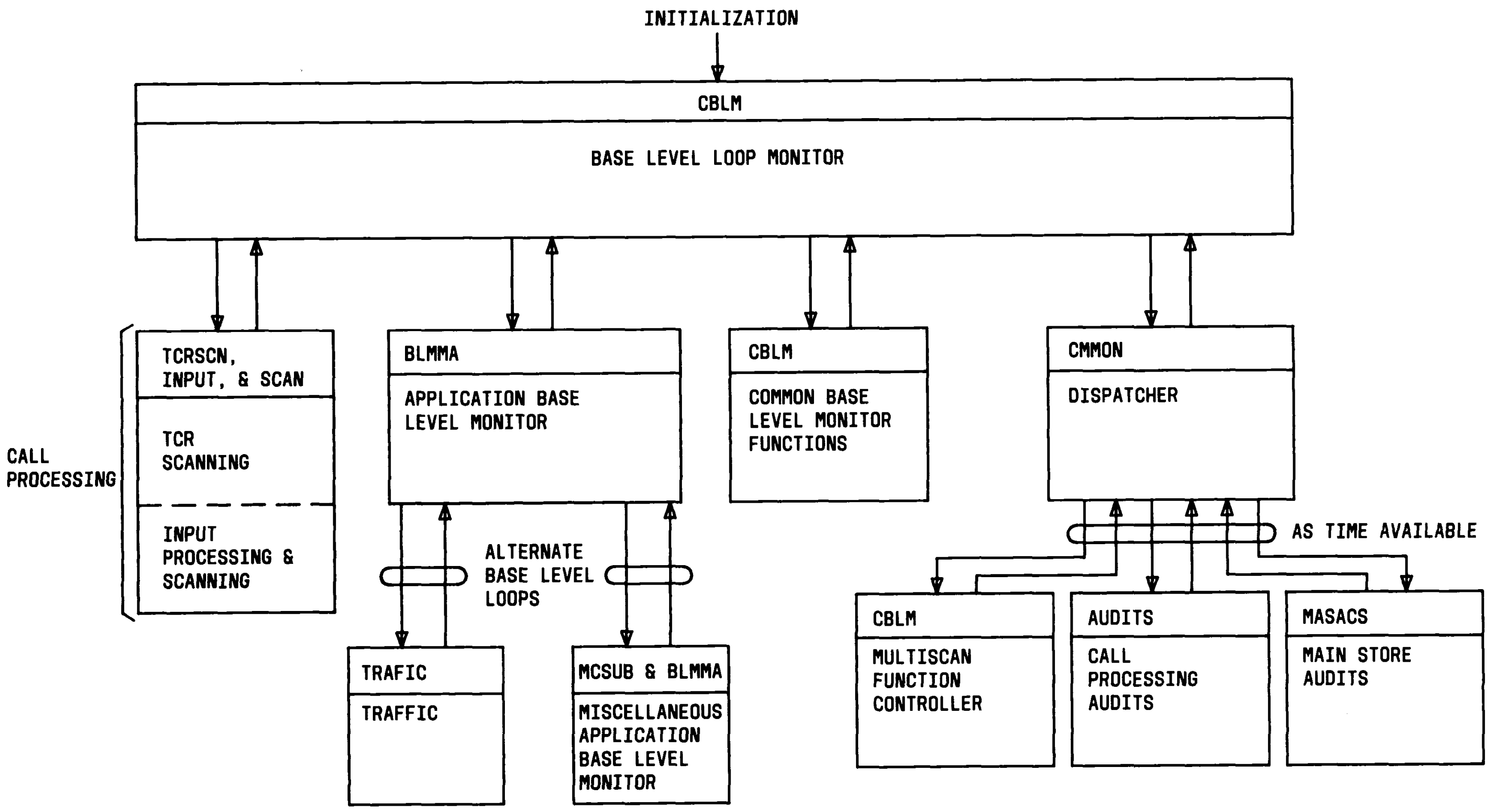

Fig. 1—Base Level Loop

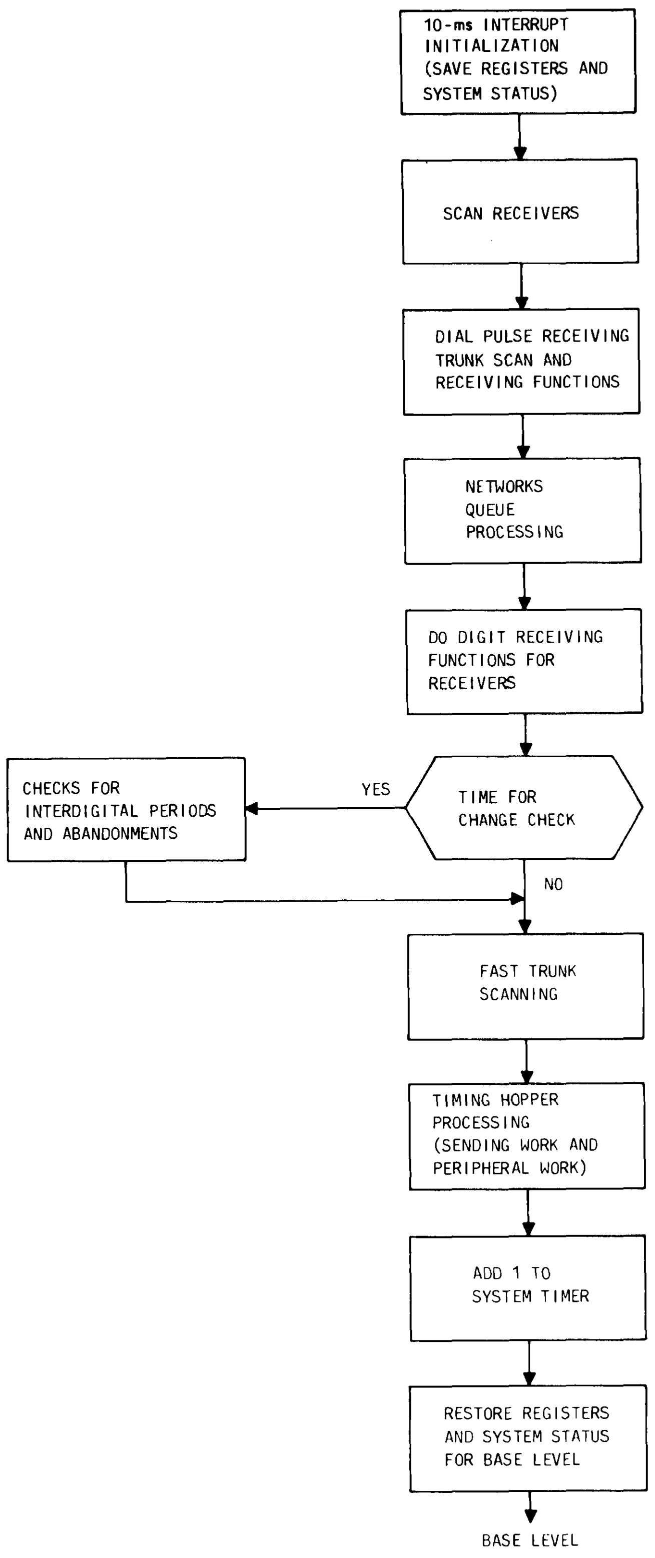
10-ms INTERRUPT INITIALIZATION (SAVE REGISTERS AND SYSTEM STATUS)
SCAN RECEIVERS
DIAL PULSE RECEIVING TRUNK SCAN AND RECEIVING FUNCTIONS
NETWORKS QUEUE PROCESSING
DO DIGIT RECEIVING FUNCTIONS FOR RECEIVERS
TIME FOR CHANGE CHECK
YES
CHECKS FOR INTERDIGITAL PERIODS AND ABANDONMENTS
NO
FAST TRUNK SCANNING
TIMING HOPPER PROCESSING (SENDING WORK AND PERIPHERAL WORK)
ADD 1 TO SYSTEM TIMER
RESTORE REGISTERS AND SYSTEM STATUS FOR BASE LEVEL
BASE LEVEL

Fig. 2—Timed Interrupt

TABLE A

PROGRAM IDENTIFICATION

PROGRAM NAMES	PROGRAM TITLES	PROGRAM NUMBERS
AUDITS	Audit Monitor, Audit Subroutines, and Some Audit Programs	PR-3H002
BLMMA	Application Portion of the Base Level Monitor Program	PR-3H004
CBLM	Common Base Level Monitor	PR-1C950
CINIT	Common Initialization	PR-1C952
CMMON	Common Maintenance Monitor	PR-1C963
CSYSUB	Common System Subroutine	PR-1C956
CTAPH	Common Tape Handler	PR-1C957
CTTYH	Common TTY Handler Program	PR-3H012
CUTIL	Common Utilities	PR-1C962
DIGPRO	10-Millisecond Interrupt Program—Digit Receiving and Sending	PR-3H153
FASTTK	Fast Trunk Scanning Program	PR-3H159
INPUT	Input Monitor Program	PR-3H160
MMONA	Maintenance Monitor Application	PR-3H203
POINT	Peripheral Order Interpreter	PR-3H168
QTMON	Network Queue and Timing Hopper Monitor	PR-3H171
TCRSCN	Base Level TCR Scan	PR-3H174
TRAFIC	Traffic and Plant Measurements	PR-3H008
TVADM	Test Vertical Administration Program	PR-3H178

TABLE B

COMMON SYSTEM SUBROUTINES

SUBROUTINE	TITLE	FUNCTION
BCDXBIN	BCD to Binary Conversion	Converts a 4-digit BCD number to binary
BINXBCD	Binary to BCD Conversion	Converts a binary number to BCD
TENZERO and ZEROTEN	Convert BCD Zero	Examines each digit of a 4-digit BCD number for one of two zero forms (1010 or 0000) and converts to the other form
INTBEGIN and INTBGNX	Interrupt Begin Routines	Stores system status for interrupts (INTBEGIN is for internal interrupts, INTBGNX is for interrupts from external devices)
INTEND	Interrupt End Routine	Restores system status to preinterrupt state and transfers control back to base level
UNLODIOC and LODIOC	Unload and Load I/O Channel State	Unloads or loads the two state registers (IOD and CHC) in an I/O channel
RGCHKADR	Range Check Address	Checks a store address to determine whether it is in an equipped store module and whether it is write protected or not
UNWPST and UNWPOST	Turn Off Write Protect of Store	Turns off write protection for a 4096-word block of main store (in either store)
INITST and INITOST; WPST and WPOST	Initialize MASC Including Write Protect	Initialize all on-line or off-line main store controllers and load write protect registers
STPSTUPD	Begin or Stop Store Update	Enables or disables automatic updating of the other store
UNLODMCH and LODMCH	Unload and Load Maintenance Channel State	Unloads or loads the state registers (MCHB and MCHTR) for a maintenance channel (MCH)
UPD_OTS	Update Off-Line Temporary Store From On-Line	Copies on-line temporary store to the off-line store through the use of clearing tables
GET_OTS	Copy Off-Line Temporary Store to On-Line	Copies off-line temporary store to the on-line store through the use of clearing tables
ZERO_TS	Zero Temporary Store	Zeroes temporary store by using clearing tables
CLR_16 CLR_WRDS WRT_PTRN	Multiple Store Clear or Pattern Write Subroutine	Clears block of 16 memory words Clears block of N memory words Writes a specified pattern into block of N memory words

TABLE B (Contd)

COMMON SYSTEM SUBROUTINES

SUBROUTINE	TITLE	FUNCTION
L2_N	Multiple Load Subroutines	Load registers R2 through RN starting at location specified in RA1
MOVST	Move Block in Store	Transfers block of N words from location specified in RA0 to location specified in RA1
ST2_N	Multiple Store Subroutines	Store registers R2 through RN starting at location specified in RA1
SIO	Send and Test I/O	Used to send I/O orders and/or test for a response (SIO instruction followed by TESTIO)
SMIO		SMIO instruction followed by TESTIO
TESTIO		Looks for response from peripheral device
SENDIO	Send I/O Order and Retry on Failure	Uses normal I/O send orders and retries on MTC response or error
SENDIOS		Uses normal I/O send orders and retries only on error
SENDMIO		Uses MTC send orders and retry on MTC response or error
SENDMIOS		Uses MTC send orders and retry only on error
SLDMCHB	Send and Test MCH	Sends MCH message and tests for response (RO=LDMCHB)
SLDMIRL		Sends MCH message and tests for response (RO=LDMIRL)
TMCH	Test MCH State	Gates MCH status bits for R0 for testing
SBUMCH	Send and Test Backup Maintenance Channel	Sends orders to other 3A CC via the backup MCH (uses other store write facility and other 3A CC interrupt)
INSYNC	Synchronize Base Level With Interrupt	Permits base level program to synchronize with TI (usually timed interrupts). The program can block interrupts while executing without delaying next interrupt.
INTCHK	Determine Currently Active Interrupt	Determines which interrupt is currently active
EXCMCH	Execute a Series of MCH Orders	Executes series of MCH orders specified in a table (only)
EXCMCHO		Executes series of MCH orders specified in a table and an optional first order
EXCOFLPG and INIT_OCC	Initialize Off-Line CC and Possibly Begin Execution of an Off-Line Program	Initialize off-line 3A CC and start it executing at a given address

TABLE B (Contd)

COMMON SYSTEM SUBROUTINES

SUBROUTINE	TITLE	FUNCTION
TIMOUT and TIMOUTX	Sets Up Time-Out Constant	Sets up timing constant (25-ms intervals — 25-ms minimum, 13-minute maximum)
TIMCHK	Time Check	Examines time-out constant set up by TIMOUT to determine when the specified period of time has elapsed and sets condition flop (CF) accordingly
REPT_ERR	Print Report Error TTY Message	Prints a TTY message: REPT ERR WRD A B C D E F G