Scientific Atlanta Security Monitoring System

Version A060 and A061 Description of

Operation

The SMS software consists of about 400k bytes of source code in 26 modules. After assembly and linkage it produces just under 16k of object code. The data and stack requirements are 16k bytes with an additional 4k of EEPROM data area for the subscriber bitmap and power level array. The source code is fairly well documented and should be considered the ultimate source of information when conflicting operational details are encountered.

The following pages represent a module by module description of the system. More detail will be provided in those areas that are particularly system intensive.

Module Name: ERGOCRT
Size: 11,136

This module contains most of the CRT specific interface and control routines. It is basically a series of subroutines that generate the necessary escape sequences to perform the following functions:

1) read current cursor position
2) move cursor to a given position (x,y)
3) home the cursor
4) home down the cursor
5) move cursor up one line
6) clear screen from cursor position to end of screen
6) clear screen from cursor position to end of line
7) insert a line at cursor position
8) delete a line a cursor position
8) enter and exit protect mode
9) enable and disable the keyboard
10) turn on and off video display enhacements
11) turn on (initialize) the function keys
12) write to the 25th display/status line
13) initialize the terminal
14) print current screen to printer port
15) print a line at cursor position
16) print a line of text to the printer
17) print the time and date to the printer

Module Name: MON
Size: 31,488

This module contains most of the routines that handle the SMS commands and error messages. It is called from the main menu screen (see MAIN) so it is itself a background routine. After writing the monitor status screen to the crt and setting up the function key menu and subroutine calls, it calls the routine "CMDLN" (located in CRT) that displays the command prompt and processes any command line entered by an operator (CMDLN calls ITEXT and it, in turn, calls ICHAR).If no commands have been entered, the routine waits in a loop until something happens.

Assuming a valid command has been entered, a call is executed to the address pertaining to the given command. All routines must return with the zero flag set, or MON will return to the main menu screen.

Listed below are the commands and their associated subroutines as found in MON module:

COMMAND

EXAMINE: handled by subroutine EXAM

This routine first writes a new function key menu for its own operation. It then checks to see if an address was given after the command. If so, it will begin the display at this address, otherwise it will start at address 0. At this point it simply checks the function keys 1 through 4 for further commands, which allow an operator to examine, enable or disable addresses, and exit the command.

LIST: handled by subroutine LISTO

This routine causes a sequential listing of enabled addresses to be sent to the printer. If any digit from 0 to 9 is given as an argument to the list command, the no response bit map will be listed instead of the data base bitmap. The routine simply sets the most significant bit of the LOG (log alarms) flag which serves as a signal to the task CHECKS (in CHECKS) to begin listing addresses from a bit map to the printer. The base of either the data base bitmap or the no response map is loaded into location LSTMAP. CHECKS lists this map, from lowest address to highest address, to the printer.

ACKNOWLEDGE: handled by routine ACK

This command allows operators to clear a pending alarm from the que and send the acknowledgement to the transponder. The routine has two entry points, one by way of the F8 function key, and the other via the ACK command. If the function key is used, the cursor position is first saved then restored after the call to the entry point at ACK2. Upon entery at ACK2, the crt acknowledge que (ACKCQ) is checked, and the oldest entry is removed and processed. The alarm acknowledge command is sent to the address in the entry (except for power level alarms) then the alarm count is decremented and the oldest alarm on the screen is deleted.

For the case where entry is via the ACK command with no transponder address as an argument, the sequence is the same as that just described (the cursor address is not saved and restored though). If an address is given as an argument, then the alarm acknowledge is sent to this address and nothing is removed from the ACKCQ, and nothing is changed on the screen. If additional addresses are given on the command line, they will be processed in succession.

ADD/DEL: handled by subroutines ADDR and DELETE

These commands add or delete addresses from the data base bitmap. Basically both commands share the same body of code. The difference being that either the add or delete flag (AORD) is set before the routine begins. Whether the bit is enabled or disabled is is determined in the subroutine ADORDL by examining the AORD flag. A checksum is calculated after each add or delete (CKAER) and stored at loactions PAER1-4. This is to insure database integrity.

CMC: handled by routine CMCOMM

This command simply toggles the most significant bit of the SMSFLG. This indicates to interrupt and background routines that alarms should be sent to the cmc and that communications should be periodically established between the sms and the cmc.

RETURN: handled by RETN

This routine allows for exiting from the sms monitor screen and returning to the main menu screen. It first checks the MONFLG (bits 0-4) for pending alarms. If none are pending, it checks the scanner status flags (STATSC) to see if the scan is on. If so, the most significant bit of the SMSFLG is checked to see if the cmc is enabled before allowing an exit. If any alarms are pending, or if scan is on and cmc disabled, exit is disallowed. Upon successful exit, the MONFLG is cleared, the crt alarm (CRTAQ) and status queus (CRTSQ) are flushed, and control is returned to MAIN.

RESPONSE: handled by subroutine RESPN

This command performs two functions, depending upon whether or not an argument was given with the command. If no argument was supplied, then the no response flag (bit 5 of MONFLG) is toggled. If an integer argument of between 0 - 255 is given, then this value is stored into the global variable NRTIME and will become the new no response delay time.

SWAP: handled by subroutine RFSWAP

This command causes a swap from the interrupt to the polling channel (or vice versa). This is accomplished simply by setting or resetting the global CHANBF, which various background and interrupt routines use to determine which channel is the polling channel.

SCAN: handled by subroutine SCAN

This command is another toggle. In this case the STATSC flag is checked for zero. If it is zero it means the scan was off and that it should be turned on. Before turning it on, the no-response bitmap is cleared and the alarm count (bits 0 - 4 of MONFLG) are cleared, and the alarm screen is cleared of any alarms. If the scan was on, then the STATSC flag is simply cleared.

POWER: handled by the routine PWR

This command is another dual function command. Its mode of operation depends upon the setting of the AUTOFG, which is checked upon entry. If it is zero, then at least one address is expected on the command line. This address is loaded into the RF level adjustment que (RFLAJQ) to force an auto power level adjustment procedure to occur on the given address. Additional addresses supplied on the command line are loaded into the RFLAJQ one after the other, as long as there is room. If the AUTOFG is non-zero, it is set to the value 0fh as a signal to the SEND routine, at which point a jump is made to that routine to complete the processing of the command. (see SEND)

READ: handled by subroutine RD

This is another command that requires an address for an argument. The address supplied, along with a read status command request is loaded into the crt acknowledge que (ACKQ) and subsequently sent out to the transponder. The resultant information is formated and displayed on the crt.

RESET: handled by subroutine RSTT

This command is one of three commands that use the common subroutine BRUTE to read the required address from the command line and load it into the ACKQ. The actual command, in this case a reset command, is loaded into the BC registers before calling BRUTE. Upon receiving this command, the transponder will execute a tight loop which will allow the watch-dog circuitry to timout and trigger the reset line on the device. This command will not be executed by a transponder if it has any alarms latched into its internal alarm register.

TEST: handled by subroutine TEST

The test command is exactly like the RESET command in that it simply loads the BC registers with the command code that is to be sent to the transponder, then jumps to routine BRUTE. BRUTE looks at the command line for the address(s) and loads it into the ACKQ then puts the command into the DATAQ for transmission by the interrupt routine INTO.

DIAL: handled by subroutine DIAL

Like RESET and TEST, DIAL loads the command code that (in this case, the code to cause the dialer trigger to toggle) is to be sent to the transponder into the BC registers. A jump is then performed to the routine BRUTE, which gets the address(s) from the command line and loads it into ACKQ. It then takes the command in BC and stuffs it into the DATAQ.

SEND: handled by subroutine SEND

The SEND command allows any given 16 bit data pattern (supplied in decimal form) to be sent to any particular address. It reads the command line first for the transponder address, then the data to be sent. After the data is obtained, a check of the AUTOFG is made. If it shows a signal from the POWER command (see POWER) then the data is checked for a valid power level value of between 0 - 16. This value is then positioned properly into the data word, then stuffed into the DATAQ. If the AUTOFG does not indicate that this is a power adjust operation, then the data (as obtained from the command line) is stuffed into the DATAQ.

In the case of the power level command processing, the value sent to the transponder is also stored into a four bit field in EEPROM. This four bit field is pointed to by the address of the transponder given on the command line. This is done so that the global power adjust background routine can send this unique value to each device upon being invoked by the GLOBAL command. (see HELP - GLOBAL).

See the description of the HELP module for additional commands and their associated subroutines.

Module Name: NORCHK
Size: 3,456

This module basically contains the single subroutine NORCHK. It is called from various background tasks so that it gets a chance to process no responses almost continually, if any exist. What it does upon entry, is check the no response que (TIMEQ + NORSPQ) for any pending no responses. If there are any in the que, then the time that the oldest was entered is checked against the current time. If it has been in the que longer than the time specified in the global NRTIME, it is removed from the que. At this point, the MONFLG is checked (specifically bit 5) to see if no response reporting is on or off. If off, then the image is updated to reflect the no response status, and the routine loops back to check the TIMEQ again. If no response reporting is on, then a no response alarm for the given address is loaded into the alarm que (ALARMQ).

This routine sets a counter to 10 upon entry, and decrements it each time through the loop. When the counter (NRTODO) reches zero the routine will exit. It will also exit without doing anything if there is nothing in the TIMEQ - meaning that there are no no responses to be processed.

Module Name: Power
Size: 6,844

This routine is a background task that only runs under the following conditions:

1) Once on startup.
2) When commanded to by the cmc.
3) When a GLOBAL command is issued from the sms.

All it does is cycle through the AER bitmap (data base) and load any enabled addressees found into the RF level adjustment que (RFLAJQ). This will cause an adjustment to be performed on that address by the background task PWRCAL. Whether this adjustment will be an automatic adjustment or a manual adjustment, depends upon the setting of the AUTOFG. (see AUTO command and COMMON).

This module is fairly well documented in the source listing and is generally self explanatory. Therefore no further duscussion will be persued here.

Module Name: QMAN
Size: 11,782

This module contains all of the subroutines that are required for manipulating the various queus used throughout the sms. All of the queus are FIFO, and have a maximum depth of 255 entries. Each entry is a 16 bit entity with byte 0 holding the upper, or most significant byte, and byte 1 holding the least significant byte of the transaction. In the case where the entry contains a transponder address, the most significant four bits (bits 4-7) of byte zero contain a four bit tag. This tag identifies the type or origin of the transaction. See TASKS for a listing of the tag bits and their meanings.

This module is well documented and does not require further explanation here. Further detail may be obtained directly from the source code listing. Below is a list of the subroutines contained within QMAN:

OPENQ: Initializes a que to zero entries.
CLOSEQ: De-activates a que, so that it appears to be full.
FREEZQ: Stops all que activity. No transactions may go in or out.
STARTQ: Re-activitaes a que which is frozen.
CHECKQ: Returns the number of transactions currently in a que.
ENTERQ: Places a transaction into a que. Returns space remaining.
LEAVEQ: Retreives a que entry. Also returns space remaining.
VIEWQ: Returns with a copy of the next available transaction.
FINDXQ: Finds a que entry and removes it.
PURIFY: Removes the nth entry of a que.

Module Name: RTC
Size: 9,216

This module contains four routines that deal with the real time clock interrupt (Multibus INT7, generated by the 8253 timer chip located on the CPU board). The first, named (appropritely) INT7, is the actual interrupt service routine for the timer interrupt, which is generated every 50 ms. The time of day clock global SECOND is incremented every 50 interrupts, at which time the INCTIM routine is called to update the time of day clock. The time is updated to the crt every 8 seconds.

INT7 also calls the routine TASK on every interrupt. This is the task manager which rotates through the list of background tasks and schedules them in "round-robbin" fasion. Due to the frequency of the interrupt, and the asyncronous nature of the system, each task is limited to a 25 ms. time slice. The background routine SCRTY is started every other interrupt, which results in it having the greatest allocation of CPU time.

The module also contains the routine CKINIT which is called on startup, and initializes the 8253 timer 0 for continuous operation.

Below is the list of background tasks and the interrupt cycle they are allocated:

TASK	CYCLE
SCRTY	1
QUSORT	2
SCRTY	3
PWRCAL	4
SCRTY	5
CMC	6
SCRTY	7
UPSTAT	8
SCRTY	9
POWER	10
SCRTY	11
FUNPRC	12
SCRTY	13
CRTPRC	14
SCRTY	15
CHECKS	16

Module Name: SCRTY
Size: 13,312

This module contains the dominant SCRTY background task, which runs every 100 ms. It basically monitors the queues and flags associated with the transponder scanning operation. Upon entry it checks the scanner status byte (STATSC) to see if the scan is on or off. If the scan is off, it simply exits with no action taken, otherwise it proceeds to check the STATSC flag for the initialization signal. If initialization is required (ie. on startup, or after a scan-off scan-on operation) the queues are cleared along with image ram and the temporary voting block. It then increments the global SCNCNT which is always reset to zero by the transmit routine. If this byte overflows, then SCRTY assumes that the scan has stopped for some reason and signals an E004 condition.

Next, the alarm and repoll queus are checked to see if they are full (or closed). If so, they are further checked for space remaining. If either has space remaining they are restarted. If both queus were restarted, then the INT0 routine is also restarted.

Following this, the poll que (POLLQ) is checked for space remaining. If it is closed or frozen but has space remaining it is restarted. If it contains zero transactions it is stoked up with valid addresses from the AER table. The subroutine that performs this operation (FETCH) checks the global flag PECK to see whether a PEEK command is in operation. A value of 3 or greater indicates that a PEEK command has been executed and the POLLQ is stoked with the address contained in the memory location PEEKAD. This causes continuous polling to a single address for signal measurement.

This module also contains two subroutines that are called from other modules. FILTER is a routine that checks to see if a no response transaction is a result of a valid poll from the AER table, or whether it was a result of noise. If the address is not valid, the alarm is discarded, otherwise it is processed as an ordinary alarm. The routine MAPMSK takes a transponder address and uses it to form an index into the AER (bitmap). The routine returns both the index address, and a mask with a bit set corresponding to the given address.

Module Name: SERIAL
Size: 24,832

This module contains all the serial I/O routines needed to support the interrupt driven CMC and CRT interfaces. All I/O is fully buffered for efficient interfacing with the rest of the system. The module is particularly well documented, and therefore additional detail will no be given here. Below is a listing of the subroutines contained in this module and a brief description of their operation.

INT3: This is the crt transmitter interrupt service routine. It handles the transmission of characters to the crt from the ring buffer allocated for output characters (TXCRTB). When the buffer is empty, it shuts off its on interrupt.

INT4: This is the crt receiver interrupt service routine. It services the interrupt generated by the receiver side of the 8251 USART mounted on the piggy-back module on the CPU card. It buffers incomming characters, checks for XON/XOFF pacing and calls a routine called FUNKEY that checks for function key characters so that they can be processed separately.

INT5: This interrupt service routine handles the transmission of characters from the cmc transmit buffer (TXCMCB) out to the cmc port. It will shut off its on interrupt when the buffer becomes empty, and signals the communications handler via the sms flags (SMSFLG) that the process is complete.

CHKSPC: This routine simply checks and returns the remaining space left in the crt transmitter buffer.

CHECK: This routine checks for available space in any of the other I/O ring buffers.

ICHAR: This routine reads a single character from the crt input buffer and returns the character to the caller. If a character is not available in the buffer, the routine waits in a loop until one is entered from the keyboard.

OCHAR: This routine puts a character into the output buffer for terminal transmission. If the output routine is shut down it will prompt (boot) it to start. If there is no room in the output buffer, it will wait in a loop until space becomes available.

OUTPUT: This routine loads a character into the output buffer for transmission to the cmc. Like OCHAR, it prompts the output routine (in this case INT5) if it is not running. It will also wait if the output buffer is full.

STRING: This routine transmits a string of characters to the crt until a null is found. I calls OCHAR.

AUTOBD: This routine initializes the baud rate clock for the crt USART. It checks for two successive escape characters to indicate the baud rate is correct. It switches between 1200 and 4800 baud until the above condition is met.

SLEEP: This routine re-initializes the baud rate clock to 4800 baud and suspends operation of the crt. Typing two or more escape characters will re-awaken the crt monitor functions.

INITIO: This routine is called on system start up. It initializes both the cmc port and the crt port USARTs for proper asyncronous operation. It then initilaizes the ring buffers.

PACE: This is the terminal pacing routine. It is called by ICHAR and checks for the ^Q ^S (XON/XOFF) charcaters from the terminal. Upon receiving the XOFF character, it shuts down the crt transmitter. Upon receiving an XON character the transmitter will be restarted.

Module Nmae: TASKS
Size: 28,032

This module contains three of the background routines listed in RTC. In addition, it contains three subroutines that decode and display alarm and status data on the crt of the sms. One final routine, CLRFKY, is used to clear any function key service routines and flags so that no service occurs. Listed below are routines contained in this module along with an operational description.

QSORT: This is a background task that sorts the alarms passed from scanner routines via the ALARMQ into one of the following output queus:

1) CMC communications que (CMCQ)
2) CRT status que (CRTSQ)
3) CRT alarm que (CRTAQ)

If a que is full, then the alarm is re-entered into the ALARMQ. If either the crt or the cmc is not active, the transaction is trashed rather than loaded into the appropriate que. The following tags (in the upper four bits of the transaction address) are used to direct the disposition of transactions as they are pulled out of the ALARMQ:

TAG	DESCRIPTION
0000	New alarm from scanner (see ALMCHK)
0001	Reque of alarm destined for the CMCQ
0010	Reque of alarm destined for the CRTAQ
0011	Status for the CMCQ
0101	Read status register status for CMCQ
0110	Read status register status for CRTSQ
0111	Status for the CRTSQ
1000	New power level alarm
1001	Reque of power level alarm destined for the CMCQ
1010	Reque of power level alarm destined for the CRTAQ
1011	No response reporting shut down alarm
1100	Reque of no response alarm for CMCQ
1101	Reque of no response alarm for CRTAQ

The remaining two tag values are spares, and are not used.

CRTPRC: This routine is another background task that simply calls two subroutines: CRTALM and CRTSTA. Both routines check their respective queus for pending alarms or status (CRTAQ + CRTSQ). If an alarm is present it is removed and the indicated alarm or status information is formated for display to the crt.

The actual decoding and formating of alarm information is done by the subroutine ALRMTX. If bit 6 of the global MONFLG is set, then the alarm information is displayed in binary format. This may be useful for debugging purposes.

FUNPRC: The function key processor background task checks for a function key flag that has been set (previously by the INT4 receiver interrupt service routine - see SERIAL). If it finds a flag for a given function key set, it looks for a non-zero address to call to process that particular key. This allows function keys F5, F6, F7 and F8 to operate as interrupt driven tasks, and F1 through F4 to be used as application keys for the operator interface.

CLRFKY: This subroutine is called on startup, and zeros all function key flags and service routine addresses.

Module Name: UPSTAT
Size: 11,264

UPSTAT is another background task scheduled by the real time clock task manager. It has the job of monitoring and decoding various system status flags and updating the status fields on the crt. It first checks MONFLG to see if the monitor is active. If so, it calls CHKSPC (see SERIAL) to insure adequate room is available in the crt output buffer for all the status information and enhancement escape codes that will be required. Given that the monitor is active and there is room in the buffer, the following flags are checked against the values stored when they were last checked:

1) STATSC - 0 means scan is on, non-zero means off or error
2) SMSFLG - bit 7=0 means cmc is on, else cmc is off
3) CHANBF - 0 means polling channel, else interrupt channel
4) MONFLG - bits 0 - 4 contain number of alarms on screen
5) MONFLG - bit 5=0 means no response reporting is on, else off
6) NRSCNT - display # of no responses if any (if bit 5 MONFLG=0)
7) STUCNT - display # of transponders currently being scanned

This module also contains a routine called INSTAT, that is called upon entry into the monitor screen from the main menu screen. It initializes the status display with reverse video and default values for the various fields.

Module Name: INT2
Size: 8,448

This is the interrupt service routine that handles any interrupts from the 64k RAM/EEPROM board. The ready signal from all of the EEPROMS are wire-ored to produce one interrupt signal. The SAVE and VERIFY commands initiate operations that require interrupt processing by INT2. The global SAVFLG indicates whether the current operation is part of a SAVE command (non-zero, and value 1-5) or whether it is due to the execution of a VERIFY command (value 128-129). This flag is checked when the routine is first entered, right after the interrupt latch is reset by performing an I/O write to port 50h.

If the SAVFLG indicates that the interrupt is the result of a SAVE operation, the global SAVCNT is decremented and checked for a zero value. A zero value indicates that the end of one of each of the four AER maps has been reached. The value of the SAVFLG determines which current AER map: when it hits 4, and the SAVCNT reaches 0, programming is done. The four checksum bytes are then loaded one at a time. After the last checksum byte has been programmed, the checksum is recalculated for the map in EEPROM. The value thus obtained is compared to the values programmed in. If they match, a successful save operation has been performed and the message is displayed on the crt screen. If the checksums do not match, an error message is displayed, along with an error code.

The other possible reason INT2 would be triggered is during a VERIFY command from the main menu. The VERIFY command sets the SAVFLG equal to 80h then writes the compliment of the first byte of U4 to trigger the start of the EEPROM test. Upon entry (with SAVFLG bit 7=1), a branch is made to the routine UTTER which basically handles the EEPROM test. The current byte being tested is stored in the global SAVCNT. This is compared to what was read and if they match, the value is complimented and written back. This process repeats until every byte has been written/read and re-written again. This allows for a non destructive go-nogo test of the EEPROM. See HELP for more information on the VERIFY command.

Module Name: CHECKS
Size: 6,912

CHECKS is another background task that performs two basic operations:

1) prints a listing of enabled addresses or no responses to the printer.
2) Updates the BIN, LOG and AUTO flags to the crt status line.

Upon entry, it checks the LOG flag. If bit 7=1, then a LIST command is in operation and it proceeds to build a one line buffer of addresses to list to the printer. This is done by searching through the bitmap, pointed to by location LSTMAP, for up to 100 addresses, or until a one line buffer to be printed is completed. It then exits, and picks up where it left off on the next go around to print the line. This process continues until the entire bitmap has been searched, and the set bits evaluated and listed.

If bit 7 of LOG=0, then nothing is going on with the LIST command so a branch to the routine STATUS occurs. This routine performs the same activities as UPSTAT. That is, it examines certain flags and interprets the results to the status fields on the crt. Specifically, it checks the following flags:

1) LOG = 0 means Logging is off, else on.
2) MONFLG - bit 6 = 1 means binary status reporting on, else off
3) AUTOFG = 0 means auto power level mode on, else off.

As done in the routine UPSTAT, current values are compared against previous values as stored in a last-check buffer. If any change has occurred, the field is updated. Otherwise no data is output to the crt.

Module Name: HELP
Size: 12,672

This module contains some additional sms monitor commands, and is genericaly part of the module MON. It was created when additional commands were added to the scanner, and rather than make MON any larger than it is, it was decided to put the additional commands in a separate module. See MON for background.

Listed below are the commands and their associated subroutines as found in module HELP:

COMMAND

HELP: handled by subroutine HELP

The HELP command generates a menu of available scanner commands on the alarm screen of the crt. It requires most of the alarm screen to display the menu, so if more than 3 alarms are pending, the menu will not be displayed.

SAVE: handled by subroutine SAVE

This command causes the current contents of the AER maps (database bitmap) to be written to a non-volatile save area in EEPROM. The actual save programming operation is performed by INT2 (see INT2) but it is initiated by the SAVE command. Basically what occurs here is as follows:

1) A warning message is first displayed to request an affirmative response before beginning the save operation. This is because once the operation has begun, it will over-write any existing data.

2) The interrupt mask is updated to enable INT2 interrupts.

3) The save flag is set up to signal INT2 that a save is in progress.

4) The first byte from AER1 is read and stored into the EEPROM save area.

When the byte has been programmed, the EEPROM chip will trigger its ready line which will cause the INT2 to occur. Then the interrupt routine will read and write the next byte and so on. At this time, the SAVE routine itself is through. INT2 will report the success or failure of the save operation.

LOAD: handled by subroutine LOAD

This routine will load a bitmap in from the EEPROM save area to the AER map in RAM. It is the compliment of the SAVE command. Before loading the database however, it will run a checksum on the EEPROM table to be sure of the validity of the data. If this is successful, the AER maps and parity bytes will be loaded from the image in EEPROM. A message will be displayed as to the results of the operation.

VERIFY: handled by subroutine VERIFY

This routine is available from the main menu screen only. It allows the user to perform a simple, non-destructive memory test on the RAM (at C000h) and EEPROM (at 6000h + 8000h). While it is not a definitive memory test, it will find bad chips, chips with bent pins, or faulty address or data buss drivers on board. It will call out failing memory by U number, as per the schematic. After running a RAM test, it sets up the flags for INT2 and enables the interrupt via the interrupt mask. It then reads the first byte from EEPROM, stores it in the SAVCNT, then writes a complimented version back the the EEPROM. At this time INT2 will complete the testing while VERIFY waits by repetitively calling DELAY (see LIB). Upon each return from DELAY, VERIFY will check the value of the "read" pointer. If it has not changed for two consecutive cycles, then it assumes that the operation is complete. It then moves down to check the value of the SAVFLG. A zero value here indicates that the test terminated normally and that INT2 reported the results. Otherwise, it is assumed that there is a failure in one or more of the EEPROM chips so, based on the current address pointer, an error message is displayed, calling out the faulty EEPROM.

AUTO: handled by subroutine AUTO

This command toggles the AUTOFG on and off. This flag is checked during execution of the POWER command (see MON) to indicate what method of power adjustment is to be used, and how the command itself will execute. It is also checked in the subroutines PWRCAL and PWRCHK (see COMMON) to indicate whether or not the auto power adjust mechanism should be used. If the the flag is zero then auto mode is on, if it is non-zero, manual mode is on.

PEEK: handled by subroutine PEEK

The PEEK command causes a single transponder to be polled exclusively for about five seconds so that its return signal can be measured. When executed, it loads the address given as an argument into the global PEEKAD. It then stores a non-zero value

(greater than 3) into the global PECK, after which it calls DELAY for 5 seconds. During the delay, the PECK flag is examined in the subroutine FETCH in the SCRTY background routine (see SCRTY). If the flag is greater than 3, the address at PEEKAD is loaded repeatedly into the POLLQ. After the return from DELAY, the PECK flag is set to 1 and another 2 second delay is entered which allows the addressee to drain fully from the POLLQ. The PECK flag is then reset to zero and the routine returns.

GLOBAL: handled by routine SWEEP

The GLOBAL command simply sends a signal to the background task POWER via the global PWRADJ, which causes it to start loading enabled addresses into the RFLAJQ for power adjustements. The AUTOFG flag is checked first to determine whether auto or manual mode is in operation. Currently the routine will not initiate a global power level adjustment process if in auto mode.

Module Name: INTVEC
Size: 1,697

This is an absolute module which contains interrupt vectors starting at location 40h, and also provides the jump instruction to the entry point in MAIN after system reset.

The eight jump vectors, corresponding to Multibus interrupts 0 - 8, are lisred below:

INT0 - Polling channel transmit and receive interrupt. Generated by 8253 timer located on the scanner interface board.

INT1 - Interrupt channel receiver interrupt from interface board.

INT2 - Program ready interrupt fro 64k RAM EEPROM board.

INT3 - Crt receiver USART interrupt.

INT4 - Crt transmitter interrupt.

INT5 - Cmc receiver USART interrupt.

INT6 - Cmc transmitter interrupt.

INT7 - Real time clock (task manager) interrupt.

Module Name: GLOBAL
Size: 8,704

This module contains most of the globally declared variables and data areas (Queus), although several globals are defined in other modules. The storage allocation for all of the queus and tables are located in GLOBAL. The module is fairly well self explanatory.

Module Name: DBINIT
Size: 4,224

This module contains subroutines that are used to clear and initialize the various tables and data areas. These routines are called from various modules at various times, although they are primarily activated on system initialization. The following lists the subroutines within this module:

FILMEM: This subroutine is used by other routines to initialize a given block of memory to some known value, usually zero.

DBINIT: This is the main routine that initializes most of the system flags and tables.

SCNCLR: This routine clears the image ram and temporary voting bloack array to zero's (no alarms pending).

ZNRMAP: This routine clears the no response bit map. It is called after executing a scan-off/scan-on sequence.

Module Name: MAIN
Size: 11,392

MAIN is the first module entered following a system power-up or reset. It contains the main initialization routines in addition to the master background routine (MAIN) and the main menu screen commands. The following describes each subroutine contained within this module:

INIT:

This is the entry point at power up or reset. It is basically a series of calls to other subroutines that result in the initialization of all system functions. Below is a list of activites performed by INIT, with the actual initialization routines or pertinent globals in parenthesis:

1) Stack pointer
2) Real time clock chip (CKINIT)
3) The serial cmc ports (INITIO)
4) The parallel ports 0 an 1 (INITPT)
5) The interrupt registers (8259) (INITIR)
6) CMC communications (INITC)
7) The function keys - flags and calls (CLRFKY)
8) The system error flag (ERRFLG)
9) The crt terminal functions (INITTM)
10) The databases and queues (DBINIT)
11) The crt serial port (SLEEP)
12) Output the power-up message (CPWRUP)

INITPT:

This short routine initializes the parallel I/O ports on the CPU card.

INITIR:

This subroutine initializes the interrupt registers of the 8259 chip, and initializes the interrupt mask (INTMSK).

INITTM:

This routine sets all the flags associated with terminal operations.

MAIN:

This is the master background routine. Upon entry it checks the crt status flag (CRT) for active status. If not active it simply loops back and continues to check for crt activity. When the crt becomes active (i.e. two or more escapes are typed) The terminal is initialized and the greeting message is displayed on the crt for a few seconds, and listed on the printer if it is enabled. The main menu screen is then painted on the crt along

with the function key menu. Finally, a call to CMDLN is made where it will wait for input from the keyboard.

GREET:

This subroutine is called from MAIN for the sole purpose of outputing a greeting message announcing the name of the system and who built the thing.

MAINFK:

This is another subroutine called upon entry into MAIN that initializes the function key menu on the bottom of the screen, and sets up the addressees for function key commands.

TOD:

This routine handles the TIME command. It reads command line input and sets the time clock accordingly.

DOY:

This subroutine handles the DATE command. It reads command line input and sets the date for internal software clock.

SLP:

This subroutine services the SLEEP command. It terminates terminal operations and clears the screen. After the sleep command has been executed, two or more escapes will be required to re-activate the terminal.

Module Name: LIB
Size: 16,768

This module contains a library of general purpose subroutines used throughout the scanner. The module is very well documented and the subroutines are rather generic and straight forward. Therefore no amount of time will be devoted to a lengthy description of the module. The source code listing is more than adequate as a reference. Below is a listing and brief description of the subroutines contained in LIB:

BINBCD: Binary to BCD conversion.
BCDDGT: BCD digit generation routine.
BINDEC: Binary to ASCII decimal conversion.
BINHEX: Binary to ASCII Hexadecimal conversion.
BINBIN: Binar to ASCII binary conversion.
DECBIN: ASCII decimal string to binary conversion.
HEXBIN: ASCII hexadecimal string to binary conversion.
INHEX: ASCII Hex characters (2) to binary byte conversion.
BITADR: Create bit address and bit nmask for a bitmap.
BITSET: Set a bit in a bitmap.
BITCLR: Clear a bit in a bitmap.
BITTOG: Toggle a bit in a bitmap.
BITTST: Test bit in a bit map (Z flag set)
DELAY: 1 to 128 second time delay subroutine.

Module Name: INT0
Size: 15,488

This is the interrupt handling routine that syncronizes the scanning of the transponders. It processes interrupts generated by the 8253 programmable timer located on the scanner interface board. The same interrupt signals a need for a transmit or a receive - i.e. every other interrupt will be a receive.

Upon entry, the interrupt status flag is checked (STAT0) to determine whether the interrupt signifies a receive or a transmit. If bit 3 is set, it indicates transmit so a jump is made to XMIT, otherwise a receive is assumed and processing continues. Assuming a receive, the next thing is to decrement the global that determines the number of receives due (RXPEND). Then the address of the transponder that is being read, is retrieved from the transponder receive transaction queue (STURXQ). Next, the message received bit (bit 4, port EAh) is checked to determine if the expected message was received. If so, the [alarm] data is read in. A call to the subroutine ALMCHK (see ALMCHK) is then made to check the data for any pending alarm condition and process accordingly. Following that, a call to the routine PWRCHK (see COMMON) is made to check the power level.

At this point the 8253 is reloaded with the value to be down counted until the next interrupt, then the routine exits.

In the case of a transmit, the following would take place:

The ACKQ, REPOLQ and the POLLQ are checked (in that order) to obtain the next transaction (address) to be sent out. Failing to find anything to send, the routine turns off the RX and TX pending flags and exits. Given that a transaction is found, The TX pending (TXPEND) flag is set, and the device address is clocked out to the interface board to be sent. Next, the tag bits of the transaction are examined to determine what sort of command should be sent out (normal poll request, power level, test, etc.). In the case of a power level (AUTO mode) adjust command, the value to be sent exists in a four bit field in the image RAM for the particular address. This value is fetched and sent out in the command word. Finally, the command is toggled out to the interface board and the STURXQ is loaded with the address of the device just processed. The CHANBF flag is then checked to see which channel is being used for polling. In the normal case where the P channel is being used, The RXPEND flag would now be incremented and the 8253 clock on the interface board is loaded with the down count for the next interrupt. If the I channel was being used, the 8253 is set and the RCVEXP bit in image is set to let the channel I routine (INT1) know that the address has been polled and a response is expected (see INT1).

Module Name: INT1
Size: 6,400

This is the I channel receive interrupt handling routine. It basically has two modes of operation. The first is the normal case where polling is done on the P channel. In this case, only transponders that have an alarm pending will transmit on the interrupt channel. The reception of such a message will cause an interrupt on INT1. The address of the interrupting device will be read and loaded into the REPOLLQ so that the P channel will be able to process the data as if the alarm were detected on the P channel. The [alarm] data in this case is discarded in favor of processing on the P channel.

If the I channel is being used as the polling channel, the RCVEXP bit in image RAM is checked. If it is not set (meaning this receive was not expected) then the transmission is considered to be noise. Otherwise, the address is compared to the first entry in the STURXQ for a match. A match indicates the correct response and the routine ALMCHK is called to process any alarms that might be indicated. Finally, the power level is checked via a call to PWRCHK, then the routine exits.

Module Name: ALMCHK
Size: 14,976

This is the main alarm checking and processing subroutine. It is called from INT0 and INT1 (when polling on I channel). The routine basically checks for the existance of an alarm condition on the input data. This is determined by comparing the data just received from the device, with the alarm image stored in system RAM. If there is any difference, it is considered an alarm. If no alarm condition is indicated (received data and image RAM match) the routine simply exits. Assuming there is a miscompare (alarm condition), the data is filtered for the existance of all alarms. If all of the alarm bits are set, it is considered a bogus alarm. Next, the process byte of the image RAM is checked to see if it indicates that a verification repoll sequence is active. If so, a branch to the verification code at label NXTONE is performed where the 2 out of 3 repoll sequence will proceede. Otherwise, this is a brand new alarm, and space is allocated in the temporary voting block for the verification repoll sequence. The image RAM process byte is updated to show that verification is in progress, the alarm data is stored in the temporary voting block, and the address is loaded into the REPOLQ to be re polled.

In the case where the verification repoll sequence is in progress, the status byte also indicates the repoll count for the verification sequence. Less than three causes another repoll to be queued, otherwise this is the last repoll therefore the results of the 3 polls must be tallied to find two that agree. In the case where none of the repoll data agree, the situation is called invalid due to noise and processed as a no response. In the typical case, the validated alarm data along with the originating transponder address is stuffed into the ALARMQ to be processed by the by the background tasks.

If the validated alarm is a no response alarm, it will be processed differently. First, the corresponding bit in the no response map is checked to see if the address is already in the no response queue. If it is, then the routine simply exits. If the bit is not set, meaning this is a new no response, the bit is set in the map. Next, MONFLG global is checked for the status of no response reporting. If no response reporting is on, the transaction is placed in the no response queue. If the que is full, the no response reporting is turned off (bit 5, MONFLG) and an alarm message is loaded into the ALARMQ.

If the ALARMQ has only enough room left for 3 or less entries, the REPOLQ is closed until more space is available. If the REPOLQ is low on space then the POLLQ will be frozen until space becomes available. The queues will be re-opened by the SCRTY background task as required. (see SCRTY).

Module Name: CLI
Size: 6,692

This module contains the command line interpreter, used for prossesing commands entered on the main and monitor screens. It compares the string (command) entered at the command line with a list of commands pointed to by an array of pointers. If a match is found, the address parameter from the command table is returned. This address paramter is the address of the subroutine that will process the given command. The command line pointer will be left pointing to the delimiter at the end of the command string. The called routine then may further parse the command line for parameters.

Module Name: CMC
Size: 38,144

This module contains all of the routines required to communicated with the CMC (currently an HP-1000 mini-computer). It is quite well documented and fairly straight forward, so no amount of time will be spent here other than to list the names of the actual routines conatined withing the module. Refer to the source code listing for details on CMC.

Below is a list if the routines in CMC, and what they do:

INT6: CMC receiver interrupt service routine.
CMCM: CMC/SMS communications monitor background tasks (see RTC).
INITC: Initialization routine for CMC communications.
TXNEXT: Transmit message generator.
XMIT: Transmits actual messages to the CMC.
TXSAVE: Save transmit message in a buffer in case of failure.
TXRCVR: Recover failed transmit message from buffer.
CPWRUP: SMS power up message to CMC.
CMCPRC: CMC to SMS command processor routine.

Module Name: COMM
Size: 4,992

This module contains a collection of routines that enable the SMS crt to become an SMS to CMC communications monitor. The messages originating from the CMC are displayed in reverse video, while the SMS messages to the CMC are shown in normal video. All of the data is converted to hexadecimal representation for diplay. It also is rather well documented and only the routines contained within this module will be listed here.

COMM: Entry routine; screen initialization & command line processing.

ECHOTX: SMS message handler/processor.

ECHORX: CMC message handler/processor.

ECHO: Root handler of ECHOTX and ECHORX.

Module Name: COMMON
Size:20,864

This module contains seven routines that are used in the various scanning and interrupt routines and the SCRTY background task. Below is a listing of these rotuines ans a description of their operation:

PWRCAL:
This routine is actually a background task itself (see RTC). Its primary job is to check the RFLAJQ for transactions. If there are transactions present, it calculates the necessary adjustment and loads the power adjust transaction into the repol queue. The tag bits of the transaction indicate whether it is the first adjustment or a subsequent adjustment. In the case of the first adjustment, a branch is taken down to the label INIADJ. Here, the AUTOFG is checked to determine whether auto or manual power level procedure is in effect. If auto mode is indicated, a fixed value of attenuation, midway between maximum and minimum, is asserted and the adjustment command is loaded into the repoll queue. If manual mode is in effect, the address is used to index into the EEPROM power level storage area to get the 4 bit value stored there during the most recent PO command for that address. This value is loaded into the power adjust command and stuffed into the repoll queue. In both cases the process byte in image RAM is updated to show that a power level adjust is in progress.

In the case where the adjustment is a second or subsequent adjustment, the branch to INIADJ is not taken, and the process byte in image RAM is checked for the direction of the power level discrepancy (if any) so that an approriate successive approximation adjustmant can be calculated. This then is built into a new power adjust command and stuffed into the repoll que. If satisfactory power adjustment cannot be obtained in 4 tries, then an alarm message is loaded in the ALARMQ for that particular address, and the bit corresponding to the address is set in the PWRMAP, otherwise the bit is cleared.

XNDEX:
This routine will index to the status byte of temporary storage used during repol verification of an alarm.

PWRCHK:
This routine is called by either the P or I channel receiver routine to check the received power level for a particular address. It first checks to see if adjustment is in progress for this address. Then it checks the bit in the PWRMAP associated with this address. If this bit is set, it does not bother to proceed further. Otherwise, it reads in the power level from port E8h. Before checking the value, it looks at the AUTOFG to see whether auto power level adjustment is in effect.

If auto mode is not in effect, a value of 05h, (RF OK) is forced over the actual value that was read. The value (either the actual one read, or the 05h) is checked in a lookup table to determine whether it is high or low. The new value to send is calculated and another power adjust transaction is loaded into the RFLADJQ if it is still not within limits. If the adjustement procedure originated via the PO command, a report is sent to the crt as to the results of the adjustment.

ERRHAND:

This routine reports hard system errors (i.e.E004) to the LED display and the CMC (if connected).

GENQUS:

This routine opens and initializes all of the ques used throughout the SMS.

LEDOUT:

This routine clocks a four digit BCD value in BCDE registers into the front panel LED display.

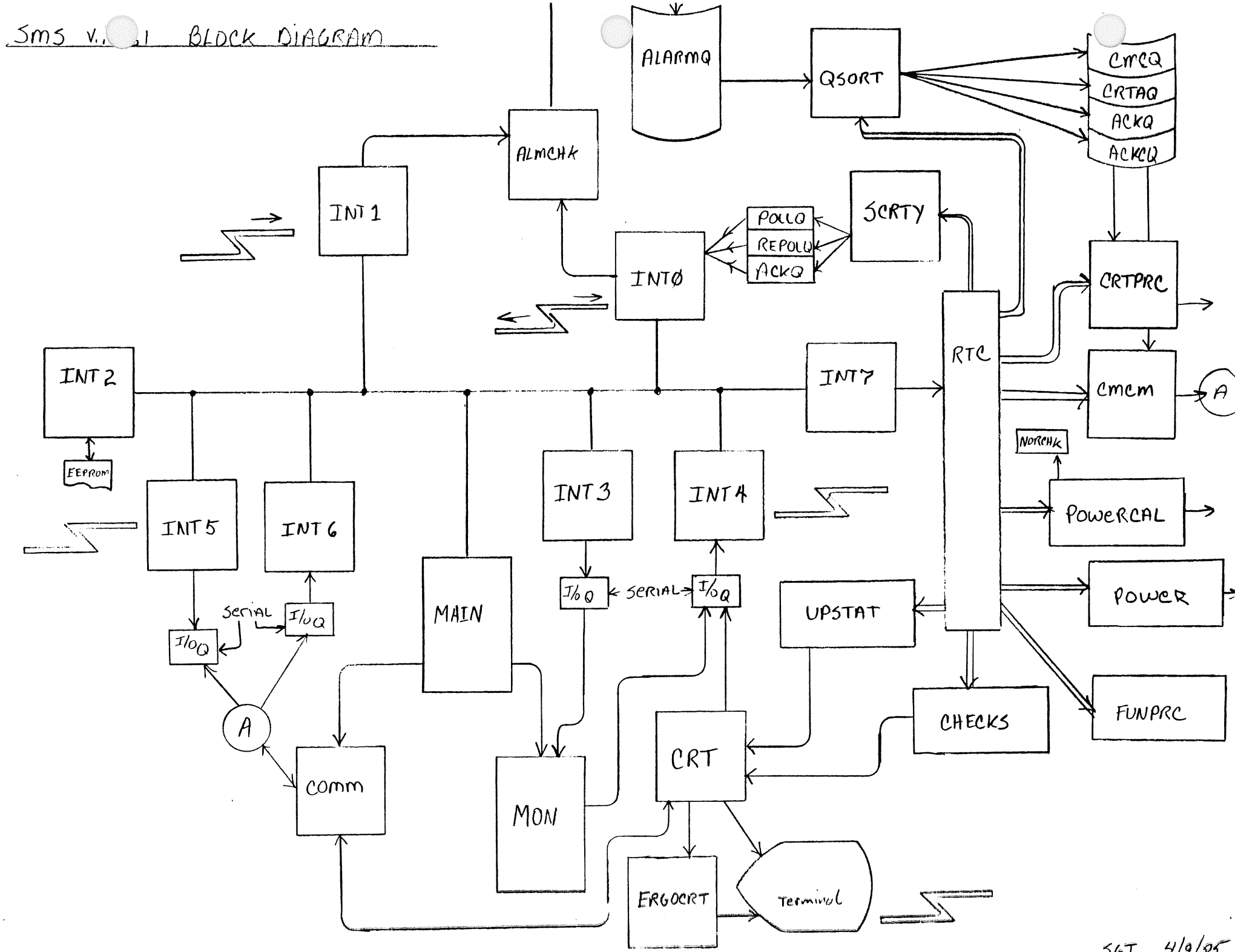
CANCEL:

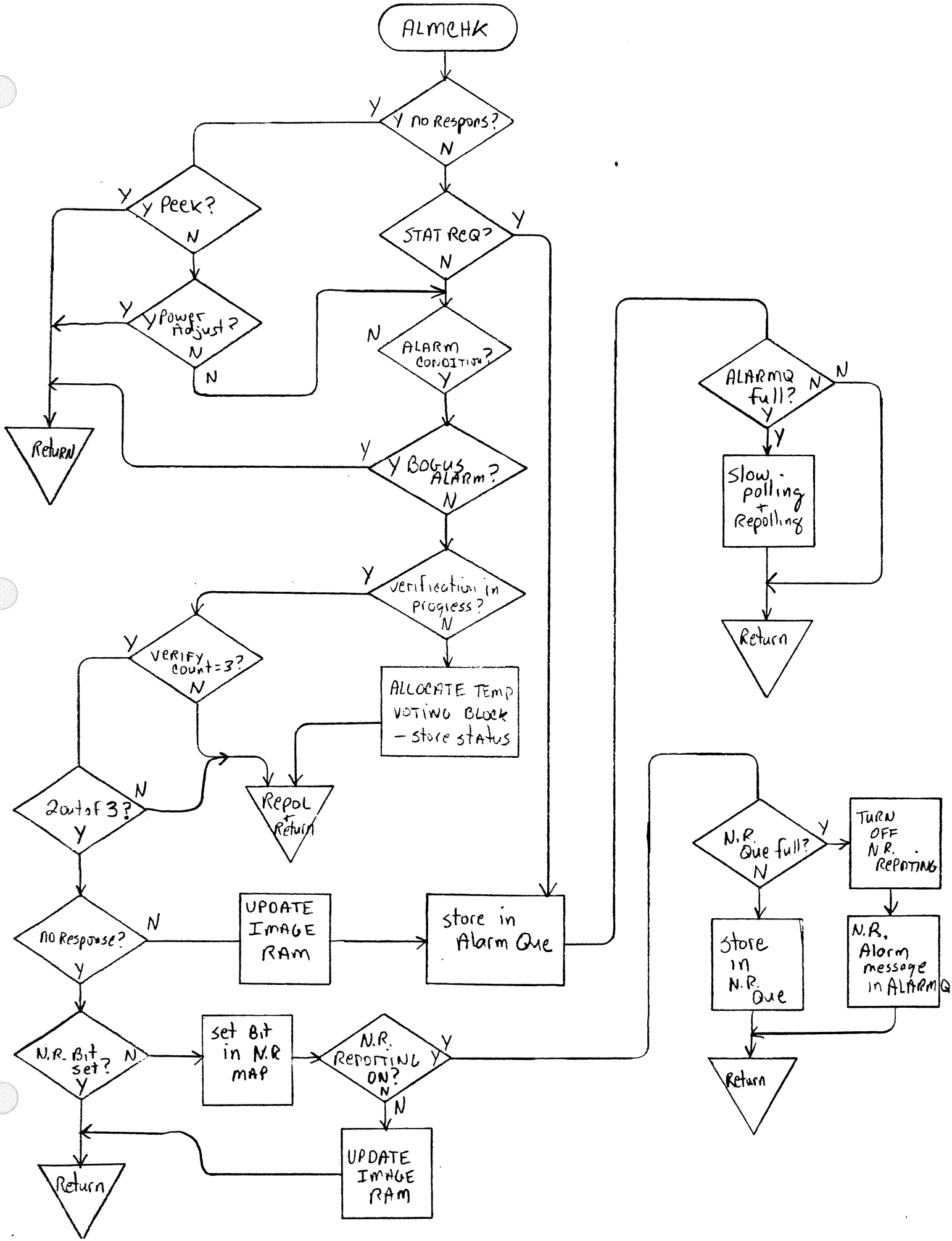
A short routine to cancel a power level adjustement in progress, and clear all associated flags and storage.

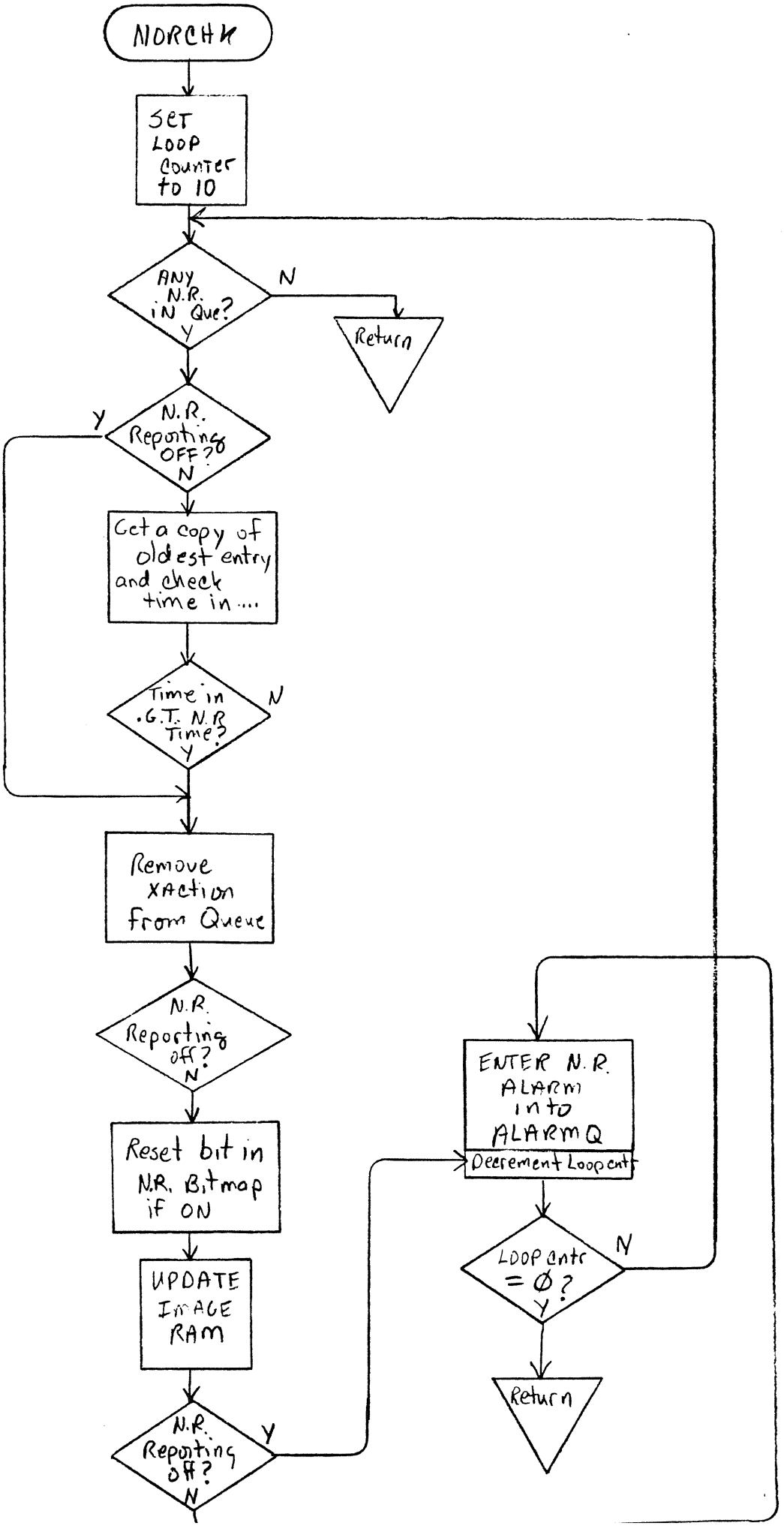
Module Name: CRT
Size: 26,752

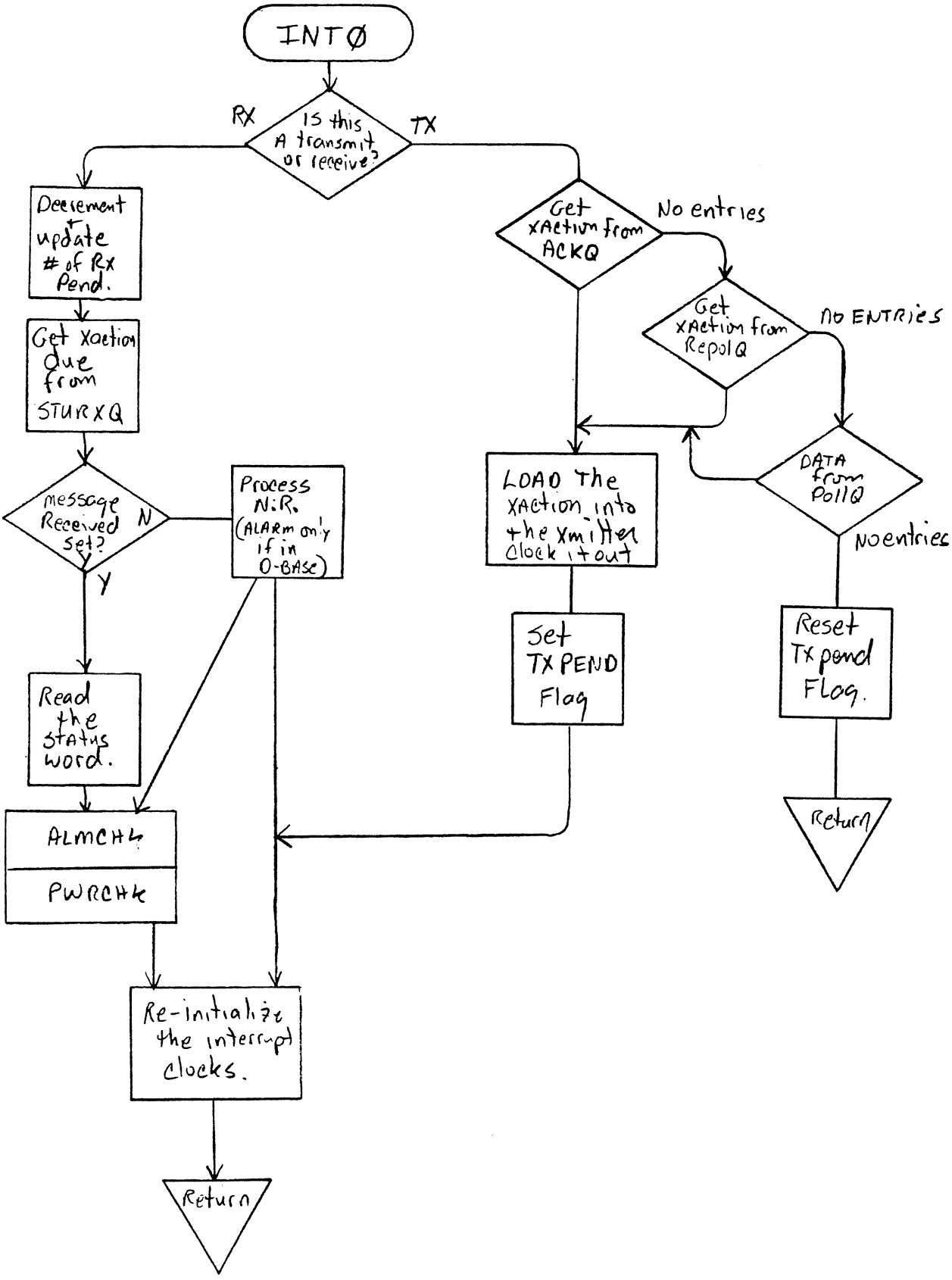
This is a well documented group of general purpose subroutines used for operation and manipulation of a terminal interface. All of the routines a rather straight forward and do not require a detailed description here. Below is a list of the routines contained in this module:

READ: Read a character from an input string.
WRITE: Write a character to terminal and keep track of cursor.
ITEXT: Read string from crt buffer into memory.
TEXT: Write a null terminated string to the crt.G
MSPACE: Output multiple spaces to the crt.
SLASH: Output an ASCII slash to the crt.
SPC: Output a single ASCII space to the crt.
PROMPT: Set up the command line with the prompt.
HEAD: Output menu heading.
MENU: Output menu directory.
CMDLN: Command line handler.
SKIPCM: Skip command within primary input string.
LASTDL: Get the last delimiter from input string.
RDVALU: Read a decimal value from primary input string.
NEXTWD: Find next word in input string.
UPTIME: Output date and time to the terminal.
TIME: Output the time to the terminal
DAYTE: Output the date to the terminal.
FUNKEY: Function key sort routine.


SMS V.[illegible].1 BLOCK DIAGRAM
ALARMQ
QSORT
CMCQ
CRTAQ
ACKQ
ACKCQ
ALMCHK
INT 1
INTØ
POLLQ
REPOLLQ
ACKQ
SCRTY
CRTPRC
RTC
INT 2
INT 7
CMCM
A
EEPROM
NORCHK
INT 5
INT 6
INT 3
INT 4
POWERCAL
I/OQ
SERIAL
I/OQ
MAIN
I/OQ
SERIAL
I/OQ
UPSTAT
POWER
A
COMM
MON
CRT
CHECKS
FUNPRC
ERGOCRT
Terminal
SGT 4/9/85


ALMCHK
Y no respons?
Y
N
Y Peek?
Y
N
STAT REQ?
Y
N
Y Power Adjust?
Y
N
N
ALARM CONDITION?
N
Y
Return
Y BOGUS ALARM?
Y
N
Y verification in progress?
Y
N
Y VERIFY count=3?
Y
N
ALLOCATE TEMP VOTING BLOCK - store status
2 out of 3?
N
Y
Repol + Return
no response?
N
Y
UPDATE IMAGE RAM
store in Alarm Que
N.R. Bit set?
N
Y
set Bit in N.R MAP
N.R. REPORTING ON?
Y
N
UPDATE IMAGE RAM
Return
ALARMQ full?
N
Y
Y
Slow polling + Repolling
Return
N.R. Que full?
Y
N
TURN OFF N.R. REPORTING
Store in N.R. Que
N.R. Alarm message in ALARMQ
Return


NORCHK
Set Loop Counter to 10
ANY N.R. in Que?
N
Y
Return
N.R. Reporting OFF?
Y
N
Get a copy of oldest entry and check time in....
Time in .G.T. N.R. Time?
N
Y
Remove XAction from Queue
N.R. Reporting off?
N
Reset bit in N.R. Bitmap if ON
UPDATE IMAGE RAM
N.R. Reporting off?
Y
N
ENTER N.R. ALARM into ALARM Q
Decrement Loop cntr
LOOP cntr = Ø?
N
Y
Return


INTØ
Is this A transmit or receive?
RX
TX
Decrement update # of Rx Pend.
Get Xaction due from STURXQ
message Received set?
N
Y
Process N.R. (ALARM only if in O-BASE)
Read the status word.
ALMCHK
PWRCHK
Re-initialize the interrupt clocks.
Return
Get xAction from ACKQ
No entries
Get xAction from RepolQ
NO ENTRIES
DATA from PollQ
No entries
LOAD the XAction into the xmitter clock it out
Set TX PEND Flag
Reset Tx pend Flag.
Return