MODULE SPECIFICATION

NAME:

MIDAS

FUNCTION:

This is the mainline of the system. It does initialization of system parameters, and contains the main control loop. Display window update, speed control, dial and function key reading, system message displays and calls to the display and simulation routines are done in the loop. The command parser is also contained as an internal subroutine. It parses and executes commands read from the keyboard in command mode. These commands are either executed within the command parser, or external command handlers are called to execute them. A system initialization routine is also called as an internal routine. This initializes display and simulation variables, and calls two other external initialization routines to set up the display and simulation.

CALLED BY:

GMS

PARAMETERS:

none

ENTRY POINTS:

MIDAS, GETOK, MIUPDAT

NORMAL EXITS:

Re-IPL's GMS on halt command from user.

ERROR CONDITIONS:

Exits on bad initialization of display.

INTERNAL ROUTINES CALLED:

COMPARSE: Contains command parser and calls to command handlers. Certain smaller commands (CLEAR, RESET, DEBUG, GMS, GO and END) are handled directly by COMPARSE.

GETOK: (GLOBAL ROUTINE) Parses the command input buffer into tokens, which are returned in a global buffer area (TOKEN).

MIUPDAT: (GLOBAL ROUTINE) Updates display window parameters and polls function keys. Keys 0-7 are used for system control and are handled here. All other keys are free to be used in the simulation. A mask is set to reflect these key hits.

INITSYS: Calls system initialization routines to initialize display and simulation. Puts user in command environment.

EXTERNAL ROUTINES CALLED:

PROSTART, QUERYDYN, PROTERM: PROCRUSTES routines

MIDASBUG, MSBINT, MIDINIT, MISIMINT, MIDISPLY, MIDASIM, MIFILE, MILOAD MISET, MIMEMM, MISAVE, MIRECL: MIDAS processors

SEGDELET, MULTIPAC, TYPEOUT, SEGLOAD, SVCCA: GMS utilities

EXTERNAL DATA REFERENCED:

MIDASCB: MIDAS common data area
PROCRUST: PROCRUSTES common data area
BHEADER: PROCRUSTES buffer header
STATREC: MIDAS simulation status record.

STORAGE UTILIZATION:

Dynamic storage allocated for auxilliary status record and memory page. Automatic storage allocated for saved status record.

ALGORITHM:

After system initialization, the main control loop is entered, where window parameters, simulation speed, and function keys are read. Depending on one of six operating modes, the command parser/processor is called (COMMAND mode), the simulation routines and display routines are called (REAL TIME, NORMAL, and SINGLE STEP modes), or just the simulation routines are called (DEBUG mode). The loop terminates on the user set halt flag, and the system is cleared, dynamic storage freed, and GMS is re-IPLed.

MODULE SPECIFICATION

NAME:

MIDISPLY

FUNCTION:

This is the main display routine for MIDAS. It implements a repetoire of five different animation effects in response to a set of pseudo display orders created by the simulator. This routine is called once per cycle of the main control loop in MIDAS, or once for every time the simulation is called. This routine is also called in the initialization of the display, from the RECALL and replay utilities when system status is changed, and from the SET utility, when a data element is altered. The five functions it performs are: BRIGHTEN or DIM a control line type device, FLASH or CHANGE the contents of a register type device, and MOVE data across a bus type device. These devices (single bit control lines, one byte registers, two byte register pairs, two byte address registers, single bit flag registers, and one and two byte busses) are referred to by symbolic addresses in the pseudo display orders generated by the simulation. It produces linked blocks of these orders, which are executed in three phases by the display program: a set up phase for all operations, a dynamic phase for MOVES, and a cleanup phase for FLASHES and MOVES.

The display program operates in two modes. The normal mode allows for full display of all requested animation effects. A brief mode, used for altering the display in initializations, SET and RECALL utilities, and REAL TIME mode display, consists of an abbreviated set of animation effects wherein the operation of the FLASH and MOVE pseudo orders are simplified.

CALLED BY:

MIDAS, INITSYS, MISET

PARAMETERS:

A pointer to a linked list of blocks of pseudo display orders is passed in the global variable MIDISTOP.

NORMAL EXITS:

The routine returns to its caller after updating the display.

ERROR CONDITIONS:

A bad display command given to PROCRUSTES causes aborting of the display process.

INTERNAL ROUTINES CALLED:

CHREG: Changes data in register type devices, when passed the device identifier of that register.

CONVASC: Does hex to ASCII conversion for CHREG.

EXTERNAL ROUTINES CALLED:

OPENIMAG, ADDSUBLK, CALLDDIM, SIMHTRAN, QUERYDYN, DELIMAGE: PROCRUSTES routines

MIUPDAT: Window and function key update routine.

EXTERNAL DATA REFERENCED:

MIDASCB: MIDAS common data area.
PROCRUST: PROCRUSTES common data area.
BHEADER: PROCRUSTES buffer header
STATREC: MIDAS simulation status record.

STORAGE UTILIZATION:

Dynamic storage allocation is done for dynamic display data lists.

GLOBAL ASSUMPTIONS & SIDE EFFECTS:

The type of animation done (normal or brief effects) depends on the global system mode variable (MIMODE).

ALGORITHM:

The pseudo display orders are interpreted in a loop. For BRIGHTEN operations, a dynamic line image is set up for the brightened line which makes a call to the line image (the image is overdrawn for brightening). This image is only allocated once. If the image is dimmed after brightening, the CALL sublk is made a NEXT sublk (no-oped). If it is then brightened once again, the NEXT sublk is once again made a CALL sublk. The DIM operation checks for one of these dynamic CALL sublks to the line image. If it is present, the CALL is no-oped (made a NEXT sublk).

For FLASH and CHANGE operations, data in the status record is converted to ASCII and the display data is changed in the named register sublk. For FLASH operations, the blink attribute is set, and a dynamic list element is created with the name of the call with the blink attribute set. In brief mode, FLASH is treated identically to change.

For MOVE operations, a dynamic display list is set up containing interpolation pairs that cause the bus data to be displayed incrementally along a path between two coordinates. The display list also contains pointers and length information for changing source and destination registers, and for displaying data along two separate paths (so that the data paths may turn corners). The register that is the bus source is changed. In brief mode, the destination register data is changed as well, and no move takes place.

The second phase of the display program is a loop where the data in move operations is moved across the bus path.

In brief mode neither this phase, nor the following one is executed, and return is made directly to the calling routine. In normal mode, a dynamic display list is allocated for FLASH and MOVE operations, as described above. In this second phase, FLASH elements are ignored. This phase consists of a loop, the length of which is determined by a counter set by the user (MINTVAL) with the speed dial. Based on the setting of this dial, the incremental moves along the bus are made in such a way that the total move is completed at the end of the loop. Thus, for each traverse of the loop, the interpolation pairs are recomputed for each active bus element on the basis of increments calculated in phase 1. The window parameter and function keys are also polled in this loop, to allow user control during this display phase.

The final phase of the display consists of a loop where the dynamic display list elements set up in phase 1 are dfreed. For FLASH operations, the blink attribute on the register image is reset. For MOVE operations, data is changed in the destination registers. At the end of this loop, dynamic images are closed and control is returned to the caller.

MODULE SPECIFICATION

NAME:

MISET

FUNCTION:

This is the register/line set utility for MIDAS. It implements the SET command and is called by the command parser. Using a set of pre-defined symbolic names, the user may change the data or state of any simulated register, line, or flag. The user specifies the idname, and the data in hexadecimal (for registers) or the keywords Set or Reset (for flags and lines). The data maintained in the simulation status record is then changed and the display processor is called to update the display.

CALLED BY:

COMPARSE

PARAMETERS:

none

ENTRY POINTS:

MISET, CONVHEX

NORMAL EXITS:

Returns to caller on completion of command.

ERROR CONDITIONS:

Invalid data or id name causes command to be ignored.

INTERNAL ROUTINES CALLED:

CONVHEX: (GLOBAL ROUTINE) Converts an EBCDIC string of up to four bytes to a hex constant.

EXTERNAL ROUTINES CALLED:

MIDISPLY, GETOK

EXTERNAL DATA REFERENCED:

MIDASCB: MIDAS common data area
STATREC: MIDAS simulation status record

ALGORITHM:

GETOK is called to parse the command string into an idname and data. The global symbol table MISYMTAB is then searched. If the symbol is found, the status record entry is changed with the new data and a pseudo display order is built. MIDISPLY is called in brief mode, to update the display.

MODULE SPECIFICATION

NAME:

MIMEMM

FUNCTION:

This routine is the memory display and alter utility. It is called from the command parser, to implement the DISPLAY and STORE commands. For DISPLAY the user specifies an address in simulated memory, and the routine displays the contents of that address and several bytes following it in memory, in the system prompt buffer. For STORE, the user specifies an address and a variable length string of hexadecimal data. The routine converts the data and stores it starting at the specified locations and displays it as described above

CALLED BY:

COMPARSE

PARAMETERS:

R9=0 for STORE, R9=1 for DISPLAY

ENTRY POINTS:

MIMEMM

NORMAL EXITS:

This routine returns to the caller when the command is completed

ERROR CONDITIONS:

Bad address or data causes the command to be ignored.

INTERNAL ROUTINES CALLED:

CONVEBC, CONVERT: Hexadecimal to EBCDIC conversion routines.

EXTERNAL ROUTINES CALLED:

GETOK, CONVHEX

EXTERNAL DATA REFERENCED:

MIDASCB: MIDAS common data area

ALGORITHM:

GETOK parses the command string into an address and data which are converted to hexadecimal. For display, the contents of memory starting at the specified address for 12 bytes is displayed in a buffer in six groups of two bytes, after conversion to EBCDIC. For store, the specified data is converted and stored starting at the specified address. The contents of memory starting at the specified address is displayed as described above. If the given address is outside the bounds of memory, the command is ignored. If invalid data is found, valid data preceeding it is stored, and the rest of the data string is ignored. Data is parsed

in groups of four hexadecimal digits.

MODULE SPECIFICATION

NAME:

MIFILE

FUNCTION:

This is the file handling routine for MIDAS. It is called by the command parser to implement disk I/O for the FILE, LOAD, SAVE, and RECALL commands. The pointer to the appropriate data or buffer area is placed in an FCB along with the logical record length and the filename and filetype. The read or write operation is then performed, along with a FINIS on that file.

CALLED BY:

COMPARSE

PARAMETERS:

none

ENTRY POINTS:

MIFILE, MILOAD, MISAVE, MIRECL

NORMAL EXITS:

This routine returns to its caller after a FINIS is done on the file.

ERROR CONDITIONS:

File not found on reads, and disk errors on writes cause an error message to be displayed in the prompt buffer and control to return to the caller.

INTERNAL ROUTINES CALLED:

none

EXTERNAL ROUTINES CALLED:

RDBUF, WRBUF, FINIS: GMS disk I/O routines.

EXTERNAL DATA REFERENCED:

MIDASCB: MIDAS common data area
STATREC: MIDAS simulation status record

GLOBAL ASSUMPTIONS & SIDE EFFECTS:

The current file name is contained in the global variable MICURFIL.

ALGORITHM:

Depending on the entry, the appropriate buffer pointer and record length is put in the FCB. The appropriate filetype (MIDA for SAVE and RECALL; and MEMM for LOAD and FILE) is moved into the FCB. The filename in MICURFIL is checked. If it is non-blank, it is moved into the FCB, otherwise the default filename ..TEMP.. is moved into the FCB. Then

either a RDBUF or WRBUF is issued. If it is successful, a FINIS is done and control is returned to the caller.

MODULE SPECIFICATION

NAME:

MIDINIT

FUNCTION:

This routine initializes the basic display by interpreting an initialization module and calling the appropriate PROCRUSTES routine to create the picture.

CALLED BY:

MIDAS

PARAMETERS:

none

ENTRY POINTS:

MIDINIT

NORMAL EXITS:

Returns to calling routine when initialization is complete.

ERROR CONDITIONS:

Initialization errors or errors from PROCRUSTES routines cause return to calling routine with a non-zero return code in R3 and a message typed at the Teleray terminal.

EXTERNAL ROUTINES CALLED:

OPENIMAG, CALDDIM, ADDSUBLK, ADDMARK: PROCRUSTES routines.

EXTERNAL DATA REFERENCED:

MIDASCB: MIDAS common data area

MIBDF: Display initialization module.

ALGORITHM:

The initialization module is an external CSECT. It contains blocks which contain an operation code, a length and data. The routine goes into a loop which gets the op code for the block, which can specify one of eleven initialization operations. The operation is then done, which entails calling a PROCRUSTES image creation or manipulation routine and passing it data contained in the block, or allocating entries in one of three tables to be used in locating line, register or bus type data elements in the data structure. The routine continues until an end flag is hit, when it returns to the calling routine.