Midas Binary Tape Format

Midas binary paper and magnetic tapes use exactly the same format. A record on magnetic tape corresponds to a block of punching in paper tape. There is blank tape both before and after each record on magnetic tape and each block on paper tape. All references to block in what follows should be interpreted as record for magnetic tape.

With the exception of the single word JMP instruction now punched by MIDAS, all blocks begin with two position and length words and end with a sum check word. There are never more than 64 data words between, making a maximum block length of 67 words (105 octal). The number of data words may always be found by subtracting the first word from the second. The sum check word contains the sum modulo $2^{18}-1$ of all other words in the block, including the first two.

TYPES OF BLOCKS

There are four different types of blocks. The types are distinguished by the first two bits of the first or second word of the block. (The first two bits of the second word are always the same as those of the first.) The four types of records are:

Bits	Type	Remarks
00	Absolute	
01	Relocatable	Also called LAC block because LAC = 200000
10	Library	Also called ADD block because ADD = 400000
11	Jump	JMP=600000, ~~not yet implemented~~

ABSOLUTE BLOCKS

The 16 bits remaining in the first word contain the address in core where the first data word is to be stored. The 16 bits of the second word, therefore, contain the address just following the one in which the last data word will be stored.

If you read in a program in extend mode, the reader leader (loading routine) will store absolute blocks in the bank they address. Addresses 0 - 7777 will go in bank 0, addresses 10000 - 17777 in bank 1, etc. The loading program itself will be in the bank addressed by the address extension switches.

If you read in a program in the normal mode, the reader leader (loading routine) will store absolute blocks whose addresses are between 0 and 7777, inclusive, in the bank it is in. Both the loading routine routine and the information read will be in the bank indicated in the address extension switches. Blocks with addresses 1XXXX and 3XXXX will cause eratic operation, and blocks with addresses 2XXXX will be treated as if the 2 were missing.

For example, read in a tape with absolute blocks bank zero addresses (0 - 7777). The address extension togs are set to bank 3. If you are in normal mode, both loading routine and program go in bank 3_x. If you are in extend mode, the loading routine goes in bank 3, but the program goes in bank 0.

RELOCATABLE BLOCKS

In order that binary programs may be added to each other without need to recompile them. MIDAS has provision for punching relocatable binary tapes. In reading the relocatable blocks, the MIDAS LINKING LOADER changes the binary values punched in the tape so that the program runs at a location other than that for which it was originally assembled.

Relocatable programs may have "ENTRY" points, and "EXITS". A routine is said to have an EXIT if it refers to a register not a part of itself. For example, a routine to solve quadratics might refer to a square root routine, e.g. jsp SQRT. A reference to get data is also called an EXIT, even though the program doesn't exit there, e.g. lac TIME.

A routine is said to have an entry any time it gives the definition of some symbolic quantity for use outside itself. For example, the square root routine would have an entry of the form SQRT, dap sqrtx. The time routine would have a register labled TIME, 0.

The entries and exits of a routine are punched by MIDAS in symbolic form in the binary tape. The linking loader remembers the definition of all entries it has so far found, and the locations where as-yet-undefined exits are used. When a new entry is defined, all the exits which use it are fixed up to work correctly. On command, the linking loader will read standard routines off of the library tape to define any entries called for by the routines used. For example, one can simply call for SQRT without including it in the binary tapes loaded because it will be defined on the library tape.

The relocatable blocks, then, contain binary information, symbolic information, and information which lets the linking loader discriminate between them. In a relocatable routine, the tape is punched as if the routine were going to be located starting at 0. Thus the first block of the tape will start out with the words 200000 (relocatable block, starting at 0) followed by 200000+N where N is the number of words in the first block before the sum check. N is always $10\phi_8$ or less.

3

Next in the relocatable block is a word made up of pairs of bits which tell whether the next nine items are:

00	Symbolic
01	Absolute, not to be changed during relocation
10	Relocated by subtracting ~~initial address~~
11	Relocated by adding ~~initial address~~

The pairs of bits in this discrimination word are read from left to right. The items they refer to may occupy from one, two, or three registers of space on the tape. Following the nine items will be another discrimination word, and so on until the place for the sum check is reached. Premature arrival at a sum check may cause part of the last discrimination word to be unused because the next block will start off with its own discrimination word.

The nine items following the discrimination word are treated as follows:

01 Absolute item: One storage word, transferred directly from tape to core.

11 Relocated by adding

10 Relocated by subtracting

The initial position of this routine in core is added (subtracted) to the value punched on tape to get a result stored in core.

00 Symbolic. Two words make up the symbol. In the first are punched the RIGHT 3 letters of the symbol in mod 50_8 notation. In the second are punched the LEFT 3 letters of the symbol in mod 50_8 notation. The two free bits of the first word are decoded to mean:

00 An EXIT, add or subtract value of exit (when found) according as the leftmost bit of the second word is 0=add 1=sub.

01 An absolute entry. Value stored in the third word is the value of the entry.

10 Minus relocation entry. Value stored in the third word minus the relocation* is taken as the value of the entry.

11 Plus relocation entry. Value stored in the third word plus the relocation is taken as the value of the entry. This is the normal entry for use in subroutines.

* In adding or subtracting for relocation, a 12 bit address is used if the leftmost bit of the second symbolic word is 0, but a 16 bit address is used if the leftmost bit is a 1.

Notice that symbolic information on the tape does not place information in core for the routine being read, but merely signals the linking loader about the linkages to set up.

LIBRARY

The first word of a library block is 400000. The second word contains 400000 + N where N is the number of words punched before the sum check. $N \leq 100_8$. The data words of the library block are taken in pairs. Each pair contains a six letter name in mod 50_8 notation. When the linking loader is scanning the library tape, it looks at these names to see if it needs any of the routines named. If not, it skips whatever follows the library block unitl it finds another library block or a jump block. If the linking loader needs anything named in the library block,it reads whatever follows and adds it to the binary in core.

The names that appear in a library block need not correspond to the names of the program which follows it. The names in a library block are not defined as entrys by the linking loader. Therefore, if it is known that the quadratic routine will need the square root routine, the square root routine should be given the names "SQRT" and "QUADRA". In this way, calling QUADRA will automatically put in the SQRT routine so that later on the tape QUADRA can use it.

If more than 32 (40_8 - 100/2) names are used in a library block, a library continuation block will follow it. The continuation block will start with 400100.

JUMP BLOCK

The jump block has identical format to the relocatable block, except that only one storage word will appear. This word will be a jmp instruction which will be modified by the exits given. The jump will start the object program.