Microsoft® Windows

Software Development Kit

Programmer's Reference

Version 2.0

Microsoft Corporation

Document No. 050051053-200-I02-1087
Part No. 00475

Figure 7.1 shows the schematic of a 10 × 12 pixel character:

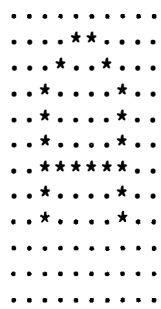

Figure 7.1 10 × 12 Pixel Character

The bytes are given here in pairs, each pair corresponding to one scan line (10 pixels and 6 bits of padding):

```
0000  0B00  1200 4100 4100 4100 3F00 4100 4100 0000 0000 0000
```

Note that the second byte of each pair in this particular case is always zero.

7.3 Executable-File Header Format

An executable file contains Windows code and data, or Windows code, data, and resources. A header, with information about segment entry points, stack offsets, and stack sizes, is added to the beginning of this executable file. The loader uses this header information when it loads the executable segments in memory. The header is broken into two parts: an old-style header and a new-style header.

7.3.1 Old-Style Header

The old-style header (WINSTUB) contains information that the loader expects for a DOS executable file. It contains a stub program that the loader can place in memory when necessary, it points to the new-style header, and it contains a relocation table.

Figure 7.2 illustrates the four distinct parts of the old-style executable header:

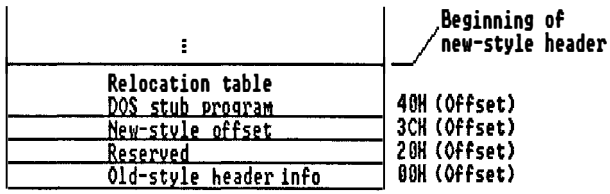

Figure 7.2 WINSTUB—Windows Old-Style Header

The loader uses the word value at offset 18H in the old-style header to determine whether a new-style header follows; if this value is 40H, the loader assumes that a new-style header will be found. (This value, 40H, is the address of the relocation table.) If a new-style header does exist, the long-word value found at offset 3CH in the old-style header gives the relative byte offset from the start of the old-style header to the start of the new-style header.

7.3.2 New-Style Header

Because Windows executable files are often larger than one segment (64K), additional information (that doesn't appear in the old-style header) is required so that the loader can properly load each segment. The new-style header was developed to provide the loader with this information automatically.

Sections 7.3.3 through 7.3.10 describe each of the eight entries in the new-style header. Each section contains a description of an entry and information about the various components of an entry in the new-style header.

7.3.3 New-Style Header Information Block

The first entry in the new-style header contains general header information, as well as information that Windows copies to the module table. (The module table contains information that the linker uses for dynamic linking.) Windows copies the information from location (relative offset) 00H to

the end of the information block into the module table. The following list describes the general information found in this block; the locations are relative to the beginning of the header-information block:

Location	Content
0H	Signature word. N is the low-order byte. E is the high-order byte.
2H	Version number of the linker.
3H	Revision number of the linker.
4H	Offset of the entry table (relative to the beginning of the new-style header).
6H	Number of bytes in the entry table.
8H	CRC-32 of the entire contents of the file (with the subsequent words taken as 00 during the calculation).
CH	Keyword. 0000H NOAUTODATA 0001H SINGLEDATA (Solo). 0002H MULTIPLEDATA (Instance). 2000H Errors detected at link time. 8000H Library module. The SS:SP information is invalid and CS:IP points to an initialization procedure called with AX equal to the module handle. This initialization procedure must execute a far return to the caller, with AX not equal to zero indicating success and AX equal to zero indicating failure to initialize. DS is set to the library's data segment if the SINGLEDATA keyword is set. Otherwise, DS is set to the caller's data segment. Only executable programs that have their SINGLEDATA keyword set may be dynamically linked. If SINGLEDATA is set, MULTIPLEDATA must be cleared.
EH	Segment number of the automatic data segment (index into the segment table). EH is set to zero if the SINGLEDATA and MULTIPLEDATA bits are cleared.

10H	Initial size (in bytes) of the dynamic heap added to the data segment. This word equals zero if there is no local allocation.
12H	Initial size (in bytes) of the stack added to the data segment. This word equals zero if SS does not equal DS.
14H	Segment number:offset of CS:IP.
18H	Segment number:offset of SS:SP. Segment number is an index into the module's segment table. The first entry in the segment table is segment number 1. If SS equals the automatic data segment and SP equals zero, the stack pointer is set to the top of the automatic data segment, just below the additional heap area.
1CH	Number of entries in the segment table.
1EH	Number of bytes in the nonresident-name table.
20H	Offset of the segment table, relative to the beginning of the new *.exe* header.
22H	Offset of the resource table, relative to the beginning of the new *.exe* header.
24H	Offset of the resident-name table, relative to the beginning of the new *.exe* header.
26H	Offset of the module-reference table, relative to the beginning of the new *.exe* header.
28H	Offset of the imported-names table, relative to the beginning of the new *.exe* header.
2AH	Offset of the nonresident-name table, relative to the beginning of the file.
2EH	Number of movable entry points.
30H	Shift count of the logical-sector alignment, log (base 2) of the segment sector size (default 9).
32H	Number of reserved segments.
34H	10 dup (0) reserved.

7.3.4 Segment Table

The segment table contains entries for each segment in the executable file. The following subentries appear within each entry; their locations are relative to the beginning of the entry:

Location	Description
0H	Logical-sector offset (*n* byte) to the contents of the segment data, relative to the beginning of the file. Zero means no file data.
2H	Length of the segment in the file (in bytes). Zero means 64K.
4H	Keyword. 0000H CODE Code-segment type. 0001H DATA Data-segment type. 0007H TYPE_MASK Segment-type field. 0010H MOVEABLE Segment is not fixed. 0040H PRELOAD Segment will be preloaded; read-only if this is a data segment. 0100H RELOCINFO Set if segment has relocation records. F000H DISCARD Discard priority.
6H	Minimum allocation size of the segment (in bytes). Total size of the segment. Zero means 65,536.

7.3.5 Resource Table

The resource table follows the segment table and contains entries for each resource in the executable file. The following list describes the subentries within each resource entry; their locations are relative to the beginning of the entry:

Location	Description
0H	Alignment shift count for resource data.
2H	Type ID: it is integer type if the high-order bit is set (8000H); otherwise, it is the offset to the type string, relative to the beginning of the resource table. If the type ID equals zero, it marks the end of the resource records.

4H	Number of resources for this type.
6H	Reserved.
	Number of resources/copies of the resource entry.
AH	Offset to the contents of the resource data, relative to the beginning of the file; the offset is in terms of alignment units specified at the beginning of the resource table.
CH	Length of the resource in the file (in bytes).
EH	Keyword.
	0010H MOVEABLE Resource is not fixed.
	0020H PURE Resource can be shared.
	0040H PRELOAD Resource is preloaded.
10H	Resource ID. It is integer type if the high-order bit is set (8000H); otherwise, it is the offset to the resource string, relative to the beginning of the resource table.
12H	Reserved.
	Resource type and name strings are stored at the end of the resource table. Note that these strings are *not* null-terminated.
16H	Length of type or name; equals zero if end of resource table.
17H	Text of type or name; case-sensitive.

7.3.6 Resident-Name Table

The resident-name table follows the resource table, and contains module names. Strings in the resident-name table are case-sensitive; they are *not* null-terminated. The following list describes the subentries and their locations with respect to the beginning of each entry:

Location	Description
00H	Length of string. This byte equals zero if there are no more strings in the table.
01H to XXH	Text of character string.
XX + 1H	Ordinal number (index into entry table).

7.3.7 Module-Reference Table

The module-reference table follows the resident-name table. Each entry contains an offset for the module-name string within the imported-names table; each entry is two bytes long.

7.3.8 Imported-Name Table

The imported-name table follows the module-reference table. This table contains the names of modules that were imported by the executable file. Each entry is composed of a one-byte field that contains the length of the string, followed by any number of characters. The strings are *not* null-terminated.

7.3.9 Entry Table

The entry table follows the imported-name table. This table contains bundles of entry-point ordinal values. (The ordinal value of the first entry point is the number 1.) The loader scans the bundles, searching for a specific entry point and, upon finding the entry point, multiplies that point's ordinal value by the entry size in order to index the entry properly.

The linker forms bundles in the most dense manner it can, under the restriction that it cannot reorder entry points to improve bundling. The reason for this restriction is that other *.exe* files may refer to entry points within this bundle by their ordinal value, as described in the following table:

Location	Description
0H	Number of entries in this bundle. All records in one bundle are either movable or refer to the same fixed segment. This byte equals zero if there are no more bundles in the entry table.
1H	Segment indicator for this bundle.
	000 Unused.
	0FFH Movable segment; segment number is in the six-byte entry shown in following list:
	DB Keyword.
	0000 0001 Set if the entry is exported.
	0000 0010 Set if the segment uses shared data segments.

INT 3FH.

DB Segment number.

DW Offset.

Other—Segment number of the fixed segment.

If this is a fixed segment, the entries are three bytes:

DB Keyword.

0000 0001 Set if the entry is exported.

0000 0010 Set if the entry uses a global (shared) data segment.

The following assembly-language instruction must be the first instruction in the prologue of this entry:

```
MOV AX,DS-value
```

This may be set only for SINGLEDATA library modules.

DW Offset.

7.3.10 Nonresident-Name Table

The nonresident-name table follows the entry table. The first entry in a nonresident-name table is a module description. Strings in the nonresident-name table are case-sensitive and are *not* null-terminated. Each entry in the table is of variable size. The following list describes the subentries and their locations with respect to the beginning of each entry:

Location	Description
00H	Length of string. This byte equals zero if there are no more strings in the table.
01H to XXH	Text of character string.
XX + 1H	Ordinal number (index into entry table).