Chapter 12. Object File Formats

This chapter describes the format of several object files.

12.1. ELF/DWARF Object Format

The TASKING VX-toolset for TriCore produces objects in the ELF/DWARF 3 format.

The ELF/DWARF Object Format for the TriCore toolset follows the convention as described in the *TriCore Embedded Application Binary Interface* [Infineon].

For a complete description of the ELF and DWARF formats, please refer to the *Tool Interface Standard (TIS)*.

12.2. Intel Hex Record Format

Intel Hex records describe the hexadecimal object file format for 8-bit, 16-bit and 32-bit microprocessors. The hexadecimal object file is an ASCII representation of an absolute binary object file. There are six different types of records:

- Data Record (8-, 16, or 32-bit formats)
- End of File Record (8-, 16, or 32-bit formats)
- Extended Segment Address Record (16, or 32-bit formats)
- Start Segment Address Record (16, or 32-bit formats)
- Extended Linear Address Record (32-bit format only)
- Start Linear Address Record (32-bit format only)

By default the linker generates records in the 32-bit format (4-byte addresses).

General Record Format

In the output file, the record format is:

| : | length | offset | type | content | checksum |
|---|--------|--------|------|---------|----------|
|---|--------|--------|------|---------|----------|

where:

is the record header.

length is the record length which specifies the number of bytes of the content field. This

value occupies one byte (two hexadecimal digits). The linker outputs records of 255 bytes (32 hexadecimal digits) or less; that is, *length* is never greater than 0xFF.

TASKING VX-toolset for AURIX Development Studio User Guide (non-commercial)

offset is the starting load offset specifying an absolute address in memory where the

data is to be located when loaded by a tool. This field is two bytes long. This field is only used for Data Records. In other records this field is coded as four ASCII

zero characters ('0000').

type is the record type. This value occupies one byte (two hexadecimal digits). The

record types are:

| Byte Type | Record Type |
|-----------|-------------------------------------|
| 00 | Data |
| 01 | End of file |
| 02 | Extended segment address (not used) |
| 03 | Start segment address (not used) |
| 04 | Extended linear address (32-bit) |
| 05 | Start linear address (32-bit) |

content is the information contained in the record. This depends on the record type.

checksum is the record checksum. The linker computes the checksum by first adding the

binary representation of the previous bytes (from length to content). The linker then computes the result of sum modulo 256 and subtracts the remainder from 256 (two's complement). Therefore, the sum of all bytes following the header is

zero.

Extended Linear Address Record

The Extended Linear Address Record specifies the two most significant bytes (bits 16-31) of the absolute address of the first data byte in a subsequent Data Record:

| : 02 0000 04 upper_address checksul | : |
|-------------------------------------|---|
|-------------------------------------|---|

The 32-bit absolute address of a byte in a Data Record is calculated as:

```
( address + offset + index ) modulo 4G
```

where:

address is the base address, where the two most significant bytes are the upper_address

and the two least significant bytes are zero.

offset is the 16-bit offset from the Data Record.

index is the index of the data byte within the Data Record (0 for the first byte).

Example:

| :020000 | 0400FFFB |
|---------|-----------------|
| | _ checksum |
| | _ upper_address |
| | _ type |

```
| |_ offset
|_ length
```

Data Record

The Data Record specifies the actual program code and data.

| : | length | offset | 00 | data | checksum |
|---|--------|--------|----|------|----------|
|---|--------|--------|----|------|----------|

The *length* byte specifies the number of *data* bytes. The linker has an option (--hex-record-size) that controls the length of the output buffer for generating Data records. The default buffer length is 32 bytes.

The offset is the 16-bit starting load offset. Together with the address specified in the Extended Address Record it specifies an absolute address in memory where the data is to be located when loaded by a tool.

Example:

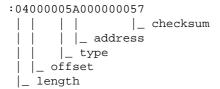
Start Linear Address Record

The Start Linear Address Record contains the 32-bit program execution start address.

| : | 04 | 0000 | 05 | address | checksum |
|---|----|------|----|---------|----------|
|---|----|------|----|---------|----------|

With linker option --hex-format=S you can prevent the linker from emitting this record.

Example:



End of File Record

The hexadecimal file always ends with the following end-of-file record:

