libGIS

1.0.5

Generated by Doxygen 1.7.1

Sat Feb 5 2011 19:38:39

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 AtmelGenericRecord Struct Reference

`#include <atmel_generic.h>`

Data Fields

- uint32_t address
- uint16_t data

3.1.1 Detailed Description

Structure to hold the fields of an Atmel Generic record.

3.1.2 Field Documentation

3.1.2.1 uint32_t AtmelGenericRecord::address

The 24-bit address field of the record.

3.1.2.2 uint16_t AtmelGenericRecord::data

The 16-bit data field of the record.

The documentation for this struct was generated from the following file:

- atmel_generic.h

3.2 IHexRecord Struct Reference

`#include <ihex.h>`

Data Fields

- uint16_t address
- uint8_t data [IHEX_MAX_DATA_LEN/2]
- int dataLen
- int type
- uint8_t checksum

3.2.1 Detailed Description

Structure to hold the fields of an Intel HEX8 record.

3.2.2 Field Documentation

3.2.2.1 uint16_t IHexRecord::address

The 16-bit address field.

3.2.2.2 uint8_t IHexRecord::checksum

The checksum of this record.

3.2.2.3 uint8_t IHexRecord::data[IHEX_MAX_DATA_LEN/2]

The 8-bit array data field, which has a maximum size of 256 bytes.

3.2.2.4 int IHexRecord::dataLen

The number of bytes of data stored in this record.

3.2.2.5 int IHexRecord::type

The Intel HEX8 record type of this record.

The documentation for this struct was generated from the following file:

- ihex.h

3.3 SRecord Struct Reference

`#include <srecord.h>`

Data Fields

- uint32_t address
- uint8_t data [SRECORD_MAX_DATA_LEN/2]
- int dataLen

- int type
- uint8_t checksum

3.3.1 Detailed Description

Structure to hold the fields of a Motorola S-Record record.

3.3.2 Field Documentation

3.3.2.1 uint32_t SRecord::address

The address field. This can be 16, 24, or 32 bits depending on the record type.

3.3.2.2 uint8_t SRecord::checksum

The checksum of this record.

3.3.2.3 uint8_t SRecord::data[SRECORD_MAX_DATA_LEN/2]

The 8-bit array data field, which has a maximum size of 32 bytes.

3.3.2.4 int SRecord::dataLen

The number of bytes of data stored in this record.

3.3.2.5 int SRecord::type

The Motorola S-Record type of this record (S0-S9).

The documentation for this struct was generated from the following file:

- srecord.h

Chapter 4

File Documentation

4.1 atmel_generic.h File Reference

Low-level utility functions to create, read, write, and print Atmel Generic binary records.

`#include <stdio.h>`

`#include <stdint.h>`

`#include <stdlib.h>`

`#include <string.h>`

Data Structures

- struct AtmelGenericRecord

Enumerations

- enum **_AtmelGenericDefinitions** {
  ATMEL_GENERIC_RECORD_BUFF_SIZE = 16, **ATMEL_GENERIC_ADDRESS_LEN** = 6, **ATMEL_GENERIC_DATA_LEN** = 4, **ATMEL_GENERIC_SEPARATOR_OFFSET** = 6,
  ATMEL_GENERIC_SEPARATOR = ':' }
- enum AtmelGenericErrors {
  ATMEL_GENERIC_OK = 0, ATMEL_GENERIC_ERROR_FILE = -1, ATMEL_GENERIC_ERROR_EOF = -2, ATMEL_GENERIC_ERROR_INVALID_RECORD = -3,
  ATMEL_GENERIC_ERROR_INVALID_ARGUMENTS = -4, ATMEL_GENERIC_ERROR_NEWLINE = -5 }

Functions

- int New_AtmelGenericRecord (uint32_t address, uint16_t data, AtmelGenericRecord *genericRecord)
- int Read_AtmelGenericRecord (AtmelGenericRecord *genericRecord, FILE *in)

- int Write_AtmelGenericRecord (const AtmelGenericRecord *genericRecord, FILE *out)
- void Print_AtmelGenericRecord (const AtmelGenericRecord *genericRecord)

4.1.1 Detailed Description

Low-level utility functions to create, read, write, and print Atmel Generic binary records.

Author

Vanya A. Sergeev <vsergeev@gmail.com>

Date

February 2011

Version

1.0.5

4.1.2 Enumeration Type Documentation

4.1.2.1 enum AtmelGenericErrors

All of the possible error codes the Atmel Generic record utility functions may return.

Enumerator:

ATMEL_GENERIC_OK Error code for success or no error.

ATMEL_GENERIC_ERROR_FILE Error code for error while reading from or writing to a file. You may check errno for the exact error if this error code is encountered.

ATMEL_GENERIC_ERROR_EOF Error code for encountering end-of-file when reading from a file.

ATMEL_GENERIC_ERROR_INVALID_RECORD Error code for error if an invalid record was read.

ATMEL_GENERIC_ERROR_INVALID_ARGUMENTS Error code for error from invalid arguments passed to function.

ATMEL_GENERIC_ERROR_NEWLINE Error code for encountering a newline with no record when reading from a file.

4.1.3 Function Documentation

4.1.3.1 int New_AtmelGenericRecord (uint32_t *address*, uint16_t *data*, AtmelGenericRecord * *genericRecord*)

Sets all of the record fields of an Atmel Generic record structure. Note that the Atmel Generic record only supports 24-bit addresses.

Parameters

address The 24-bit address of the data.

data The 16-bit word of data.

genericRecord A pointer to the target Atmel Generic record structure where these fields will be set.

Returns

ATMEL_GENERIC_OK on success, otherwise one of the ATMEL_GENERIC_ERROR_ error codes.

Return values

ATMEL_GENERIC_OK on success.

ATMEL_GENERIC_ERROR_INVALID_ARGUMENTS if the record pointer is NULL.

4.1.3.2 void Print_AtmelGenericRecord (const AtmelGenericRecord * *genericRecord*)

Prints the contents of an Atmel Generic record structure to stdout. The record dump consists of the address and data fields of the record.

Parameters

genericRecord A pointer to the Atmel Generic record structure.

Returns

Always returns ATMEL_GENERIC_OK (success).

Return values

ATMEL_GENERIC_OK on success.

4.1.3.3 int Read_AtmelGenericRecord (AtmelGenericRecord * *genericRecord*, FILE * *in*)

Reads an Atmel Generic record from an opened file.

Parameters

genericRecord A pointer to the Atmel Generic record structure that will store the read record.

in A file pointer to an opened file that can be read.

Returns

ATMEL_GENERIC_OK on success, otherwise one of the ATMEL_GENERIC_ERROR_ error codes.

Return values

ATMEL_GENERIC_OK on success.

ATMEL_GENERIC_ERROR_INVALID_ARGUMENTS if the record pointer or file pointer is NULL.

ATMEL_GENERIC_ERROR_EOF if end-of-file has been reached.

ATMEL_GENERIC_ERROR_FILE if a file reading error has occured.

ATMEL_GENERIC_INVALID_RECORD if the record read is invalid (record did not match specifications).

4.1.3.4 int Write_AtmelGenericRecord (const AtmelGenericRecord * *genericRecord*, FILE * *out*)

Writes an Atmel Generic record to an opened file. Note that the Atmel Generic record only supports 24-bit addresses, so only 24-bits of the address stored in the Atmel Generic record structure that genericRecord points to will be written.

Parameters

genericRecord A pointer to the Atmel Generic record structure.

out A file pointer to an opened file that can be written to.

Returns

ATMEL_GENERIC_OK on success, otherwise one of the ATMEL_GENERIC_ERROR_ error codes.

Return values

ATMEL_GENERIC_OK on success.

ATMEL_GENERIC_ERROR_INVALID_ARGUMENTS if the record pointer or file pointer is NULL.

ATMEL_GENERIC_ERROR_FILE if a file writing error has occured.

4.2 ihex.h File Reference

Low-level utility functions to create, read, write, and print Intel HEX8 binary records.

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
```

Data Structures

- struct IHexRecord

Enumerations

- enum **_IHexDefinitions** {

  IHEX_RECORD_BUFF_SIZE = 768, **IHEX_COUNT_OFFSET** = 1, **IHEX_COUNT_LEN** = 2, **IHEX_ADDRESS_OFFSET** = 3,

  IHEX_ADDRESS_LEN = 4, **IHEX_TYPE_OFFSET** = 7, **IHEX_TYPE_LEN** = 2, **IHEX_DATA_OFFSET** = 9,

  IHEX_CHECKSUM_LEN = 2, **IHEX_MAX_DATA_LEN** = 512, **IHEX_ASCII_HEX_BYTE_LEN** = 2, **IHEX_START_CODE_OFFSET** = 0,

  IHEX_START_CODE = ':' }

- enum IHexErrors {
  IHEX_OK = 0, IHEX_ERROR_FILE = -1, IHEX_ERROR_EOF = -2, IHEX_ERROR_INVALID_RECORD = -3,
  IHEX_ERROR_INVALID_ARGUMENTS = -4, IHEX_ERROR_NEWLINE = -5 }
- enum IHexRecordTypes {
  IHEX_TYPE_00 = 0, IHEX_TYPE_01, IHEX_TYPE_02, IHEX_TYPE_03,
  IHEX_TYPE_04, IHEX_TYPE_05 }

Functions

- int New_IHexRecord (int type, uint16_t address, const uint8_t *data, int dataLen, IHexRecord *ihexRecord)
- int Read_IHexRecord (IHexRecord *ihexRecord, FILE *in)
- int Write_IHexRecord (const IHexRecord *ihexRecord, FILE *out)
- void Print_IHexRecord (const IHexRecord *ihexRecord)
- uint8_t Checksum_IHexRecord (const IHexRecord *ihexRecord)

4.2.1 Detailed Description

Low-level utility functions to create, read, write, and print Intel HEX8 binary records.

Author

Vanya A. Sergeev <vsergeev@gmail.com>

Date

February 2011

Version

1.0.5

4.2.2 Enumeration Type Documentation

4.2.2.1 enum IHexErrors

All possible error codes the Intel HEX8 record utility functions may return.

Enumerator:

IHEX_OK Error code for success or no error.

IHEX_ERROR_FILE Error code for error while reading from or writing to a file. You may check errno for the exact error if this error code is encountered.

IHEX_ERROR_EOF Error code for encountering end-of-file when reading from a file.

IHEX_ERROR_INVALID_RECORD Error code for error if an invalid record was read.

IHEX_ERROR_INVALID_ARGUMENTS Error code for error from invalid arguments passed to function.

IHEX_ERROR_NEWLINE Error code for encountering a newline with no record when reading from a file.

4.2.2.2 enum IHexRecordTypes

Intel HEX8 Record Types 00-05

Enumerator:

IHEX_TYPE_00 Data Record
IHEX_TYPE_01 End of File Record
IHEX_TYPE_02 Extended Segment Address Record
IHEX_TYPE_03 Start Segment Address Record
IHEX_TYPE_04 Extended Linear Address Record
IHEX_TYPE_05 Start Linear Address Record

4.2.3 Function Documentation

4.2.3.1 uint8_t Checksum_IHexRecord (const IHexRecord * *ihexRecord*)

Calculates the checksum of an Intel HEX8 IHexRecord structure. See the Intel HEX8 specifications for more details on the checksum calculation.

Parameters

ihexRecord A pointer to the Intel HEX8 record structure.

Returns

The 8-bit checksum.

4.2.3.2 int New_IHexRecord (int *type*, uint16_t *address*, const uint8_t * *data*, int *dataLen*, IHexRecord * *ihexRecord*)

Sets all of the record fields of an Intel HEX8 record structure.

Parameters

type The Intel HEX8 record type (integer value of 0 through 5).

address The 16-bit address of the data.

data A point to the 8-bit array of data.

dataLen The size of the 8-bit data array.

ihexRecord A pointer to the target Intel HEX8 record structure where these fields will be set.

Returns

IHEX_OK on success, otherwise one of the IHEX_ERROR_ error codes.

Return values

IHEX_OK on success.

IHEX_ERROR_INVALID_ARGUMENTS if the record pointer is NULL, or if the length of the 8-bit data array is out of range (less than zero or greater than the maximum data length allowed by record specifications, see IHexRecord.data).

4.2.3.3 void Print_IHexRecord (const IHexRecord * *ihexRecord*)

Prints the contents of an Intel HEX8 record structure to stdout. The record dump consists of the type, address, entire data array, and checksum fields of the record.

Parameters

ihexRecord A pointer to the Intel HEX8 record structure.

Returns

Always returns IHEX_OK (success).

Return values

IHEX_OK on success.

4.2.3.4 int Read_IHexRecord (IHexRecord * *ihexRecord*, FILE * *in*)

Reads an Intel HEX8 record from an opened file.

Parameters

ihexRecord A pointer to the Intel HEX8 record structure that will store the read record.

in A file pointer to an opened file that can be read.

Returns

IHEX_OK on success, otherwise one of the IHEX_ERROR_ error codes.

Return values

IHEX_OK on success.

IHEX_ERROR_INVALID_ARGUMENTS if the record pointer or file pointer is NULL.

IHEX_ERROR_EOF if end-of-file has been reached.

IHEX_ERROR_FILE if a file reading error has occured.

IHEX_INVALID_RECORD if the record read is invalid (record did not match specifications or record checksum was invalid).

4.2.3.5 int Write_IHexRecord (const IHexRecord * *ihexRecord*, FILE * *out*)

Writes an Intel HEX8 record to an opened file.

Parameters

ihexRecord A pointer to the Intel HEX8 record structure.

out A file pointer to an opened file that can be written to.

Returns

IHEX_OK on success, otherwise one of the IHEX_ERROR_ error codes.

Return values

IHEX_OK on success.

IHEX_ERROR_INVALID_ARGUMENTS if the record pointer or file pointer is NULL.

IHEX_ERROR_INVALID_RECORD if the record's data length is out of range (greater than the maximum data length allowed by record specifications, see IHexRecord.data).

IHEX_ERROR_FILE if a file writing error has occured.

4.3 srecord.h File Reference

Low-level utility functions to create, read, write, and print Motorola S-Record binary records.

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
```

Data Structures

- struct SRecord

Enumerations

- enum **_SRecordDefinitions** {

  SRECORD_RECORD_BUFF_SIZE = 768, **SRECORD_TYPE_OFFSET** = 1, **SRECORD_TYPE_LEN** = 1, **SRECORD_COUNT_OFFSET** = 2,

  SRECORD_COUNT_LEN = 2, **SRECORD_ADDRESS_OFFSET** = 4, **SRECORD_CHECKSUM_LEN** = 2, **SRECORD_MAX_DATA_LEN** = 64,

  SRECORD_MAX_ADDRESS_LEN = 8, **SRECORD_ASCII_HEX_BYTE_LEN** = 2, **SRECORD_START_CODE_OFFSET** = 0, **SRECORD_START_CODE** = 'S' }
- enum SRecordErrors {

  SRECORD_OK = 0, SRECORD_ERROR_FILE = -1, SRECORD_ERROR_EOF = -2, SRECORD_ERROR_INVALID_RECORD = -3,

  SRECORD_ERROR_INVALID_ARGUMENTS = -4, SRECORD_ERROR_NEWLINE = -5 }
- enum SRecordTypes {

  SRECORD_TYPE_S0 = 0, SRECORD_TYPE_S1, SRECORD_TYPE_S2, SRECORD_TYPE_S3,

  SRECORD_TYPE_S4, SRECORD_TYPE_S5, SRECORD_TYPE_S6, SRECORD_TYPE_S7,

  SRECORD_TYPE_S8, SRECORD_TYPE_S9 }

Functions

- int New_SRecord (int type, uint32_t address, const uint8_t *data, int dataLen, SRecord *srec)
- int Read_SRecord (SRecord *srec, FILE *in)
- int Write_SRecord (const SRecord *srec, FILE *out)
- void Print_SRecord (const SRecord *srec)
- uint8_t Checksum_SRecord (const SRecord *srec)

4.3.1 Detailed Description

Low-level utility functions to create, read, write, and print Motorola S-Record binary records.

Author

Vanya A. Sergeev <vsergeev@gmail.com>

Date

February 2011

Version

1.0.5

4.3.2 Enumeration Type Documentation

4.3.2.1 enum SRecordErrors

All possible error codes the Motorola S-Record utility functions may return.

Enumerator:

SRECORD_OK Error code for success or no error.

SRECORD_ERROR_FILE Error code for error while reading from or writing to a file. You may check errno for the exact error if this error code is encountered.

SRECORD_ERROR_EOF Error code for encountering end-of-file when reading from a file.

SRECORD_ERROR_INVALID_RECORD Error code for error if an invalid record was read.

SRECORD_ERROR_INVALID_ARGUMENTS Error code for error from invalid arguments passed to function.

SRECORD_ERROR_NEWLINE Error code for encountering a newline with no record when reading from a file.

4.3.2.2 enum SRecordTypes

Motorola S-Record Types S0-S9

Enumerator:

SRECORD_TYPE_S0 Header record, although there is an official format it is often made proprietary by third-parties. 16-bit address normally set to 0x0000 and header information is stored in the data field. This record is unnecessary and commonly not used.

SRECORD_TYPE_S1 Data record with 16-bit address

SRECORD_TYPE_S2 Data record with 24-bit address

SRECORD_TYPE_S3 Data record with 32-bit address

SRECORD_TYPE_S4 Extension by LSI Logic, Inc. See their specification for more details.

SRECORD_TYPE_S5 16-bit address field that contains the number of S1, S2, and S3 (all data) records transmitted. No data field.

SRECORD_TYPE_S6 24-bit address field that contains the number of S1, S2, and S3 (all data) records transmitted. No data field.

SRECORD_TYPE_S7 Termination record for S3 data records. 32-bit address field contains address of the entry point after the S-Record file has been processed. No data field.

SRECORD_TYPE_S8 Termination record for S2 data records. 24-bit address field contains address of the entry point after the S-Record file has been processed. No data field.

SRECORD_TYPE_S9 Termination record for S1 data records. 16-bit address field contains address of the entry point after the S-Record file has been processed. No data field.

4.3.3 Function Documentation

4.3.3.1 uint8_t Checksum_SRecord (const SRecord * *srec*)

Calculates the checksum of a Motorola S-Record SRecord structure. See the Motorola S-Record specifications for more details on the checksum calculation.

Parameters

srec A pointer to the Motorola S-Record structure.

Returns

The 8-bit checksum.

4.3.3.2 int New_SRecord (int *type*, uint32_t *address*, const uint8_t * *data*, int *dataLen*, SRecord * *srec*)

Sets all of the record fields of a Motorola S-Record structure.

Parameters

type The Motorola S-Record type (integer value of 0 through 9).

address The 32-bit address of the data. The actual size of the address (16-,24-, or 32-bits) when written to a file depends on the S-Record type.

data A pointer to the 8-bit array of data.

dataLen The size of the 8-bit data array.

srec A pointer to the target Motorola S-Record structure where these fields will be set.

Returns

SRECORD_OK on success, otherwise one of the SRECORD_ERROR_ error codes.

Return values

SRECORD_OK on success.

SRECORD_ERROR_INVALID_ARGUMENTS if the record pointer is NULL, or if the length of the 8-bit data array is out of range (less than zero or greater than the maximum data length allowed by record specifications, see SRecord.data).

4.3.3.3 void Print_SRecord (const SRecord * *srec*)

Prints the contents of a Motorola S-Record structure to stdout. The record dump consists of the type, address, entire data array, and checksum fields of the record.

Parameters

srec A pointer to the Motorola S-Record structure.

Returns

Always returns SRECORD_OK (success).

Return values

SRECORD_OK on success.

4.3.3.4 int Read_SRecord (SRecord * *srec*, FILE * *in*)

Reads a Motorola S-Record record from an opened file.

Parameters

srec A pointer to the Motorola S-Record structure that will store the read record.

in A file pointer to an opened file that can be read.

Returns

SRECORD_OK on success, otherwise one of the SRECORD_ERROR_ error codes.

Return values

SRECORD_OK on success.

SRECORD_ERROR_INVALID_ARGUMENTS if the record pointer or file pointer is NULL.

SRECORD_ERROR_EOF if end-of-file has been reached.

SRECORD_ERROR_FILE if a file reading error has occured.

SRECORD_INVALID_RECORD if the record read is invalid (record did not match specifications or record checksum was invalid).

4.3.3.5 int Write_SRecord (const SRecord * *srec*, FILE * *out*)

Writes a Motorola S-Record to an opened file.

Parameters

srec A pointer to the Motorola S-Record structure.

out A file pointer to an opened file that can be written to.

Returns

SRECORD_OK on success, otherwise one of the SRECORD_ERROR_ error codes.

Return values

SRECORD_OK on success.

SRECORD_ERROR_INVALID_ARGUMENTS if the record pointer or file pointer is NULL.

SRECORD_ERROR_INVALID_RECORD if the record's data length (the SRecord.dataLen variable of the record) is out of range (greater than the maximum data length allowed by record specifications, see SRecord.data).

SRECORD_ERROR_FILE if a file writing error has occured.

Index