Intel Hex Encoder/Decoder Class

Generated by Doxygen 1.7.1

Thu Mar 1 2012 23:43:48
## Contents

1 Main Page ................................................. 1
   1.1 Introduction ........................................ 1
   1.2 Contact Information ............................... 1
   1.3 Licensing Information ......................... 1
   1.4 Image Information ............................... 2

2 Class Index ............................................. 3
   2.1 Class List .......................................... 3

3 File Index ................................................ 5
   3.1 File List ........................................... 5

4 Class Documentation .................................... 7
   4.1 intelhex Class Reference ....................... 7
      4.1.1 Detailed Description ....................... 11
      4.1.2 Constructor & Destructor Documentation ... 11
         4.1.2.1 intelhex ............................... 11
         4.1.2.2 ~intelhex ............................. 11
         4.1.2.3 intelhex ............................... 11
      4.1.3 Member Function Documentation ........... 12
         4.1.3.1 addError .............................. 12
         4.1.3.2 addWarning ........................... 12
         4.1.3.3 begin ................................. 12
         4.1.3.4 blankFill ............................. 13
         4.1.3.5 blankFill ............................. 13
         4.1.3.6 blankFill ............................. 13
         4.1.3.7 blankFillAddressLowByte ............ 13
         4.1.3.8 blankFillAddressLowByte .......... 13
         4.1.3.9 blankFillRandom ....................... 13
4.1.3.10 blankFillRandom ........................................... 13
4.1.3.11 currentAddress ........................................ 13
4.1.3.12 decodeDataRecord ..................................... 13
4.1.3.13 end ....................................................... 13
4.1.3.14 endAddress .............................................. 14
4.1.3.15 getData .................................................. 14
4.1.3.16 getData .................................................. 14
4.1.3.17 getNoErrors ............................................ 14
4.1.3.18 getNoWarnings ......................................... 14
4.1.3.19 getStartLinearAddress ................................. 15
4.1.3.20 getStartSegmentAddress ............................... 15
4.1.3.21 insertData .............................................. 15
4.1.3.22 insertData .............................................. 15
4.1.3.23 jumpTo .................................................. 16
4.1.3.24 linearAddressingOn ................................... 16
4.1.3.25 operator= .............................................. 16
4.1.3.26 overwriteData ......................................... 16
4.1.3.27 overwriteData ......................................... 16
4.1.3.28 popNextError ......................................... 16
4.1.3.29 popNextWarning ...................................... 17
4.1.3.30 segmentAddressingOn ................................ 17
4.1.3.31 setStartLinearAddress ................................. 17
4.1.3.32 setStartSegmentAddress ............................... 18
4.1.3.33 startAddress ........................................... 18
4.1.3.34 stringToHex .......................................... 18
4.1.3.35 ucToHexString ....................................... 18
4.1.3.36 ulToHexString ....................................... 19
4.1.3.37 ulToString ........................................... 19
4.1.3.38 verboseOff ............................................ 19
4.1.3.39 verboseOn .............................................. 19

4.1.4 Friends And Related Function Documentation .................. 20
4.1.4.1 operator<< ............................................. 20
4.1.4.2 operator>> ............................................. 20

4.1.5 Member Data Documentation .................................. 20
4.1.5.1 csRegister ............................................ 20
4.1.5.2 eipRegister ............................................ 20
CONTENTS

4.1.5.3 exists ......................................................... 21
4.1.5.4 foundEof ..................................................... 21
4.1.5.5 ihContent ................................................... 21
4.1.5.6 ihErrors ..................................................... 21
4.1.5.7 ihIterator ................................................... 21
4.1.5.8 ihReturn ..................................................... 21
4.1.5.9 ihWarnings .................................................. 21
4.1.5.10 ipRegister .................................................. 21
4.1.5.11 msgError ................................................... 22
4.1.5.12 msgWarning ............................................... 22
4.1.5.13 noOfErrors ................................................ 22
4.1.5.14 noOfWarnings ............................................. 22
4.1.5.15 segmentAddressMode .................................... 22
4.1.5.16 segmentBaseAddress ................................... 22
4.1.5.17 startLinearAddress ..................................... 23
4.1.5.18 startSegmentAddress ................................... 23
4.1.5.19 verbose ................................................... 23

5 File Documentation ................................................. 25

5.1 intelhexclass.cpp File Reference ................................. 25
  5.1.1 Detailed Description ....................................... 26
  5.1.2 Enumeration Type Documentation ........................... 26
    5.1.2.1 intelhexRecordType .................................. 26
  5.1.3 Function Documentation .................................... 26
    5.1.3.1 operator<< ........................................... 26
    5.1.3.2 operator>> ........................................... 27

5.2 intelhexclass.h File Reference ................................ 27
  5.2.1 Detailed Description ....................................... 28
Chapter 1

Main Page

1.1 Introduction

The Intel HEX File class module is designed to encode, decode and manipulate the content of Intel HEX format files commonly generated by most toolchains for embedded processors and microcontrollers.

It uses standard C++ streams to decode files and store them in memory, and encode data stored in memory back into an Intel HEX format file. Once the file content is in memory, the content can then be manipulated using the available API.

With this class it is possible to create tools that can compare Intel HEX files, fill empty space with desired values, splice two or more files together to name a few possibilities.

1.2 Contact Information

For more information and the latest release, please visit this projects home page at http://codinghead.github.com/Intel-HEX-Class To participate in the project or for other enquiries, please contact Stuart Cording at codinghead@gmail.com

1.3 Licensing Information

Copyright (c) 2012 Stuart Cording
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.4 Image Information

Image chosen for this project comes from 'Henkster'. Original image is from http://www.sxc.hu/photo/504350 on stock.xchng.

Author

Stuart Cording aka CODINGHEAD

Note

No notes to date (19th Jan 2012)
Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

intelhex (Class to decode, encode and manipulate Intel HEX format files) . . . . . . . . . . . 7
Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

intelhexclass.cpp ....................................................... 25
intelhexclass.h .......................................................... 27
Chapter 4

Class Documentation

4.1 intelhex Class Reference

Class to decode, encode and manipulate Intel HEX format files.

#include <intelhexclass.h>

Collaboration diagram for intelhex:

Public Member Functions

- intelhex ()
intelhex Class Constructor.

• ~intelhex ()
  intelhex Class Deconstructor.

• intelhex (const intelhex &ihSource)
  intelhex Class Copy Constructor.

• intelhex & operator= (const intelhex &ihSource)
  intelhex Class Assignment Operator.

• void begin ()
  Moves the address pointer to the first available address.

• void end ()
  Moves the address pointer to the last available address.

• void jumpTo (unsigned long address)
  Moves the address pointer to the desired address.

• unsigned long currentAddress ()
  Returns the current segment base address.

• bool startAddress (unsigned long *address)
  Returns the lowest address currently available.

• bool endAddress (unsigned long *address)
  Returns the highest address currently available.

• bool getData (unsigned char *data)
• bool getData (unsigned char *data, unsigned long address)
• bool insertData (unsigned char data)
• bool insertData (unsigned char data, unsigned long address)
• void overwriteData (unsigned char data)
• void overwriteData (unsigned char data, unsigned long address)
• bool blankFill (unsigned char data)
• bool blankFill (unsigned char const data, unsigned long sizeOfData)
• bool blankFill (unsigned char const data, unsigned long sizeOfData, unsigned long endAddress)
• bool blankFillRandom ()
• void blankFillRandom (unsigned long endAddress)
• bool blankFillAddressLowByte ()
• void blankFillAddressLowByte (unsigned long endAddress)
• unsigned long getNoWarnings ()
  Returns number of unread warning messages.

• unsigned long getNoErrors ()
  Returns number of unread error messages.
• bool popNextWarning (string &warning)
  Pop next warning message from the list of warnings.

• bool popNextError (string &error)
  Pop next error message from the list of errors.

• bool getStartSegmentAddress (unsigned short *ipRegister, unsigned short *csRegister)
  Returns segment start address for the IP and ES registers.

• bool getStartLinearAddress (unsigned long *eipRegister)
  Returns segment linear address for the EIP register.

• void setStartSegmentAddress (unsigned short ipRegister, unsigned short csRegister)
  Sets the segment start address for the IP and CS registers.

• void setStartLinearAddress (unsigned long eipRegister)
  Sets the segment start address for the EIP register.

• void segmentAddressingOn ()
  Turns on segment addressing mode during encoding.

• void linearAddressingOn ()
  Turns on linear addressing mode during encoding.

• void verboseOn ()
  Turns on textual output to cout during decoding.

• void verboseOff ()
  Turns off textual output to cout during decoding.

Private Member Functions

• unsigned char stringToHex (string value)
• string ulToHexString (unsigned long value)
• string ucToHexString (unsigned char value)
  Converts an unsigned char to a string in HEX format.

• string ulToString (unsigned long value)
  Converts an unsigned long to a string in DEC format.

• void decodeDataRecord (unsigned char recordLength, unsigned long loadOffset, string::const_iterator data)
  Decodes the data content of a data record.

• void addWarning (string warningMessage)
  Add a warning message to the warning message list.

• void addError (string errorMessage)
  Add an error message to the error message list.
Private Attributes

• map<unsigned long, unsigned char> ihContent
  Container for decoded Intel HEX content.

• map<unsigned long, unsigned char>::iterator ihIterator
  Iterator for the container holding the decoded Intel HEX content.

• pair<map<unsigned long, unsigned char>::iterator, bool> ihReturn
  Pair for the container holding the decoded Intel HEX content.

• unsigned long segmentBaseAddress
  Stores segment base address of Intel HEX file.

• struct {
    unsigned short csRegister
    unsigned short ipRegister
    bool exists
  } startSegmentAddress

  Stores the content of the CS/IP Registers, if used.

• struct {
    unsigned long eipRegister
    bool exists
  } startLinearAddress

  Stores the content of the EIP Register, if used.

• struct {
    list<string> ihWarnings
    unsigned long noOfWarnings
  } msgWarning

  Structure to hold warning messages.

• struct {
    list<string> ihErrors
    unsigned long noOfErrors
  } msgError

  Structure to hold error messages.

• bool foundEof
  Note that EOF record is found.

• bool verbose
  Select verbose mode.

• bool segmentAddressMode
  Select segment address mode.
4.1 intelhex Class Reference

Friends

- ostream & operator<< (ostream &dataOut, intelhex &ihLocal)
  
  Output stream overload operator.

- istream & operator>> (istream &dataIn, intelhex &ihLocal)
  
  Input stream overload operator.

4.1.1 Detailed Description

Class to decode, encode and manipulate Intel HEX format files. The Intel HEX class allows the user to stream in the content of an Intel HEX file so that its content can be analysed more easily than trying to decode the Intel HEX file in a text editor. In conjunction with a suitable application it is possible to create content, analyse content and even compare the content of files with one another.

Definition at line 82 of file intelhexclass.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 intelhex::intelhex ( ) [inline]

intelhex Class Constructor.

Important initialisation steps performed here:

- clear segment base address to zero
- clear all x86 start address registers to zero
- note that there are, as yet, no errors or warnings
- note that the EOF record has not yet been found
- set verbode mode to ’false’ (default)

Definition at line 371 of file intelhexclass.h.

4.1.2.2 intelhex::~intelhex ( ) [inline]

intelhex Class Deconstructor.

Currently the deconstructor is intentially empty.

Definition at line 398 of file intelhexclass.h.

4.1.2.3 intelhex::intelhex ( const intelhex & ihSource ) [inline]

intelhex Class Copy Constructor.

Currently the copy constructor is intentially empty.

Definition at line 408 of file intelhexclass.h.
4.1.3 Member Function Documentation

4.1.3.1 void intelhex::addError ( string errorMessage ) [private]

Add an error message to the error message list.

Parameters

errorMessage - the text to be added for this error

Definition at line 223 of file intelhexclass.cpp.

4.1.3.2 void intelhex::addWarning ( string warningMessage ) [private]

Add a warning message to the warning message list.

Parameters

warningMessage - the text to be added for this warning

Definition at line 206 of file intelhexclass.cpp.

4.1.3.3 void intelhex::begin ( ) [inline]

Moves the address pointer to the first available address.

The address pointer will be moved to the first available address in memory of the decoded file or of the data the user has inserted into memory for the purpose of encoding into the Intel HEX format.

See also

defined()
4.1.3.4  bool intelhex::blankFill ( unsigned char  data  )
4.1.3.5  bool intelhex::blankFill ( unsigned char ∗const  data, unsigned long  sizeOfData  )
4.1.3.6  void intelhex::blankFill ( unsigned char ∗const  data, unsigned long  sizeOfData, unsigned long  endAddress  )
4.1.3.7  bool intelhex::blankFillAddressLowByte ( )
4.1.3.8  void intelhex::blankFillAddressLowByte ( unsigned long  endAddress  )
4.1.3.9  void intelhex::blankFillRandom ( unsigned long  endAddress  )
4.1.3.10 bool intelhex::blankFillRandom ( )
4.1.3.11 unsigned long intelhex::currentAddress ( ) [inline]

Returns the current segment base address.
Current address will be returned.

See also
   
   jumpTo()

Return values

Current  address being pointed to.

Definition at line 550 of file intelhexclass.h.

4.1.3.12 void intelhex::decodeDataRecord ( unsigned char  recordLength, unsigned long  loadOffset, string::const_iterator  data  ) [private]

Decodes the data content of a data record.
Takes the data element of a data record in string format, converts each 2 char element into a single byte and
then inserts that byte of data into the ihContent STL map.

See also
   
   encodeDataRecord()

Parameters

recordLength  - Number of bytes in this record as extracted from this line in the Intel HEX file
loadOffset  - The offset from the segment base address for the first byte in this record
data  - The data content of the record in a string

Definition at line 240 of file intelhexclass.cpp.

4.1.3.13 void intelhex::end ( ) [inline]

Moves the address pointer to the last available address.
The address pointer will be moved to the last available address in memory of the decoded file or of the data the user has inserted into memory for the purpose of encoding into the Intel HEX format.

See also

begin()

Note

This function has no effect if no file has been as yet decoded and no data has been inserted into memory.

Definition at line 517 of file intelhexclass.h.

4.1.3.14 bool intelhex::endAddress ( unsigned long ∗ address ) [inline]

Returns the highest address currently available.

Returns the last address that appears in the memory if there is data present. If not, no value will be returned.

Parameters

address - variable to hold address requested

Return values

true - address existed and returned value is valid
false - address did not exist and returned valid is not valid

See also

startAddress()

Definition at line 597 of file intelhexclass.h.

4.1.3.15 bool intelhex::getData ( unsigned char ∗ data, unsigned long address )

4.1.3.16 bool intelhex::getData ( unsigned char ∗ data )

4.1.3.17 unsigned long intelhex::getNoErrors ( ) [inline]

Returns number of unread error messages.

Number of unread error messages will be returned.

See also

popNextWarning(), getNoWarnings(), popNextError()

Definition at line 663 of file intelhexclass.h.

4.1.3.18 unsigned long intelhex::getNoWarnings ( ) [inline]

Returns number of unread warning messages.

Number of unread warning messages will be returned.
See also

`popNextWarning()`, `getNoErrors()`, `popNextError()`

Definition at line 651 of file `intelhexclass.h`.

### 4.1.3.19 **bool intelhex::getStartLinearAddress ( unsigned long ∗ eipRegister ) [inline]**

Returns segment linear address for the EIP register.

If this value exists, they will be returned. If not, the function returns false.

**Parameters**

- `eipRegister` - variable to store EIP register’s value

**Return values**

- `true` - EIP register has defined value
- `false` - EIP register do not contain value

See also

`getStartSegmentAddress()`, `setStartSegmentAddress()`, `setStartLinearAddress()`

Definition at line 771 of file `intelhexclass.h`.

### 4.1.3.20 **bool intelhex::getStartSegmentAddress ( unsigned short ∗ ipRegister, unsigned short ∗ csRegister ) [inline]**

Returns segment start address for the IP and ES registers.

If these values exist, they will be returned. If not, the function returns false.

**Parameters**

- `ipRegister` - variable to store IP register’s value
- `csRegister` - variable to store CS register’s value

**Return values**

- `true` - IP and CS registers have defined values
- `false` - IP and CS registers do not contain values

See also

`getStartLinearAddress()`, `setStartSegmentAddress()`, `setStartLinearAddress()`

Definition at line 745 of file `intelhexclass.h`.

### 4.1.3.21 **bool intelhex::insertData ( unsigned char data, unsigned long address )**

### 4.1.3.22 **bool intelhex::insertData ( unsigned char data )**

Inserts desired byte at the current address pointer.

Inserts byte of data at the current address pointer.
Parameters

    data - data byte to be inserted

See also

    startAddress()

4.1.3.23 void intelhex::jumpTo ( unsigned long address ) [inline]

Moves the address pointer to the desired address.
Address pointer will take on the requested address.

See also

    currentAddress()

Parameters

    address - Desired new address for the address pointer

Definition at line 536 of file intelhexclass.h.

4.1.3.24 void intelhex::linearAddressingOn ( ) [inline]

Turns on linear addressing mode during encoding.
Uses the Linear Address Record during encoding.
Definition at line 832 of file intelhexclass.h.

4.1.3.25 intelhex& intelhex::operator= ( const intelhex & ihSource ) [inline]

intelhex Class Assignment Operator.

Implements the assignment operator so that the content of the Intel HEX file in memory can be copied
to another `intelhex` variable. You may want to keep a copy of the original data in memory and only
manipulate a copy-

Parameters

    ihSource - intelhex variable to be assigned to new variable

Return values

    pointer to variable to which value is to be assigned

Definition at line 448 of file intelhexclass.h.

4.1.3.26 void intelhex::overwriteData ( unsigned char data )

4.1.3.27 void intelhex::overwriteData ( unsigned char data, unsigned long address )

4.1.3.28 bool intelhex::popNextError ( string & error ) [inline]

Pop next error message from the list of errors.
Next error message is returned from the list of errors. If there are no more errors in the list, no string will be returned unchanged.

Parameters

- **error** - variable to store error string to be returned

Return values

- **true** - more error messages are available
- **false** - no more error messages are available

See also

getNoWarnings(), getNoErrors(), popNextError()

Definition at line 712 of file intelhexclass.h.

---

### 4.1.3.29 bool intelhex::popNextWarning ( string & warning ) [inline]

Pop next warning message from the list of warnings.

Next warning message is returned from the list of warnings. If there are no more warning in the list, the string will be unchanged.

Parameters

- **warning** - variable to store warning string to be returned

Return values

- **true** - more warning messages are available
- **false** - no more warning messages are available

See also

getNoWarnings(), getNoErrors(), popNextError()

Definition at line 681 of file intelhexclass.h.

---

### 4.1.3.30 void intelhex::segmentAddressingOn ( ) [inline]

Turns on segment addressing mode during encoding.

Uses the Segment Address Record during encoding.

Definition at line 822 of file intelhexclass.h.

---

### 4.1.3.31 void intelhex::setStartLinearAddress ( unsigned long eipRegister ) [inline]

Sets the segment start address for the EIP register.

Allows user to define or redefine the contents of the EIP register

Parameters

- **eipRegister** - desired EIP register value
4.1.3.32 void intelhex::setStartSegmentAddress ( unsigned short ipRegister, unsigned short csRegister ) [inline]

Sets the segment start address for the IP and CS registers. Allows user to define or redefine the contents of the IP and CS registers

Parameters

- ipRegister - desired IP register value
- csRegister - desired CS register value

See also

getStartLinearAddress(), getStartSegmentAddress(), setStartLinearAddress()

Definition at line 793 of file intelhexclass.h.

4.1.3.33 bool intelhex::startAddress ( unsigned long * address ) [inline]

Returns the lowest address currently available. Returns the first address that appears in the memory if there is data present. If not, no value will be returned.

See also

deadAddress()

Parameters

- address - variable to hold address requested

Return values

- true - address existed and returned value is valid
- false - address did not exist and returned valid is not valid

Definition at line 569 of file intelhexclass.h.

4.1.3.34 unsigned char intelhex::stringToHex ( string value ) [private]

Definition at line 92 of file intelhexclass.cpp.

4.1.3.35 string intelhex::ucToHexString ( unsigned char value ) [private]

Converts an unsigned char to a string in HEX format. Takes the received parameter and converts it into its equivalent value represented in ASCII and formatted in hexadecimal. Return value is a 2 character long string, prefaced with '0' where necessary.
Parameters

value - a value between 0x00 and 0xFF

Return values

- 2-character long string

Note

Alpha characters are capitalised.

See also

stringToHex(), ulToHexString(), ulToString()

definition at line 189 of file intelhexclass.cpp.

4.1.3.36 string intelhex::ulToHexString ( unsigned long value ) [private]

definition at line 155 of file intelhexclass.cpp.

4.1.3.37 string intelhex::ulToString ( unsigned long value ) [private]

Converts an unsigned long to a string in DEC format.

Takes the received parameter and converts it into its equivalent value represented in ASCII and formatted in
decimal. Return value will never be longer than a 48 character long string.

Parameters

value - value to be converted

Return values

- ASCII string representation of value

See also

stringToHex(), ulToHexString(), ucToHexString()

definition at line 172 of file intelhexclass.cpp.

4.1.3.38 void intelhex::verboseOff ( ) [inline]

Turns off textual output to cout during decoding.

No output to cout during decoding of Intel HEX files.

definition at line 853 of file intelhexclass.h.

4.1.3.39 void intelhex::verboseOn ( ) [inline]

Turns on textual output to cout during decoding.

Per record single line output to cout during decoding of Intel HEX files.

definition at line 843 of file intelhexclass.h.
4.1.4 Friends And Related Function Documentation

4.1.4.1 ostream& operator<<( ostream & dataOut, intelhex & ihLocal ) [friend]

Output stream overload operator.
Operator overloaded to encode any data held in memory into the Intel HEX format for storage on disk

See also
   operator>>(()

Parameters
   dataOut - Output stream for to store the decoded file information
   ihLocal - Points to this class so that friend function has access to private class members

Return values
   - pointer to output stream

Definition at line 814 of file intelhexclass.cpp.

4.1.4.2 istream& operator>>( istream & dataIn, intelhex & ihLocal ) [friend]

Input stream overload operator.
Operator overloaded to decode data streamed in from a file in the Intel HEX format into memory

See also
   operator<<()

Parameters
   dataIn - Input stream for the encoded file information
   ihLocal - Points to this class so that friend function has access to private class members

Return values
   - pointer to input stream

Definition at line 304 of file intelhexclass.cpp.

4.1.5 Member Data Documentation

4.1.5.1 unsigned short intelhex::csRegister

Definition at line 170 of file intelhexclass.h.

4.1.5.2 unsigned long intelhex::eipRegister

Definition at line 188 of file intelhexclass.h.
4.1 intelhex Class Reference

4.1.5.3 bool intelhex::exists

Definition at line 172 of file intelhexclass.h.

4.1.5.4 bool intelhex::foundEof [private]

Note that EOF record is found.
Used to note that the EOF record was found in order to ensure that it doesn’t appear twice during encoding.
Definition at line 229 of file intelhexclass.h.

4.1.5.5 map<unsigned long, unsigned char> intelhex::ihContent [private]

Container for decoded Intel HEX content.
STL map holding the addresses found in the Intel HEX file and the associated data byte stored at that address
Definition at line 125 of file intelhexclass.h.

4.1.5.6 list<string> intelhex::ihErrors

Definition at line 219 of file intelhexclass.h.

4.1.5.7 map<unsigned long, unsigned char>::iterator intelhex::ihIterator [private]

Iterator for the container holding the decoded Intel HEX content.
Definition at line 131 of file intelhexclass.h.

4.1.5.8 pair<map<unsigned long, unsigned char>::iterator,bool> intelhex::ihReturn [private]

Pair for the container holding the decoded Intel HEX content.
This is used to acquire the result of an attempt to insert new data into ihContent. Since the ihContent is a map STL, it can’t allow data to be assigned to the same address more than once. In this way we can ensure that no address in a file is falsely assigned data more than once.
Definition at line 142 of file intelhexclass.h.

4.1.5.9 list<string> intelhex::ihWarnings

Definition at line 204 of file intelhexclass.h.

4.1.5.10 unsigned short intelhex::ipRegister

Definition at line 171 of file intelhexclass.h.
4.1.5.11  struct { … } intelhex::msgError  [private]

Structure to hold error messages.
Holds error messages generated during encoding/decoding process and number of messages currently present in system

Parameters

   ihErrors  - list of error messages as strings
   noOferrors  - no of error messages still present in the list

4.1.5.12  struct { … } intelhex::msgWarning  [private]

Structure to hold warning messages.
Holds warning messages generated during encoding/decoding process and number of messages currently present in system

Parameters

   ihWarnings  - list of warning messages as strings
   noOfWarnings  - no of warning messages still present in the list

4.1.5.13  unsigned long intelhex::noOfErrors

Definition at line 220 of file intelhexclass.h.

4.1.5.14  unsigned long intelhex::noOfWarnings

Definition at line 205 of file intelhexclass.h.

4.1.5.15  bool intelhex::segmentAddressMode  [private]

Select segment address mode.
If true, use the segment addressing mode when encoding files. otherwise the default linear address mode will be used. Please refer to Intel’s Hexadecimal Object File Format Specifiation for further information.
Definition at line 247 of file intelhexclass.h.

4.1.5.16  unsigned long intelhex::segmentBaseAddress  [private]

Stores segment base address of Intel HEX file.
The segment base address is a 32-bit address to which the current load offset (as found in a Data Record line of the Intel HEX file) is added to calculate the actual address of the data. The Data Records can only point to a 64kByte address, so the segment base address expands the addressing to 4GB. This variable always holds the last address accessed.
Definition at line 154 of file IntelHexClass.h.
4.1 intelhex Class Reference

4.1.5.17  struct { ... } intelhex::startLinearAddress  [private]

Stores the content of the EIP Register, if used.

Used to store the content of the EIP Register for HEX files created for x386 Intel processors. This information is retrieved from the Start Linear Address Record. The found element defines if this register holds valid data or not.

Parameters

eipRegister  - content of the EIP register
exists  - defines if a value for the above register has been written (true) or not (false)

4.1.5.18  struct { ... } intelhex::startSegmentAddress  [private]

Stores the content of the CS/IP Registers, if used.

Used to store the content of the CS and IS Register for HEX files created for x286 or earlier Intel processors. This information is retrieved from the Start Segment Address Record. The found element defines if these registers hold valid data or not.

Parameters

csRegister  - content of the CS register
ipRegister  - content of the IP register
exists  - defines if values for the above registers have been written (true) or not (false)

4.1.5.19  bool intelhex::verbose  [private]

Select verbose mode.

Used during development to display messages as the incoming data stream is decoded

Definition at line 237 of file intelhexclass.h.

The documentation for this class was generated from the following files:

- intelhexclass.h
- intelhexclass.cpp
Chapter 5

File Documentation

5.1 intelhexclass.cpp File Reference

#include <iostream>
#include <string>
#include <vector>
#include <cstdio>
#include "intelhexclass.h"

Include dependency graph for intelhexclass.cpp:

Enumerations

- enum intelhexRecordType {
  DATA_RECORD, END_OF_FILE_RECORD, EXTENDED_SEGMENT_ADDRESS, START_-
Functions

- istream & operator\texttt{\textgreater\textgreater} (istream &dataIn, intelhex &ihLocal)
- ostream & operator\texttt{\textless\textless} (ostream &dataOut, intelhex &ihLocal)

5.1.1 Detailed Description

Definition in file intelhexclass.cpp.

5.1.2 Enumeration Type Documentation

5.1.2.1 enum intelhexRecordType

Possible record types for Intel HEX file.

List of all possible record types that can be found in an Intel HEX file.

Enumerator:

\begin{verbatim}
DATA_RECORD
END_OF_FILE_RECORD
EXTENDED_SEGMENT_ADDRESS
START_SEGMENT_ADDRESS
EXTENDED_LINEAR_ADDRESS
START_LINEAR_ADDRESS
NO_OF_RECORD_TYPES
\end{verbatim}

Definition at line 79 of file intelhexclass.cpp.

5.1.3 Function Documentation

5.1.3.1 ostream& operator\texttt{\textless\textless} ( ostream & dataOut, intelhex & ihLocal)

Operator overloaded to encode any data held in memory into the Intel HEX format for storage on disk.

See also

operator\texttt{\textgreater\textgreater}()

Parameters

- \texttt{dataOut} - Output stream for to store the decoded file information
- \texttt{ihLocal} - Points to this class so that friend function has access to private class members

Return values

- pointer to output stream

Definition at line 814 of file intelhexclass.cpp.
5.2 intelhexclass.h File Reference

5.1.3.2 istream\& operator\triangleright\triangleright ( istream \& dataIn, intelhex \& ihLocal )

Operator overloaded to decode data streamed in from a file in the Intel HEX format into memory

See also

operator\triangleright\triangleright()

Parameters

\texttt{dataIn} - Input stream for the encoded file information

\texttt{ihLocal} - Points to this class so that friend function has access to private class members

Return values

- pointer to input stream

Definition at line 304 of file intelhexclass.cpp.

5.2 intelhexclass.h File Reference

\#include \textless \texttt{iostream}\textgreater
\#include \textless \texttt{map}\textgreater
\#include \textless \texttt{list}\textgreater

Include dependency graph for intelhexclass.h:
This graph shows which files directly or indirectly include this file:

![Graph showing dependencies between files]

### Classes

- **class intelhex**
  
  *Class to decode, encode and manipulate Intel HEX format files.*

### 5.2.1 Detailed Description

#### Author

Stuart Cording aka CODINGHEAD

A class to handle the encoding, decoding and manipulation of an Intel HEX format file as generated by many tool chains for embedded processors and microcontrollers.

This class is constructed based upon the definition given in the document 'Hexadecimal Object File Format Specification', Revision A, January 6, 1988, © 1998 Intel Corporation.

#### Note

See the git versioning notes for version information

Definition in file `intelhexclass.h`. 
Index

∼intelhex  
   intelhex, 11

addError  
   intelhex, 12

addWarning  
   intelhex, 12

begin  
   intelhex, 12

blankFill  
   intelhex, 12, 13

blankFillAddressLowByte  
   intelhex, 13

blankFillRandom  
   intelhex, 13

csRegister  
   intelhex, 20

currentAddress  
   intelhex, 13

DATA_RECORD  
   intelhexclass.cpp, 26

decodeDataRecord  
   intelhex, 13

eipRegister  
   intelhex, 20

derm  
   intelhex, 13

END_OF_FILE_RECORD  
   intelhexclass.cpp, 26

dataAddress  
   intelhex, 14

eXists  
   intelhex, 20

EXTENDED_LINEAR_ADDRESS  
   intelhexclass.cpp, 26

EXTENDED_SEGMENT_ADDRESS  
   intelhexclass.cpp, 26

foundEof  
   intelhex, 21

getData  
   intelhex, 14

gtNoErrors  
   intelhex, 14

getNoWarnings  
   intelhex, 14

getStartLinearAddress  
   intelhex, 15

getStartSegmentAddress  
   intelhex, 15

ihContent  
   intelhex, 21

ihErrors  
   intelhex, 21

ihIterator  
   intelhex, 21

ihReturn  
   intelhex, 21

ihWarnings  
   intelhex, 21

insertData  
   intelhex, 15

intelhex, 7

∼intelhex, 11

addError, 12

addWarning, 12

begin, 12

blankFill, 12, 13

blankFillAddressLowByte, 13

blankFillRandom, 13

csRegister, 20

currentAddress, 13

decodeDataRecord, 13

eipRegister, 20

derm, 13

dataAddress, 14

exists, 20

foundEof, 21

getData, 14

gtNoErrors, 14

getNoWarnings, 14

getStartLinearAddress, 15

getStartSegmentAddress, 15

ihContent, 21

ihErrors, 21
ihIterator, 21
ihReturn, 21
ihWarnings, 21
insertData, 15
intelhex, 11
ipRegister, 21
jumpTo, 16
linearAddressingOn, 16
msgError, 21
msgWarning, 22
noOfErrors, 22
noOfWarnings, 22
operator<<, 20
operator>>, 20
operator=, 16
overwriteData, 16
popNextError, 16
popNextWarning, 17
segmentAddressingOn, 17
segmentAddressMode, 22
segmentBaseAddress, 22
setStartLinearAddress, 17
setStartSegmentAddress, 18
startAddress, 18
startLinearAddress, 22
startSegmentAddress, 23
stringToHex, 18
ucToHexString, 18
ulToHexString, 19
ulToString, 19
verbose, 23
verboseOff, 19
verboseOn, 19
intelhexclass.cpp, 25
DATA_RECORD, 26
END_OF_FILE_RECORD, 26
EXTENDED_LINEAR_ADDRESS, 26
EXTENDED_SEGMENT_ADDRESS, 26
intelhexRecordType, 26
NO_OF_RECORD_TYPES, 26
operator<<, 26
operator>>, 26
START_LINEAR_ADDRESS, 26
START_SEGMENT_ADDRESS, 26
intelhexclass.h, 27
intelhexRecordType
intelhexclass.cpp, 26
ipRegister
intelhex, 21
jumpTo
intelhex, 16
linearAddressingOn
intelhex, 16
msgError
intelhex, 21
msgWarning
intelhex, 22
NO_OF_RECORDTYPES
intelhexclass.cpp, 26
noOfErrors
intelhex, 22
noOfWarnings
intelhex, 22
operator<<
intelhex, 20
intelhexclass.cpp, 26
operator>>
intelhex, 20
intelhexclass.cpp, 26
operator=
intelhex, 16
overwriteData
intelhex, 16
popNextError
intelhex, 16
popNextWarning
intelhex, 17
segmentAddressingOn
intelhex, 17
segmentAddressMode
intelhex, 22
segmentBaseAddress
intelhex, 22
setStartLinearAddress
intelhex, 17
setStartSegmentAddress
intelhex, 18
START_LINEAR_ADDRESS
intelhexclass.cpp, 26
START_SEGMENT_ADDRESS
intelhexclass.cpp, 26
startAddress
intelhex, 18
startLinearAddress
intelhex, 22
startSegmentAddress
intelhex, 23
stringToHex
intelhex, 18
ucToHexString
intelhex, 18
ulToHexString
    intelhex, 19
ulToString
    intelhex, 19

verbose
    intelhex, 23
verboseOff
    intelhex, 19
verboseOn
    intelhex, 19