

ALHexManip 1.3 Usage - Short Overview Examples

SUBSTITUTE	File_in1	WITH	..\..\..\..\Bin\L7Bw02temp.hex	
SUBSTITUTE	File_out1	WITH	..\..\..\..\Bin\L7Bw02temp.app	
NEW_BUFFER	Buf_A000_to_AFFF	FROM_FILE	File_in1	-useBufferRange={0x000, 0x34} -setByteOrder=LittleEndian -useAreas={{0x00, 0x34}} -changeStartAddressTo=0x0000A000
-- other Keywords (all options from above can be used) --				
NEW_BUFFER	Buf_File1	FROM_FILE_BINARY	File_in1	same options possible as above
NEW_BUFFER	Buf_B000_to_BFFF	FROM_BUFFER	Buf_A000_to_AFFF	same options possible as above
NEW_BUFFER	Buf_C000_to_DFFF	FROM_BUFFERS	{Buf_A000_to_AFFF, Buf_B000_to_BFFF}	same options possible as above
NEW_BUFFER	Buf_C000_to_C002	FROM_VALUES	{0x80, 0x90, 0x00}	same options possible as above
NEW_BUFFER	Buf_F000_to_FFFF	EMPTY		same options possible as above
NEW_CRC	MyCRC	FROM_BUFFER	Buf_B000_to_BFFF	-setType=Crc8_OnesComplement -useAreas={{0xB000, 0x5}}
-- other Keywords (all options from above can be used) --				
NEW_CRC	MyCRC2	FROM_BUFFERS	{Buf_B000_to_BFFF, Buf_A000_to_AFFF}	same options possible as above
NEW_HASH	-- other Keywords (similar to CRC) --			
FILL_BUFFER	Buf_C000_to_DFFF	WITH_FILLER	{0x80, 0x90, 0x00}	-useAreas={{0xC111, 0x40}}
FILL_BUFFER	Buf_C000_to_DFFF	WITH_VALUES	{0x80, 0x90, 0x00}	-useAreas={{0xC111, 0x40}}
FILL_BUFFER	Buf_C000_to_DFFF	WITH_STRING	"Some Text like IDs"	-useAreas={{0xC111, 0x40}}
FILL_BUFFER	Buf_C000_to_DFFF	WITH_CRC	MyCRC	-setByteOrder=BigEndian -useAreas={{0xC111, 0x40}}
PRINT_BUFFER	Buf_F000_to_FFFF			-useAreas={{0xF000, 0xF0010}}
PRINT_TEXT	{The Crc Value is: MyCRC}			-resolveSymbolsAsHexValue={MyCRC}
WRITE_FILE	File_out1	WITH_BUFFERS	{Buf_A000_to_AFFF Buf_B000_to_BFFF Buf_F000_to_FFFF}	-setExecutionAddress=0x0A000 -setFormat=IntelHexExtended
CALL_APP	echo	WITH_ARGUMENTS	{Hello World! MyCRC2}	-resolveSymbolsAsHexValue={MyCRC2}

Parameter Reference - supported values:

`-setType=` : `Crc32_Afs2` (which is default), `Crc32_Afs2`, `Crc32_Standard`, `Crc32_Vector_AUTOSAR`, `Crc32_NIO_AUTOSAR_P4`, `Crc8_Standard`, `Crc32_Ethernet_Mpeg2`, `Crc8_OnesComplement`, `Crc8_TwosComplement`, `Crc8_Eeprom`, `Crc16_OnesComplement_BE`, `Crc16_TwosComplement_BE`, `Crc16_Standard`, `Crc16_Legacy_Lit456_Led2`, `Crc16_Ccitt`, `Crc16_ibm_sdlc`
: `Hash_Gost_256`, `Hash_Md2`, `Hash_Md4`, `Hash_Md5`, `Hash_Md6`, `Hash_Ripemd128`, `Hash_Ripemd160`, `Hash_Ripemd256`, `Hash_Ripemd320`, `Hash_Sha1`, `Hash_Sha2_224`, `Hash_Sha2_256`, `Hash_Sha2_384`, `Hash_Sha2_512`, `Hash_Sha3_224`, `Hash_Sha3_256`, `Hash_Sha3_384`, `Hash_Sha3_512`

: `InterpretAsLittleEndian`, `InterpretAsBigEndian`
`-setByteOrder=`: `BigEndian` (which is default), `LittleEndian`
`-setFormat=` : `IntelHexExtended` (which is default), `IntelHexSegment`, `RawBinary`, `MotorolaS[_Zahl]`, `EmbeddedC_Assembler[_Zahl]` [] = optional

Erklärung für:

- a) `-useBufferRange={0xC111, 0x40}`: Erzeugt einen Buffer, der bei Adresse 0xC111 beginnt und die Länge 0x40 hat.
- b) `-useAreas={{0xC111, 0x40},{0xD111, 0x40}}` Erzeugt einen **Buffer**, der bei 0xC111 beginnt und bis 0xD111+0x40 geht. Die Areas `{{0xC111, 0x40},{0xD111, 0x40}}` werden aus dem Infile eingelesen, der Rest dazwischen wird mit 0 initialisiert.
- c) `-useBufferRange={0x00, 0xE000}`
`-useAreas={{0xC111, 0x40},{0xD111, 0x40}}` Erzeugt einen fixen Buffer der bei 0x0000 anfängt und bis 0x0000+0xE000 geht. Die Areas `{{0xC111, 0x40},{0xD111, 0x40}}` werden aus dem Infile eingelesen in diesen Buffer übertragen, der Rest wird mit 0 initialisiert.
- d) `-useAreas={{0xC111, 0x40},{0xD111, 0x40}}`
`-changeStartAddressTo=0x0000A000` Erstellt einen Buffer von 0xC111 bis 0xD111+0x40. Die Areas `{{0xC111, 0x40},{0xD111, 0x40}}` werden aus dem Infile eingelesen in diesen Buffer übertragen, der Rest wird mit 0 initialisiert. Hinterher wird die Startadresse des Buffers (0xC111) auf 0xA000 redefiniert.

Info:

- Arithmetic mode (21.10.2008)! Calculate dynamic hexvalues by using `+`, `-`, `*`, `/` or `%` operators. Calculation is from left side to right side. Braces are supported.
- Print Text supports calculation by using brackets e.g. `[0xB000 + 0x5]`
- Full description is available in Src under `ALHexManip_Vs2013\ALHexManip\ALHexManipManual.txt`
- HXM Syntax Highlighter is available for VS 2008:
\\dereulnquest1\Install\C++\VS\Plugins\HexSyntaxHighlighting_Vs2008\HxmSyntaxHighlighting_Vs2008.exe