```
//#############################################################################
//        ALHexManip 2008 – 2021 COMMAND LINE VERSION DESCRIPTION
//#############################################################################
//
//CONTENT:  - 0. EXAMPLE CONFIGURATION FILE FOR ALHexManip
//          - 1. COMPLETE KEYWORD DESCRIPTION
//          - 2. SELF RUNNING SCRIPT EXAMPLE
//
//
//#############################################################################
//        0. EXAMPLE CONFIGURATION FILE FOR ALHexManip
//#############################################################################

NEW_BUFFER     @Buf_0000_to_0009  FROM_VALUES {0x01, 0x02, 0x03, 0x04}
                                    -useBufferRange={0x0000,0xA}
                                    -setByteOrder=BigEndian

NEW_BUFFER     @Buf               FROM_VALUES {0x01, 0x02, 0x03, 0x04}
                                    -useBufferRange={0x0000,0xA}
                                    -setByteOrder=LittleEndian

NEW_CRC        @MyCrcs            FROM_BUFFER    @Buf_0000_to_0009
                                    -setType=Crc32_Afs2

FILL_BUFFER    @Buf_0000_to_0009  WITH_CRC       @MyCrcs
                                    -useAreas={{0x00,0x4}}

FILL_BUFFER    @Buf               WITH_CRC       @MyCrcs
                                    -useAreas={{0x00,0x4}}

WRITE_FILE     ASDF2_BIG.c        WITH_BUFFERS   {@Buf_0000_to_0009}
                                    -setFormat=EmbeddedC_Assembler
                                    -setExecutionAddress=.Test

WRITE_FILE     ASDF2_LITTLE.c     WITH_BUFFERS   {@Buf}
                                    -setFormat=EmbeddedC_Assembler
                                    -setExecutionAddress=.Test

PRINT_BUFFER  @Buf_0000_to_0009     -useAreas={{0x00, (0x01+((3-0x03)+2)*2)-3}}

//#############################################################################
//        1. COMPLETE KEYWORD DESCRIPTION
//#############################################################################
//Foreword: - Lists begin with { and end with }
//          - List elements are separated by , (comma) or (semicolon) ;
//            or spaces
//          - Spaces are ignored
//          - Numbers are always interpreted as hex values
//          - For every command are always all [options] usable
//          - [options] are optional and do not have to be specified
//          - [options] begin with -,
//             followed by the option name (naming like variables),
//             followed by = (spaces are allowed)
//             followed by the option setting name
//          - Program arguments which are conform to the pattern a=b
//            will be usable in the script. They are treated like SUBSTITUTES
```

```
//
//
//-------------------------------------------------------------------------------
//COMMAND  SUBSTITUTE
//-------------------------------------------------------------------------------
//SUBSTITUTE    [searchString]          WITH            [replacement]
//  Will replace every occurance of searchString with the replacement
//
//  Type of [searchString] = any alphanum char
//  Type of [replacement]  = any char except a newline
//
//
//Examples:
//   SUBSTITUTE     File_in1           WITH
//                                         ..\..\..\..\Bin\L7Bw02temp.app
//
//   NEW_BUFFER     @Buf_A000_to_AFFF  FROM_FILE
//                                          File_in1
//
// =>NEW_BUFFER     @Buf_A000_to_AFFF  FROM_FILE
//                                         ..\..\..\..\Bin\L7Bw02temp.app
//
//
//-------------------------------------------------------------------------------
//COMMAND  NEW_BUFFER
//-------------------------------------------------------------------------------
//NEW_BUFFER    [nameForNewBuffer]  FROM_VALUES       [valueList]        [options]
//  Creates a Buffer from a list of hex values
//
//NEW_BUFFER    [nameForNewBuffer]  FROM_BUFFER       [nameABuffer]      [options]
//  Creates a Buffer by using values from another buffer
//
//NEW_BUFFER    [nameForNewBuffer]  FROM_BUFFERS      [nameABufferList]  [options]
//  Creates a Buffer by using values from multiple buffers
//
//NEW_BUFFER    [nameForNewBuffer]  EMPTY                                [options]
//
//  Creates a Buffer by using no values
//
//NEW_BUFFER    [nameForNewBuffer]  FROM_FILE         [filePath]         [options]
//  Creates a Buffer by using values imported from a file
//
//NEW_BUFFER    [nameForNewBuffer]  FROM_FILE_BINARY  [filePath]         [options]
//  Creates a Buffer by using values imported from a file in RAW Binary Mode
//
//  Type of [nameForNewBuffer] = any combination of alphanumeric characters
//                                 or a variableName
//  Type of [valueList]        = a list of byte values, like:
//                                    {0x80, 0x90, 0x00}
//                                    {0x80 0x90 0x00}
//                                    {80, 90, 00}
//                                    { 0x80;    90,   0x00 }
//  Type of [nameABuffer]      = is the name of a previously created buffer,
//                                 by the command NEW_BUFFER
//                                 createdBufferA
//  Type of [nameABufferList]  = is a list of names previously created buffers.
```

```
//                                  { createdBufferA,
//                                    createdBufferB,
//                                    createdBufferC  }
//  Type of [filePath]         = is a list of anychar, or a word, which means
//                               that, as long as the path contains no spaces
//                               you can ignore {}
//                               Files could have IntelHex, MotorolaS
//                               or RawBinary Format. Common types are
//                               automatically determined by the file suffix.
//                               Complicated path an file names (also with
//                               space(s) in path names can be used as
//                               follows.
//                                  C:\test\first_way.hex
//                                  "C:\test me\"second_way.hex
//                                  {C:\test me\third_way.hex}
//  [options]
//    -useBufferRange=[bufferRange]
//    -setByteOrder=[ordering]
//    -useAreas=[memoryAreaList]
//    -changeStartAddressTo=[startAddress]
//
//  Type of [bufferRange]      = Touple-List with startOffset and size
//                               It defines the size of the created Buffer.
//                               If it is empty, than the size is determined,
//                               by the lowest and highest offset in
//                               the [memoryAreaList]
//                                  {0x000, 0x34}
//  Type of [ordering]         = Enumeration Type which can be defined
//                               to determine in which byteOrder a crc
//                               is inserted in the buffer possible
//                               values are:
//                                  BigEndian    (which is default)
//                                  LittleEndian
//  Type of [memoryAreaList]   = A List of memoryRegions
//                               Every memoryRegion consists of a startOffset
//                               and size.
//                               If a [memoryAreaList] is specified,
//                               only values in this range will be used in
//                               the buffer.
//                               If this option is not specified, the whole
//                               content will be used.
//                               A [memoryAreaList] looks like:
//                                  {{0x00, 0x34}}
//                                  {{0xA000, 0x0180} {0xAF80, 0x0080}}
//  Type of [startAddress]     = The startAddress gets used, after the buffer
//                               was filled with all values. It changes
//                               the relative addressing. Therefore the value
//                               will be assigned to the first Address.
//                               The [startAddress] is a single value like:
//                                   0x0000A000
//                                   0
//
//
//-------------------------------------------------------------------------------
//COMMAND  NEW_CRC
//-------------------------------------------------------------------------------
```

```
//NEW_CRC     [nameForNewCrc]   FROM_BUFFER  [nameABuffer]        [options]
//  Creates a Crc Value from a range of values contained in a buffer
//
//NEW_CRC     [nameForNewCrc]   FROM_BUFFERS [nameABufferList]    [options]
//  Creates a Crc Value from a range of values contained in a buffers
//
//  Type of [nameForNewCrc]    = any alphanumeric characters / or a variableName
//                                  createdCrcA
//  Type of [nameABuffer]      = is the name of a previously created buffer,
//                               by the command NEW_BUFFER
//                                  createdBufferA
//  Type of [nameABufferList]  = is a list of names previously created buffers.
//                                  { createdBufferA,
//                                    createdBufferB,
//                                    createdBufferC  }
//
//  [options]
//    -setType=Crc32_Afs2
//    -useAreas=[memoryAreaList]
//
//  Type of [crcType]          = Enumeration Type which can be defined
//                               to determine the calculation method
//                               possible values are:
//                                  Crc32_Afs2 (which is default),
//                                  Crc32_Afs2,
//                                  Crc32_Standard,
//                                  Crc32_Vector_AUTOSAR,
//                                  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
//
//  Type of [memoryAreaList]   = A List of memoryRegions
//                               Every memoryRegion consists of a
//                               startOffset and size.
//                               If a [memoryAreaList] is specified,
//                               only values in this range will be used in
//                               the buffer.
//                               If this option is not specified, the whole
//                               content will be used.
//                               A [memoryAreaList] looks like:
//                                  {{0x00, 0x34}}
//                                  {{0xA000, 0x0180} {0xAF80, 0x0080}}
//
//
//-------------------------------------------------------------------------
//COMMAND  NEW_HASH
//-------------------------------------------------------------------------
//NEW_HASH
```

```
//  Creates a Hash Value from a range of values contained in a buffer
//  NEW_HASH has the same description as NEW_CRC
//
//  [options]
//    -setType=Hash_Sha2_256
//    -useAreas=[memoryAreaList]
//
//  Type of [hashType]         = Enumeration Type which can be defined
//                               to determine the calculation method
//                               possible values are:
//                                  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
//
//
//------------------------------------------------------------------------------
/COMMAND  FILL_BUFFER
//------------------------------------------------------------------------------
//FILL_BUFFER [nameForNewBuffer] WITH_FILLER  [fillPattern]    [options]
//  fills the buffer with a background pattern, which is written to a byte
// (if there is no value assigned in a given byte)
//
//FILL_BUFFER [nameForNewBuffer] WITH_CRC     [nameForCrc]     [options]
//
//  inserts a previously calculated crc in the buffer
//
//FILL_BUFFER [nameForNewBuffer] WITH_HASH    [nameForHash]    [options]
//
//  inserts a previously calculated hash in the buffer
//
//FILL_BUFFER [nameForNewBuffer] WITH_VALUES  [valueList]      [options]
//  inserts a list of byte values in the buffer
//
//  Type of [nameForNewBuffer] = any alphanumeric character or a variableName
//                                  createdBufferA
//  Type of [fillPattern]      = = a list of byte values, like:
//                                  {0x80, 0x90, 0x00}
//                                  {0x80 0x90 0x00}
//                                  {80, 90, 00}
//                                  { 0x80;    90,   0x00 }
//  Type of [nameForCrc]       = is the name of a previously created Crc,
```

```
//                                by the command NEW_CRC
//                                   createdCrcA
//  Type of [nameForHash]      = is the name of a previously created Hash,
//                                by the command NEW_HASH
//                                   createdHashA
//  Type of [valueList]        = a list of byte values, like:
//                                    {0x80, 0x90, 0x00}
//                                    {0x80 0x90 0x00}
//                                    {80, 90, 00}
//                                    { 0x80;    90,   0x00 }
//
//  [options]
//    -useAreas=[memoryAreaList]
//
//  Type of [memoryAreaList]   = A List of memoryRegions
//                                Every memoryRegion consists of a
//                                startOffset and size.
//                                If a [memoryAreaList] is specified,
//                                only values in this range will be used in
//                                the buffer.
//                                If this option is not specified, the whole
//                                content will be used.
//                                A [memoryAreaList] looks like:
//                                   {{0x00, 0x34}}
//                                   {{0xA000, 0x0180} {0xAF80, 0x0080}}
//
//
//------------------------------------------------------------------------------
//COMMAND  PRINT_BUFFER
//------------------------------------------------------------------------------
//PRINT_BUFFER [nameABuffer]                          [options]
//  outputs the content of a buffer to the output stream (console or build log)
//
//  Type of [nameABuffer]      = is the name of a previously created buffer,
//                                by the command NEW_BUFFER
//                                   createdBufferA
//
//  [options]
//    -showAreas=[memoryAreaList]
//
//  Type of [memoryAreaList]   = A List of memoryRegions
//                                Every memoryRegion consists of a
//                                startOffset and size.
//                                If a [memoryAreaList] is specified,
//                                only values in this range will be used in
//                                the buffer.
//                                If this option is not specified, the whole
//                                content will be used.
//                                A [memoryAreaList] looks like:
//                                   {{0x00, 0x34}}
//                                   {{0xA000, 0x0180} {0xAF80, 0x0080}}
//
//
//------------------------------------------------------------------------------
//COMMAND  PRINT_CRC
//------------------------------------------------------------------------------
```

```
//PRINT_CRC [nameACrc]
//  outputs the content of a crc to the output stream (console or build log)
//
//  Type of [nameACrc]      = is the name of a previously created crc,
//                            by the command NEW_CRC createdCrcA
//
//-------------------------------------------------------------------------------
//COMMAND  PRINT_HASH
//-------------------------------------------------------------------------------
//PRINT_HASH [nameAHash]
//  outputs the content of a hash to the output stream (console or build log)
//
//  Type of [nameAHash]     = is the name of a previously created hash,
//                            by the command NEW_HASH createdHashA
//
//-------------------------------------------------------------------------------
//COMMAND  PRINT_TEXT
//-------------------------------------------------------------------------------
//PRINT_TEXT [echoText]    [options]
//  outputs the content of [echoText] to the output stream (console or build log)
//
//  Type of [echoText]      = is a List of alphanum chars  like:
//                                {This is an example}
//                                "This is an example"
//                                This_is_an_example
//
//                          Calculation of Values is possible by using
//                          square brackets inside like:
//                              {Crc is at this: [0x300 + 0x433 - 43] address}
//
//                          This will result in the text:
//                               Crc is at this: 0x708 address
//
//  [options]
//    -resolveSymbolsAsHexValue=[crcNameList]
//
//  Type of [crcNameList]         =  A list of Crc variable names,
//                                   which are referenced in the [args]
//                                   and should be dynamically resolved
//                                   by the lexical parser.
//
//-------------------------------------------------------------------------------
//COMMAND  WRITE_FILE
//-------------------------------------------------------------------------------
//WRITE_FILE    [filePath]  WITH_BUFFERS  [nameABufferList]       [options]
//  Exports bufferInformation into a file. Default output is IntelHex
//
//  Type of [filePath]         = is a list of anychar, or a word, which
//                               means that, as long as the path contains
//                               no spaces, you can ignore {}
//                               Till now the files could have IntelHex,
//                               MotorolaS or RawBinary Format.
//                               Most types can be automatically determined
//                               by the file suffix.
//                                  C:\test\first_way.hex
//                                  "C:\test me\"second_way.hex
```

```
//                                {C:\test me\third_way.hex}
//
//  Type of [nameABufferList]  = is a list of names previously created buffers.
//                                { createdBufferA,
//                                  createdBufferB,
//                                  createdBufferC  }
//
//  [options]
//    -setExecutionAddress=[executionAddress]
//    -setFormat=[formatSetting]
//
//  Type of [executionAddress] = The executionAddress from where the
//                               Application gets loaded. If nothing
//                               is specified, this information will not
//                               be exported.
//                               The [executionAddress] is a single value like:
//                                   0x0000A000
//                                   0
//  Type of [formatSetting]    = Enumeration Type which can be set to override
//                               the default ExportFormat
//                               allowed values are:
//                                  IntelHexExtended    (which is default)
//                                  IntelHexSegment
//                                  RawBinary
//                                  MotorolaS
//
//
//-------------------------------------------------------------------------------
//COMMAND  CALL_APP
//-------------------------------------------------------------------------------
//CALL_APP   [AppPath]  WITH_ARGUMENTS  [args]       [options]
//  Executes an external program (such as DOS echo or any other program)
//
//  Type of [AppPath]         =  Application
//                               A list of anychar, or a word, which means
//                               that, as long as the path contains no spaces,
//                               you can ignore {}
//
//  Type of [args]            =  The arguments
//                               A list of anychar, or a word, which means
//                               that, as long as the path contains no spaces,
//                               you can ignore {}
//                                 { blah blah d=4}
//                                 blah_blah
//
//  [options]
//    -resolveSymbolsAsHexValue=[crcNameList]
//
//  Type of [crcNameList]        =  A list of Crc variable names
//                                  which are referenced in the [args]
//                                  and should be dynamically resolved
//                                  by the lexical parser.
//
//EXAMPLE:
//  CALL_APP  echo  WITH_ARGUMENTS  {Hello World!}
//
```

```
//



//
//#############################################################################
//        2. SELF RUNNING SCRIPT EXAMPLE
//#############################################################################
//
SUBSTITUTE         FirstOut             WITH            first.hex
SUBSTITUTE         SecondOut            WITH            second.hex
SUBSTITUTE         IntelHexExtendedOut  WITH            IntelHexExtendedOut.hex
SUBSTITUTE         IntelHexSegmentOut   WITH            IntelHexSegmentOut.hex
SUBSTITUTE         RawBinaryOut         WITH            RawBinaryOut.raw
SUBSTITUTE         MotorolaS_Out        WITH            MotorolaS_Out.s

SUBSTITUTE         App                  WITH            echo


// -----------------------
// Part 1
// -----------------------
PRINT_TEXT
{MY SAMPLE text is: Buffer Creation and Fill Test:
   Some Buffers are created with values and filled with a mask.}

// if useBufferRange is not submitted, the min, max of all areas will
// be used if useBufferRange and useAreas is not submitted and it is
// FROM_FILE, FROM_BUFFER, the whole content will be used
// if useBufferRange and useAreas is not submitted and it
// is EMPTY => it will be an EMPTY BUFFER offset=0, size=0
//
```