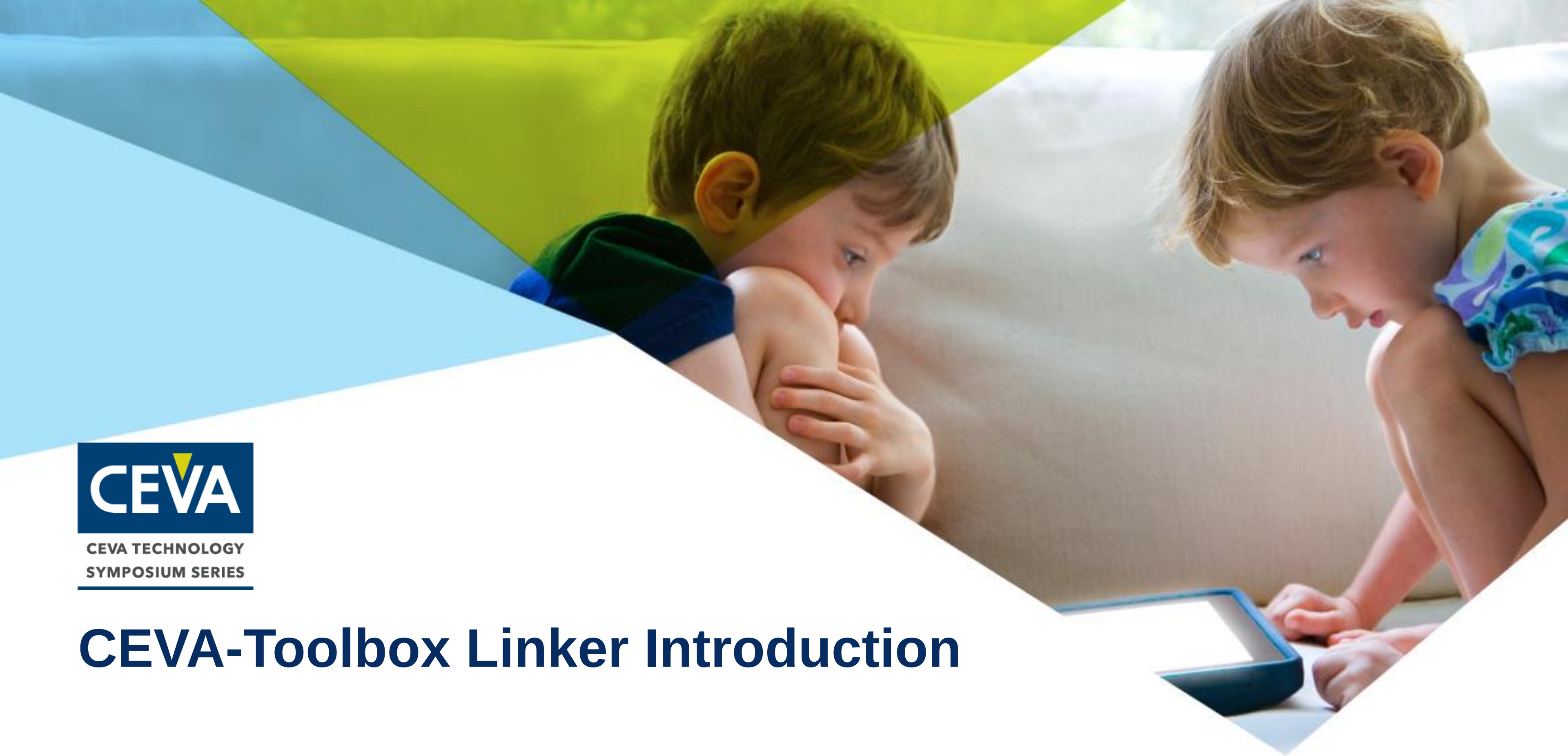


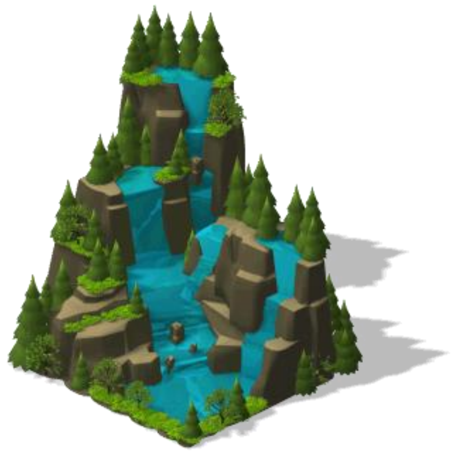
CEVA TECHNOLOGY SYMPOSIUM SERIES

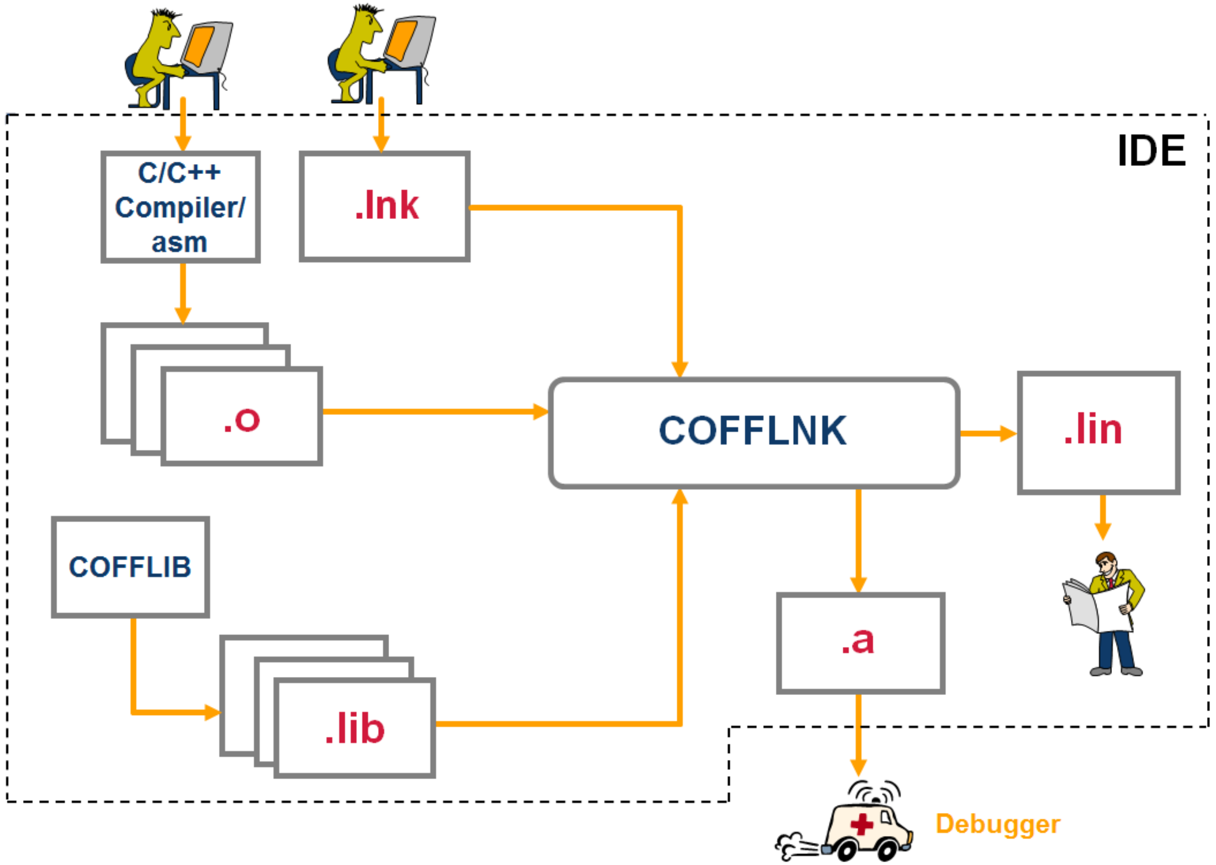
CEVA-Toolbox Linker Introduction

2016

www.ceva-dsp.com

CEVA-Toolbox Linker Flow





CEVA-Toolbox Linker Command Line Options

- Command line invocation
  - cofflnk [options] [LnkFile]
- Generate full data and code mapping information Including code and data memory holes
  - -m
- Automatic alignment to all sections by this defaults: 0x04 for data, 0x20 for code
  - -alignAllSections
- Invoke Macr Pre Processing before the linkinig process
  - -p[,MPP-options]
- Add functions reference table into the listing file containing all function calls of each function
  - -funcRef

CEVA-Toolbox Linker Command Line Options

- List of object files to be linked
  - -obj <obj1,obj2,…>
- List of libraries files to be linked
  - -lib <lib1,lib2,…>
- Print detailed section information, such as address and size of variables and functions of each section
  - -secInfo

CEVA-Toolbox Linker Script File Example

objects:

List of File Names (.o)

libraries:

List of File Names

Classes:

List of Memory Classes

class1:

List of sections with
Opt Location, Alignment & Attrib

⇨

sec [LocExpr] [AlignExpr] [Attrib]

sec [LocExpr] [AlignExpr] [Attrib]

sec [LocExpr] [AlignExpr] [Attrib]

....

sec [LocExpr] [AlignExpr] [Attrib]

...

classN:

List of sections with
Opt Location, Alignment & Attrib

CEVA-Toolbox Linker Command Line Options

```
objects:                              ; List of object files
    crt0.o                            ; Must be the first one in the list
    MyObject1.o                       ; Located in current directory
    ..\..\otherpath\MyObject3.o       ; Relative access usage
    %MyWork%\MyObject4.o              ; Environment variable usage
    crtn.o                            ; Must be the last one in the list

;; Order of system libraries IS IMPORTANT and should NOT be changed
libraries:                            ; List of Library files
<TOOLS PATH>\libs\fileio.lib
<TOOLS PATH>\libs\libc.lib
<TOOLS PATH>\libs\libios.lib
<TOOLS PATH>\libs\libioe.lib
<TOOLS PATH>\libs\tk1lib.lib
```

CEVA-Toolbox Linker Command Line Options

```
;; User libraries should follow the system libraries
        MyLib1.lib                                  ; Located in current directory
        c:\mypath\subdirectory\MyLib2.lib           ; Direct access usage
        ..\..\otherpath\MyLib3                      ; Relative access usage
        %MyWork%\MyLib4                             ; Environment variable usage

;; List of memory class (data and/or code) declarations
classes:
       eprom_class  [c:8000, c:8fff     ]           ; Code range
        xram_class   [d:0000, d:03ff     ]
        yram_class   [d:fc00 , d:ffff      ]

;; User defined code class
eprom_class:
        section1 at lo                              ; Locate section1 at lowest location
        section2                                    ; Locate section2 next to section1
        section3 at 0x8a00                          ; Locate section3 at 0x8a00 specific address
```

Libraries

- Libraries are not linked unless required
  - Only when there is a reference that cannot be resolved from the user objects
  - In that case, the Linker links only the required object files from the   corresponding library
- Since Compiler system libraries include one function per object file, it ensures that the minimal necessary code is inserted
  - For this reason, it is advised to create user libraries in the same way
  - Don’t forget to split global buffers to separate files as well
- Using the -libRefInfo Linker switch will provide a list of the symbols   origin.
  - Each symbol that was declared in an object that was located in a library and the Linker used this object in the linking process will be listed in a special table in the .lin file called
    “LIBRARY SYMBOLS REFERENCE INFORMATION”
    that will hold the object and library names next to the symbol name

Sections Mapping Attributes: 'at'

The linker must locate the section at the indicated address expression

Example:

data:

SecA at 0x100

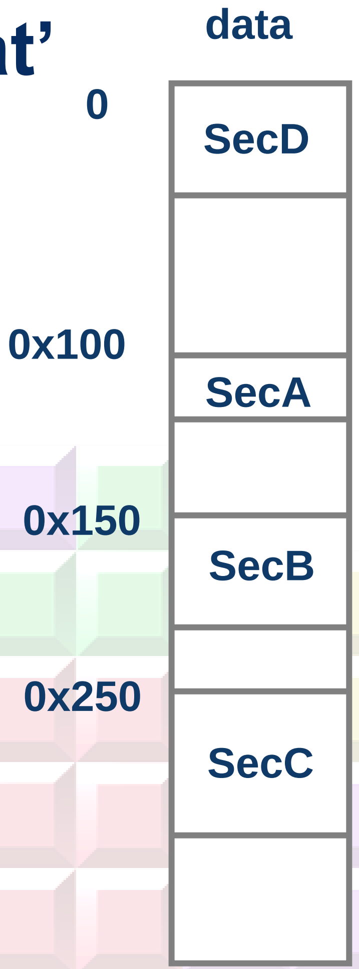
SecB at SecA+0x50

SecC at next(SecB+0x100, 0x200)

SecD at lo

Note: The 'at' directive can be used together with any of the following directives. In such a case, the Linker tries to locate the section exactly at the address denoted by the other directive.

Example: 'at lo' instructs the Linker to locate the section exactly at the lowest address of the memory class

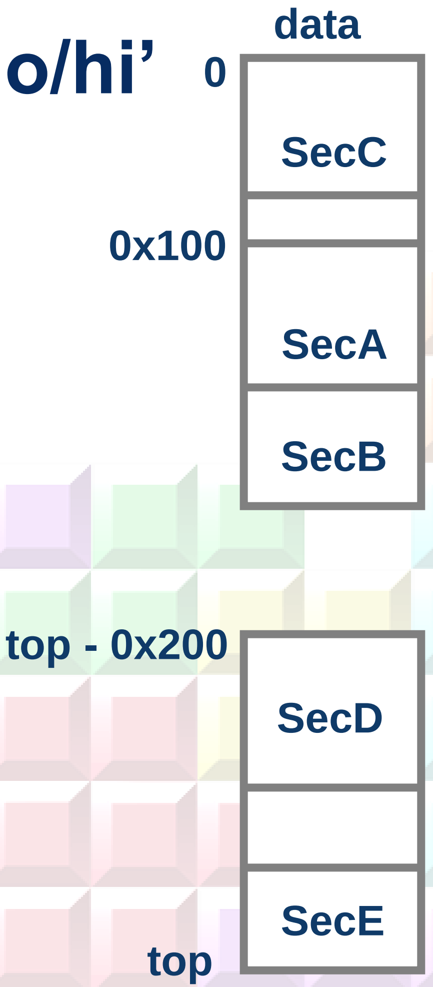


Sections Mapping Attributes: ‘lo/hi’

The Linker starts searching from the lowest/highest address of the memory class

Example:

```
data:
    SecA at 0x100         ; locate SecA at 0x100
    SecB lo               ; SecB is bigger than 0x100
    SecC at lo            ; force SecC at class beginning
    SecD at hi -0x200     ; force SecD at class top - 0x200
    SecE hi               ; start searching from top
```

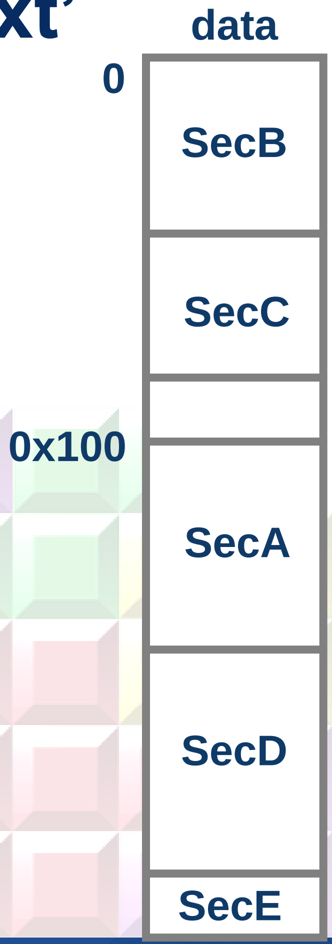


Sections Mapping Attributes: ‘next’

- next directive is used for locating the section at the next available memory hole that fits the section size, starting from the last mapped section’s last address
- next(list) indicates the Linker to search for a suitable memory hole fitting the section size following the maximal address represented by a list of expressions

Example:

```
SecA at 0x100
SecB lo                  ; assume SecB size is 0x50
SecC next                ; assume SecC size is 0x30
SecD next(SecA, SecB)    ; locate secD following
                         ; SecA, SecB – end addresses are taken
SecE                     ; default next is assumed
```



Sections Mapping Attributes: ‘align’

Forces the Linker to place the relevant section at an address that is an integer multiple of a numeric constant

Example:

```
SecA  at     0x180             ; assume (SecA < 0x80) && (SecB > 0x80)
   SecB  align 0x100           ; searching a fitting hole for the section
                               ; starts at 0x200 (and continues every
                               ; multiple of 0x100)
```

Note:

Alignment requirement can be forced in Assembly level as well by the Assembler’s .ALIGN directive

Sections Mapping Attributes: ‘size’

- Used to extend a predefined section or to create a new section with a specified size
- When used to create a new section, this section will have a noload attribute
- Can be implemented on sections defined in CRT0 file
  - such as: MALLOC_SECT_, STACK_SECT_, etc, according to the specific application’s requirements
- Can be used as space holder for reserved addresses, specifically in Emulation

Sections Mapping Attributes: ‘size’

Example:

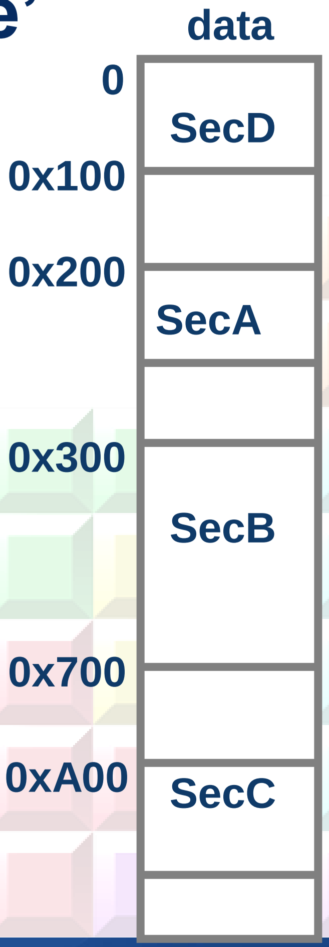
data:

SecA at 0x200

SecB at SecA+0x100 size 0x400

SecC at next(SecB+0x700, 0x200)

SecD at lo size 0x100



Sections Mapping Attributes: ‘smallest’

- Instructs the Linker to search for the smallest memory hole in the memory class that the section is to be located in that can fit the section

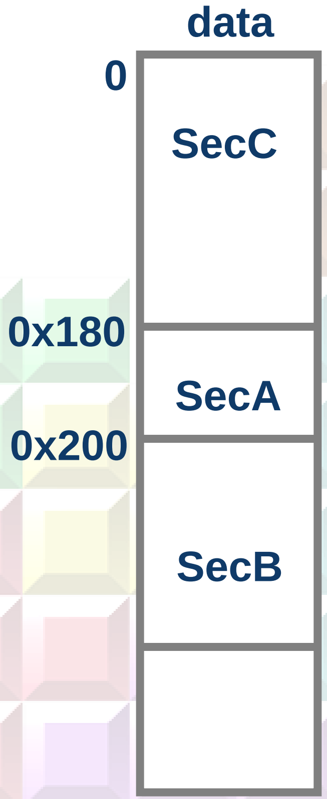
Example:

data:

SecA at 0x180

SecB align 0x100

SecC smallest ; assuming SecC size is 0x180, locate at 0



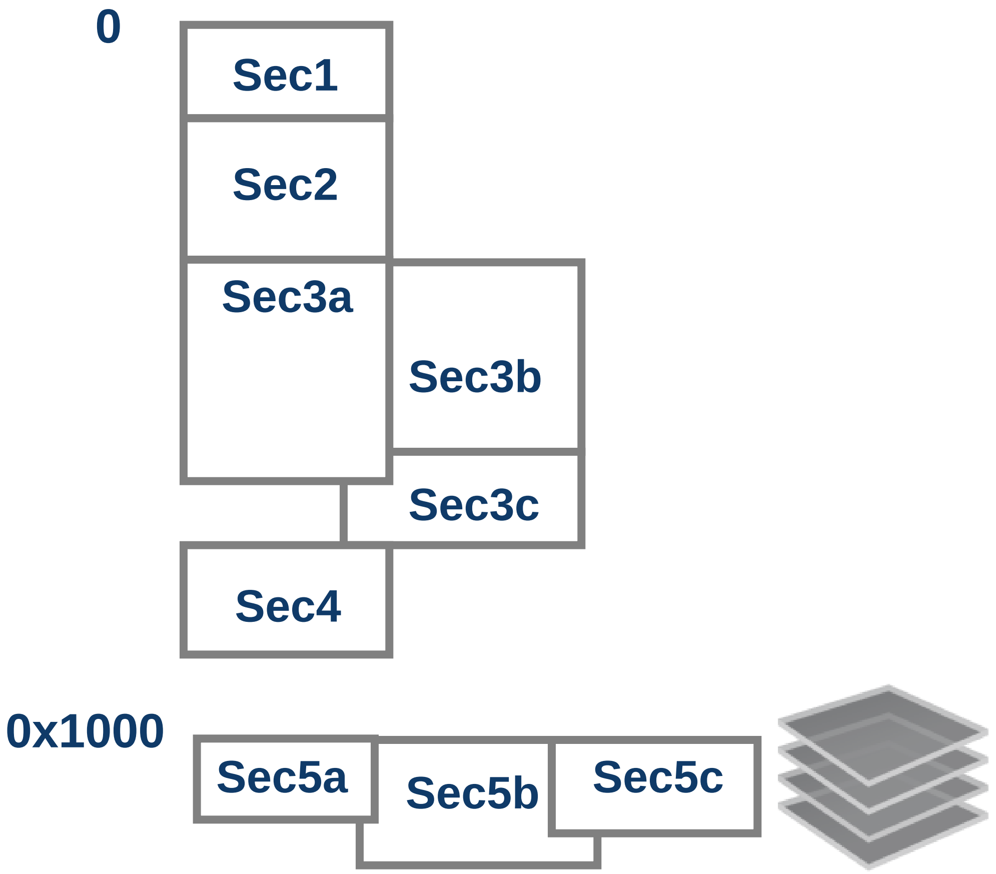
Sections Mapping Attributes: ‘clone’

- The clone attribute enables location of a section multiple times in the same memory type
  - e.g: data, code_ext
- Section cloning can be used when a section should be mapped in both RAM and ROM, in the same memory type/class
- Used by the Compiler for initializing initialized variables on reset
  - const_data and .data sections
- Example

```
data:
const data lo                              ; Map the original section
const_data_clone  next clone const_data    ; Map the clone section using the same name as
                                           ; in the Assembly file  (const_data_clone)
```

Sections Overlay

```
data:
    Sec1
    Sec2
    {          ;open overlay group
      Sec3a
      Sec3b    at  Sec3a  noload
      Sec3c    noload
    }          ;close overlay group
    Sec4
    {          ;open overlay group
      Sec5a      at   0x1000
      Sec5b      at   Sec5a noload
      Sec5c      at   Sec5a noload
    }          ;close overlay group
```



Sections Mapping Tip

Maximizing memory utilization in the linking process

- Map, in the Linker script file, only sections that must be located in specific location
  - Leave all the rest of the sections for the Linker's automatic mapping
- Use both -sortUnmentioned and -mapUnmentionedSmallest options optimizes memory mapping of the unspecified sections
  - *cofflnk -sortUnmentioned -mapUnmentionedSmallest [other options] file.lnk*
  - Appear in the objects but are not mapped in the Linker's script file
  - Achieves good automatic utilization of memory holes

CEVA

CEVA TECHNOLOGY SYMPOSIUM SERIES

THANK YOU

www.ceva-dsp.com