This document is for information and instruction purposes. Mentor Graphics reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Mentor Graphics to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Mentor Graphics products are set forth in written agreements between Mentor Graphics and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Mentor Graphics whatsoever.

MENTOR GRAPHICS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

MENTSOR GRAPHICS SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF MENTOR GRAPHICS CORPORATION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

RESTRICTED RIGHTS LEGEND 03/97

U.S. Government Restricted Rights. The SOFTWARE and documentation have been developed entirely at private expense and are commercial computer software provided with restricted rights. Use, duplication or disclosure by the U.S. Government or a U.S. Government subcontractor is subject to the restrictions set forth in the license agreement provided with the software pursuant to DFARS 227.7202-3(a) or as set forth in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clause at FAR 52.227-19, as applicable.

Contractor/manufacturer is:
Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.
Telephone: 503.685.7000
Toll-Free Telephone: 800.592.2210
Website: www.mentor.com
SupportNet: supportnet.mentor.com/
Send Feedback on Documentation: supportnet.mentor.com/doc_feedback_form

TRADEMARKS: The trademarks, logos and service marks ("Marks") used herein are the property of Mentor Graphics Corporation or other third parties. No one is permitted to use these Marks without the prior written consent of Mentor Graphics or the respective third-party owner. The use herein of a third-party Mark is not an attempt to indicate Mentor Graphics as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A current list of Mentor Graphics’ trademarks may be viewed at: www.mentor.com/trademarks.

End-User License Agreement: You can print a copy of the End-User License Agreement from: www.mentor.com/eula.
## Table of Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C01</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Harness Complexity and Composites</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Managing Variants: Modular Harnesses</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Modular Harness Design Methods and Flows</td>
<td>4</td>
</tr>
<tr>
<td>C02</td>
<td>Module code and design definition</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Module Codes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Module Code Management</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Module Designs</td>
<td>10</td>
</tr>
<tr>
<td>C03</td>
<td>Module Assignation</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Assignation Process</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Assignation Methods</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Clearing Assignations</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Special Cases</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Modular Breakdown</td>
<td>15</td>
</tr>
<tr>
<td>F01</td>
<td>Function based assignation</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Overview of Function-based Assignation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Standard” Assignation</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>“Mandatory” Assignation</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>“Mandatory, Factor” Assignation</td>
<td>24</td>
</tr>
<tr>
<td>F02</td>
<td>Technical Assignation</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Overview of Technical Assigniation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Technical Module Code Creation</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Combination Validation and Filtering</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Technical Assignation Process</td>
<td>36</td>
</tr>
<tr>
<td>F03</td>
<td>Rule based functional assignation</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Overview of Rule-based Functional Assignation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Contextual Rule-based Assignation</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Global Rule-based Assignation</td>
<td>42</td>
</tr>
</tbody>
</table>
Chapter C01
Introduction to Modular Harness Design

Table of Content:

<table>
<thead>
<tr>
<th>Item</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harness Complexity and Composites</td>
<td>1</td>
</tr>
<tr>
<td>Managing Variants: Modular Harnesses</td>
<td>2</td>
</tr>
<tr>
<td>Modular Harness Design Methods and Flows</td>
<td>4</td>
</tr>
</tbody>
</table>
Harness Complexity and Composites

In electrical system design, customers face a common challenge in striking a balance between offering a wide range of products and product options, and managing the wide range of electrical configurations needed to support the product range.

Composite Harnesses are the answer to that challenge. A composite harness is a (typically virtual) superset which collates all variants of a harness in one container – the composite harness.

This technique is widely used in the automotive industry. The principal aim of this technique is to minimize administration and change effort by managing all variants of a harness within a single design.

Capital HarnessXC has the ability to depict a composite harness and automatically generate the details of every variant of it as and when they are required via the use of Option Expressions on wires and components; depicting product features needed in each variant. Derivative Harnesses (or simply, Derivatives) are generated through the process of Composite Breakdown. Each derivative represents a fully functional variant of the composite harness that contains a subset of its content.

The screenshots below demonstrate how a composite is broken-down into derivatives as currently available in Capital HarnessXC.
A variant management problem therefore arises for OEMs and their harness suppliers in this approach: for each optional feature added to a harness, the number of variants is doubled, and an increasing number of variants means a fairly large overhead of managing their part numbers and option associations.

So an alternative methodology was needed to better manage composite harnesses in a way that reduces the overhead of variant management for highly variable harnesses.

**Managing Variants: Modular Harnesses**

Modular harness design techniques address the overhead of variant management experienced in derivatives.

A composite superset describing all variants of a harness is still used, but instead of breaking this down to finished versions (derivatives), it is broken down into partly-finished *Modules*. On their own, modules are not complete fully-functioning harnesses, but sub-assemblies that correspond to partial vehicle functions. These sub-assemblies are assembled together as and when required to form the final harness in a vehicle.
A (functional) module usually has a 1-1 relationship with its corresponding function. A derivative on the other hand has a number of “applicable options” associated with it. This leads to a very important distinction and benefit to modular techniques over derivatives: Variant Management.

Modular harness design significantly reduces the overhead of managing harness variants by transforming the problem of managing complex derivative combinations based on option content into one of managing simple modules.

To illustrate this advantage, imagine a scenario where 2 optional vehicle functions are available:
- A: Electric windows
- B: Heated seats

In a derivative approach, all combinations of the options are calculated, resulting in four derivatives that need to be managed:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Derivative</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>D1 (Base: no optional features present)</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>D2 (Heated seats)</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>D3 (Electric Windows)</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>D4 (Heated seats &amp; Electric Windows)</td>
</tr>
</tbody>
</table>

This means that 2 derivatives are needed for a given set of options.

In contrast, in a modular approach, there would only be two modules to manage, one corresponding to each function:
- Module A: Electric Windows
- Module B: Heated Seats

With the two modules available on the shelf for usage, once a certain configuration is determined, the modules needed to complete the harness (Module A and/or Module B) are assembled on the spot into the harness.

As the number of options/functions increase, the difference can be seen more clearly. The modular approach significantly reduces the variant management overhead.
Modular Harness Design Methods and Flows

Functional Modules

Given the benefits that a Modular Composite design flow offers, a lot of OEMs are employing a modular composite design flow to save on variant management costs.

Customer-Specific Vehicles (known as KSK) is an application of this technique at major OEMs. In such business models, OEMs list the different functions that a vehicle provides and customers can choose any given combination out of that list for a fully-configurable vehicle, for example heated seats, electric windows, etc.

To achieve this on a harness by harness basis, customers design their harnesses using “Functional Modules” which correspond to vehicle functions. Once a configuration definition is determined, the required modules can be assembled into one harness.

Production Modules

From the point of view of manufacturing of the harness, Tier-1 harness suppliers are also interested in this technique, but from a production assembly perspective. Whether it is a finished version of a harness, or just a sub-module, managing the process of the production of this assembly is a challenge. The derivatives or sub-modules can be split into smaller unfinished pieces that can then be produced individually to suit manufacturing constraints and maximize the efficiency of production resources. These will then be assembled together to produce the final product which will be delivered to the customer (the OEM).

OEM-Supplier Flows

Various techniques exist to manage harness design and production. Using Capital ModularXC different flows of data can be implemented between the OEM and their Tier-1 harness suppliers. These are depicted below.
Functional Module Approach:

- OEM
  - Motivation
    - Customer Specific Vehicles (KSK)
    - Derivative Vehicles
  - Design
    - Functional Modules
    - Production Modules
- T1
  - Engineering
    - Functional Modules
    - Production Modules
  - Manufacture
    - Functional Modules
    - Production Modules

Functional and Production module Approach:

- OEM
  - Motivation
    - Customer Specific Vehicles (KSK)
    - Derivative Vehicles
  - Design
    - Functional Modules
    - Production Modules
  - Manufacture
    - Functional Modules
    - Production Modules
- T1
  - Engineering
    - Functional Modules
    - Production Modules
  - Manufacture
    - Functional Modules
    - Production Modules
  - Design
    - Derivatives
      - Derivatives
# Chapter C02
Module Designs and Module Codes

<table>
<thead>
<tr>
<th>Table of Content:</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Codes ..................................</td>
<td>6</td>
</tr>
<tr>
<td>Module Code Management ..................</td>
<td>6</td>
</tr>
<tr>
<td>Module Designs ................................</td>
<td>10</td>
</tr>
</tbody>
</table>
Module Codes

Module Codes represent the different features or functions that are available in the electrical system. For example, in a vehicle, electric windows and heated seats are functions that have module codes representing them in the project.

Some module codes are mandatory while others are optional. Mandatory module codes (also called “base” or “core” module codes) are necessary for the operation of a harness and so are present on all finished versions of that harness. Optional ones can selectively be included in a harness configuration.

Module codes have relationships which define the way in which they are related. These relationships affect the inclusion and exclusion of modules when others are included.

Module relationship types include:

- **Inclusivity**: where the presence of one module code mandates the presence of others. For instance, a CD-changer mandates a CD-player.
- **Exclusivity**: where a module code cannot exist with others. For instance, Left-Hand Drive and Right-Hand Drive or Diesel and Petrol Engines.

Module Codes can be:

- Functional Module Codes: known as FM Codes
- Production Module Codes: known as PM Codes

Module Code Management

**Project Preferences**

Depending on the particular flow, an organization can configure a Project Preference to hide the attributes on the modular dialogs that are not used (i.e. production codes if only interested in a functional module flow). This helps to simplify the UI for the designer.
Module Behaviors

The ability to modify module design information can be controlled by the release level. If a design is at a checking or released stage, the ability to make changes to the module codes and their assignation to diagram objects should be removed. Each release level definition can have certain behaviors assigned. To prevent module code assignation changes to a design, the ‘functional module interaction and/or Production module interaction behavior should be removed for that particular release level.

Default Naming

Default naming can be configured for functional and production module codes as with all other object types.

Module Code Management Dialog

Once the setting is configured in project preferences, the set of Functional/Production module codes in the system can be edited in Capital Project, or in Capital ModularXC:

In Modular XC:

- Select Edit > Module Codes > Functional/Production Module Codes

This will launch the module code management window for the selected module code type, where module codes can be created, modified and deleted.
Within any of these dialogs, the following functions are available:

1. **Create FM/PM Module Code** / ![icon]
   Creates a module code under the selected folder.

2. **Create FM/PM Technical Module Code** / ![icon]
   Creates a Technical Module. The details of this action will be covered later in the course (Technical Assignation).

3. **Purge FM/PM Module Code**
   Finds and deletes module codes which are not used on any of the designs in the project.

4. **Create Folder**
   Creates a folder under the selected folder. This is used to organize module codes into groups.

5. **Create Combination**
   Creates a Combination of technical modules. The details of this action will be covered later in the course – (Technical Assignation).

6. **Delete**
   Deletes the selected objects (modules/folders).

7. **Sort Module Codes**
   Sorts objects under the selected folder alphanumerically (ascending/descending). Default sort order is the order in which the objects were created.

**Note**
Module Codes can be defined at system level. Users can drag and drop module codes from the System level into individual projects in Capital project.

**Note**
Design-level overrides can be enabled from the menu item: Edit / Module Codes / Enable Design Override of Functional/Production Codes.
Defining Module Code Relationships

Module relationships can be defined in one of two ways:

1. **Against a group:** sets a folder's content to be all Mutually Exclusive and/or Mandatory.

2. **Individual:** sets an individual module code's relationship to other modules. A module code can be made mandatory and/or inclusive/exclusive of other module codes.

**Exercise 1:**

⇒ Review the module code relationships as detailed in the exercise worksheet.
Module Designs

Modules are created in a similar way to derivatives; by creating a new module design under the owning composite and allocating applicable module codes to it.

To create a module design, right click on a composite which has applicable modules associated
⇒ Select New [Functional/Production] Module…

Note
The type of module designs available in the right mouse menu will be determined based on the type of module codes associated to the composite (Functional/Production/both)

A Module (design) usually has a 1-to-1 mapping with a single module so there is only one applicable module code associated with the design. However, this is not mandatory, and in some cases a module can have a number of associated module codes.

Automated module design creation and mapping

Instead of manually creating a module design, an automated process is available. Once modular assignation is complete, individual designs can be created for each individual module code.

⇒ Edit/Module Codes
⇒ Map functional/Production module codes to designs
⇒ Click Automap
⇒ Each module code will have a modular design created

Exercise 2:
⇒ Create module designs as detailed in the exercise worksheet
Chapter C03
Module Code Assignation Overview

Table of Content:

<table>
<thead>
<tr>
<th>Assignation Process</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignation Methods</td>
<td>12</td>
</tr>
<tr>
<td>Clearing Assignations</td>
<td>15</td>
</tr>
<tr>
<td>Special Cases</td>
<td>15</td>
</tr>
<tr>
<td>Modular Breakdown</td>
<td>16</td>
</tr>
</tbody>
</table>
**Assignation Process**

In a modular design flow, the composite needs to be tagged to indicate how the different objects will be distributed on different modules so that when two modules come together on a certain configuration of the harness, one, and only one instance of that object is found. This tagging process is called "module code assignation".

In this process, the expected start point is that the wires come in already tagged with their respective module codes. The assignation task calculates the *module code expressions* (applicable module codes) of the objects they terminate at (e.g. connectors), or pass under (e.g. clips and tubes). When a configuration is assembled, all objects within that configuration will be tagged with the relevant module code.

For example, in the diagram below all wires terminating at each connector have the same code. Assignation is straightforward in this case as the connector is present on the same module as the wires. The code from the wires is thus assigned to the connector as shown below.

When the configuration requires the presence of module F1’s wiring, the connectors will also be present.

Of course, this is a straightforward case, and hence assignation is simple.

Other cases exist, where the codes on the wires are different. A logical assessment of the codes needs to be performed to determine which module codes to tag the object with.

In the following example, wiring is present at the connector from 2 mutually exclusive modules; F1 and F2. On a given configuration, only F1 wiring is present, on another only F2 wiring is present. No configuration can have wiring from both module codes. In this case the connector needs to be tagged with F1 and F2, as it is needed for both modules.

There are more challenging cases where a logical assessment of the relationships of the codes on wires is not enough to determine the result of assignation.

For example, in the following composite a tube and a clip have wiring from two modules; A and B, passing through them. A and B have no relation to each other so the tube and the clip are required for module A alone, module B alone and when A and B exist together.
In this case the most accurate thing to do is to create and assign new codes to the tube and clip, which correspond to each of the potential code combinations. These new codes are called *Technical Codes*.

**Assignation Methods**

Module code expressions are assigned to an object via one of two methods:

- Manual
- Generated

**Manual**

In the manual method, the user selects the module codes to be assigned by editing the User FM Codes/User PM Codes fields in the object’s attributes.

⇒ To access the percentage editor, RMB on selected module codes within the module code dialog
⇒ Select Edit, then enter the percent required for each module code.
**Note**
Assignations done using Percentages do not represent the real, manufacturable modules but rather ones that can be used for initial costing purposes.

Alternatively, manual assignation can be performed by dragging and dropping unassigned design objects into different module code nodes in the Code Browser Tabs (FM Code/PM Code tab).
Assignations can be generated through a number of methods and routines using Capital ModularXC. These are:

- **Function-based**: generates module code expressions based on the module codes on related wires and the relationship between them. Four routines can be employed in this method;
  - “Standard” Assignation
  - “Mandatory” Assignation
  - “Mandatory, Factor” Assignation
  - “Technical” Assignation

- **Rule-based**: generates module code expressions based on user-defined design constraints. Two constraint types can be employed in this method;
  - Standard-constraints
  - Plug-in constraints

Each of these methods will be covered in detail in upcoming chapters. Auto generated codes are displayed in the ‘Generated’ code fields against the relevant object.
Note
The Code Browsers indicate generated as well as manual assignations on the objects. For generated assignations, object names are shown in *italics*.

If an object has both manual and generated code expressions then it is shown only under the manual codes in the code browsers.

Clearing Assignations
Manual and auto generated module code assignations can be cleared via the RMB/clear code function.

Special Cases
Two special cases exist for assignation:

1. Multicores
   a. Multicores can be assigned a module code (manually) but this is redundant as it is not required for multicore processing. Hence, multicores do not appear in the code browsers
   b. Breakdown utilizes the codes on the innercores and not the code on the multicore
   c. If all the innercores break down into the same module then the multicore will also be present. If not, the innercores get populated into the module as simple wires with no multicore. A DRC warns of inconsistent assignations between innercores of a multicore to avoid such a situation.
   d. Component assignation is driven by the innercore wire assignations and not the code assigned to the multicore

2. Assemblies
   a. Assemblies should be assigned a module code manually (required for correct processing). There is no automatic assignation for assemblies
   b. The components in the assembly get assigned separately from the assembly
   c. If the assembly and all the contents are tagged to break down into the same module then the whole assembly and its contents are populated on that module (Assembly attribute ‘Include-on-Bom’=True, Sub-components attribute ‘Include-on-Bom=False’). If not then the sub-components break-down to their target module with attribute ‘Include-on-Bom=True’ as standard non-assembly components
**Modular Breakdown**

When all assignations are performed, *Modular Breakdown* can populate module designs with applicable content from the composite based on the tags on different objects. In this process objects that were previously tagged through assignation will be examined to see if their assigned codes match the target modules' applicable codes.
Chapter F01
Function-based Assignation

Table of Content:

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of Function-based Assignation</td>
<td>17</td>
</tr>
<tr>
<td>“Standard” Assignation</td>
<td>18</td>
</tr>
<tr>
<td>“Mandatory” Assignation</td>
<td>20</td>
</tr>
<tr>
<td>“Mandatory, Factor” Assignation</td>
<td>24</td>
</tr>
</tbody>
</table>
Overview of Function-based Assignation

In function-based assignation methods, it is assumed that wires have been tagged with functional module codes. Functional-based assignation generates module code expressions based on the module codes set on related wires and the relationship between the codes. Four routines can be employed in this method:

- “Standard” Assignation
- “Mandatory” Assignation
- “Mandatory, Factor” Assignation
- “Technical” Assignation

The first three methods will be covered in this chapter with the final method; Technical Assignation being covered in a later chapter.

To launch function-based assignation on an object:
⇒ RMB on the object then select Modular / Assign Functional Module Code/Assign Production Module Code.
⇒ Alternatively, the action can be reached from the main menu item: Actions / Processing / Modular / Assign Functional Module Codes / Assign Functional Module Codes.

Multiple objects can be selected and tagged at one time. The sequence in which the system analyses the selected objects does not make a difference, with the exception of the situation where splices and multi-crimps are selected for technical assignation. This special case will be covered later as part of Technical Assignation.

To specify the required routines of assignation
⇒ Select the required routines from the FM/PM Code Assignation Configuration window

Multiple selections of routines is permitted with the exception of “Mandatory Only” and “Mandatory, Factor” which are mutually exclusive.

Selected assignation routines run in sequence (left to right in the UI) for each selected object. The system only moves on to the next routine if the current routine fails to assign. For example, for the Connector below all routines are selected, so the system will run the assignation routines in this sequence:

1. Standard
2. If failed, then: Mandatory
3. If failed, then: Factor
4. If failed, then: Technical
“Standard” Assignation

Standard Assignation is the first routine that can be run in function-based assignation. It considers only the logical relationships (inclusivity/exclusivity) between module codes found on the wires terminating at or passing through the object, then generates the resulting module code and assigns it to the object.

3 scenarios are illustrated in the following example on a connector:

1. **Same Codes**
   If the wires all have the same code then assignation is straightforward (the code is assigned to the object) as shown in the golden box below.

2. **Mutually Exclusive Codes**
   If multiple codes exist that are all mutually exclusive then the object is assigned each of the codes. This is because these codes will never exist together on the same finished harness in any configuration.
3. **Inclusive Codes**
   
   If multiple codes exist where one is inclusive of the other, the object will be tagged with the pre-requisite code, this will guarantee the object’s presence whenever any of the two module codes is present.

**Exercise 1:**

⇒ Run ‘Standard’ assignation on the harness objects as detailed in the exercise worksheet, ensuring you can explain the results of the assignation process in each case in light of code relationships defined.

⇒ Observe reporting of failed assignations in the output window
“Mandatory” Assignation

The second routine is Mandatory Assignation. It considers only the mandatory nature of the module codes found on the wires terminating at/passing through the object, then generates the resulting module code and assigns it to the object.

A successful Mandatory assignation on an object will only occur if the following conditions are satisfied, in the order listed below:

1. **Same Mandatory Codes**
   On the wires associated with the object, all mandatory codes are the same. In this case that code will be assigned.

2. **Single Mandatory Code**
   On the wires associated with the object, a single mandatory code exists that has no exclusive relationships with any other mandatory code applicable to this design. In this case that is the code that will be assigned irrespective of any other codes being analyzed for this object’s assignation.

Note that in the example above, Base1 is a mandatory code with no exclusive relationships with any other mandatory codes. This code will always be present so it is the one selected for assignation.
3. Mutually Exclusive Set of Mandatory Codes

On the wires associated with the object, a single, full set of mandatory, mutually exclusive codes exists. In this case all of the codes in the 'set' will be assigned to the object. For example, if Petrol, Diesel are all mandatory and mutually exclusive to each other (and not exclusively related to any other codes) then they are said to be a “full set of mutually exclusive codes” and hence all assigned to the object at which they exist.

In the examples above LHD and RHD are mutually exclusive; similarly, Petrol and Diesel are exclusive. Assignment was hence successful as in each case a single and full set of mutually exclusive codes was found.

If none of the above rules can be satisfied then Mandatory assignation will fail to make any assignations for the object in question.
In the example above, Petrol-Diesel form a full set, LHD-RHD form another full set, so the condition of a "single full set" has not been met, hence assignation fails.

A few more examples are illustrated below that further explain how Mandatory assignation operates:

As there is no relationship between each pair of codes (they can exist in combination) then Standard/Mandatory assignation would fail.

This would need to be resolved via the use of Assignation Factor, Technical or Manual assignation.

In this example above, Petrol-Diesel form a full set, LHD-RHD form another full set. However, there is another mandatory code present which is Base1. This code is not exclusive with any other codes on the design, so it is always applicable and hence it is assigned to the connector.

As the other mandatory codes can exist in combination Base1 is the only Mandatory code that will always be there hence it will be assigned.
The same concept from the previous example can be observed in this one too. Even though Petrol and LHD have no relationship to each other, they have a relationship to other codes in the design (Diesel, and RHD). Base1 is again, the only mandatory code with no exclusive relationships to any other mandatory codes on the design, so it is assigned to the connector.

In this situation, Petrol and LHD are present and are Mandatory…but a complete set is not present and so neither can be used. Assignation hence fails in this case.
Here, Base 1 & Base 2 are both mandatory so the connector is always going to be required but there is no information to enable the system to determine which module the connector should be assigned to, hence user intervention would be required. This particular example will be of interest for the next assignation routine; “Mandatory, Factor”

**Exercise 2:**

⇒ Run ‘Standard’ and ‘Mandatory’ assignation on the harness objects as detailed in the exercise worksheet, ensuring you can explain the results of the assignation process in each case in light of code relationships defined.
⇒ Observe reporting of failed assignations in the output window

**“Mandatory, Factor” Assignation**

This routine operates in the same way as mandatory assignation, with the additional behavior that when there is a conflict between codes on an object (when no relationships or mandatory tags exist that would easily determine assignation), an Assignation Factor setting on the code definition is used to determine assignation. The code with the highest assignation factor is chosen.
Assignations done using “Mandatory, Factor” may not represent the real, manufacturable modules but rather ones that can be used for initial costing purposes. In some cases, however, they do represent the true modules such as the case below where Base 1 & Base 2 are both mandatory so the connector is always going to be required. By using Mandatory, Factor assignation and setting up the factor setting, only one of the codes is selected and in this case the representation is accurate.

**Exercise 3:**

⇒ Set Factors on mandatory module codes as detailed in the exercise worksheet
⇒ Run ‘Standard’ and ‘Mandatory, Factor’ assignation on the harness objects as detailed in the exercise worksheet, ensuring you can explain the results of the assignation process in each case in light of code relationships defined.
⇒ Observe reporting of failed assignations in the output window
Chapter F02
Technical Assignation

Table of Content:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of Technical Assignation</td>
<td>26</td>
</tr>
<tr>
<td>Technical Module Code Creation</td>
<td>26</td>
</tr>
<tr>
<td>Combination Validation and Filtering</td>
<td>32</td>
</tr>
<tr>
<td>Technical Assignation Process</td>
<td>33</td>
</tr>
</tbody>
</table>
Overview of Technical Assignation

Where wiring with multiple codes exists at a component and the codes have no relationships, the most accurate thing to do is to create and assign new codes to the component that correspond to each of the potential code combinations.

For example, in the following situation, the connector is required for F6 alone, F7 alone and when F6 and F7 exist together.

Technical Module Code Creation

Technical module codes can be created in one of two ways:

1. Manual
2. Automatic

Manual Creation

The first step is to define technical module codes that will be used for each combination. In the previous example there are three combinations so we need three technical module codes defined.

A separate folder can optionally be created, inside which the technical codes can be created.

In this case new Technical Module codes (TM Codes) are required that correspond to each of the existing code combinations.
3 codes are created; TFM1, TFM2, TFM3

Next, combination folders need to be created. Functional module codes are placed into the combination folders and a technical module code(s) are created and mapped to the desired functional module code combinations.

To create a combination folder:
  ⇒ Highlight the combinations folder
  ⇒ Select ‘create a new combination’
  ⇒ Define a name for the combination folder
To add a functional code to a combination folder:
⇒ Locate the desired functional module code
⇒ Drag and drop the desired code into the combination folder

Once dragged, functional modules show as columns on the right hand side so the user can configure the combinations of interest

Rows are inserted to correspond to each required combination of the selected functional modules
A description can be added for each row if required.

For each row defined, a pre-defined technical module can be selected and mapped to that combination:

An automated process is available to create valid technical module codes based on a selection of functional codes:

⇒ Ensure the desired functional codes have been inserted into the combination folder
⇒ Highlight the combination folder and select “Insert Valid Rows”
⇒ Assign the desired codes for the generation process to the ‘selected’ area of the dialog

⇒ Select the “Assign TM Codes to Rows” button

This will automatically create TM codes and assign them to the generated combinations.
Automatic Creation

When technical assignation is run, Capital ModularXC can automatically generate combinations and corresponding technical modules by examining an object's related wires. If there are a number of functional modules with no relationships defined, a project preference per object type controls whether the routine should automatically generate combinations. The preference also specifies the maximum number of combinations to be generated, if generation is allowed.

For example, in the setting above, the preference is set to allow automatic generation of combinations with a limit of 5 combinations (reusing pre-defined ones first)

The “Reuse existing code” setting means that the assigned technical code will be used when the matching combination is found. Switching this to “Create new code” will trigger the creation of new codes for all combinations regardless of the existence of a matching technical code in the user’s definition.

Note

If the user has selected “Reuse code”, then the technical module code associated with a combination will be used as long as there is only one code associated. If there is more than one code, an error message will be displayed.

In the case where the number of combinations needed is greater than the maximum allowed in project preferences, the user needs to make a decision on an alternative method; either use factor to assign the object to one of the codes in question, or just use manually created combinations, assuming these are the common codes in use already.
When generation is configured the automatically generated combinations are displayed under the same location as user-defined ones and are marked with a special icon for differentiation.

The generated combination can be renamed afterwards.

### Combination Validation and Filtering

#### Combination Validation

When technical codes and associated combinations are created manually, mistakes can happen. The module code editor offers a method through which users can validate combinations and remove invalid ones.

- To highlight invalid combinations, select the “Select Invalid Rows” button.

This will highlight the invalid rows.

- To remove selected invalid rows, select the “Remove the Selected Rows” button.

#### Note

Technical modules cannot be added into a mandatory folder and a folder cannot be made mandatory if it contains technical codes. This is because of the “conditional” nature of the technical module; it depends on the presence or absence of other modules.
Combination Filtering

Combinations created using a given set of modules can be filtered to show only a subset of interest. Filtering can be done in one of two ways:

1. Filter by codes – this displays codes related to the selected code
2. Filter by design – this displays codes within a selected design

⇒ To filter by codes, right click inside a combination folder (right hand side) and click “Filter by codes…”

⇒ Choose which codes should be used in the filtered view

⇒ Click OK and the view is filtered accordingly

To filter by design:

⇒ Right click inside a combination folder (right hand side) and click “Filter by design…”

⇒ Select the applicable code type and design to be used for filtering

Select the required codes as done in the previous method. The view will be filtered based on the usage of the selected codes on the designs

⇒ To remove any filtering that is applied and revert back to the default view, select the “Turn off Filtered view” button

Exercise 1:

⇒ Create the technical codes as described in the exercise worksheet
Technical Assignation Process

Technical assignation is run as part of function-driven assignation. Typically it is selected in combination with other assignation options in the dialog.

Technical Assignation Processing Rules

The Technical module assignation process is applied to selected objects. The following steps are applied to each object:

1. Remove previously generated technical codes that were assigned to the object
2. Determine combinations & technical codes needed, based on code relationships on related wires
3. Propagate technical codes to the selected object(s)

For most objects, this is carried out as follows:

1. Blow-away previously generated technical codes
   - Remove all generated codes assigned to the object
2. Determine combinations & technical codes needed, based on code relationships on related wires
   - Collect all the driving codes for the object
   - Stop here if we have less than two driving codes as standard or mandatory assignation should be sufficient
   - Otherwise, select or generate the combinations of functional codes present (given the settings in project-preferences)
   - Assign technical codes to the combinations if they do not have them (given the settings in project-preferences)
   - Return the list of technical codes to be propagated.

Splices and Multi-crimps

Special behavior exists in the technical assignation process for splices and wire multi-crimps (multi-term’s).

- When assigning Splices and multi-crimps to technical modules the system also tags the wires with the technical module codes. Note that this tagging is done via the Generated Functional Codes field on the wires
- Modular breakdown recognises that the wires should only be added to the technical modules and NOT to the modules associated with the original functional codes on the wires
In the example above, the technical codes are calculated for the splice, and then codes that are applicable to individual wires will be applied to them as long as the technical code combination contains the original functional module codes on those wires.

- If the other end of the wire is not connected to a splice or multi-crimp, then all previously assigned technical codes are removed and the codes are regenerated.

- If the other end of the wire is connected to a splice or multi-crimp, then the only previously assigned technical codes that are removed are:
  - Those that have not been assigned from the other splice or multi-crimp
  - Those that have become invalid.

Remaining ones are maintained on the wire.

In cases where an object’s assignation includes both multi-crimps as well as splices, there is a need to determine which one of the two to assign first. For this purpose, an assignation order setting is present in project preferences to specify which is assigned first:

**Exercise 2:**

⇒ Run ‘Technical’ assignation on the harness objects as detailed in the exercise worksheet, ensuring you can explain the results of the assignation process in each case in light of preferences defined.

⇒ Observe reporting of failed assignations in the output window.
Chapter F03
Rule-based Functional Assignation

Table of Content:

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of Rule-based Functional Assignation</td>
<td>40</td>
</tr>
<tr>
<td>Contextual Rule-based Assignation</td>
<td>40</td>
</tr>
<tr>
<td>Global Rule-based Assignation</td>
<td>42</td>
</tr>
</tbody>
</table>
Overview of Rule-based Functional Assignation

As an alternative to the interaction with individual objects on the diagram, users can drive functional assignments on objects via a pre-configured set of rules defined on the design. The assignation task then becomes one of configuring these rules to match the users required settings. A single click action would start the assignation process.

Contextual Rule-based Assignation

Using the constraints “Assign Module Codes for Selected Objects” and “Assign Technical Modules for Selected Objects” the process of function-driven assignation on objects can be automated. A rule is defined with a constraint per object type to determine the routine to be used when the assignation action is run on that object.

To define a rule:

⇒ RMB on the design name and select Edit / Properties / Rules tab (alternatively it can be defined on a project level through main menu action Edit / Rules)
⇒ Select Add Constraint then select one from the available list. Once selected it can be configured

The following steps show how to configure the “Assign Module Codes for Selected Object(s)” constraint.

⇒ Highlight the constraint and click ok
⇒ Click object and select the required object from the drop down list
⇒ Click Module space and select the required module code type
⇒ Click method and select the required assignation method

The steps can be repeated for each object type as required

The constraint “Assign Module Codes for Selected Objects” is pluggable. By assigning a custom constraint to it the assignation of each selected object will be driven by the behavior from the custom constraint

Once assignation constraints have been added to the design the interactive UI no longer appears when running the assignation routine; the system just uses the constraints instead.
Global Rule-based Assignation

For a global rule-driven assignation of module codes to all design objects, a custom constraint “Assign Module Codes for All Design Objects via Plugin” can be used to drive the rule operation.

Exercise:

⇒ Clear all previously generated assignations
⇒ Redefine the assignation routines using constraints as detailed in the exercise worksheet, and run the assignation action. Ensure you can explain the results of the assignation process in each case in light of constraints defined.
Chapter F04
Interactive Assignation

Table of Content:

<table>
<thead>
<tr>
<th>Interactive Assignation</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive Assignation</td>
<td>43</td>
</tr>
</tbody>
</table>
Interactive Assignation

Once the automatic tagging process completes, some objects would usually need a more detailed study due to their special nature. The experienced process engineer would use the interactive tools to complete the assignations for these objects. This process is achieved through the use of the following facilities and steps.

1. Analysis and sorting of unassigned objects using the Code Browser and Module Explorer
2. Managing different “slicing and merging” scenarios of modules using the Scratchpad
3. Assignment of objects to their final modules using drag and drop into the Code Browser

Analyzing Unassigned Objects

Using the FM Codes tab, unassigned objects can be seen under the “Unassigned” folder. The dimmer button and slider can be used to hide/show objects more clearly on the diagram, this could be used to help differentiate unassigned from assigned objects.

In the example, the contents of FM01 are highlighted on the diagram.

The following example displays objects in the unassigned folder.
The Module Explorer enables the user to study the unassigned objects in detail. By clicking on the Module Explorer button, a new tab opens in the output window, to which objects can be dragged from the FM Codes tab (usually from under the unassigned node). Once objects are dragged into it, they can be sorted based on different criteria through its columns.

Drag data into the module explorer tool
Note
The Module Explorer can be launched as a floating window using the CTRL button and pressing on the Module Explorer button.

Objects within the explorer window can be ordered according to the number of cavities, wires or unassigned wires:
⇒ Select the required column name and click the ascending/descending button

If an object is no longer of interest it can be selected and removed from the window by clicking on the ‘remove selected’ button.

The ‘Add unassigned wires’ button will associate wires to the selected connector in the module explorer.

Exercise 1:
⇒ Refer to the exercise worksheet to investigate the functions available within the module explorer

Managing Different Scenarios
A scratchpad facility is available to help investigate different combinations of modular codes which could help users to decide whether to make certain design changes such as merging options together. The objects in question are dragged into the scratchpad to create a temporary module.

To launch the Scratchpad go to View / Scratchpad.
⇒ Add the required selection of objects to the scratchpad wither from the Code Browser or the Module Explorer

This column displays the functional or production code associated to the object. PM expression is displayed if the Functional Module explorer is active.

Click on a column header to sort that column
The content of the scratchpad can be viewed on the diagram using the dimmer and slider. This is to make sure that the content satisfies what the user wants included in the module.

![Scratchpad Diagram]

**Final Assignation**

Once the module content is satisfactory to the user, it can be assigned to a module code. This is done via drag and drop from the Scratchpad (or directly from the Module Explorer) into a pre-defined module code in the Code Browser.

**Exercise 2:**

⇒ Refer to the exercise worksheet to complete the exercise
Chapter C04
Module Relationships

<table>
<thead>
<tr>
<th>Table of Content</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifying Code Relationships</td>
<td>49</td>
</tr>
</tbody>
</table>
Identifying Code Relationships

It is useful to report on the relationships between different module codes. To launch the reporter:

⇒ Select Tools/Report Module Relationships

The Module Relationship Reporting facility can help users identify related codes in the same space, or in different spaces, highlighting the type of relationship between them.

These relationships include

**Same Code Space**

1. **Inclusive Codes**
   Where code A is inclusive of code B (B is the pre-requisite code needed for A to be present)

   In the example above, Functional Codes are used for both Source and Output.

2. **Technical Codes**
   Where codes TM1, TM2 and TM3 are technical codes corresponding to the different combinations of FM1 and FM2; if only FM1, only FM2 or FM1&FM2 are needed, the facility will indicate which technical module codes are needed in correspondence. This mapping information can be helpful in identifying a link between the original specification of module content, and the real content which would be manufactured.
In the example above, Functional Codes are selected in the Source Codes field and the Report is generated based on the Required Functional Codes. This is an example of same-code-space reporting.

**Different Code Spaces**

3. **Mapping Codes**
   Where an object is tagged with both functional and production modules, the facility helps users identify the mapping between both. Users can use this to analyze the breakdown/merging of different functional modules for costing purposes when converted to the production module space.

**Exercise 1:**

⇒ Use the Module Relationship Reporter to study the relationship between functional module combinations and corresponding technical as well as production module codes as detailed in the exercise worksheet.
Chapter C05
Module Processing

<table>
<thead>
<tr>
<th>Table of Content</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Breakdown</td>
<td>51</td>
</tr>
<tr>
<td>Reference Components</td>
<td>51</td>
</tr>
<tr>
<td>Terminal/Seal Locations</td>
<td>53</td>
</tr>
<tr>
<td>User Vs. Generated Codes</td>
<td>53</td>
</tr>
<tr>
<td>Tube Processing</td>
<td>53</td>
</tr>
</tbody>
</table>
Module Breakdown

In a similar way to Composite Breakdown, a Modular Breakdown process can break down a modular composite into individual functional and production modules (which have been created prior to the breakdown process).

The settings of the breakdown process are largely the same as Composite Breakdown for derivatives, with the exception of two items that are detailed below:

1. Reference Components
2. Terminal/Seal Location (Connector vs. Wire End)

⇒ To start Module Breakdown go to Tools / Harness Processing
⇒ Select the Module Breakdown check box
⇒ Select any other processing options needed for the composite/modules.

Reference Components

In order to enable cavity component selection on modules where the connector does not exist, Modular Breakdown will include such ‘referenced’ connectors with their attribute ‘Include on BOM’ unset.

This way, cavity components can be selected and the connector doesn’t appear on the BOM for this module.

An example of this scenario is shown below:

Connectors C-1 and C-2 belong on Module 1. There are wires on Module 2 and Module 3 that will terminate on these connectors upon assembly.
On Module 2 and Module 3, C-1 can be found as a reference component.

The connector is not assigned to Module 2 or Module 3.
Terminal/Seal Locations

An option “Terminals/Seals stay with Wires (not Connectors)” is available under harness processing options. This ensures that terminals or seals remain with the wires rather than the connectors. This option is checked by default. When this option is unchecked, terminals and seals will be added to the module design where the connector is present.

This functionality supports the ability for a module design to just contain a connector and its terminals as a kit of parts.

User Vs. Generated Codes

If a User (manual) assignation was performed on an object, this is used first by the Modular Breakdown process. Automatically assigned codes come second in priority of breakdown.

A special rule applies in Modular Breakdown for wires which were automatically assigned technical module codes. In this case, the technical codes take precedence over the user assigned ones. This situation may happen with splice or multi-crimp assignation.

Tube Processing

Tubes have some special behaviour during harness processing.

- If a tube was assigned to a mandatory module, then the bore size (width) of the tube is calculated based on worst-case scenario of wires passing through it rather than just the mandatory wiring.
- If a tube was assigned to technical codes then on each technical module, the tube would have a different bore size calculated based on the wires passing through the tube on the particular technical module in question.

Exercise:

⇒ Process module designs selecting processing options as detailed in the exercise worksheet.
Chapter C06
Styling

Table of Content:

<table>
<thead>
<tr>
<th>Module Code Attribute Styling</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Design Information Styling</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>56</td>
</tr>
</tbody>
</table>
**Module Code Attribute Styling**

The attributes highlighted in previous chapters for functional and production modules are present on all object types to be used in styling. The attributes can be accessed in the same way as other attributes today (Attribute styling, Composite Text styling, Query styling, etc.).

The attributes available in the style set are listed below:

1. Generated Functional Module Code
2. Generated Production Module Code
3. User Functional Module Code
4. User Production Module Code

The example below shows one of the attributes being added to a connector’s attributes – a typical use case for styling modular information on the diagram.

And this is how it looks on the diagram:

A question mark can also be styled under Composite Text to be displayed when the value of the attribute is null (via a visibility query). This helps to identify unassigned objects on the diagram.
And this is how it looks on the diagram:

Another example below shows the module code attributes being displayed in a wire list table.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Display Title</th>
<th>Visible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wire Number</td>
<td>Wire No.</td>
<td></td>
</tr>
<tr>
<td>User Functional Module Codes</td>
<td>FMC-user</td>
<td></td>
</tr>
<tr>
<td>User Production Module Codes</td>
<td>PMC-user</td>
<td></td>
</tr>
</tbody>
</table>

And this is how it looks on the diagram.

<table>
<thead>
<tr>
<th>Wire No.</th>
<th>FMC-user</th>
<th>PMC-user</th>
</tr>
</thead>
<tbody>
<tr>
<td>1016</td>
<td>F1</td>
<td>PM Code 2</td>
</tr>
<tr>
<td>2N-SPKR-LF-1-519</td>
<td>F6</td>
<td>PM Code 2</td>
</tr>
<tr>
<td>2N-SPKR-LF-2-521</td>
<td>F2</td>
<td>PM Code 2</td>
</tr>
<tr>
<td>2N-SPKR-LR-1-523</td>
<td>F2</td>
<td>PM Code 2</td>
</tr>
<tr>
<td>2N-SPKR-LR-2-525</td>
<td>F2</td>
<td>PM Code 2</td>
</tr>
<tr>
<td>2N-SPKR-LT-1-684</td>
<td>F2</td>
<td>PM Code 2</td>
</tr>
<tr>
<td>2N-SPKR-LT-2-686</td>
<td>F2</td>
<td></td>
</tr>
<tr>
<td>4N-FUEL-LID-563</td>
<td>F6</td>
<td></td>
</tr>
<tr>
<td>4N-INACTIVE-568</td>
<td>F7</td>
<td></td>
</tr>
<tr>
<td>4N-INACTIVE-569</td>
<td>F6</td>
<td></td>
</tr>
<tr>
<td>4N-INACTIVE-570</td>
<td>F6</td>
<td></td>
</tr>
<tr>
<td>4N-POWER-1-571</td>
<td>F6</td>
<td></td>
</tr>
<tr>
<td>4N-TRUNK-575</td>
<td>F3</td>
<td></td>
</tr>
<tr>
<td>5N-CTSY-LP-577</td>
<td>F3</td>
<td></td>
</tr>
</tbody>
</table>
Module Design Information Styling

Two new tables are available for the representation of module design information:

1. Functional Module Information
2. Functional Production Information

These two tables can be added at a design level on the composite design. The tables depict the applicable codes for each module design created under the composite.

To access the table style:

⇒ Navigate to the desired style set
⇒ Design/Decorations/tables/Add
⇒ Select the desired table and style

The following screenshots show the list of attributes that can be included in these tables.
Design-level Modular Attributes

The following attributes are available for styling on the design-level.

1. Module Parent
2. Module Child
3. Functional Module Parent
4. Functional Module Child
5. Functional Module Code Mgr Overridden
6. Production Module Parent
7. Production Module Child
8. Production Module Code Mgr Overridden
9. Num Functional Modules
10. Num Functional Module Parents
11. Num Functional Module Children
12. Num Production Modules
13. Num Production Module Parents
14. Num Production Module Children

Exercise:

⇒ Style the composite and functional/production modules as detailed in the exercise worksheet.
**Introduction**

In a "customer specific harness" flow the OEM often provides the supplier with electronic design data, typically in XML, DSI or KBL formats. The data is supplied as a single file containing full composite and module definition or as a set of separate files each containing the definition of a single module.

The recognized method for importing and managing changes to the data throughout the design lifecycle is using Change Manager into Capital ModularXC.

This section describes how to create modular harness data in Capital ModularXC and then how to identify and update that data with changes received which would be expected during the design lifecycle. The process can be run incrementally or in a single step.

Change Manager provides the mechanism to import MCAD and Module data into Capital ModularXC

⇒ **Select Bridge/Change Manager**

⇒ **Incoming Harnesses Panel**: Indicates which module files are loaded and whether they are linked

⇒ **Harness Objects Panel**: shows the bundles and components and their link status

⇒ **Process**: Will process the incoming data and place the information on the harness diagram

⇒ **Display**: Will display the preview pane for the incoming harness content

⇒ **Incoming**: Allows the user to locate and import module data and map modules once the harness has been linked

---

**Select Harness for Import**

Available harnesses will be automatically displayed if they are in the defined Harness repository. The harness repository can be defined under Incoming/Bridge Options/Import.

Alternatively select the **Incoming** button to navigate to the xml file required for import.
It is possible to view the data as shown above or in the preview pane by selecting View / Change Manager or clicking the Display tab:

**Displaying Items**

Select View / Show Labels or the icon ❂ to show or hide the text labels.

Select View / Show Splices or the icon ❃ to show or hide all splices.

Select View / Show Fixings or the icon ❄ to show or hide any fixings (clips/grommets).

Select View / Show Protections or the icon ⚥ to show or hide any protections/insulations.

Select View / Show Connectors or the icon 🔐 to show or hide any connectors.

**Query Editor**

The Filter button in the change manager dialog allows the users to quickly determine the differences between the source and target data e.g.

⇒ what has been added, deleted or changed in either model
⇒ if the lengths have changed
⇒ bundles that have been affected by a topology change

To find new bundles, connectors, splices etc, select Unlinked objects and the system will display only new Module bundles i.e. not yet linked in the Change Manager and in the Preview (unlinked objects are shown in green, whereas linked objects are colored in grey).

**Note**

Link and Unlink functions are used after the first import.
Link Objects

It is possible to manually link an object from the incoming Module to the Capital ModularXC object.

⇒ Select the object in the Change Manager pane or the preview pane
⇒ Select Actions / Link or Right Click and select Link

⇒ Select the object to link to in the Capital ModularXC design

AutoLink Objects

It is possible to automatically link objects from the incoming Module to the Capital ModularXC object.

⇒ Select the object in the Change Manager pane or the preview pane
⇒ Select Actions / Autolink or Right Click and select AutoLink

The system can automatically link objects that have matching properties such as the Object Name, Electrical ID (UID), Part Number or Length.
Attribute/Property Mismatch

There may be occasions where the objects are linked but a property is different e.g. a bundle exists in both models and has been linked but the bundle length is different (for instance, it has been changed in the source module).

This will be shown in the Change Manager as an **Attribute/Property Mismatch**.

To resolve this issue, synchronize the data by:

- Select the mismatched object/s
- Select **Actions / Synchronize** or **Right Click** and **Synchronize**

**Note**

When objects are linked, the system automatically performs the synchronization.

---

Connectivity Mismatch

There may be occasions when the layout changes e.g. a branch exists in the source module but not in Capital ModularXC, length changes etc. This will be shown as follows:

- Missing Branch: No Capital Name
- Connected Branches: connectivity Mismatch

To resolve the issue **Place** the additional branch.
Mapping Modules

For every Module Code used in incoming data set, Change Manager allows the creation/mapping of a module design and associates this to the corresponding composite design.

Since the incoming data set can contain a high number of modules the mapping action of the modules is performed in a single dialog containing all modules.

The user maps modules by selecting a context menu against the harness file or by selecting the map module icon. The modules contained in the incoming harness file are listed in the incoming column. The Capital modules are created automatically by selecting Automap, manual Capital modules are created by selecting Edit. Edit can also be used to modify Automap created modules.
Report Harness

Selecting View/Harness Report in Change Manager will display a report in the ModularXC reports window under the bridges tab. The following filterable columns are displayed.

**Severity** – Error, Warning, Info
**Operation** – Place, Autolink, Synchronize, Link, Unlink, Delete, Map Module etc
**Object type** – The type of object Harness, Module, Bundle, Splice, Wire, SpotTape etc
**Object** – Object on which operation is done
**Message** – A message to the user to inform them what has been done

<table>
<thead>
<tr>
<th>Severity</th>
<th>Operation</th>
<th>Object type</th>
<th>Object</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>Place</td>
<td>Harness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warning</td>
<td>Autolink</td>
<td>Module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td>Synchronize</td>
<td>Bundle</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Link</td>
<td>Splice</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unlink</td>
<td>Wire</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete</td>
<td>SpotTape</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Map Module</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note**
Importing or saving a harness triggers a report dialog

Exercise 1: Import Module Flow
⇒ Overview instruction as per your exercise sheet

Revising Modules

Importing revised module data and opening the mapping dialog the user is able to identify actions that need to be performed. Colors are used to indicate the status of each module.

Black - Mapped no action required
Green - Indicates that the incoming module data is different and the design requires a revision
Red - Name conflict (Ok button will be disabled)
Blue - Design is new i.e. after the automap has been used

Deletion

Once the user has identified the differences between the incoming and target data he is able to delete items in the target data individually if he wishes to validate each change or automatically if manual operations are undesirable.

The query editor can be used to identify “No longer Linked Objects” which can be deleted via a context menu.

Exercise 2: Revised Module Flow
⇒ Overview instruction as per your exercise sheet