

# EMIC CODIFY.

---

## Introducción.

EMIC es una plataforma para desarrollar todos los componentes necesarios para implementar una solución completa dentro del paradigma industria 4.0, incluyendo, dispositivos electrónicos industriales, paneles de control WEB, HMI, funciones lambda para el despliegue de modelos de ML, etc. Está compuesta por módulos de software ordenados en bibliotecas, una familia de módulos de hardware, y un entorno de desarrollo de alto nivel.

El uso de la plataforma como herramienta de desarrollo, permite acelerar los tiempos de salida al mercado y eliminar la deuda técnica inherente a todo proceso tecnológico, mejorando significativamente el método tradicional de desarrollo de producto de forma disruptiva.

El éxito de EMIC se explica en el modelo de trabajo colaborativo basado en un esquema modular, escalable, ordenado y autodescriptivo, que permite dividir las etapas necesarias para obtener soluciones complejas en tareas más simples que son llevadas a cabo por distintos actores en distintos momentos.

Por un lado desarrolladores crean bibliotecas de con el código para manejar recursos de hardware, modelos de ML, widgets para dashboard, filtros digitales, protocolos, servicios webs, etc. Estas bibliotecas incluyen metatexto para su clasificación y descripción.

EMIC interpreta el metatexto que describe el contenido de las bibliotecas y lo presenta en una interfaz de usuario como ícono o etiqueta. Permitiendo que el código sea accesible desde un entorno de programación a alto nivel para los integradores.

Al momento en que los integradores (o los mismos desarrolladores) deseen crear aplicaciones para una solución a requerimiento específico, desde el editor EMIC podrán generar un script que contenga la lógica de funcionamiento de la solución. Ese script es utilizado por otro proceso del sistema, que accede al código creado por los Desarrolladores y lo integra para obtener un resultado final, que puede ser un código ejecutable en el hardware específico, una aplicación web que corre en cualquier dispositivo, o un modelo de machine learning que puede ser ejecutado en un proceso en la nube.

Los archivos que contienen el código creado por los desarrolladores estarán alojados en repositorios de GitHub y organizados respetando las especificaciones de EMIC logrando que el sistema pueda reconocer e interpretar el contenido. Además del código con las funciones que integran cada librería se agregan comandos EMIC, que son instrucciones que el sistema ejecutará en el caso que sea invocado por el sistema, es decir cuando forme parte de una solución final.

## EMIC Codify.

Cuando el sistema EMIC fusiona el script desarrollado en la etapa de integración con las distintas librerías que integran la solución, lo hace siguiendo las indicaciones proporcionadas por los desarrolladores.

Estas indicaciones se hacen utilizando comandos en un lenguaje de programación creado especialmente para el manejo de documentos de texto y código llamado EMIC Codify.

Utilizando EMIC Codify se establece la relación entre la definición de los módulos EMIC y sus dependencias, así como las dependencias de cada archivo dentro de una biblioteca. Además este lenguaje de programación permite moldear a cada librería dentro de un proyecto específico, es decir que los archivos pueden tomar distintas formas, adaptándose automáticamente a la necesidad de cada solución.

Cuando se compila el proyecto, una vez que esta listo el script, el sistema comienza a generar la solución interpretando los comandos ubicados en un archivo llamado **generate.emic**. Estos comandos indican los pasos a seguir, incluyendo las rutas con las ubicaciones de todas las dependencias del proyecto. A medida que se invoca a los archivos para formar parte de la solución, se ejecutan los comandos EMIC que se encuentran en esos archivos.

Los comandos que con la forma **EMIC:xxxx(yyyy)** indican una acción que se ejecuta inmediatamente o evalúan una condición para determinar si el próximo bloque de código debe ser interpretado o ignorado.

El resto de la líneas de texto que no contienen comandos será enviadas a un archivo de salida que formará parte de la solución final. Aunque si dentro de este texto se encuentra una expresión de la forma **.{xxx.yyy}**, será reemplazado por otro texto que fué definido previamente.

Para acceder a los distintos archivos contenidos, tanto en el repositorio, como en el proyecto que se está editando, se usa un sistema de rutas y volúmenes lógicos. De esta manera para referirse a un archivo en particular no hace falta conocer su verdadera ubicación. Los volúmenes lógicos son:

Volumen	Referencia.
DEV:	Volumen donde se encuentran los archivos del repositorio.
TARGET:	Volumen donde se van almacenando los archivos generados en el proceso de compilación.
SYS:	Volumen donde se crean los archivos de configuración de cada aplicación.
USER:	Volumen don se ubican los archivos del usuario (integrador)

Comandos EMIC Codify:

Nota: Los términos entre corchetes son opcionales y entre dobles corchetes se pueden repetir varias veces.

setInput

Indica al sistema que se debe procesar un archivo. Luego el sistema continua ejecutando las lineas restantes del archivo que se estaba procesando. Al ejecutar el comando, se pueden definir mediante pares de clave-valor un conjunto de **macros** que serán usada como texto variable durante el procesamiento.

Sintaxis:

```
EMIC:setInput([origin:][dir/]file[,key=value])
```

Definiciones:

Nombre	Opcional	Descripción
origin	SI	volumen en el que se ubica el archivo. Si se omite se usará el volumen actual.
dir	SI	ruta del archivo.
file	NO	nombre del archivo.
key	SI	nombre de cada parámetro que será utilizado en la interpretación del archivo.
value	SI	valor tomará el parámetro que reemplaza a la clave en la interpretación del archivo.

setOutput

Establece el archivo de salida. Todo el texto generado durante el procesamiento siguiente al comando será enviado al archivo indicado, si el archivo no existe se creará en el momento que se intente escribir en él. Cada vez que se ejecuta este comando la salida actual se almacena para ser restablecida posteriormente.

Sintaxis:

```
EMIC:setOutput([target:][dir/]file)
```

Definiciones:

Nombre	Opcional	Descripción
target	SI	volumen en el que se encuentra el archivo de salida. Si se omite se usará el volumen de salida actual.
dir	SI	ruta del archivo. Si no existe, se crea. Si se omite, se usará la ruta de salida actual.
file	NO	nombre del archivo. Si el archivo no existe, lo crea

restoreOutput

Restablece es archivo de salida al destino anterior.

Sintaxis:

```
EMIC:restoreOutput
```

copy

Indica al sistema que se debe procesar un archivo y enviar el texto generado durante el procesamiento a un archivo de salida especificado, si el archivo no existe se creará en el momento que se intente escribir en él. Al ejecutar el comando, se pueden definir mediante pares de clave-valor un conjunto de **macros** que serán usada como texto variable durante el procesamiento.

Sintaxis:

```
EMIC:copy([origin:][dir1/]file1,[target:][dir2/]file2[[,key=value]])
```

Definiciones:

Nombre	Opcional	Descripción
origin	SI	volumen en el que se encuentra el archivo de entrada. Si se omite se usará el volumen.
dir1	SI	ruta del archivo de entrada. Si se omite, se usará la ruta del archivo procesado actualmente.
file	NO	nombre del archivo de entrada.
origin	SI	volumen en el que se encuentra el archivo de salida. Si se omite se usará el volumen de salida actual.
dir1	SI	ruta del archivo. Si no existe, se crea. Si se omite, se usará la ruta de salida actual.
file	NO	nombre del archivo se salida. Si el archivo no existe, se crea
key	SI	nombre de cada parámetro que será utilizado en la interpretación del archivo.
value	SI	valor tomará el parámetro que reemplaza a la clave en la interpretación del archivo.

.....

define

Define una nueva **macro** formada por una clave y un valor, que sera almacenada para su posterior utilización.

Sintaxis:

```
EMIC:define([group.]key,value)
```

Definiciones:

Nombre	Opcional	Descripción
group	SI	Nombre del grupo en que se define la macro, si se omite se usa el grupo por defecto <b>global</b>
key	NO	Clave que identifica a la <b>macro</b> .
value	NO	Texto se guarda de la <b>macro</b> *

unDefine

Borra una **macro**.

Sintaxis:

```
EMIC:unDefine([group.]key)
```

Definiciones:

Nombre	Opcional	Descripción
group	SI	Nombre del grupo en que se define la macro, si se omite se usa el grupo por defecto <b>global</b>
key	NO	Clave que identifica a la <b>macro</b> .

.{group.key}.

Es comando se usa en las partes del texto que queremos que sean sustituidos por otro texto definido previamente

Reemplaza .{**group**.key}. por el valor asignado con EMIC:define([group.]key,value).

EMIC:foreach(**group**) .{Item}. EMIC:endfor

Sintaxis:

if

```
EMIC:if(condition)
```

Definiciones:

Nombre	Opcional	Descripción
``		
``		
``		
``		
``		

Sintaxis:

```
EMIC:elif(condition)
```

Definiciones:

.....

Sintaxis:

```
EMIC:ifdef(macro)
```

Definiciones:

Nombre	Opcional	Descripción
``		
``		
``		
``		
``		

.....

Sintaxis:

```
EMIC:ifndef(macro)
```

Definiciones:

Nombre	Opcional	Descripción
``		
``		
``		
``		
``		

.....

Sintaxis:

```
EMIC:else
```

Definiciones:

.....

Sintaxis:

```
EMIC:endif
```

Definiciones:

---

`/**< */ /*RFI ModuloReferencia //RFI TAG:driverName= /*RFI`