# Erstellung neuer Bausteine für die MICAS Workbench

Um einen neuen Baustein in die Workbench einzubinden, muss dessen Spezifikation bekannt sein. Sie findet sich in der Header Datei der Implementierung.

## Voraussetzungen

* Die komplette MICAS Workbench muss ausgechekt sein <http://10.49.41.4/svn/wisag_wb/trunk>
* Siehe auch <http://10.49.41.4/svn/wisag_wb/trunk/docuware/de.wisag.automation.micas.docuware/MicasWorkbenchDeveloperSetup.docx>
* Man muss die MICAS Workbench aus Eclipse heraus starten können.

## Was muss erzeugt werden

* DML Modell für den Diagrammeditor
* Die Eclipse Extension für das DML Modell
* Das HW Modell für den Compiler, den MIC Editor und die Diagrammvalidierung

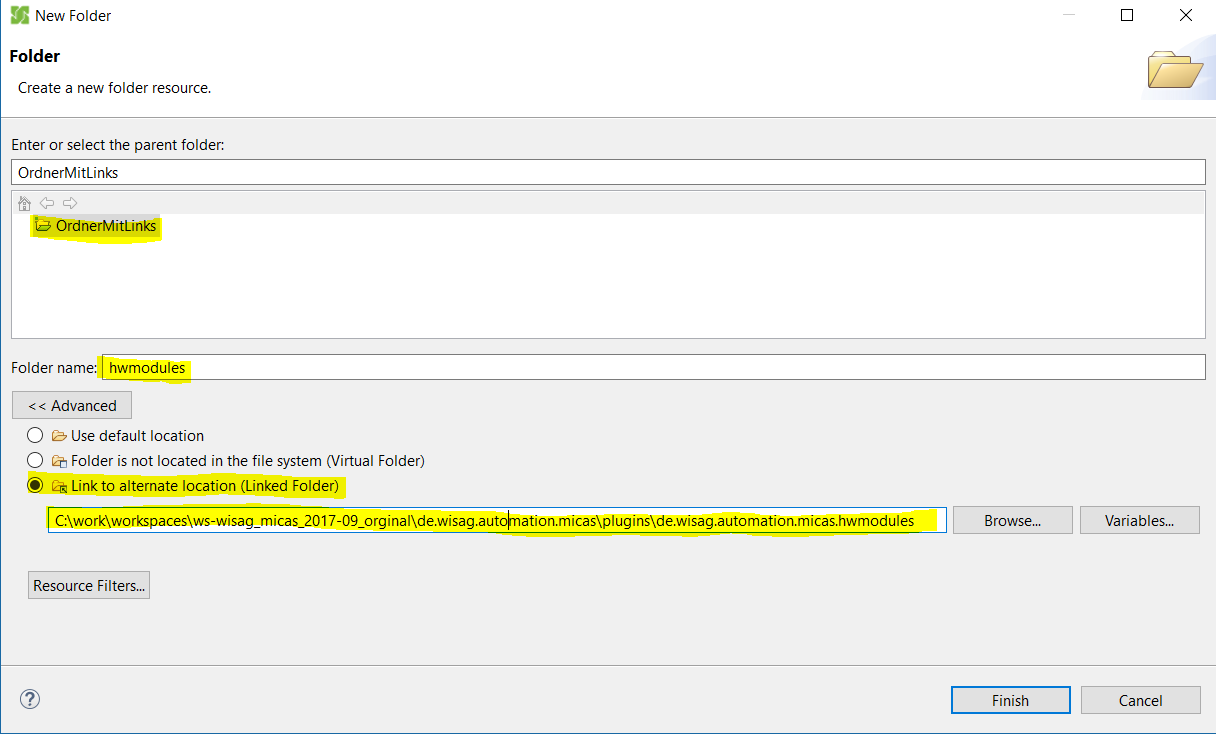
## Allgemein

Bei den Modellen handelt es sich um XMI Dateien. Eine besonders wüste Form von XML. Man kann die auch komplett händisch erstellen, hat dann jedoch das Problem der XMI ID’s. Deshalb ist zu empfehlen ein Gerüst der Dateien in der Micas Workbench zu erstellen und sie dann in der Entwicklungsumgebung nachzubearbeiten. Der Grund für die Trennung zwischen DML Modell und HW Modell ist, dass es an der Hardware Ports gibt, die im Diagramm gar nicht zu finden sind (Direct Value).

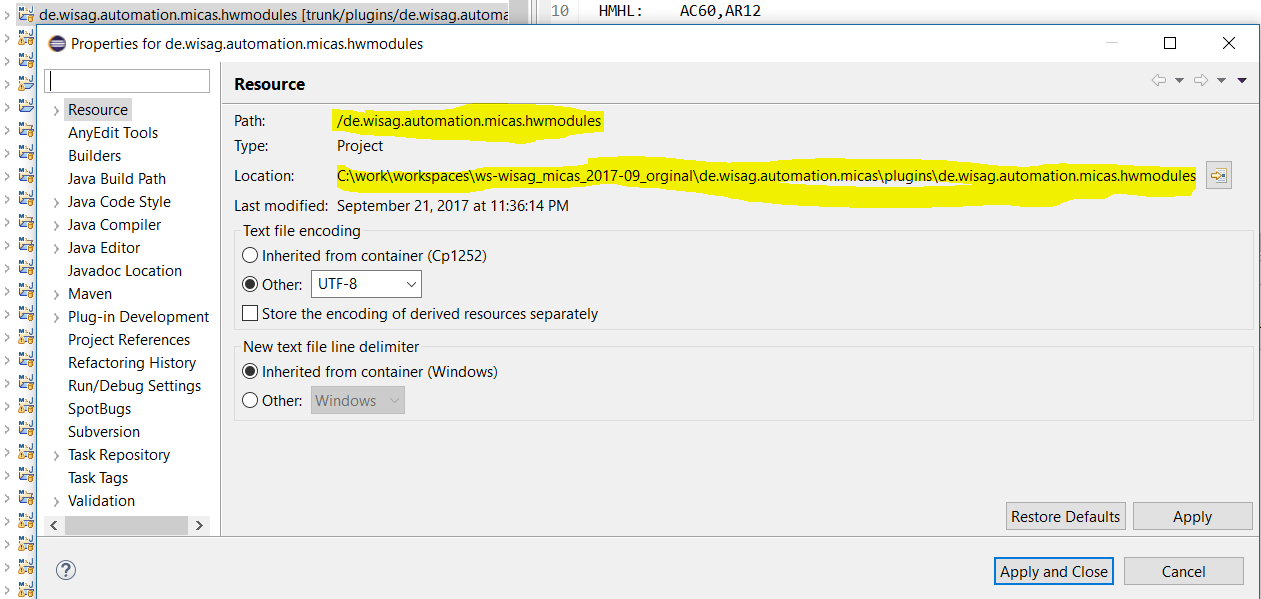
## Einbinden von externen Resourcen in ein MICAS Workbench Projekt

In der MICAS Workbench können Ordner aus dem Dateisystem in Projekte verlinkt werden. Dazu benötigt man den Physischen Pfad des Ordners.

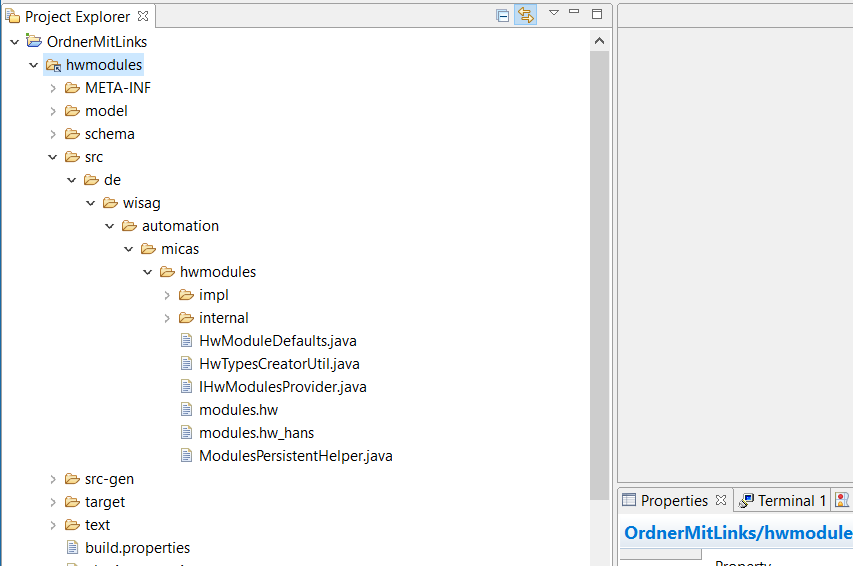
1. Man erzeugt ein leeres MICAS Projekt
2. Man Erzeugt einen neuen Ordner (New Folder) und wählt unter Advances „Link to alternate location (Linked Folder)“



Der eingetragene Pfad ist die Location des Projektes aus der Entwicklungsumgebung. Diesen erhält man über „Properties“ des entsprechenden Projektes.



Danach ist der Ordner aus der Entwicklungsumgebung ‚verlinkt‘ und man kann hineinschauen als wäre es ein normaler Ordner.



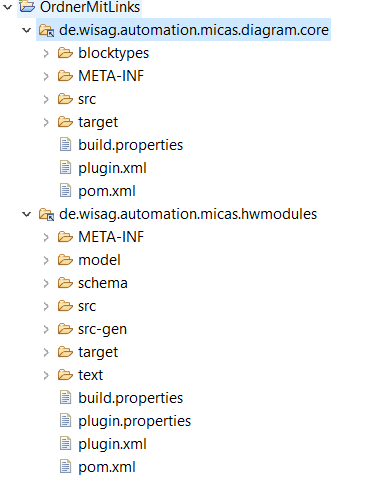
## Einbinden der nötigen Entwicklungsprojekte

Den im letzten Kapitel beschriebenen Vorgang wendet man auf folgende Projekte an

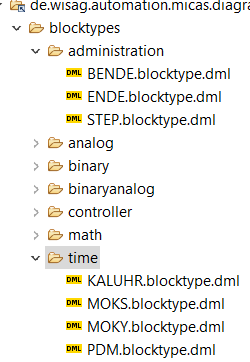
de.wisag.automation.micas.hwmodules

de.wisag.automation.micas.diagram.core

Danach enthält das Beispielprojekt die nötigen Entwicklungsprojekte in der Micas Workbench.



Im diagram.core Projekt kann man nun die Blockdefinitionen anschauen.

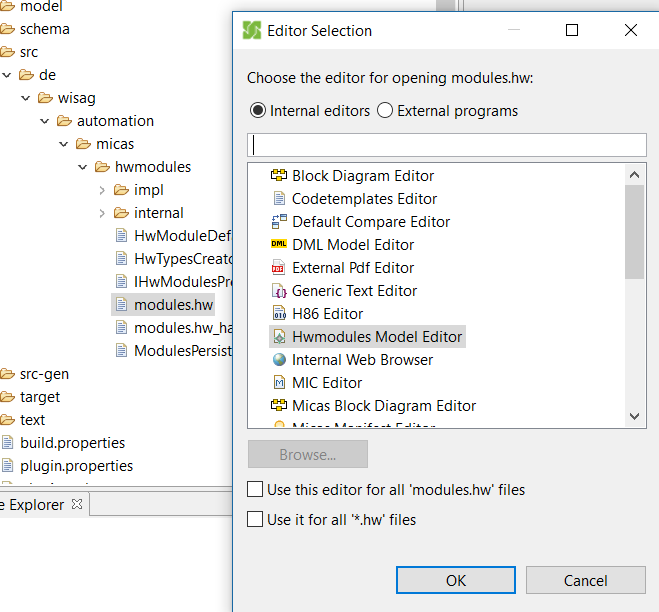


Das hwmodules Projekt enthält eine wichtige Datei, für die auch ein Editor verfügbar ist :

hwmodules/src/de/wisag/automation/micas/hwmodules/modules.hw

Diese Datei kann wie folgt geöffnet werden :

Datei anwählen->Rechte Maustaste->Open With Editor



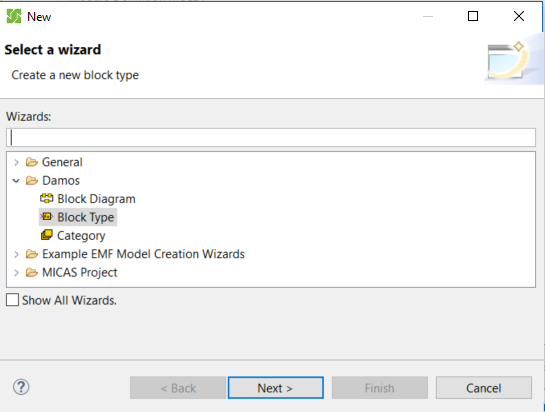
Der Hwmodules Model Editor erlaubt das Editieren der Hardwaredefinitionen für den Compiler.

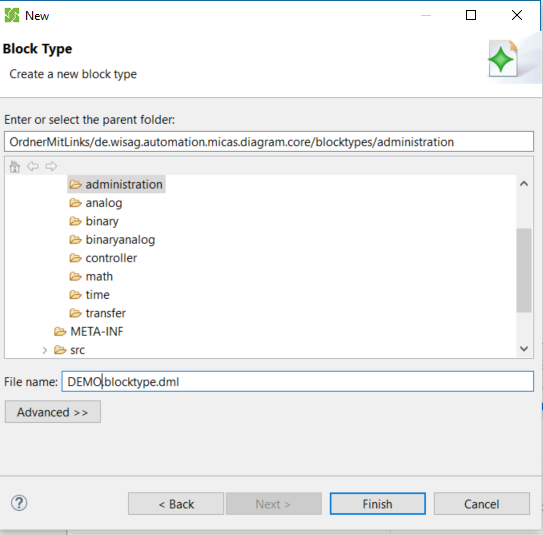
## Erstellen eines DML Blocks für Micas

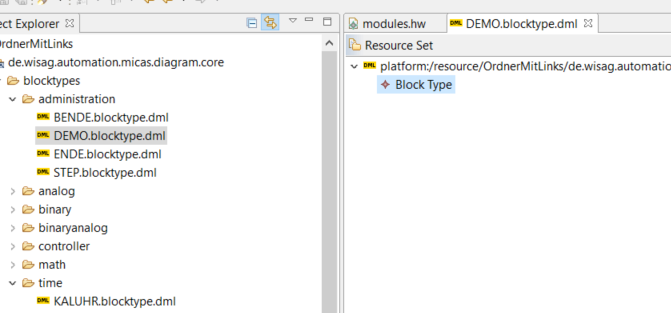
Man wählt in der Micas Workbench das diagram.core Projekt aus und dort den Ordner in dem der Block erscheinen soll.

BSP Block DEMO Category administration.

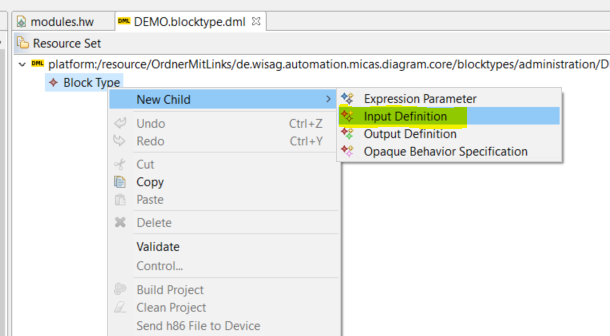
Ordner administration markieren, Rechte Maustaste, New->Other





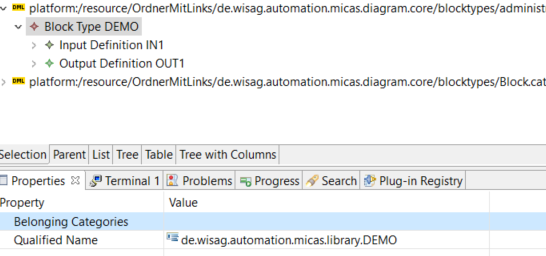


Im DML Editor können nun grob input und outputs angelegt werden. Das ist zuempfehlen, dann ist das Nachbearbeiten des XMI einfacher.



Zu jedem Port legt man Eine leere “Opaque Data Type Specification” an.

Im Block selbst trägt man im Qualified Name Den Namen ein allerdings mit einem Namespace :

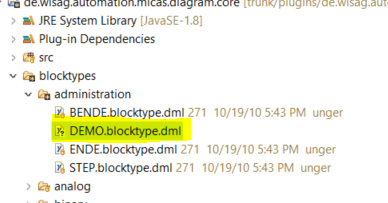


Belonging Categories bleibt erst mal leer.

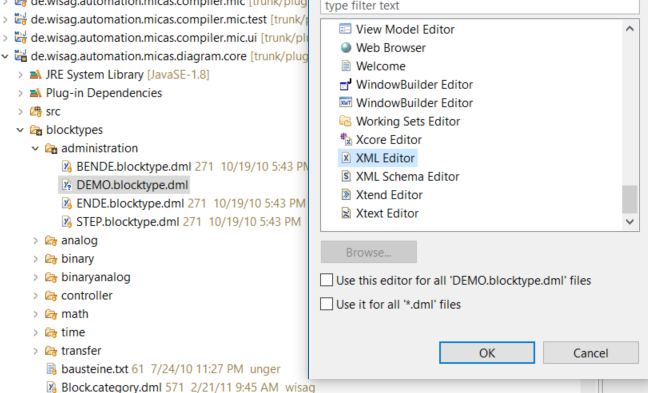
Jetzt müssen im einem XML Editor einige Anpassungen gemacht werden. DML Editor schliessen,

In Entwicklungs Eclipse wechseln und das diagram.core Projekt mit F5 refreshen.

Die neue DML Datei sollte nun sichtbar sein :

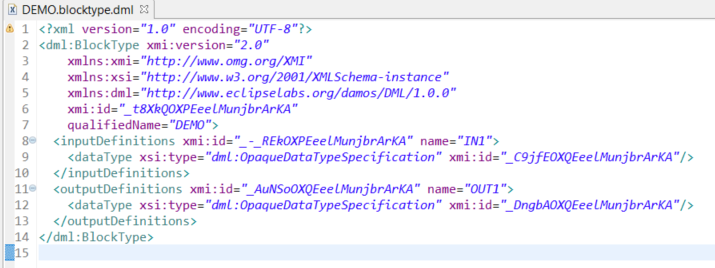


Den neuen Block nun mittels Open With

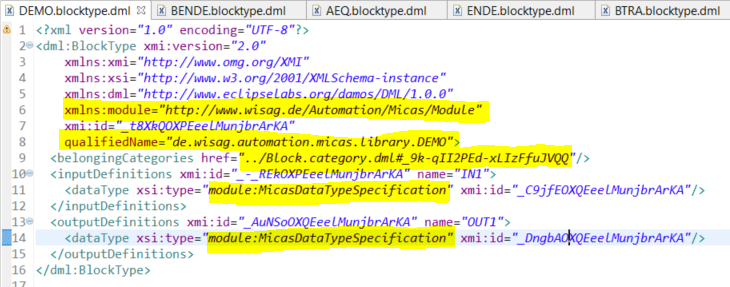


Öffnen.

Das Sieht dann etwa so aus :

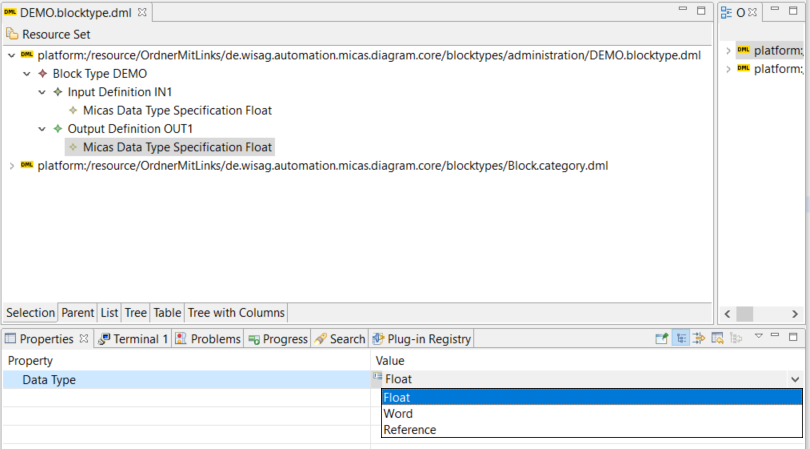


* Namespace hinzufügen xmlns:module=<http://www.wisag.de/Automation/Micas/Module>
* dataType von dml:OpaqueDataTypeSpecification auf module:MicasDataTypeSpecification ändern
* belonging Categories einfügen



Nach dem Speichern in der Entwicklungsumgebung wechselt man zurück in die MICAS Workbench und macht auf das Projekt de.wisag.automation.micas.diagram.core ein Refresh (F5).

Jetzt passen die Datentypen und es kann auch ein MICAS Datentyp zugewiesen werden.

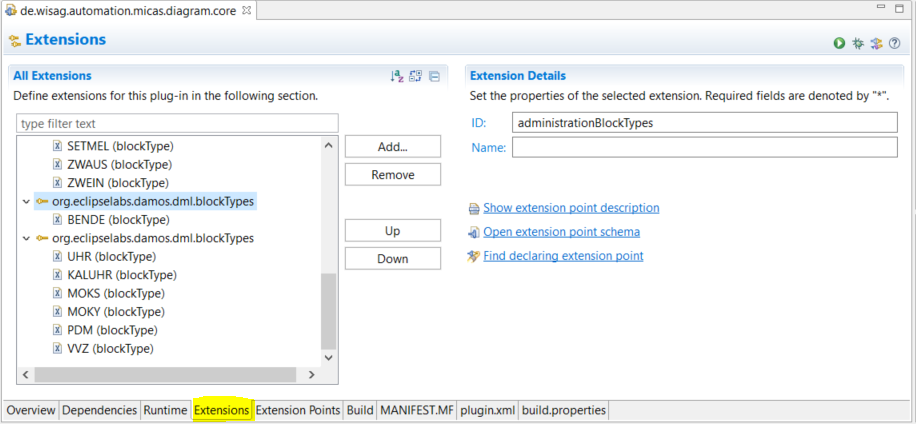


## Registrieren des neuen Blocks in der Entwicklungs eclipse

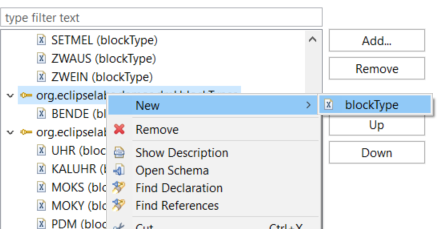
Jetzt muss der neue Block Registriert werden dazu öffnet man in der Entwicklungs eclipse

de.wisag.automation.micas.diagram.core/META-INF/MANIFEST.MF

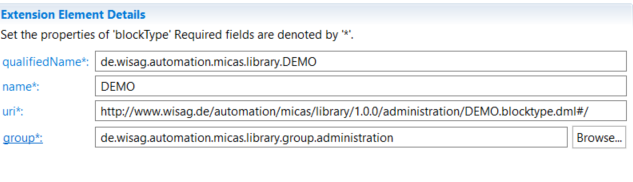
Und wählt im Manifesteditor die Seite „Extensions“ aus:



Dort erzeugt man sich einen neuen blockType



Die Felder müssen mit Daten aus der DML Datei befüllt werden.



**ACHTUNG:** in der uri stehen Teile aus dem projektpfad

In der Entwicklungs Eclipse ist der Pfad

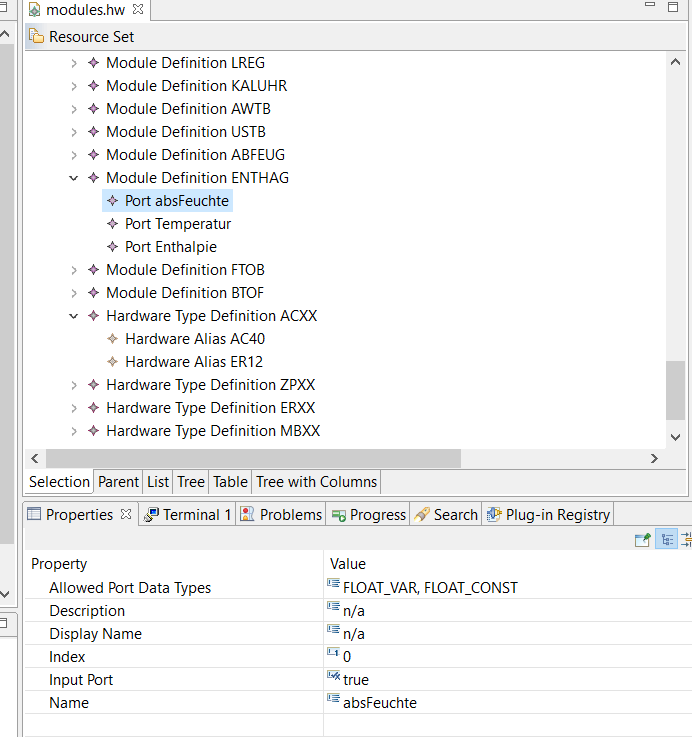
de.wisag.automation.micas.diagram.core/blocktypes/**administration/DEMO.blocktype.dml**

Damos findet so die Blöcke.

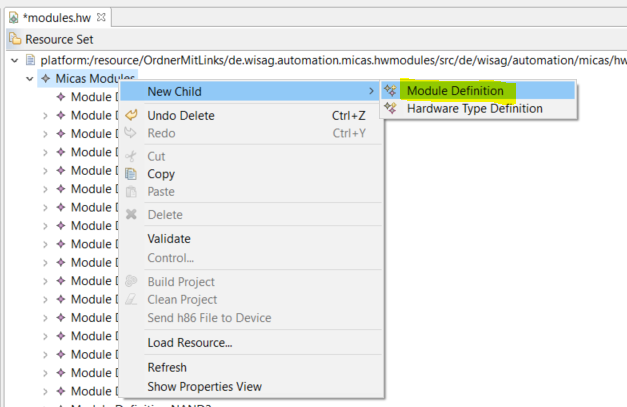
Jetzt steht der neue Block im Diagrammeditor zur Verfügung. Der Compiler weiss noch nichts davon.

## Hinzufügen der Hardwaredefinition

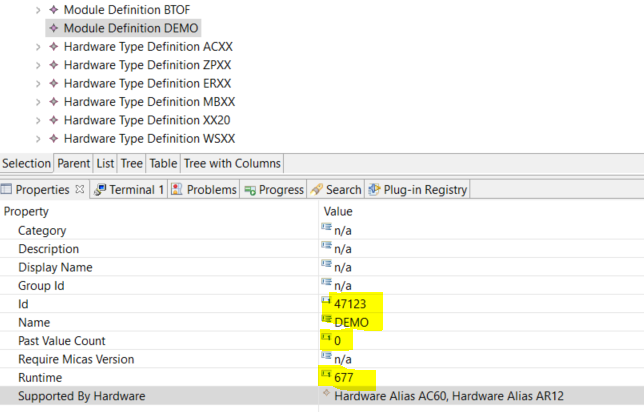
In der MICAS Workbench im verlinkten Projekt de.wisag.automation.micas.hwmodules wird nun die Datei modules.hw bearbeitet.



Wobei es zu jedem Modul eine Module Definition gibt



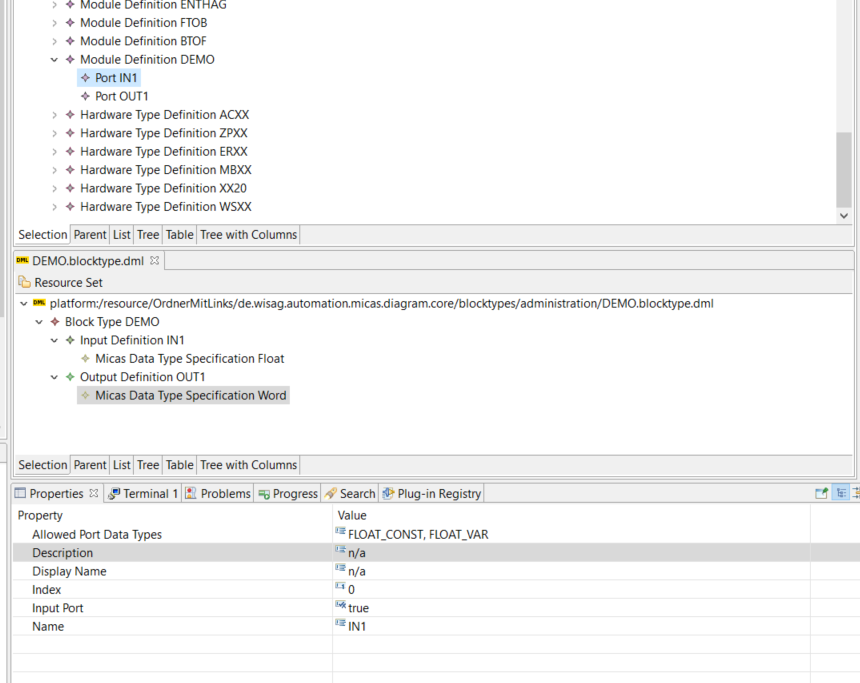
Folgende Felder müssen ausgefüllt werden:

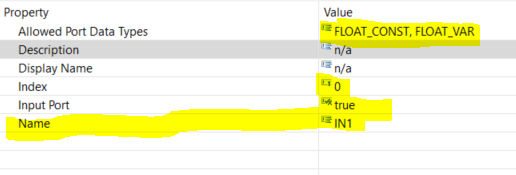


Id ist dezimal, Supported By Hardware ist zur Prüfung.

Anschliessend definiert man die Ports. Diese sind von 0..N durchnummeriert.

Praktischerweise kann man sich den DML Editor danebenlegen:



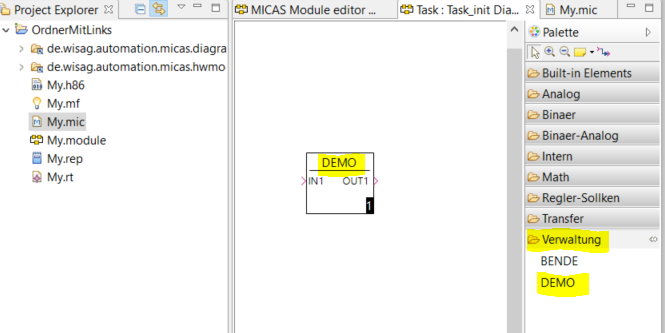


Nachdem man alle Ports angelegt hat, speichert man die Datei modules.hw, wechselt in die Entwicklungs Eclipse zurück und macht auf das hwmodules Projekt ein Refresh (F5).

## Testen eines neuen Blocks

Micas Workbench aus der Entwicklungsumgebung heraus neu Starten.

* In der Palette ist der neue Baustein in der entsprechenden Kategorie eingefügt
* Er lässt sich im Diagrammeditor verbauen.



Das generierte MIC File enthält auch den Aufruf.

