EyeQ CDD Integration

EyeQ CDD Integration

1. Scope

The scope of this document is to learn procedure and steps to be followed during integration of the EyeQ CDD with Application software. This doc will also give an InSite of regular issues faced and care to be taken during integration along with some debug steps.

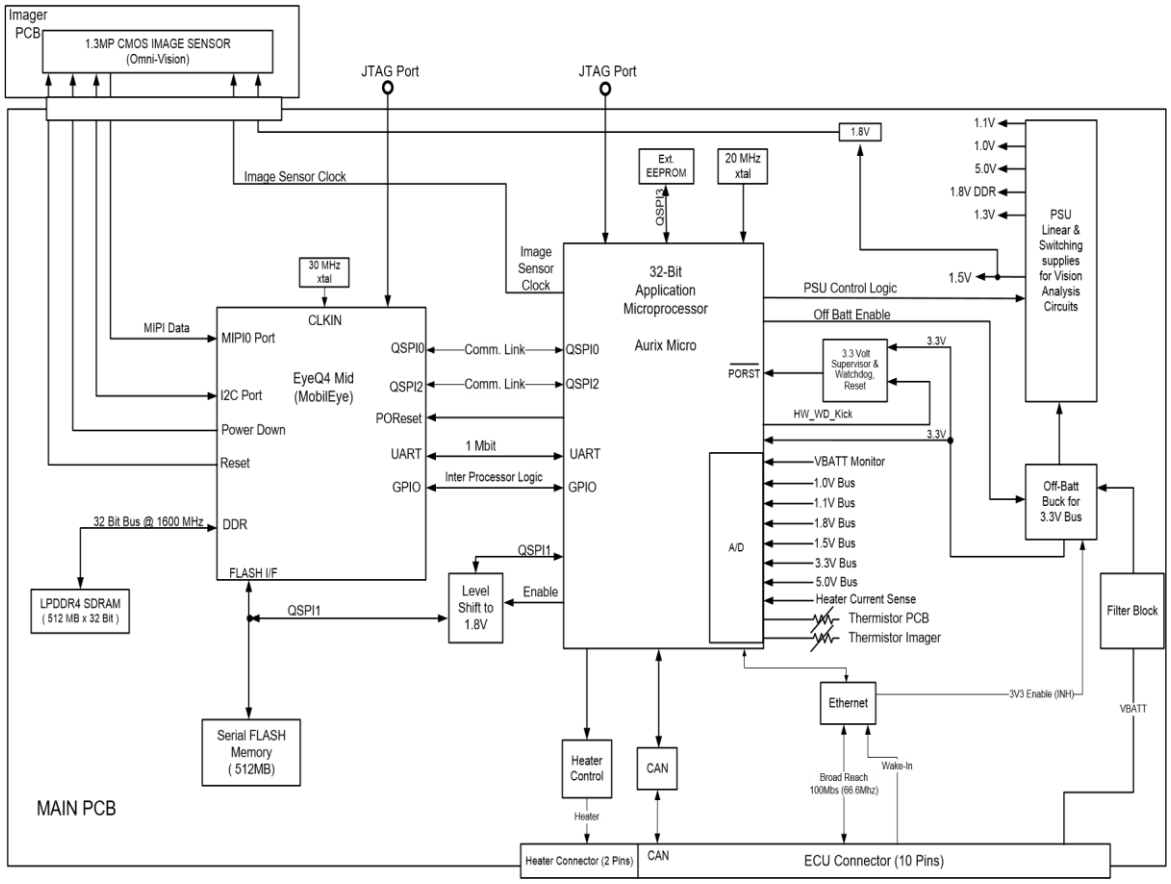
NOTE: This document gives all steps in integration process, some of steps can be ignored if there are no changes in particular step in core release.

2. Introduction to EYEQ, EYEQ CDD and Application Interaction

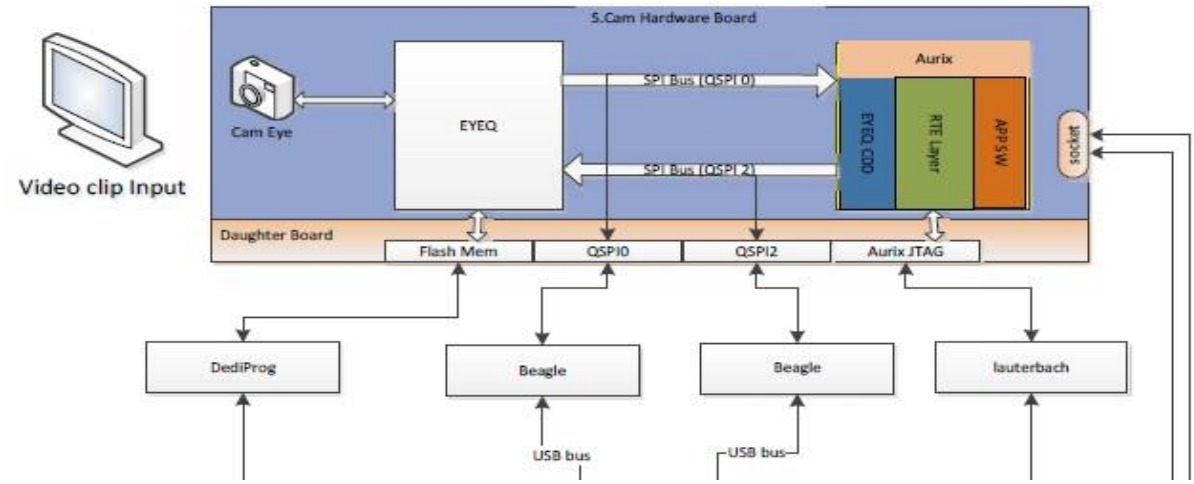
Front Camera Module consist of EYEQ on EyeQ chip (external processor or ASIC),with EYEQCDD application and application algo components on AURIX .EYEQ is generally delivered by MobilEye(ME) hence EyeQ binaries are called ME files.

ME consist of core image acquisition along with processing according to features supported by particular ME. ME interacts with EyeQ CDD using QSPI half-duplex communication on two different SPI channels usually (QSPIO and QSPI2) as shown in below diagrams.

EyeQ CDD Integration



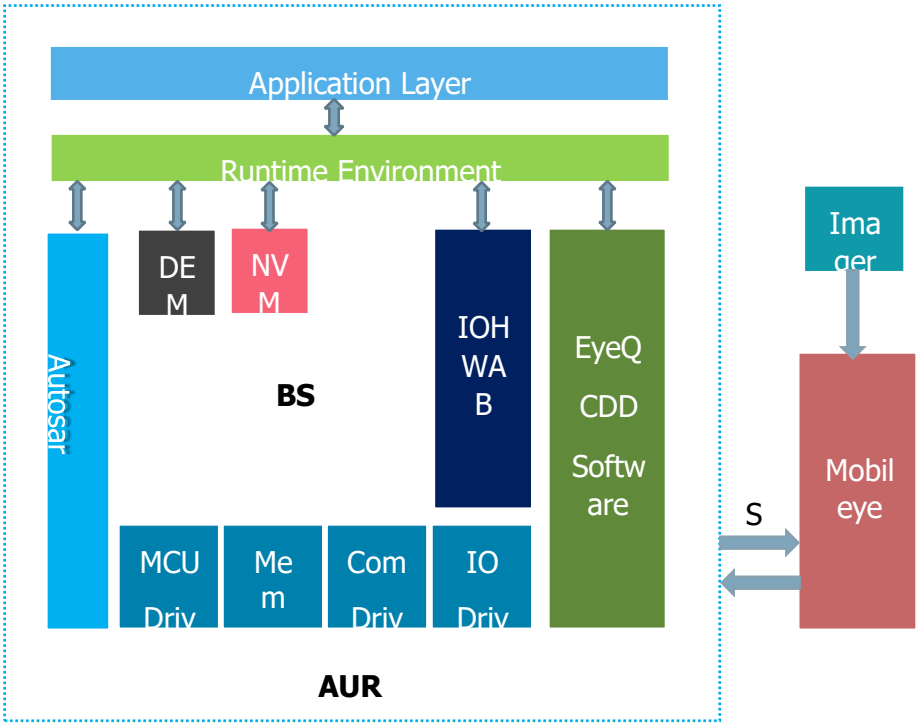
Detail Front Camera module



Front camera module block diagram with various connection

EyeQ CDD Integration

The below diagram shows the EyeQ CDD interactions with Application,BSW and the interaction between ME and Aurix. ME and Aurix are connected through SPI communication as discussed earlier to exchange the information. EyeQ CDD software will run on Aurix and handles the interactions between Aurix and ME ,it controls the power and state management of ME according to application needs. EyeQ CDD software architecture is designed based on OSI layered architecture. The power management of ME is handled through IOHWAB. DEM and NVM are configured for EyeQ CDD specific faults and blocks. EyeQ CDD integration is done with the interface connections between Application software and EyeQ CDD, NvM and Dem interface connections and mapping the schedulable entities in EyeQ CDD component to Autosar OS. The EyeQ CDD Integration process is explained in detail in below sections in this document.



EyeQ CDD Integration

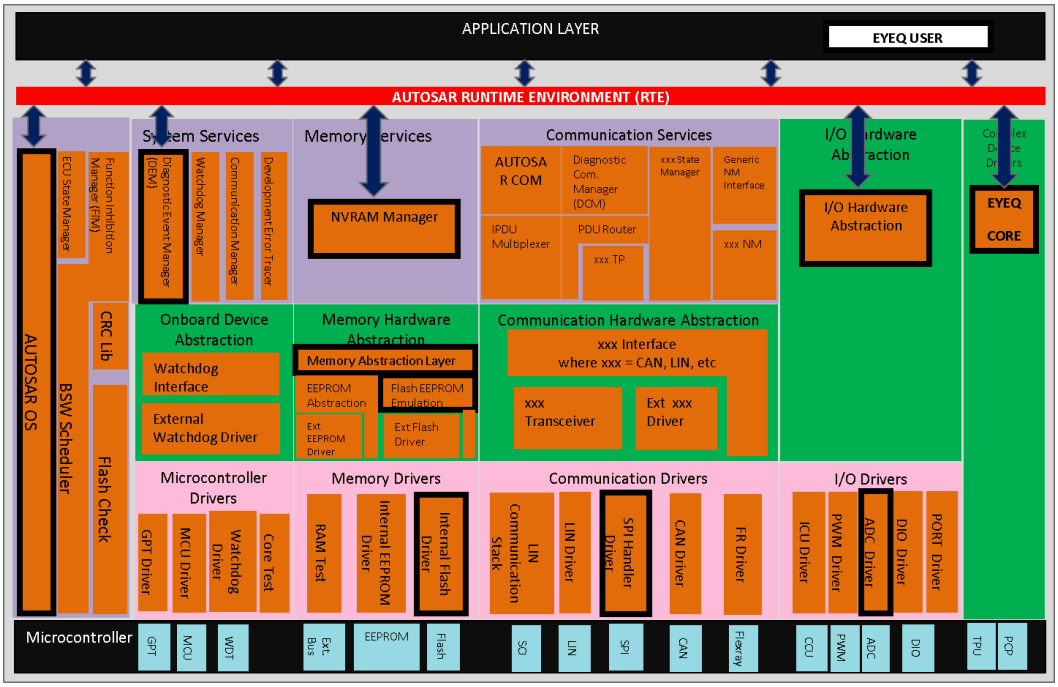


Figure 1 DAS_SCAM_4.8 Platform Architecture

The above figure shows the DAS_SCAM_4.8 platform architecture. It follows AUTOSAR_4.0.3 classic platform. EyeQ CDD component interacts with application software, IoHwAb, NvM and DEM modules. It uses the services provided by NVM, DEM and IOHWAB. It uses MCAL services to interact with ME chip. Os will trigger the periodic runnables in EyeQ CDD component and manages the execution of ISR's configured for EyeQ CDD component.

The EyeQ CDD core software contains the configuration of NVM, DEM and IOHWAB specific to EyeQ CDD module. NVM configuration consists of NVM Blocks configured for EyeQ CDD. DEM configuration contains DEM events configured for EyeQ CDD faults.

The EyeQ CDD core software integration with application software carries integration of EyeQ CDD source code, EyeQ CDD Configuration, NVM configuration specific to EyeQ CDD, DEM configuration specific to EyeQ CDD, IOHWAB, MCAL, AOUSAFE ,OS changes from core ,linker and Startup code.

EyeQ CDD Integration

3. Understanding ME Software Release

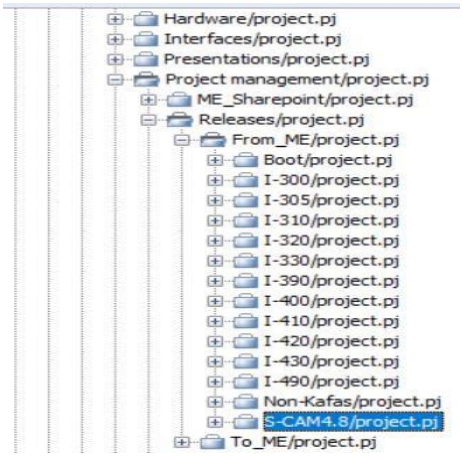
ME Software release contains binary file which will be programmed into EyeQ chip and a Release notes along with SPI protocol sheet(excel) for the current release.

Usually ME software release will be done by MobilEye when there is an update in ME software or changes suggested by the user or if there is a bug reported by the user in ME software.

ME releases can be find in following integrity path:

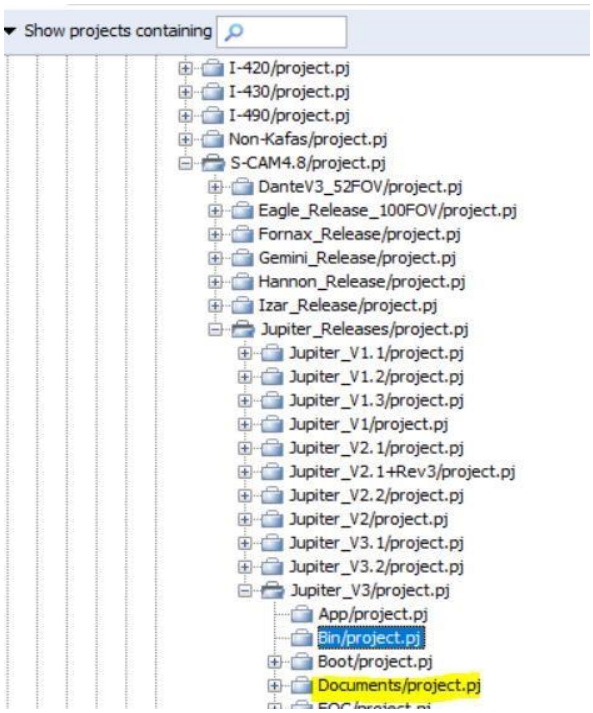
url:integrity://skobde-mks.kobde.trw.com:7001/si/viewproject?project=/DAS%2dME/020%5fCore/EyeQ4/Project%20management/Releases/From%5fME/project.pj

By clicking the above integrity link the integrity view will be as shown in below figure:



Select the variant according to project and project specific for example norma xx.

EyeQ CDD Integration



Folder structure of ME release as below

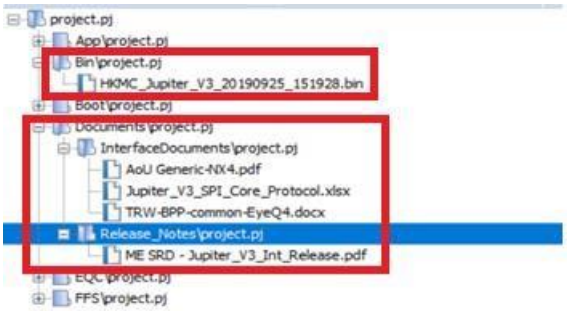
Name	Date modified	Type
App	10-03-2021 10:54	File folder
Bin	10-03-2021 11:05	File folder
Boot	10-03-2021 11:05	File folder
dbc	10-03-2021 11:06	File folder
Documents	10-03-2021 11:16	File folder
EQC	10-03-2021 11:37	File folder
FFS	10-03-2021 11:37	File folder

Inside variant folder ,bin folder contains the binary file in .bin format to be flashed to

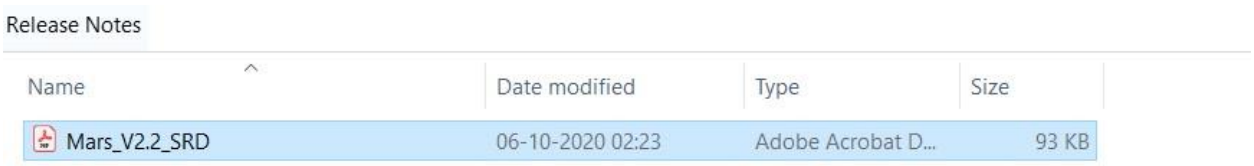
EyeQ chip. Generally .bin files are flashed using Dediprog tool if port is available. In GWM and G35 project variants these .bin files are converted into .s37 which will be explained in document and flashed using MFT tool .

Documents folder contains TRW BPP document, SPI protocol sheet (Excel) and a release notes for the current release. Sandbox view shown below.

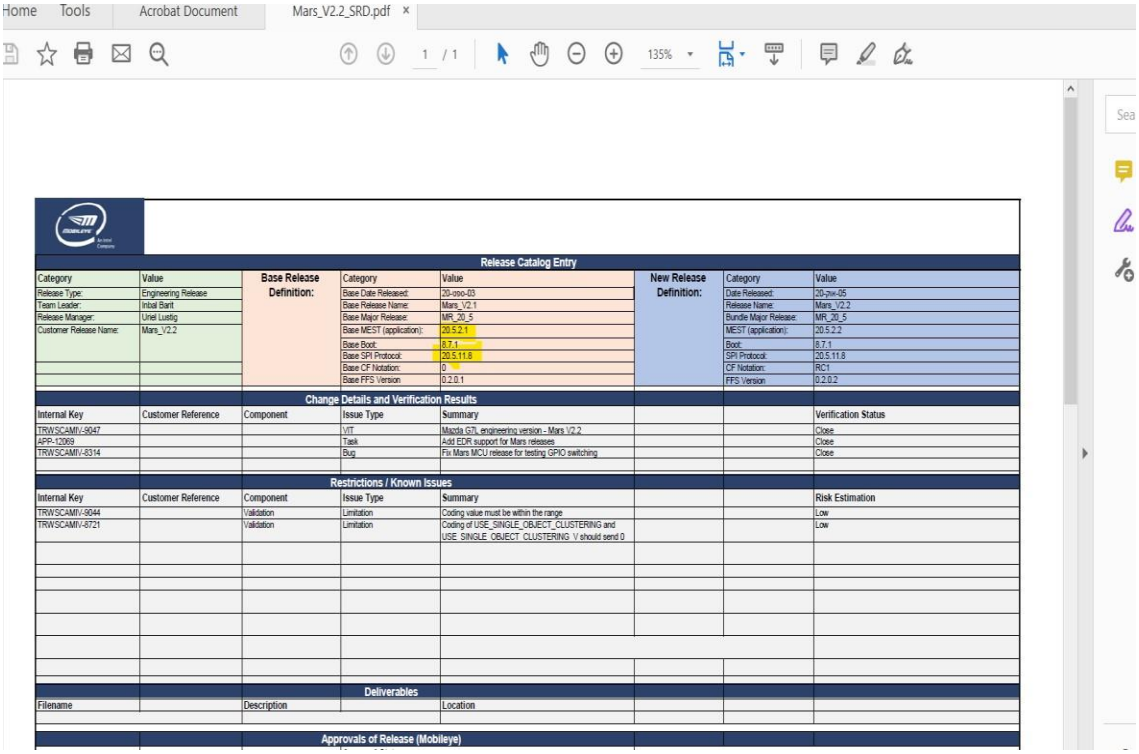
EyeQ CDD Integration



Release information is present in \Documents\Release Notes



This document gives version of Boot, MEST and SPI protocol version as below



EyeQ CDD Integration

Release Catalog Entry	
Category	Value
Base Date Released:	20-090-03
Base Release Name:	Mars_V2.1
Base Major Release:	MR_20_5
Base MEST (application):	20.5.2.1
Base Boot:	8.7.1
Base SPI Protocol:	20.5.11.8
Base CF Notation:	0
Base FFS Version	0.2.0.1

Other important file in the ME release software is meio which is present in \FFS\etc folder .This file contains important information about features supported by particular ME and other details as below .Protocol list gives all the feature supported and released in particular ME .

meio.cfg

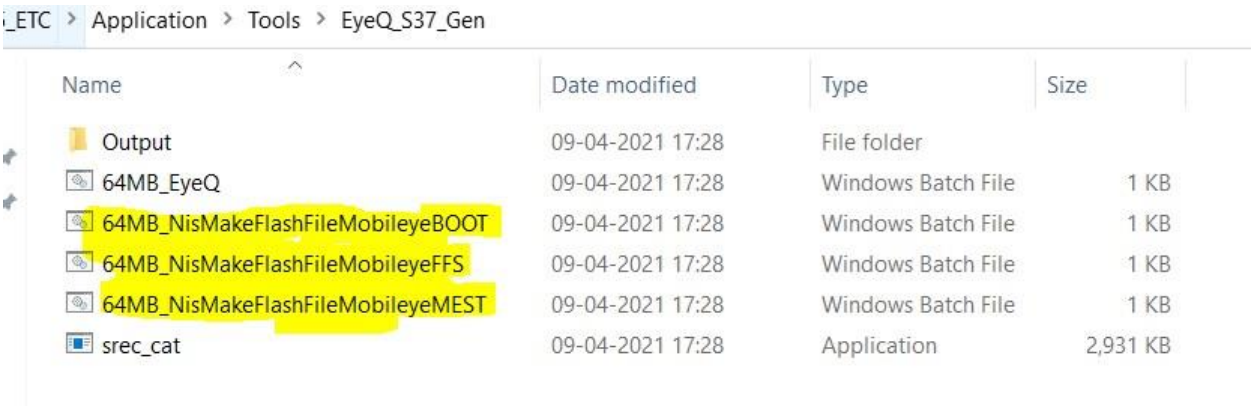
```
0
1  ##################################### SPI ########################################
2
3  [SPI]
4  enabled = true
5  sensors = { "Core" }
6  protocols = { "Core" }
7
8
9  [SPIWrite]
0  isSpiWordBigEndian = false
1
2  [SPIRead]
3  isSpiWordBigEndian = false
4
5  [Core]
6  protocolList = { "calibrationDynamic", "common", "dfa", "failSafes", "fcf_cv", "fcf_vd", "fcf_vru",
7
8
9  ################################### Other Comm. Protocols ####################
0
1  [CAN1]
2  enabled = false
3
4  [CAN2]
```

EyeQ CDD Integration

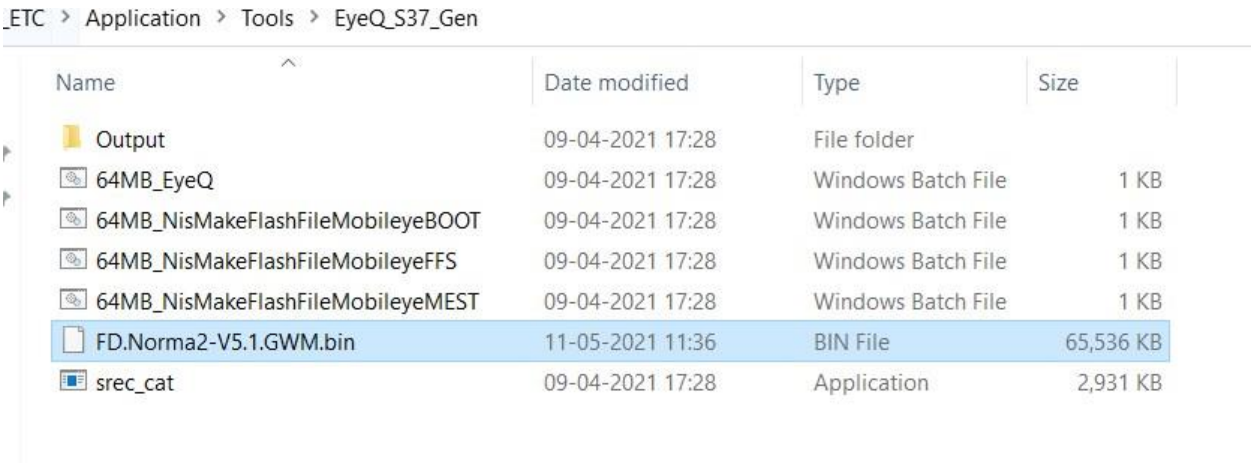
4. Converting ME binary file .bin to .S37 files .

In GWM and other variants of project .S37 of ME are flashed using MFT tool. Single binary file of ME is converted into three .s37 files BOOT,FFS and MEST files .

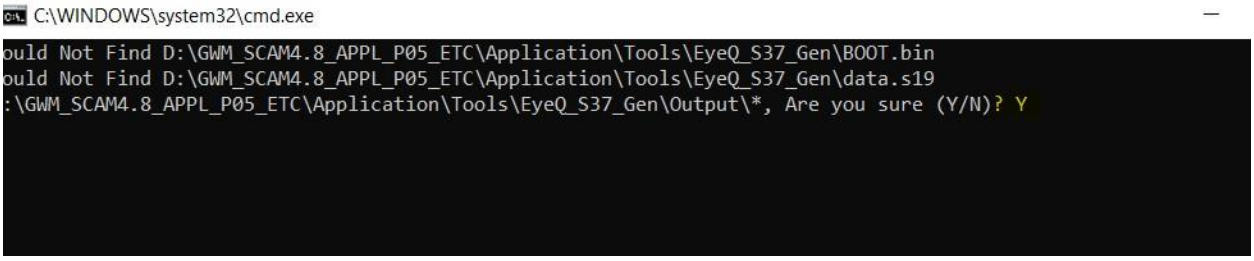
There are tools (batch files)files present in tools folder of GWM application project in path Application\Tools\EyeQ_S37_Gen as below highlighted.



Place the binary file of ME which need to be converted into S37 files in below folder as shown

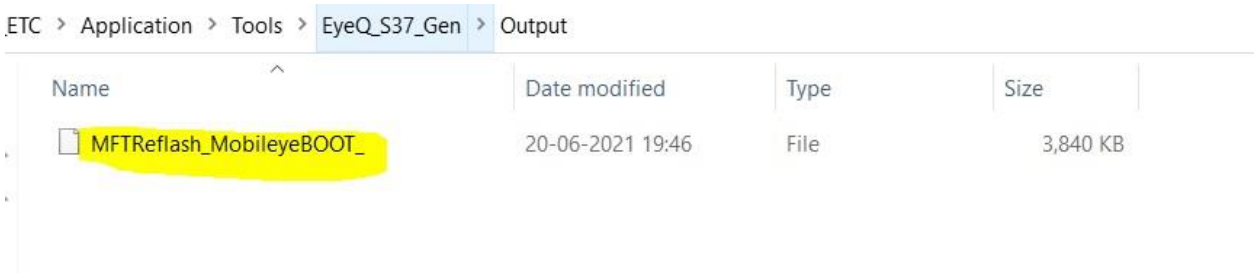


Drag and Drop the .bin on Batch files for example on 64MB_NisMakeFlashFileMobileyeBOOT.A command window will pop out as below enter Y to generate Boot s37 file

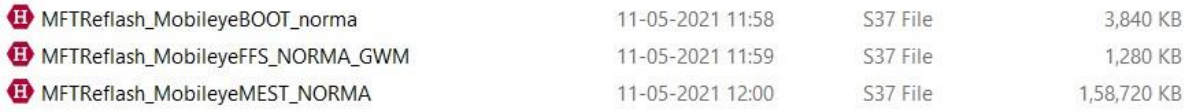


Boot files is generated in Output folder as below

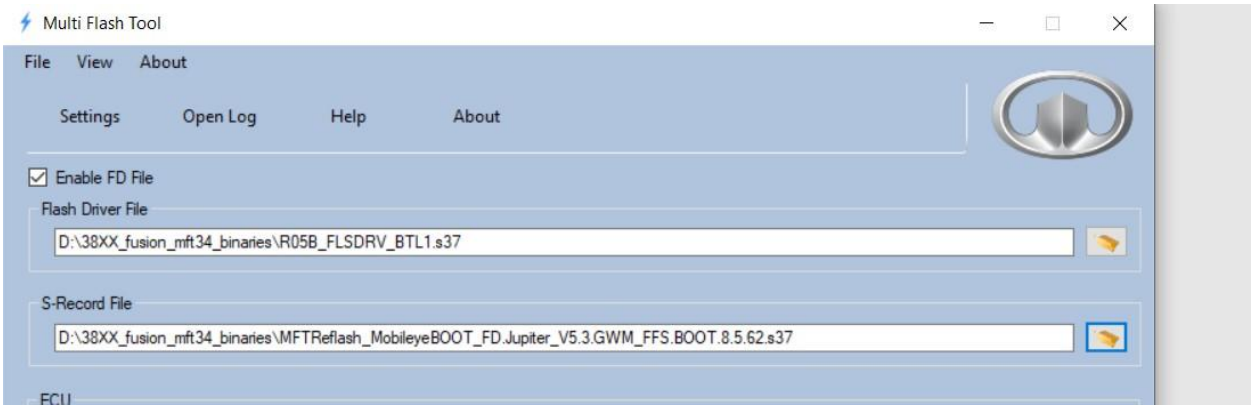
EyeQ CDD Integration



Rename file with .S37 extension and move to some other location . Similarly drag and drop FFS and MEST files on respective batch files , rename with S37 extension and move to separated folder as below.



We can flash s37 files one after other using MFT tool by selecting each at once as below one after other.



EyeQ CDD Integration

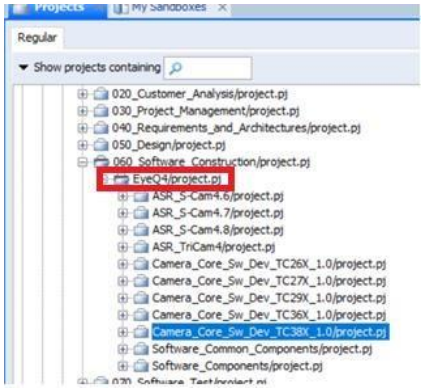
5. Understanding the Core Software(EyeQ CDD) Release

Core Software will be released by Core team. The Core Software release contains configurations and source code of EyeQ CDD, MCAL, IOHWAB and contains integration code that is useful for the platform integration.

The below link will navigate to core software releases in integrity:

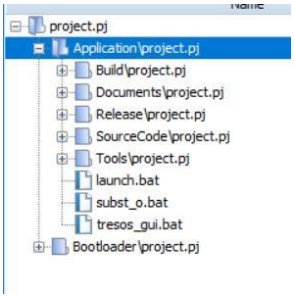
url:integrity://skobde-mks.kobde.trw.com:7001/si/viewproject?project=/DAS/040%5fTest%5fProjects/%3cCore%5fFC%3e%20Core%20Forward%20Camera%20Development/060%5fSoftware%5fConstruction/EyeQ4/projec t.pj

By clicking the above link the integrity view is shown in below picture:



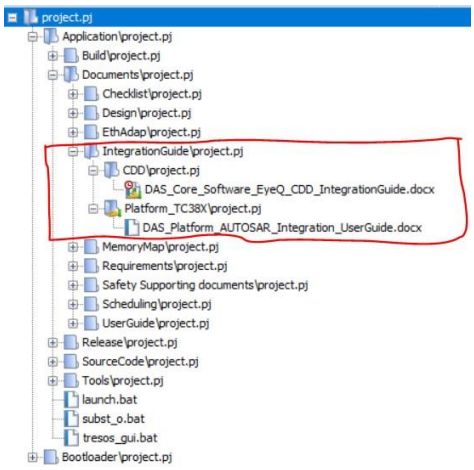
6. Understanding the Core Software(EyeQ CDD) Folder Structure

Sandbox view of Core software project is shown in below figure..

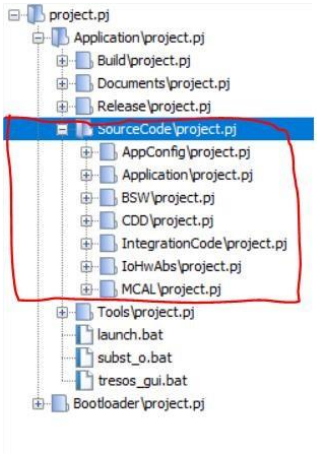


In **Documents** folder **Integration Guide** folder contains CDD integration guide and Platform integration guide.

EyeQ CDD Integration



The SourceCode folder view shown in below figure:



- AppConfig --> This folder contains EyeQ CDD, IOHWAB and PTM component configuration files(.arxml and cfg.h) which can be modified by application team.
- CDD --> This folder contains source code for EyeQ CDD Component, Production mode services, MCU_JTAG_Lock, PTM and code generators for DEM, NVM Proxy modules.
- IntegrationCode --> This folder contains source code for SafetyLib(AoUSafe), Startup code(AURIX).
- IoHwAbs --> This folder contains the IoHwAbs source code specific to EyeQ CDD component.
- MCAL --> This folder contains third party MCAL driver source code, generated configuration files(.arxml, cfg.c, cfg.h).

EyeQ CDD Integration

7. Core Software Integration Steps and Procedure

Below are steps involved in EyeQ CDD Core software integration. Details of each steps will be explained where ever it is required in documents in upcoming chapters.

Step1:

- Integrate the EyeQ_CDD ,IoHwAb, AOUSAFE files and other file from core release.
- Integrate the MCAL Changes.
- Identify & Configure the NVM blocks if any changes.
- Identify & Configure the Dem Events if any changes.
- Identify & Configure the OS if any changes.
- Identify linker changes, make files changes if any and integrated to application linker and Sconstruct file.

Step2:

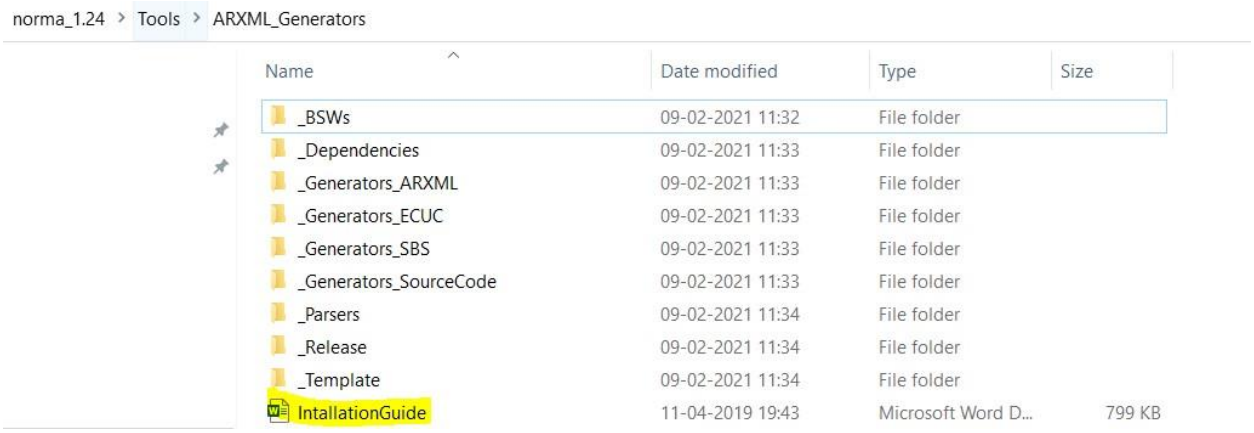
- Convert ARXMLS according to project architecute and Import EyeQ_CDD ,IoHwAb Proxy(wrapper) ,PTM related Arxml in to Developer.
- Map the EyeQ Interface related changes in Developer SWC's.
- With above changes build the SW.
- If Build is successful Test and verify the Basic Communication & EyeQ Vision Mode.

EyeQ CDD Integration

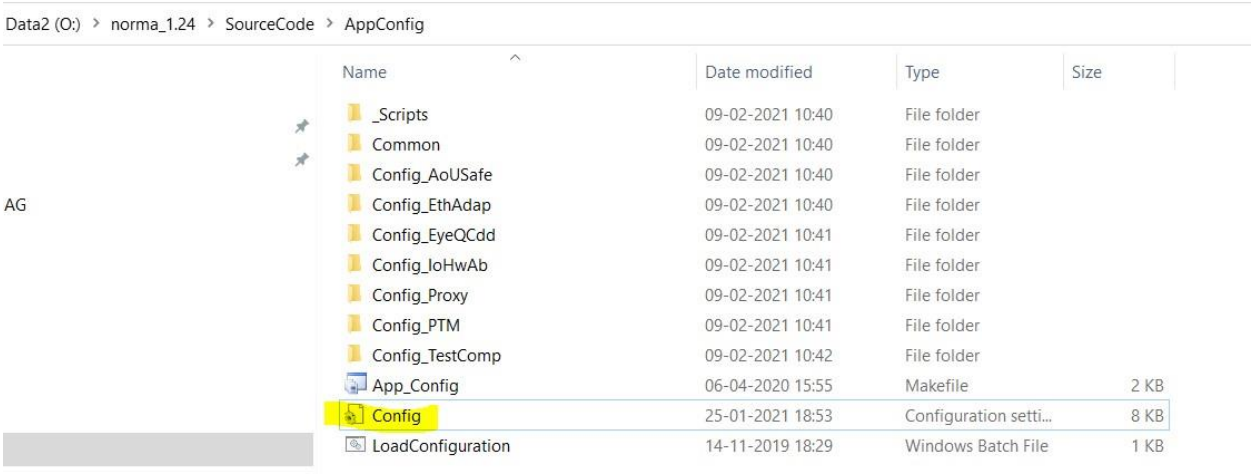
8. ARXML Generation according to Project Architecture.

ARXML delivered by core cannot be directly imported into developer as the core architecture and Application architecture may not be same. Components like NVM, DEM, EYEQCDD, IoHwAb and PTM may not be in same core in core and application. Integrator need to configure certain files and follow below steps to generate there architecture specific ARXMLS.

To install tools related to ARXML we need to follow installation guide in following path of core delivered software.

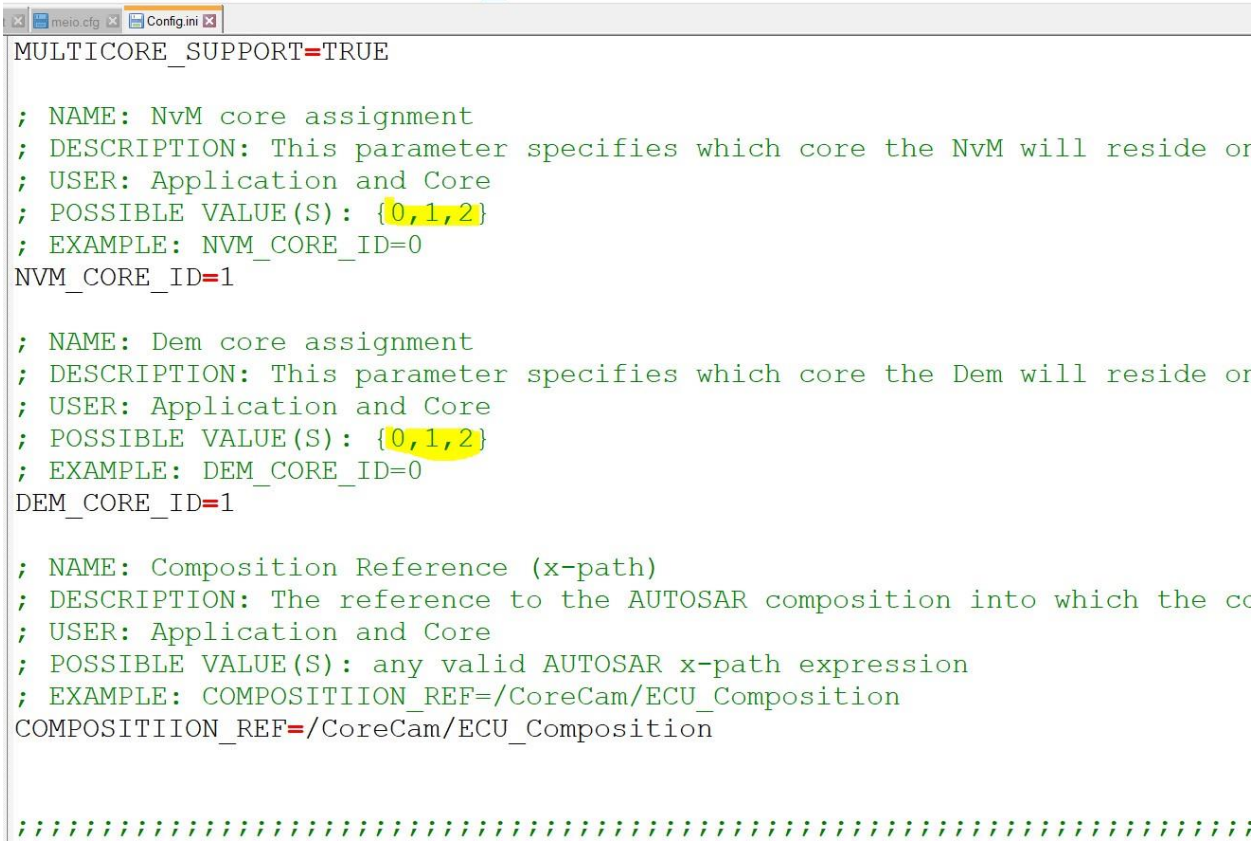


After installation of tools, generate o drive for core project by clicking on subst_o. Configuration
of a component in particular core can be done in Config.ini file at location O:\norma_1.24\SourceCode\AppConfig as shown below



Example of Config.ini is as shown below, we can see options to configure according to project architecture in core 1.2 or 3 for NVM,DEM,CDD,IOHW and PTM components.

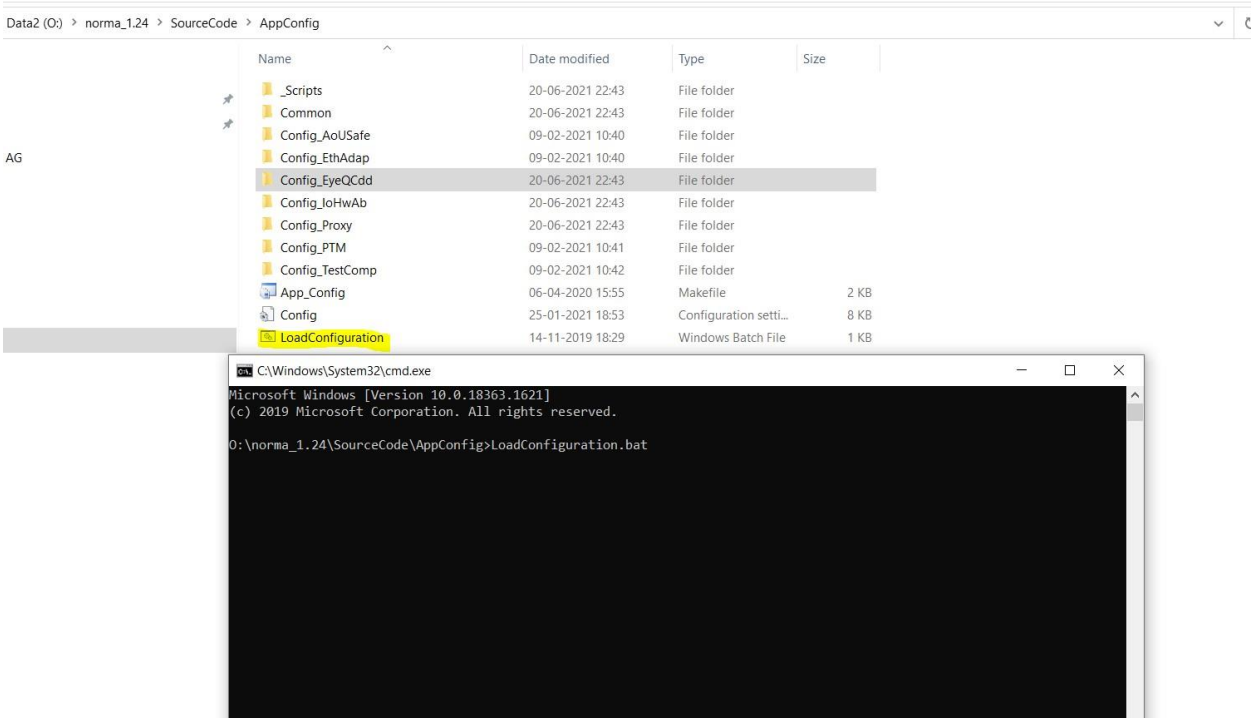
EyeQ CDD Integration



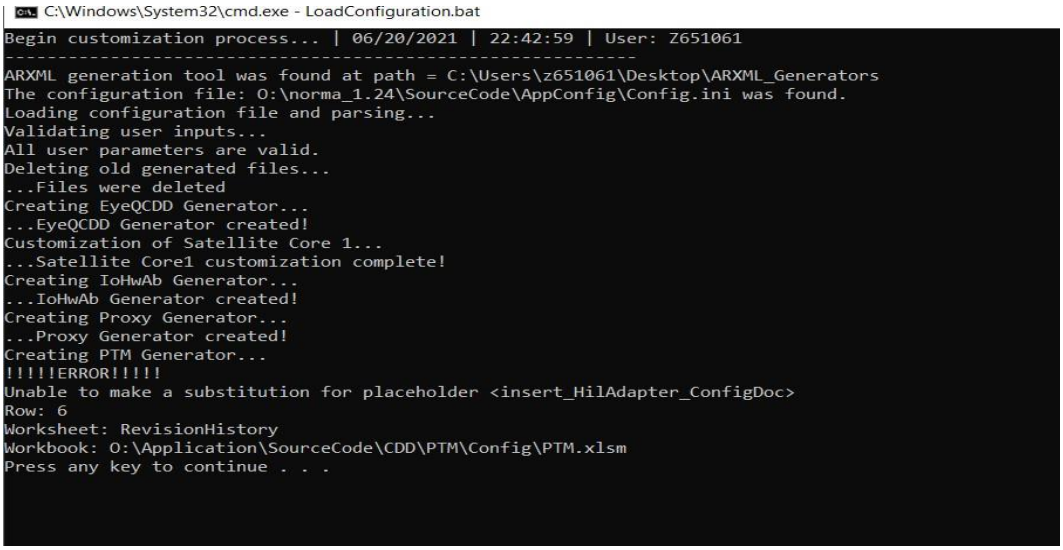
Save the file after configuring according to project architecture. We need to generate now excels sheets for CDD, PROXY, IOHW and PTM. Proxy component has NVM and DEM related ports and interfaces.

To generate excel sheets we need to run LoadConfiguration batch file in command prompt as below

EyeQ CDD Integration

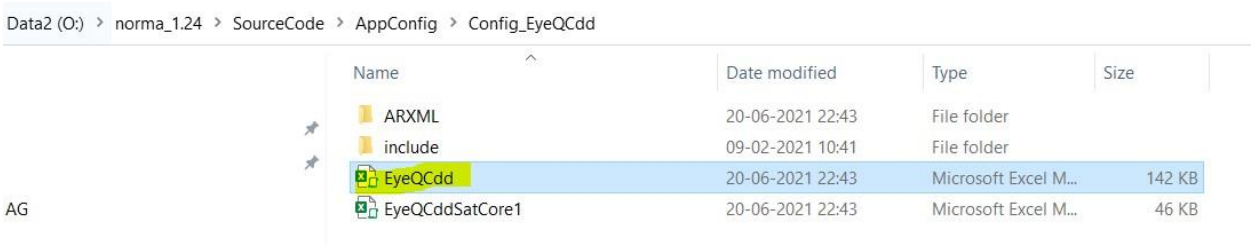


Excels for EYEQCDD,IOHWAB etc are generated and can be as seen below

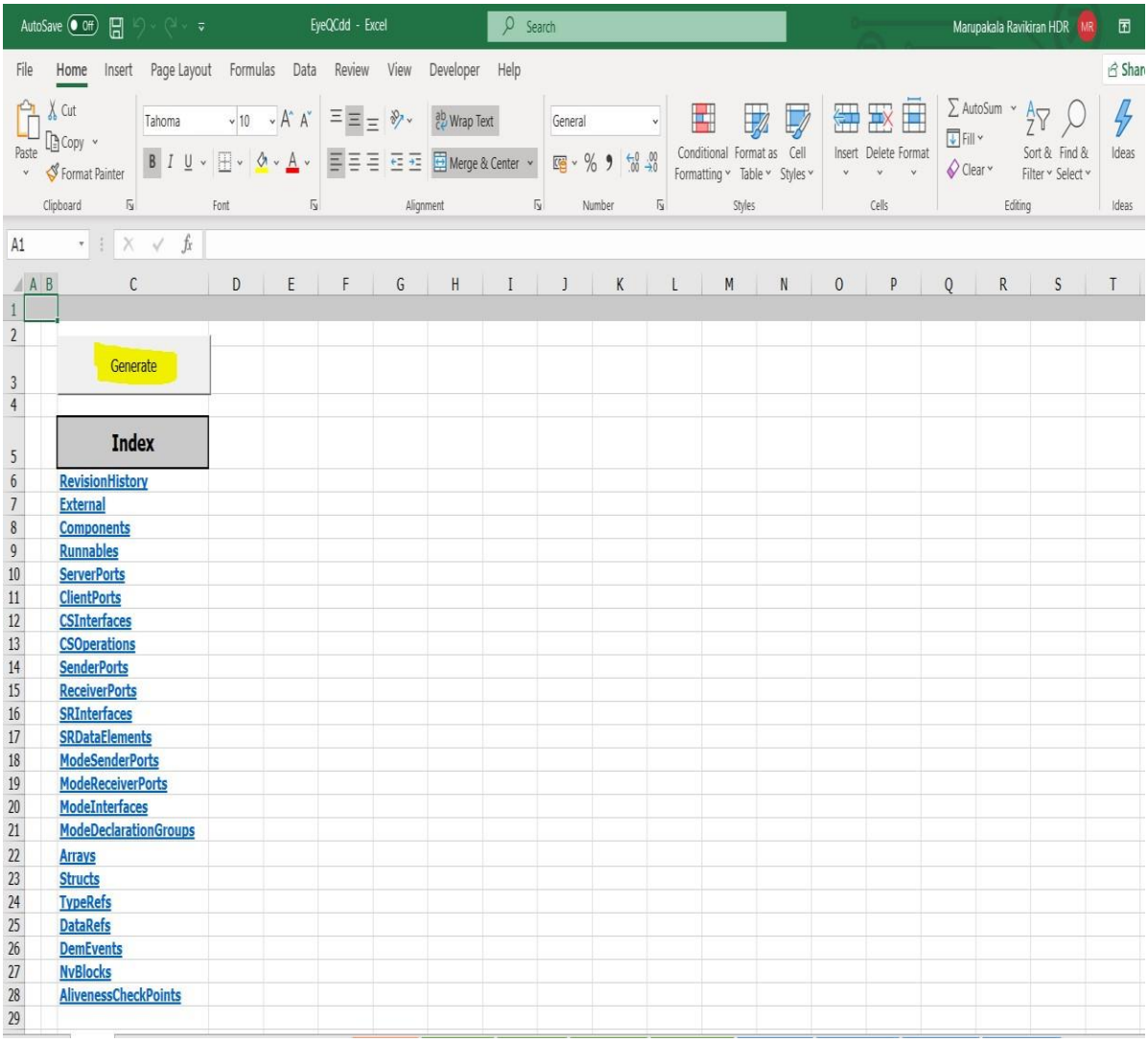


Paths of excel sheets are folders of configuration like for EYEQCDD at O:\norma_1.24\SourceCode\AppConfig\Config_EyeQCdd as below

EyeQ CDD Integration

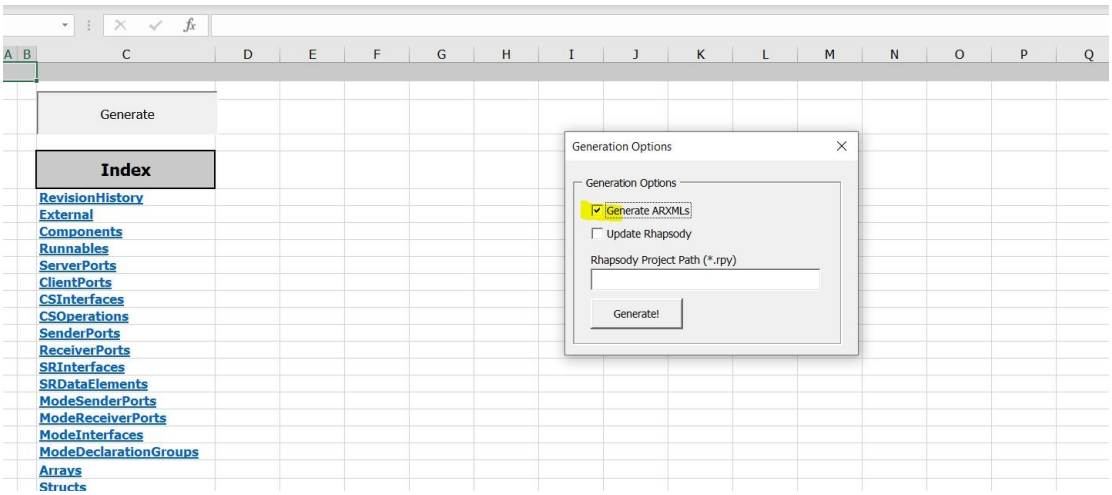


Now we need to generate arxmls by opening EyeQCDD Excel and clicking on generate option as below .



Now window will pop up select , Generate ARXML and click generate

EyeQ CDD Integration



A window will now open with generation completed.We can now see ARXML at path O:\norma_1.24\SourceCode\AppConfig\Config_EyeQCdd\ARXML forEyeQCDD .

Similarly we can generate ARXML for IOHWAB,PROXY,PTM etc components .Now ARXMLS are ready to be integrated .

EyeQ CDD Integration

9. Core Software Integration Steps in detail Static files:

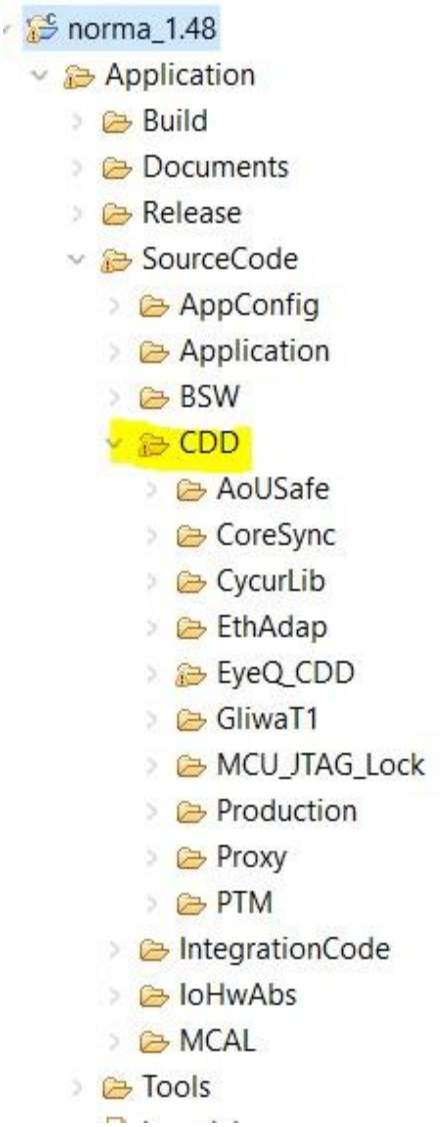
9.1 CDD Component Integration (Source code)

In core sandbox we can find CDD in below path as shown below

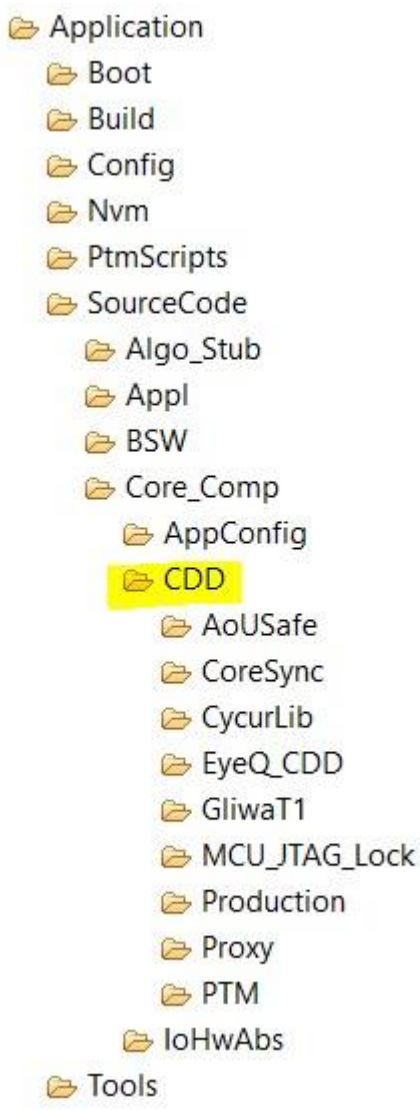
Application\SourceCode\CDD -- Core CDD path

Application sandbox of GWM camera we have CDD folder in below path as shown

Application\SourceCode\Core_Comp\CDD --GWM CDD path



Core CDD Path



Application CDD path

EyeQ CDD Integration

This folder contains the source code for EyeQ CDD, Production,CoreSync ,Cycurlib,GliwaT1,PTM,MCU_JTAG_LOCK and AOUSAFE. We need to integrate these static files from core to application.

Below are points need to be considered while integration.

1)Before merging changes from core please compare CDD changes from earlier checkpoint to identify changes.

2)We need to merge the changes which are done in current check point. Please do not overwrite any changes done in application from core .If we not sure about any point to merge ,we need to drop mail and get confirmation from application owner who has done changes(Ideally we need to take from core but in GWM this is how it is followed.)

3) PTM source code can be adapted from Application software. The destination folder in Application software may vary across the projects. PTM,GliwaT1 integration is taken care at STC.

4) EyeQ CDD, Production, CoreSync and Cycurlib need to be merged as it is from core by following above points. MCU_JTAG_LOCK feature is not used in GWM and variants. We will discuss about AOUSAFE in detail in upcoming chapters.

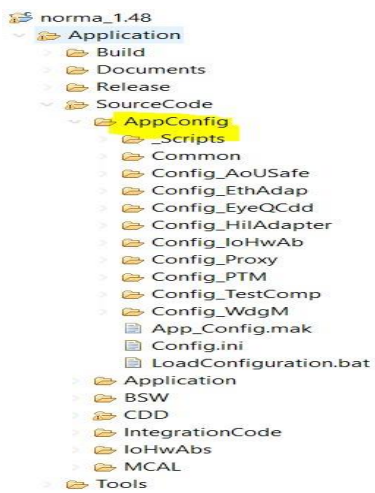
9.2 App Config Integration

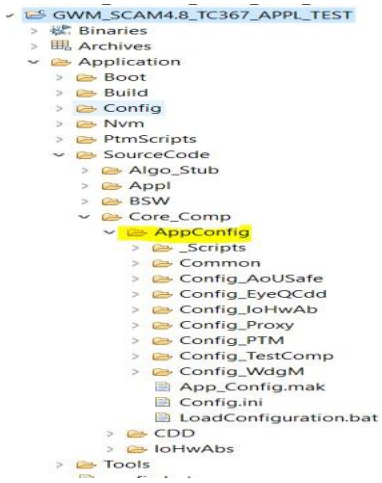
Below is path of App config folder in core

norma_1.48\Application\SourceCode\AppConfig --Core path

Below is path of App config folder in GWM application project

Application\SourceCode\Core_Comp\AppConfig --GWM application Path





AppConfig path in core AppConfig path in GWM application

This configuration can be modified according to project needs. In this folder Config_AoUSafe, Config_EyeQCDD, Config_IoHWb,Config_WdgM need to copied from core and can be modified according to project needs.

Config_Proxy contain code from ARXML generator tool which we discussed earlier chapter of ARXML generator. We need to copy generated code to application folder. Test components is not required to copy and PTM can be ignored as done at STC.

NOTE: Component configurations may need to adapt from Application software.

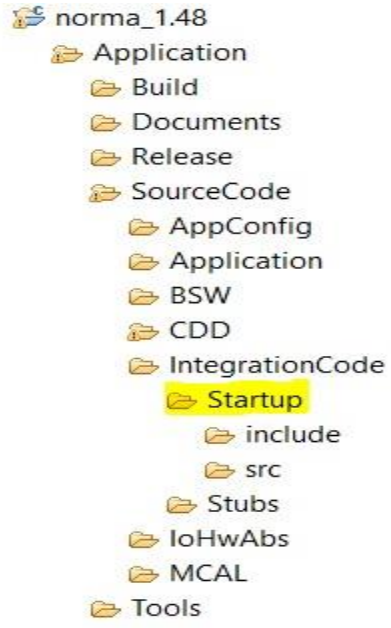
9.3 IoHwAb Integration

This folder contains the source code for IOHWAB component as explained in **Heading 5**.

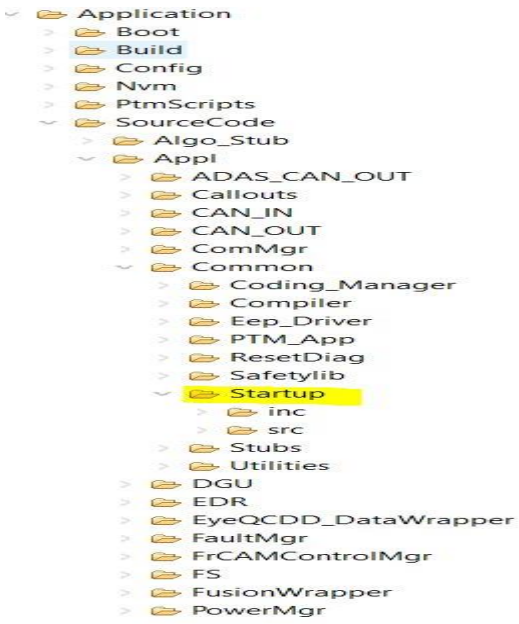
The folder **IoHwAb** should be copied from Core software(refer to **Heading 5**) to Application software. The destination folder in Application software may vary across the projects.

9.4 Startup code integration

The folder **Startup** should be copied from Core software to Application software.



Core start up path



GWM start up path

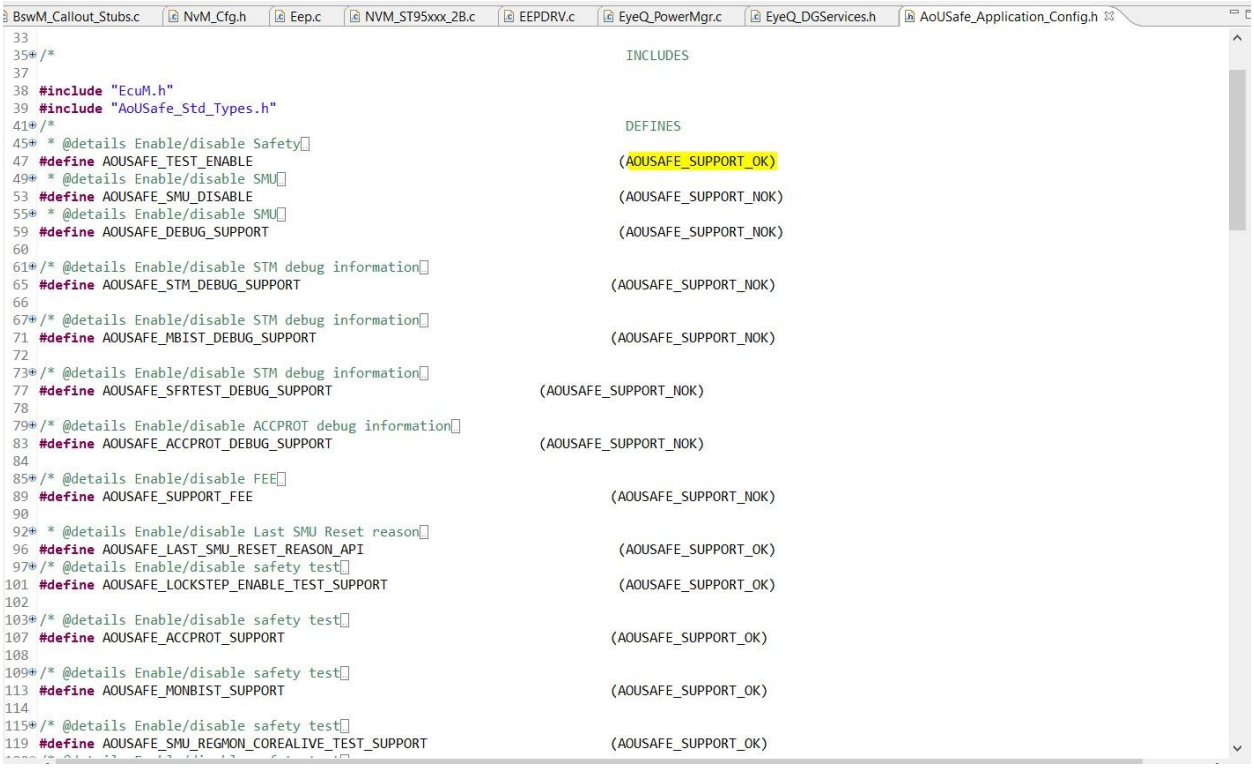
EyeQ CDD Integration

Application\SourceCode\IntegrationCode\Startup --core path
Application\SourceCode\Appl\Common\Startup – GWM path

NOTE : Some of the files may be adapted from application software(Project specific).

9.5 AoUSafe Integration:

AoUsafe folder path is in CDD folder as discussed earlier. We need to take all files as it is from core .For AoUSafe to work without issues we need to configure certain files according to project .In path "Application\SourceCode\Core_Comp\AppConfig\Config_AoUSafe\include" we need to configure AoUSafe_Application_Config.h file according to project architecture .We can enable safety or disable from this file .There are many other configuration in this file which can be modified.



The folder **AoUSafe** should be copied from Core software(refer to **Heading 5**) to Application software.

NOTE : When the core platform integration is done for the first time,it is suggested that **AoUSafe** integration can be done at last.

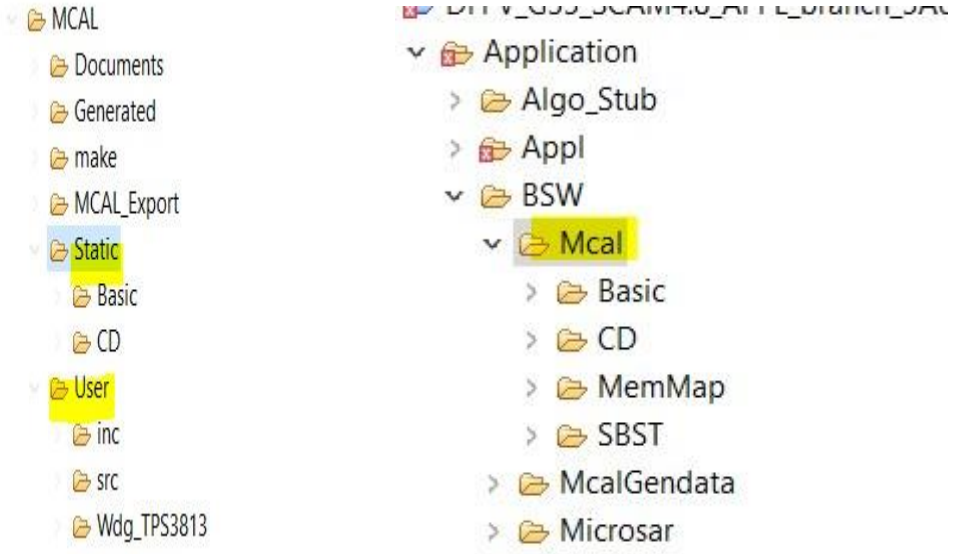
EyeQ CDD Integration

9.6 MCAL Static code Integration

The folder **MCAL(static and user should not be missed)** should be copied from Core software to Application software. The destination folder in Application software may vary across the projects.

Application\SourceCode\MCAL -core path

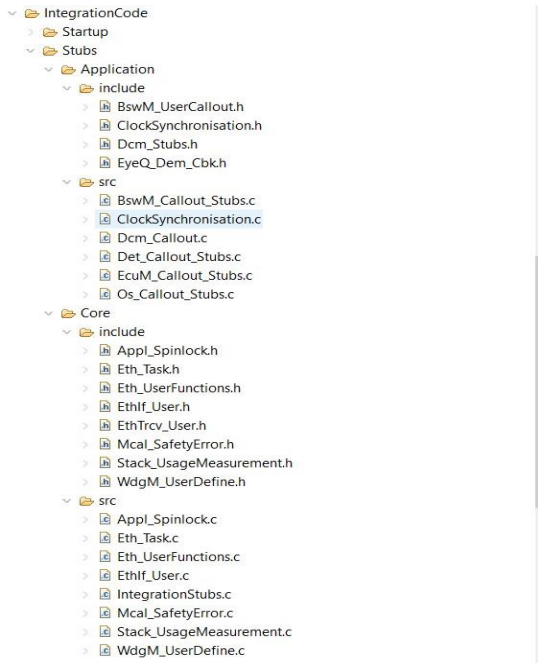
Application\BSW\Mcal -GWM path



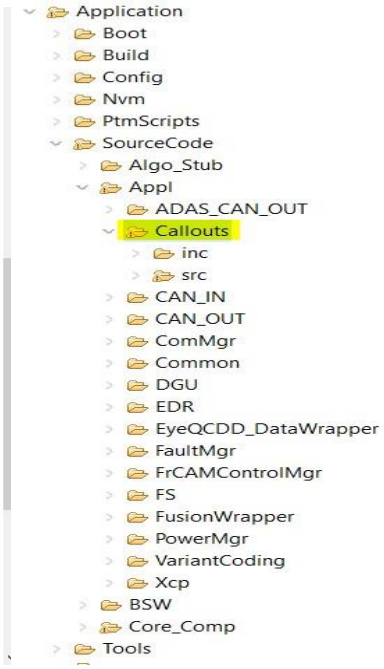
9.7 Other Static code Integration.

There are other files in Application\SourceCode\IntegrationCode\Stubs should be integrated EcuM_Callouts_Stubs,BswM_Callouts_Stubs should be carefully integrated which as satup and shutdown strategy.Stubs in GWM are at Application\SourceCode\Appl\Callouts.

EyeQ CDD Integration



Core stubs path



Gwm stubs path

10. ARXML import and BSW updates

Prerequisites

- Davinci Tools shall compatible with Vector SIP version.
- Platform compatible Tasking version should be installed.

10.1 Importing Arxmls to the Davinci Developer

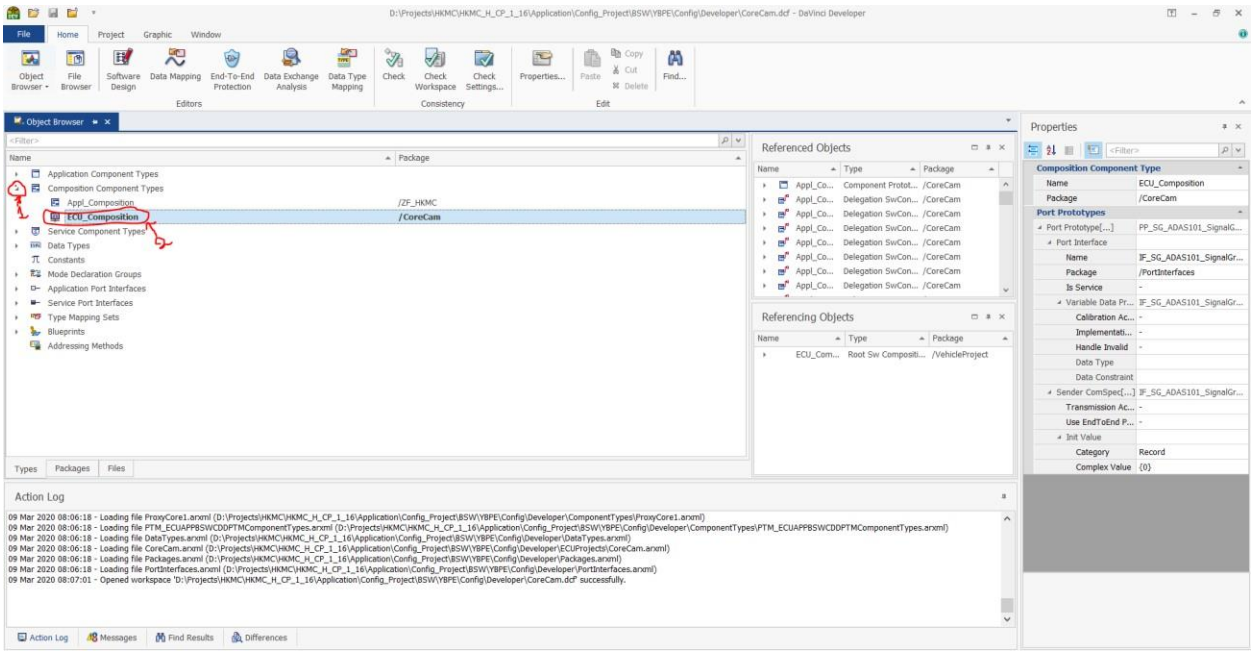
The first step in integration procedure is to import necessary Arxml files which are generated using Arxml generator procedure of EyeQ CDD, Proxy component and IOHWAB component which contains the description of the components.

These components can be overwritten on existing components or can be deleted from and imported into the project.

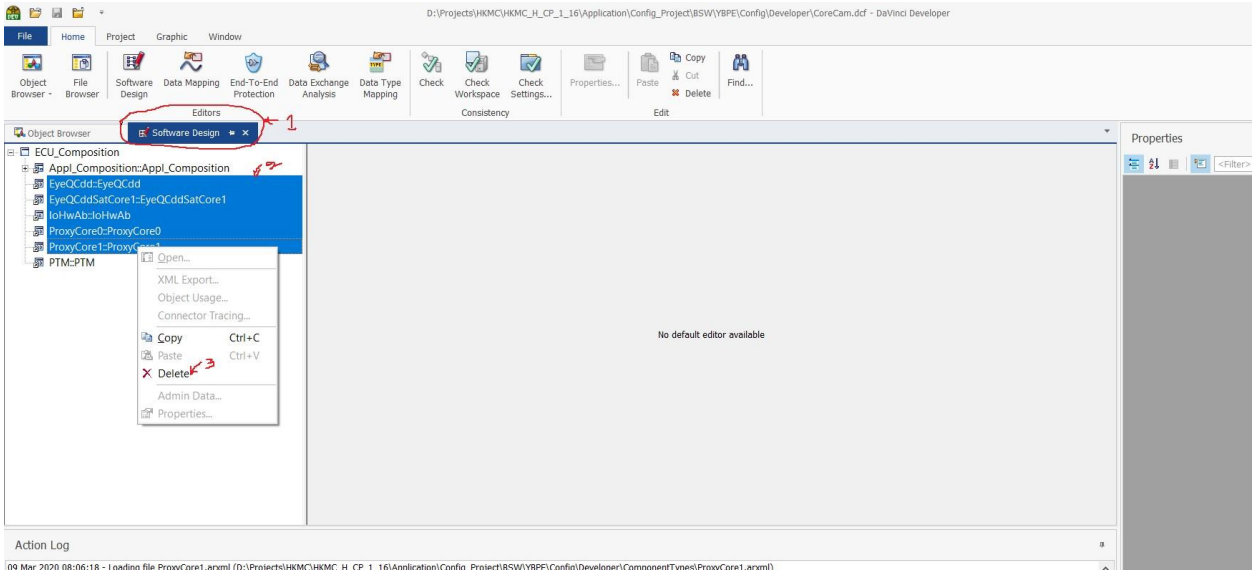
To delete components from project first delete components from ECU Composition and then delete from Application component types.

Click on Composition Component Types and then double click on **Ecu_Composition**

EyeQ CDD Integration

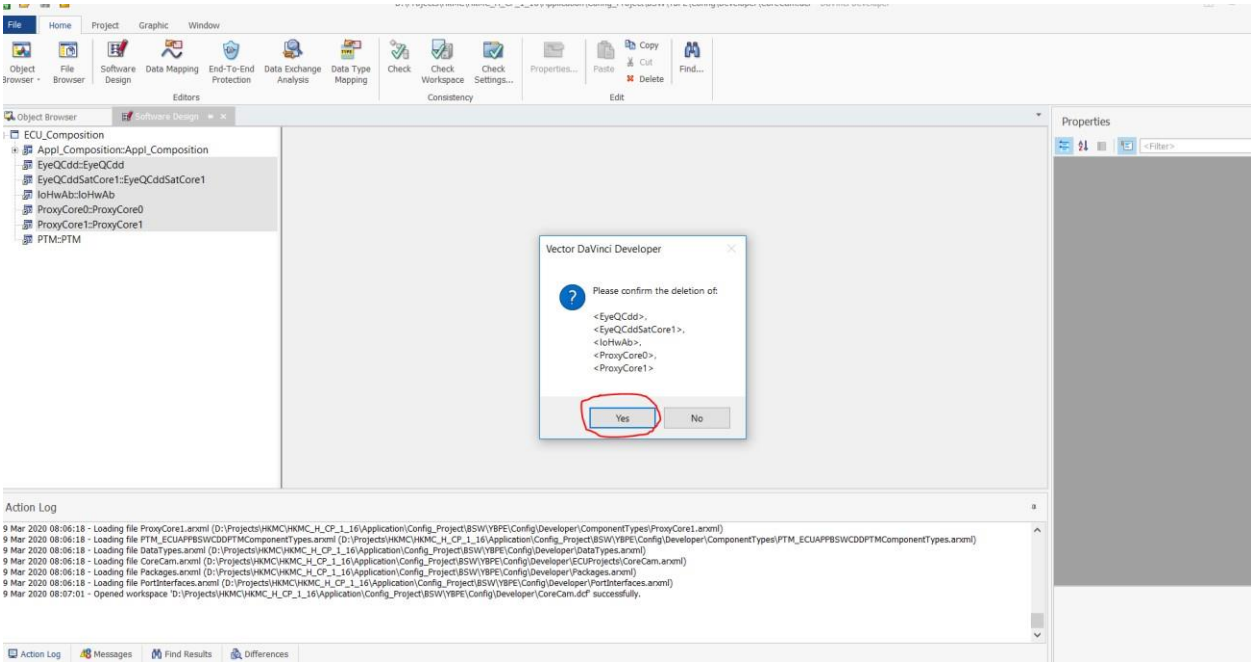


Select the EyeQ CDD core components from software design tab and right click and select delete.

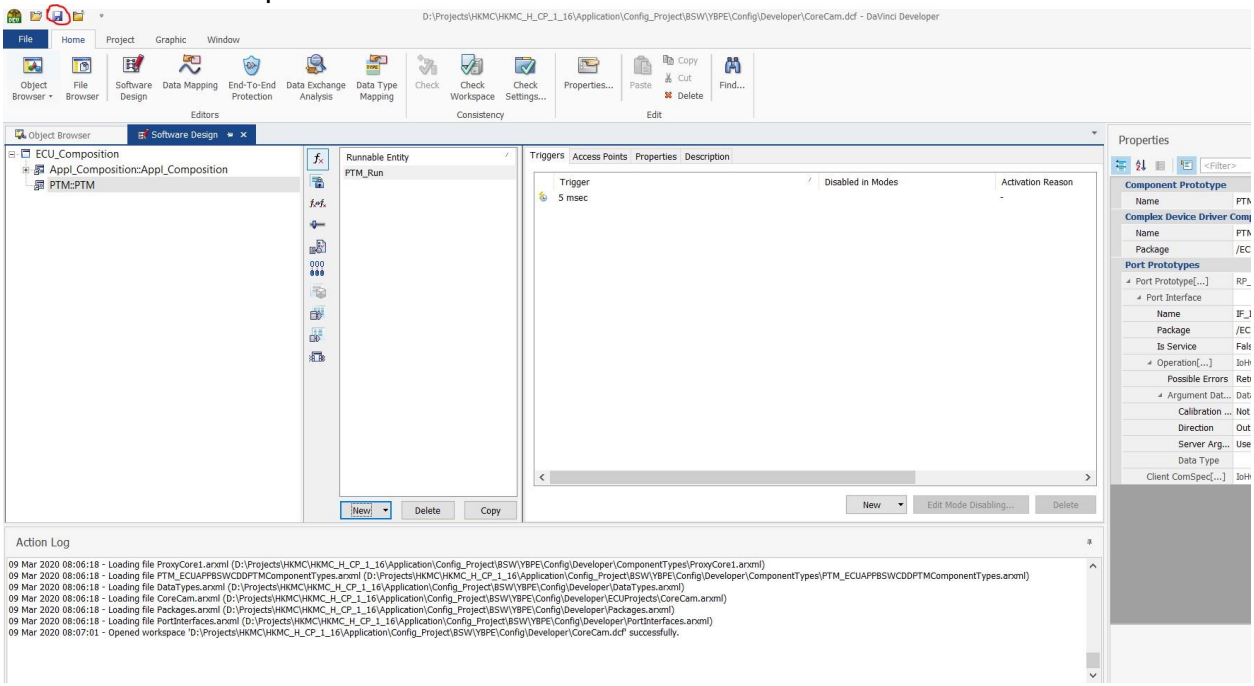


Click on Yes

EyeQ CDD Integration

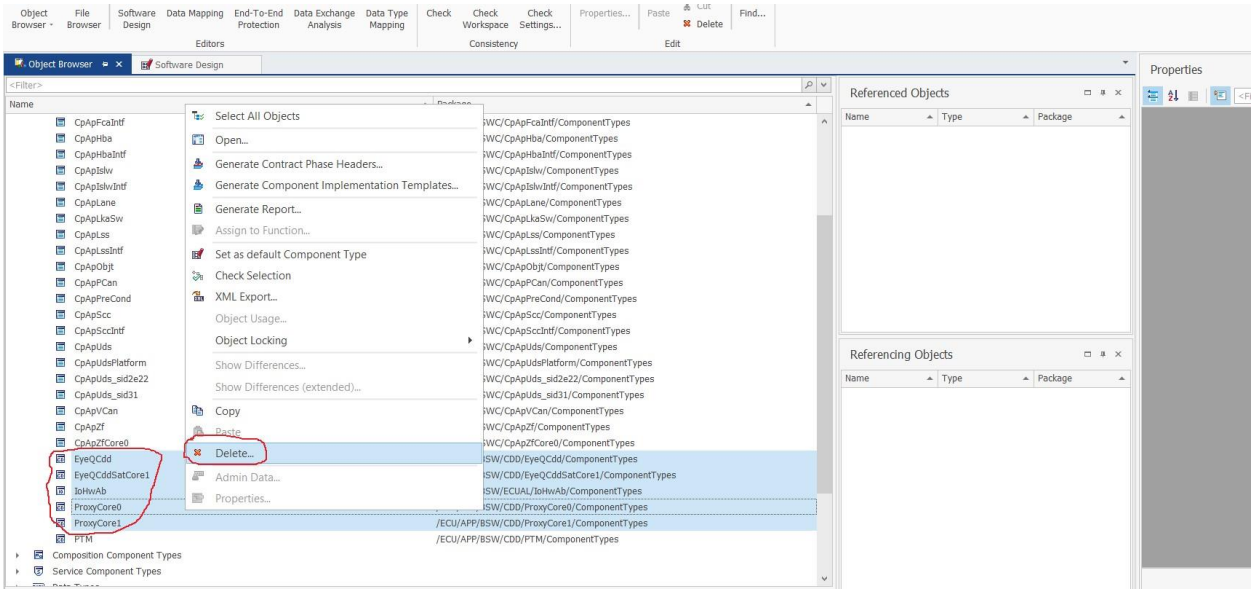


Now save the workspace.



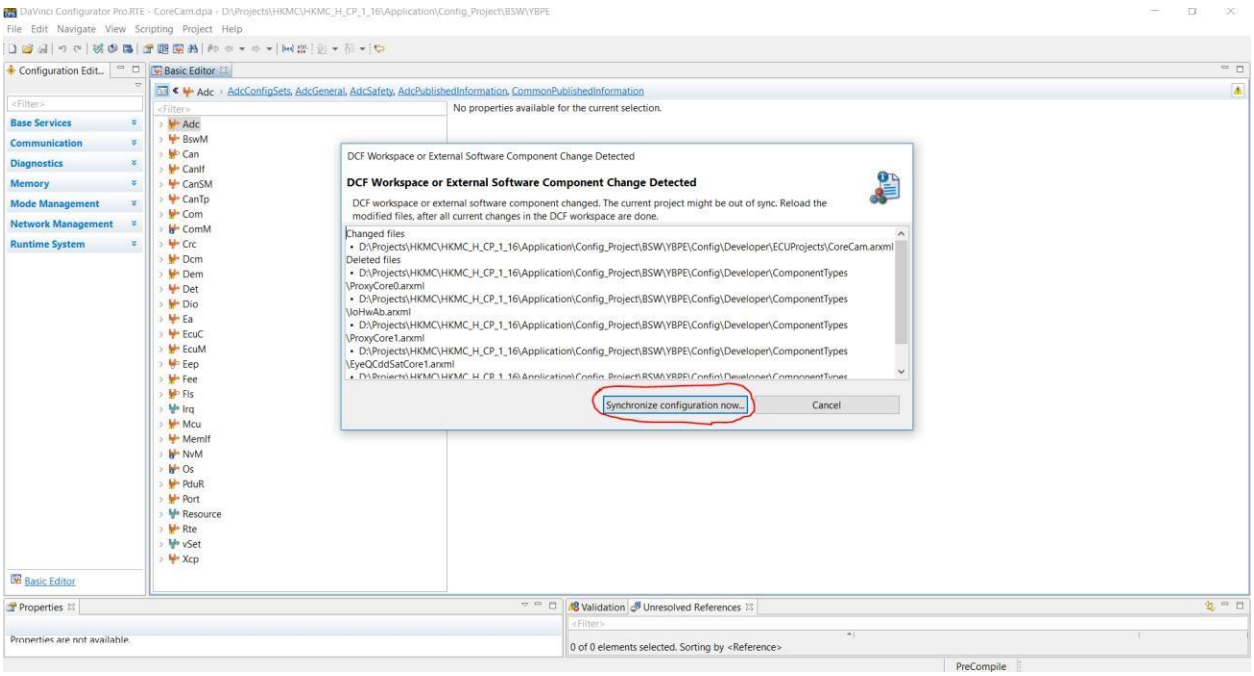
Now delete Application component types of EyeQ CDD Core from Application component types in object browser tab.

EyeQ CDD Integration



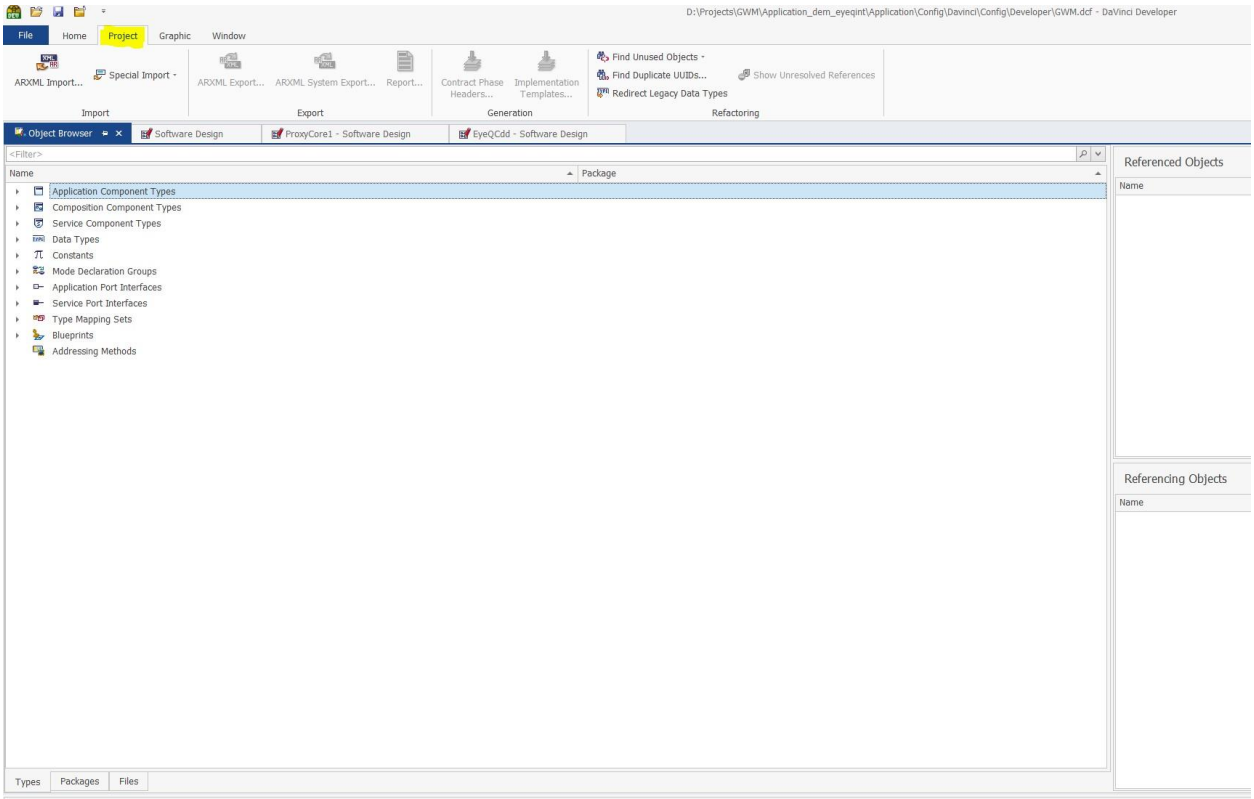
Now save the project.

Now open Configurator and the screen should apper as shown, then click on **Synchronize configuration now** and the click on save.

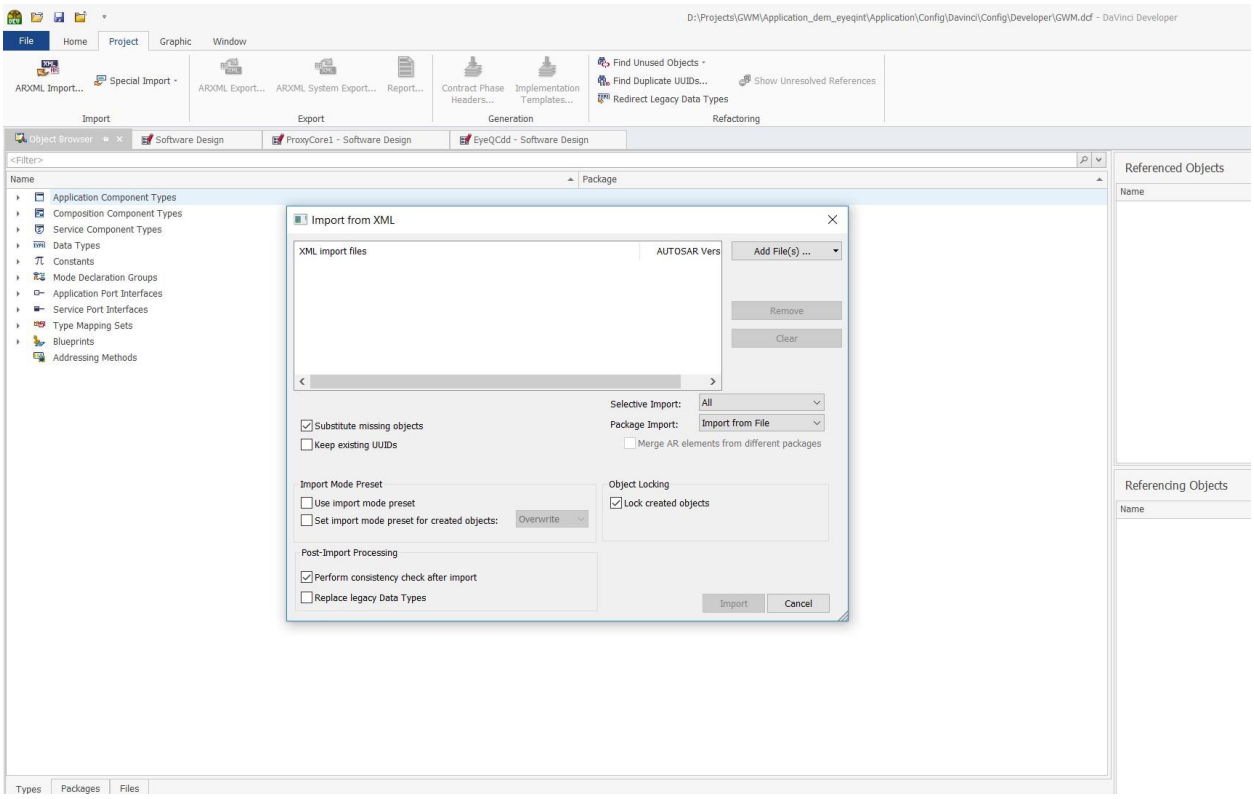


-->Open Davinci Developer project
-->Select project tab

EyeQ CDD Integration

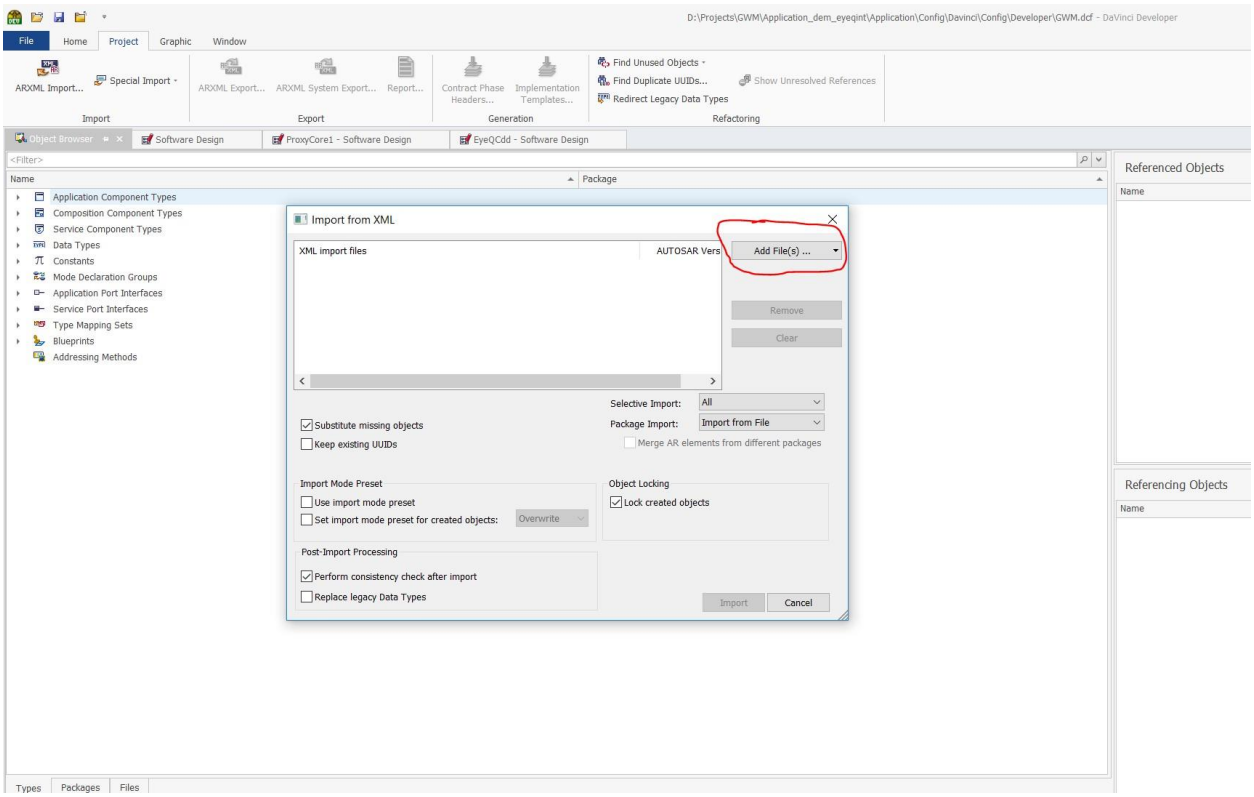


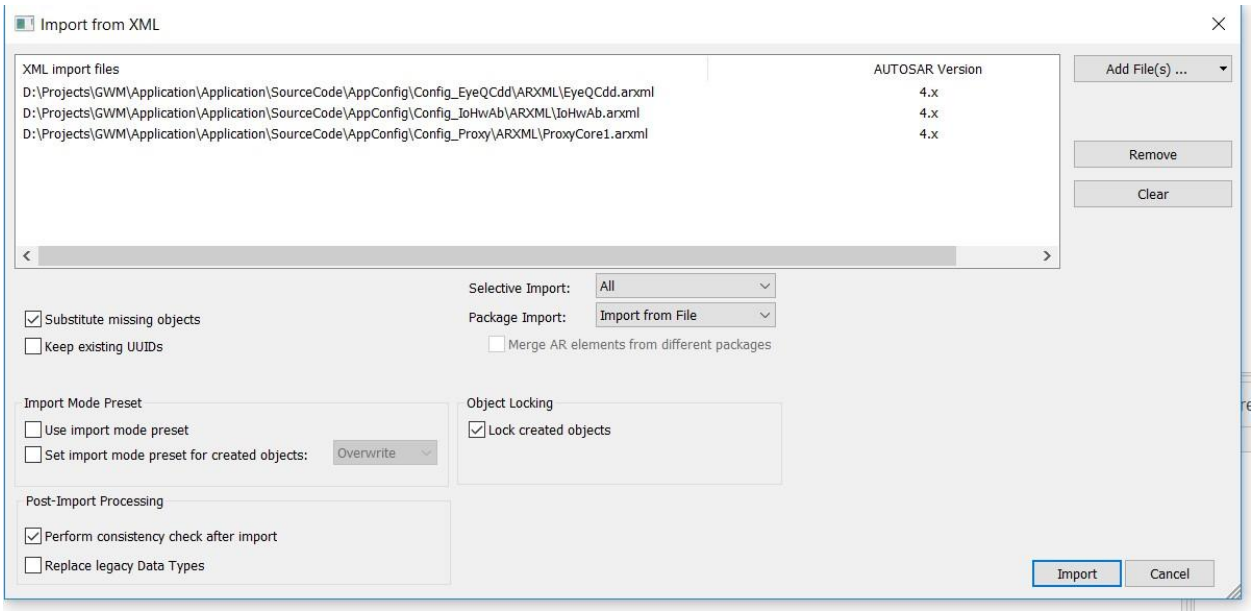
-->Click on ARXML Import



EyeQ CDD Integration

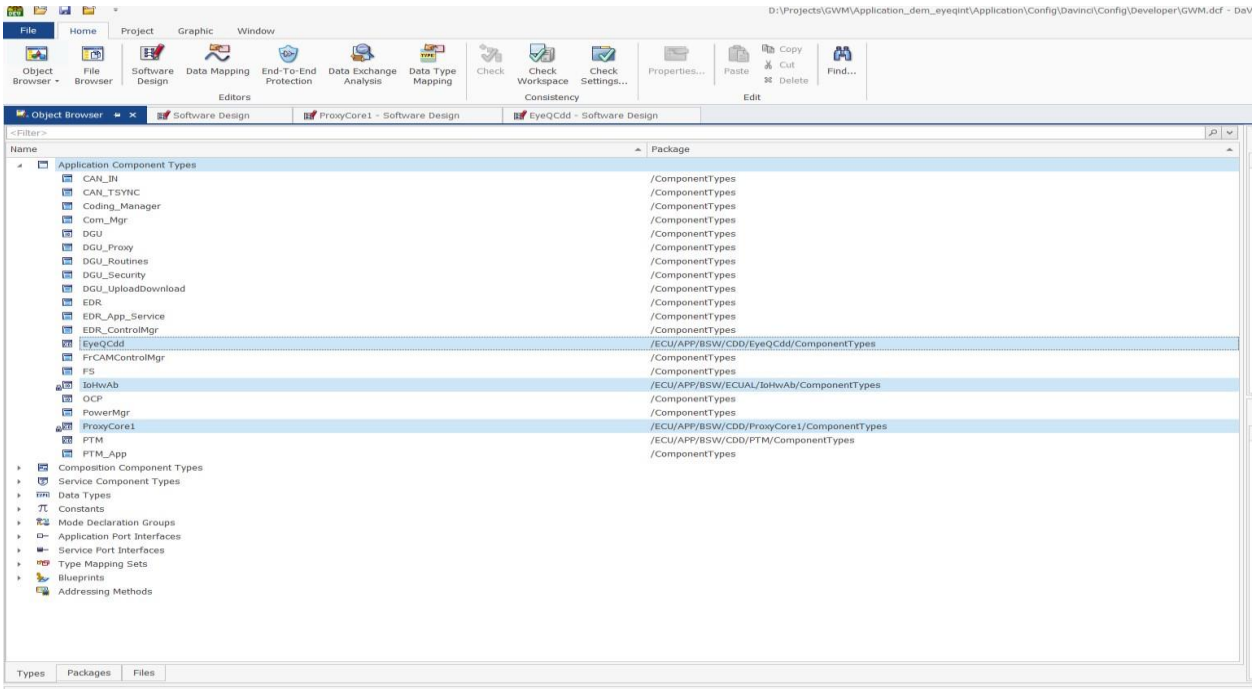
-->Click on Add Files and then select EyeQCdd.arxml, IoHwAb.arxml, ProxyCore1(or ProxyCore0 depends on architecture).arxml from AppConfig folder which are generated as per project needs by Arxml generator tool in the project and then click on import.





EyeQ CDD Integration

Once imported successfully these components should reflect in Home-->Object Browser->Application Component Types



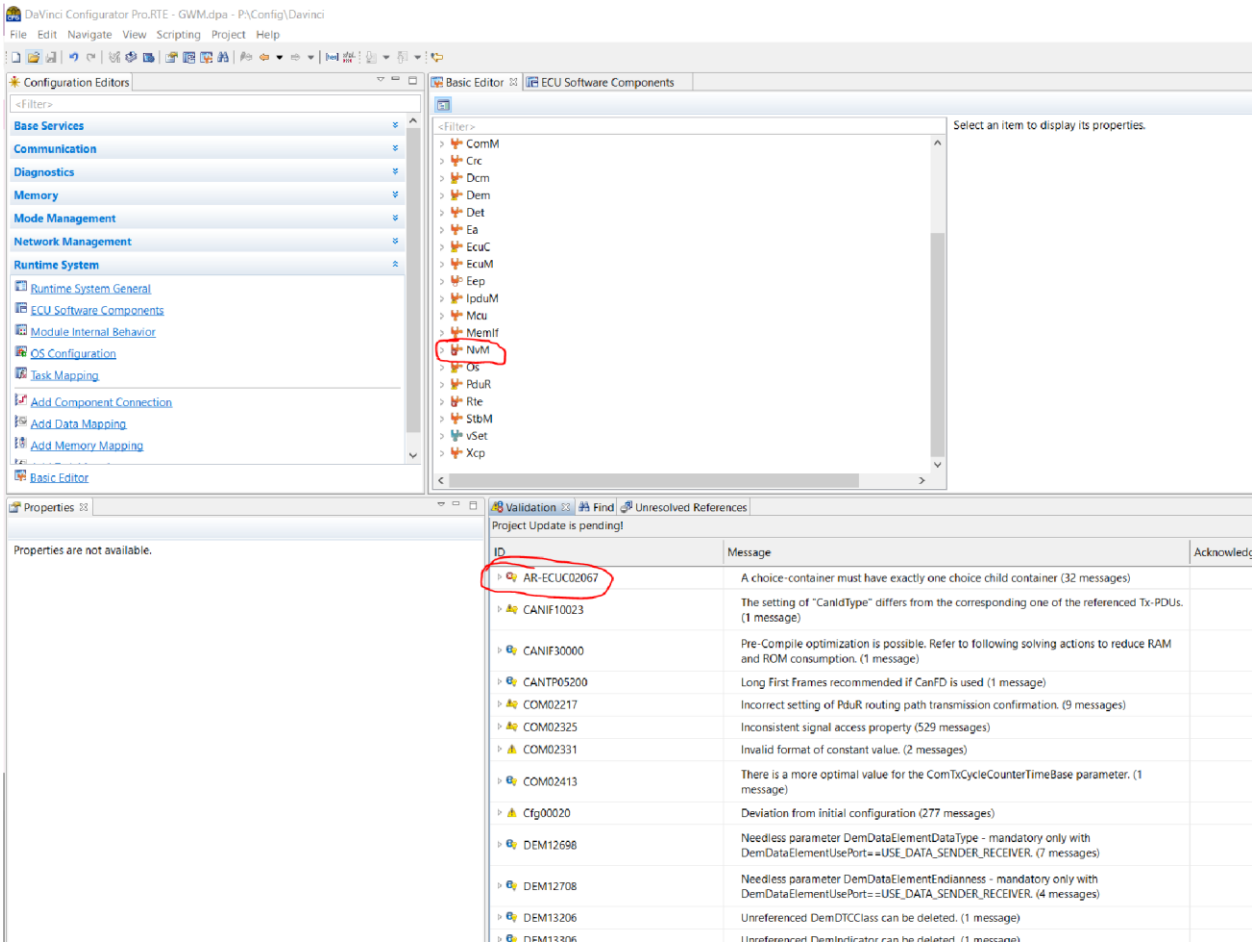
NOTE: If components are newly added the components should appear in **Application Component Types.** If components are already exists then components can be overwritten.

EyeQ CDD Integration

10.2 NvM Integration

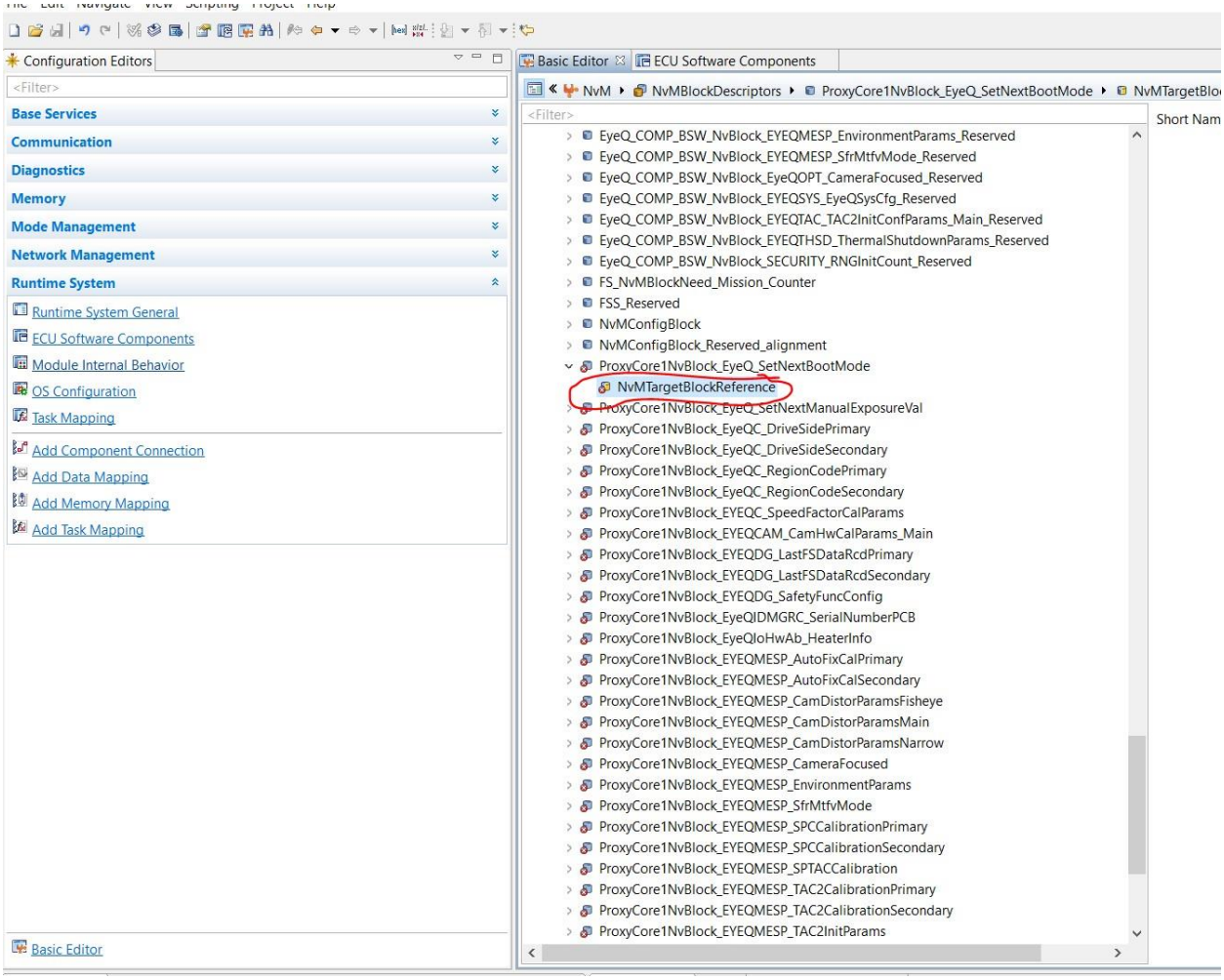
Once the arxmls are imported Davinci configurator will synchronise with the Davinci developer project. After synchronization the NvM configuration needs will be updated. Follow the below procedure to cofigure NvM.

Check for NvM errors in **validation** tab.



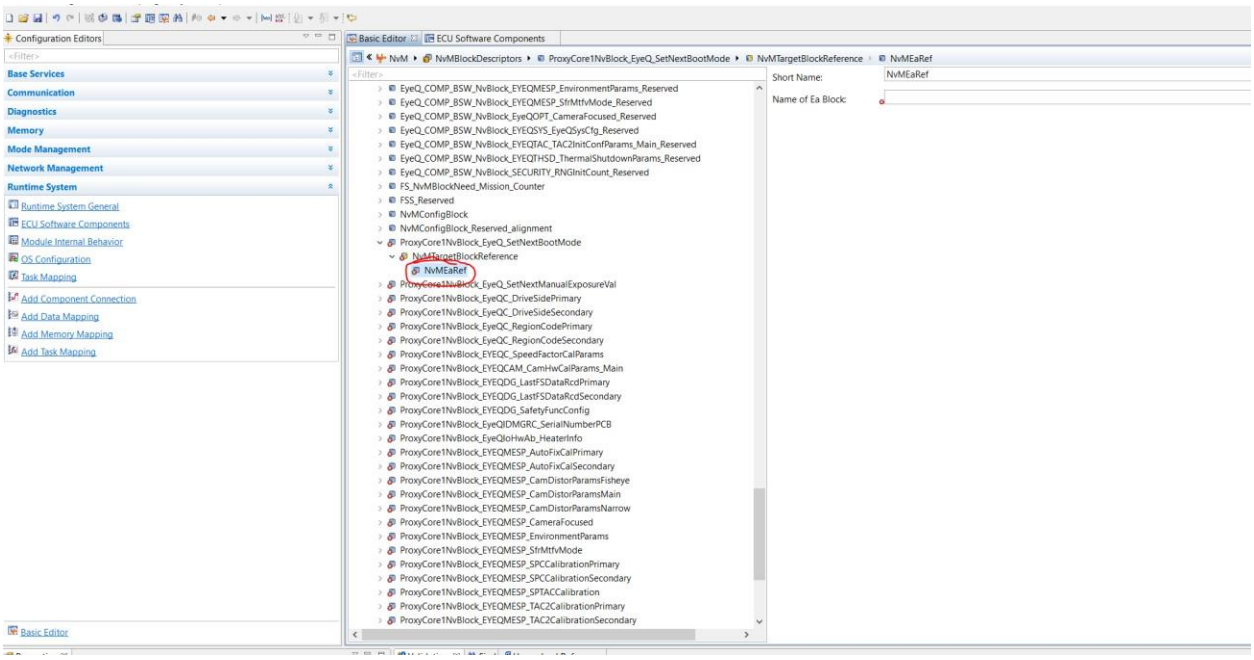
Go to faulty blocks in NVM Component and

EyeQ CDD Integration

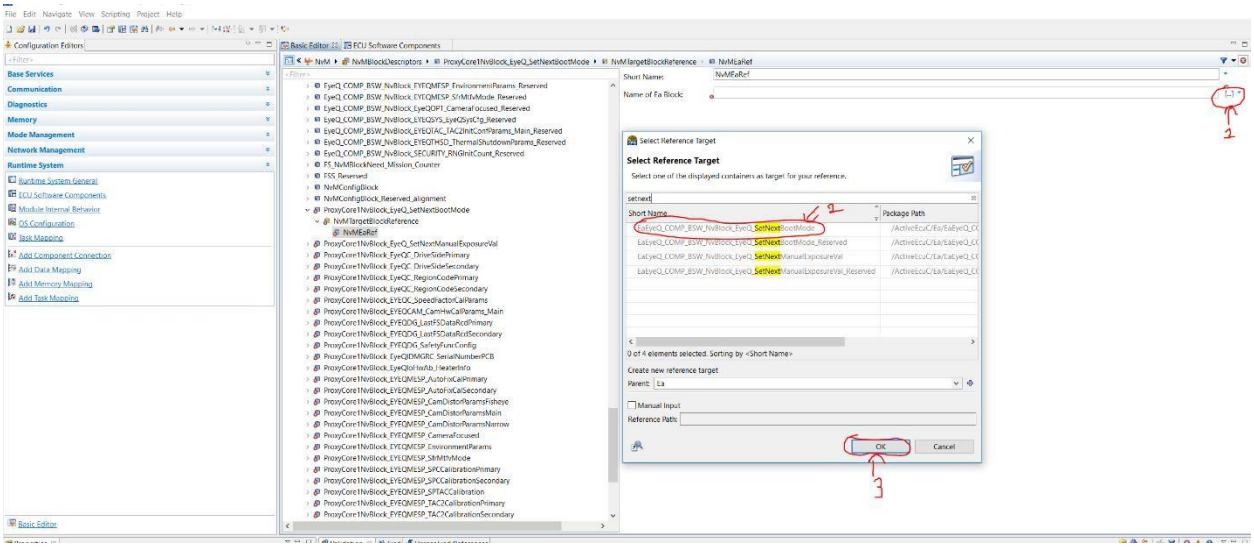


Right click on the Block reference and Choose Ea Block or FEE block based on project requirement.

EyeQ CDD Integration



Select an Ea block by clicking on select a reference target and then select the respective block.(Ea block configuration is taken for example)



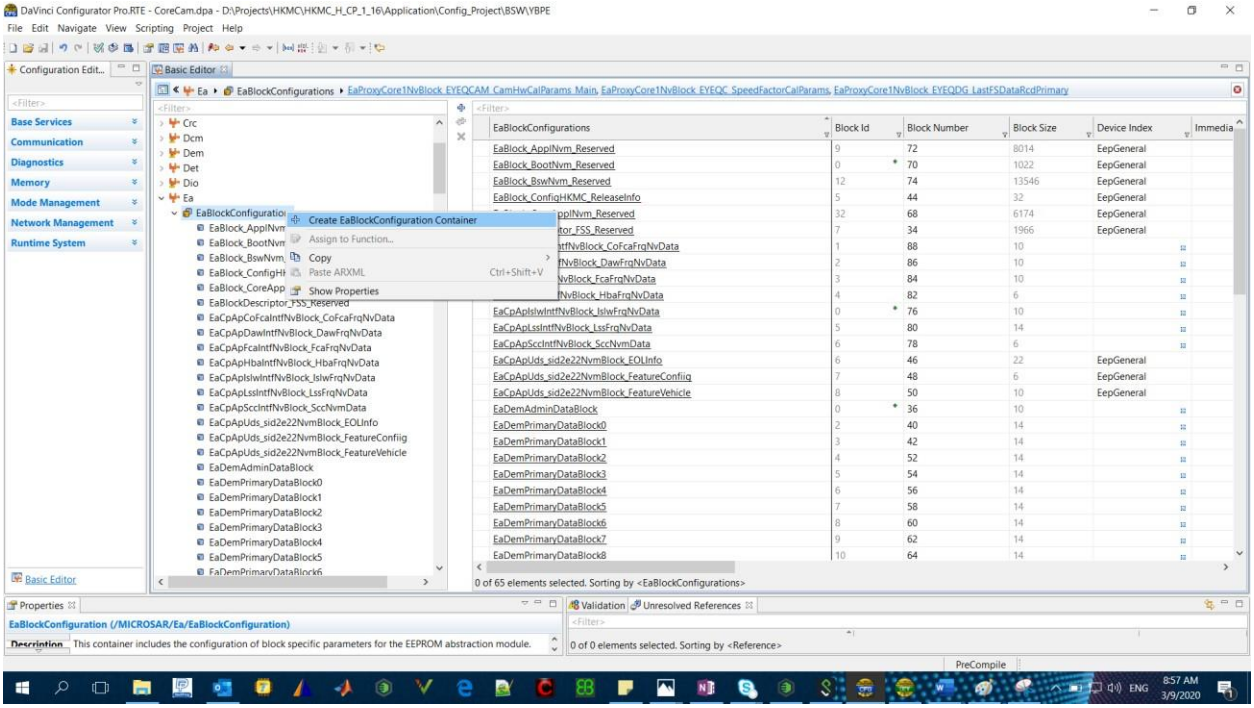
Repeat same steps for all faulty blocks and validate once finished.

If some blocks are newly added/removed in core software then create/delete Ea/Fee blocks first in Ea/Fee in EyeQ CDD CORE partition in Application software and then give the Ea/Fee block reference in NvM Configuration. Update the NvM block lengths as per CORE software requirement and adjust the Core NvM
Reserved space in EyeQ CDD CORE partition

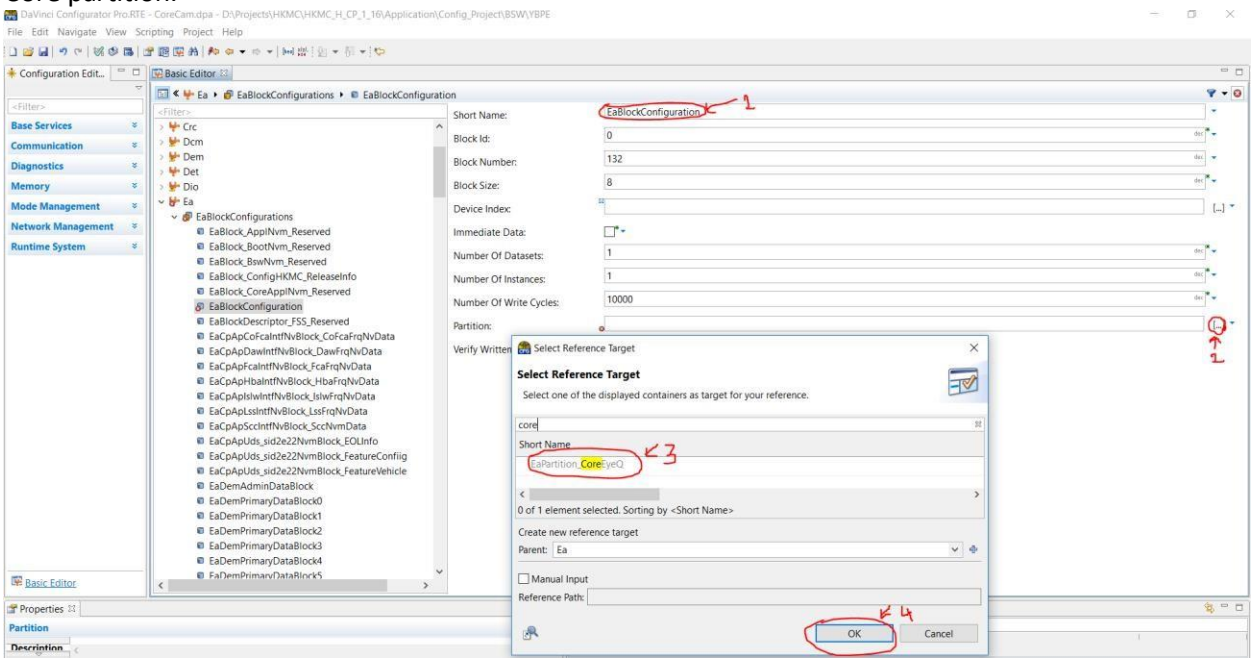
Creating and deleting Ea/Fee Blocks

EyeQ CDD Integration

In configurator open **Ea** module and the right click on **EaBlockConfigurations** and the select **CreateEaBlockConfigurationContainer.**



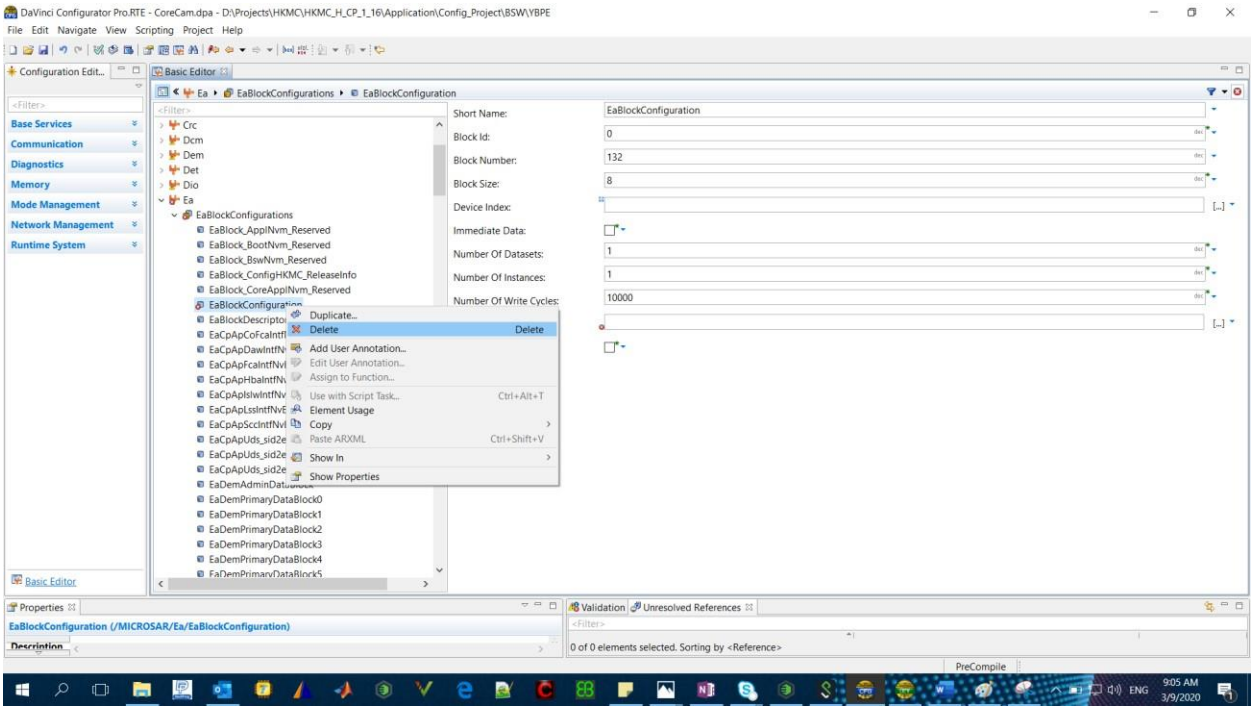
Click on created EaBlock and rename the block with Core specific block name and the select Core partition.



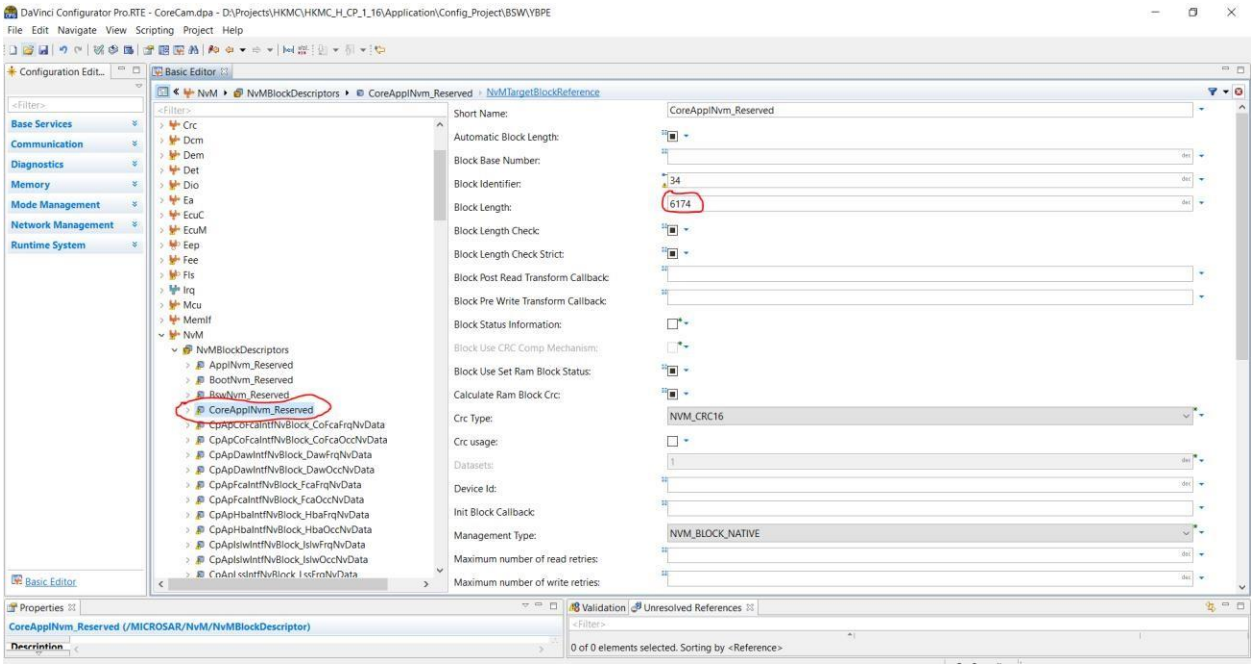
Repeat the above step for all newly created core blocks.

EyeQ CDD Integration

To Delete Ea block configuration which are removed from Core, Right click on the block to be removed and select delete.



Now adjust the Core reserved space according to new NvM layout of the Core.



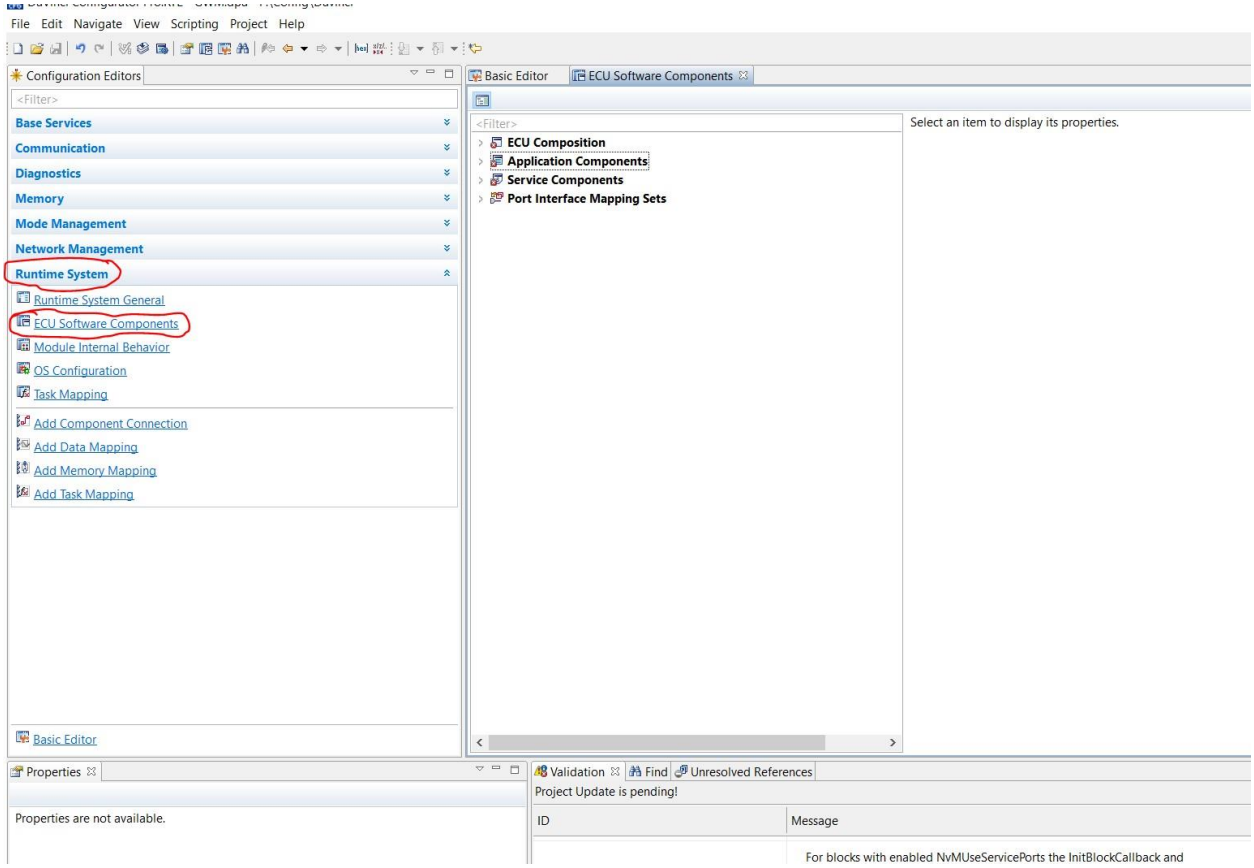
If this step is not done then the size NvM partition for Core and the sum of configured core block lengths along with Core reserved block will not be same. Then Nvm and Ea modules

will throw an error. To calculate the Core partition length, Subtract the lengths of blocks which are newly added from Core reserved block length and add the lengths of the blocks which are removed/deleted to Core reserved block length. The lengths of CRC and Start and End markers of the blocks should be included.

***Note**: To refer Core NvM Blocks the information will be given in EyeQCDD_Platform_Integration_Guide in Core software.*
After this respective changes in layout and .cfg should be done.

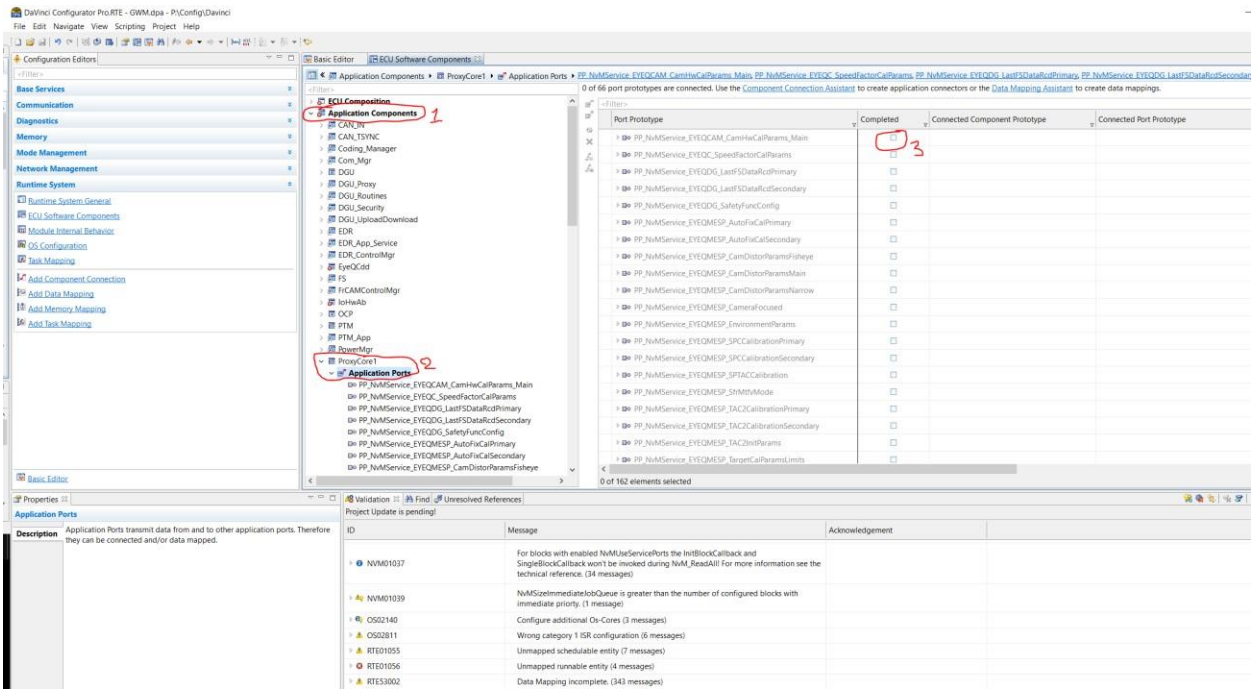
Connections to EyeQ CDD:

Click on Runtime system and then click on Ecu Software components.

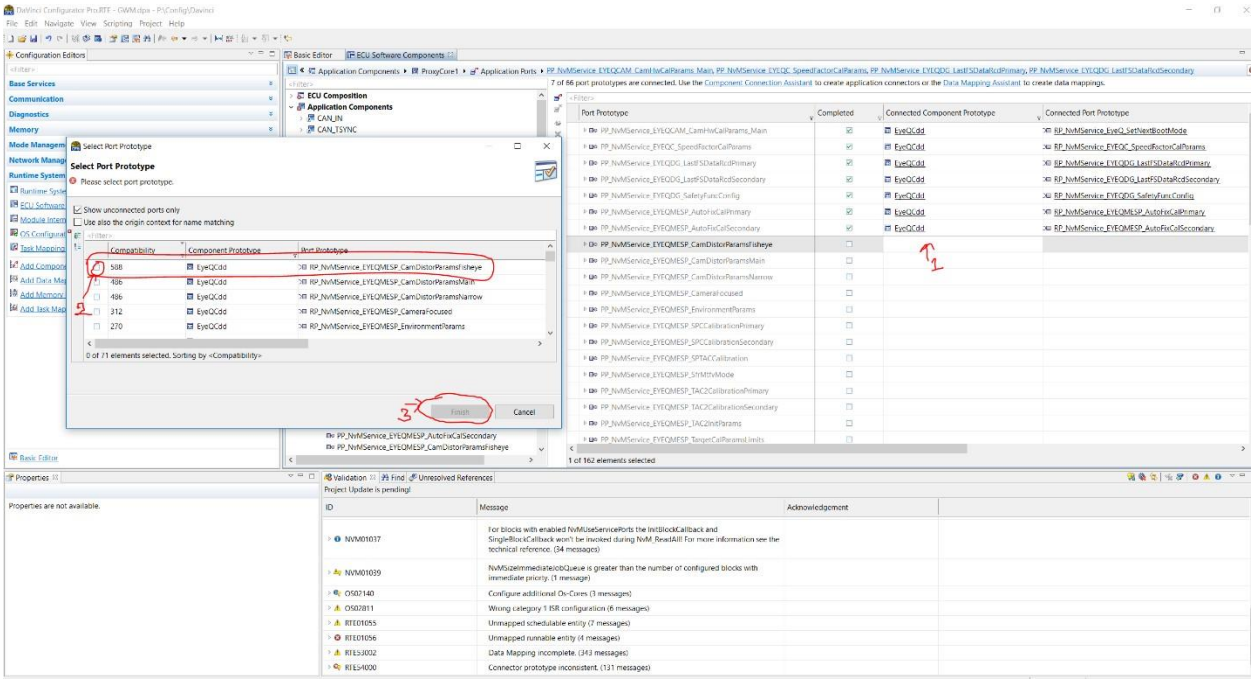


Click on Application components and then click on proxycore1(or proxycore0 depends on architecture) and the application ports and check for the connection status.

EyeQ CDD Integration



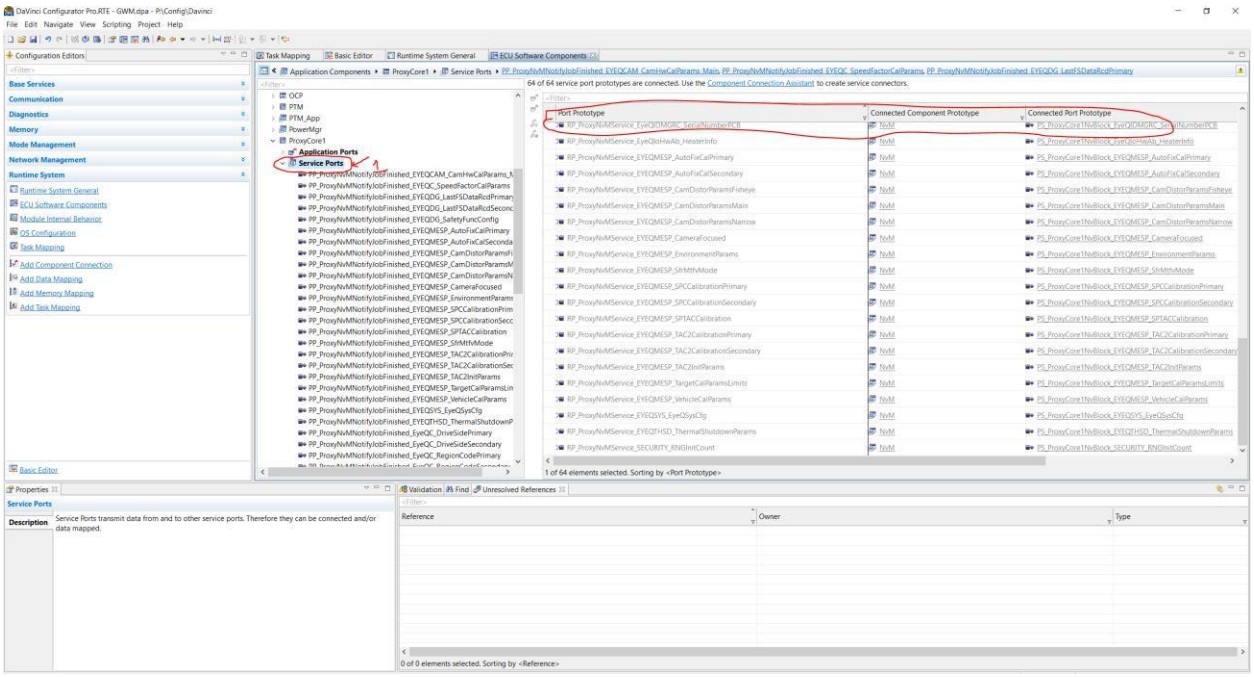
Compare with the port connections provided in core project and do the needful connections. Double click on Connected component prototype and select the port to connect and click on finish.



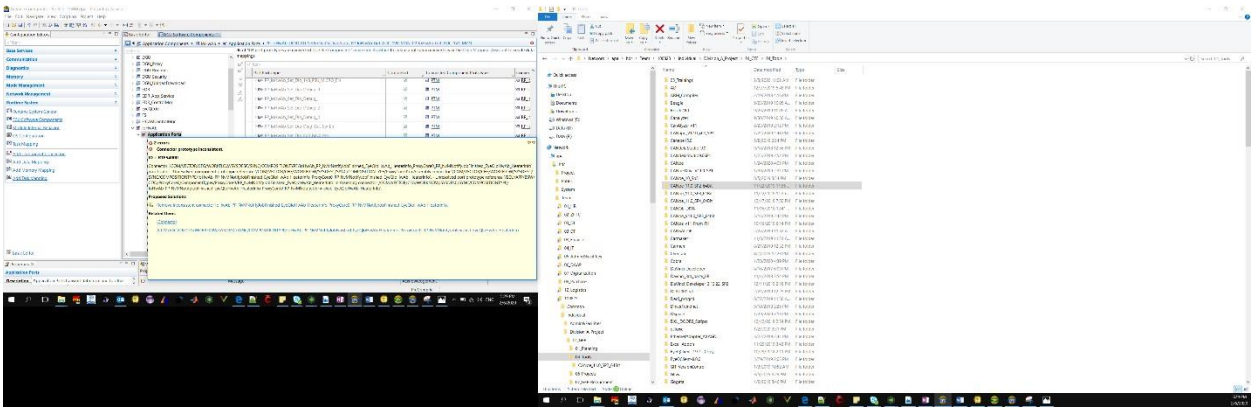
Connections to NvM:

Select Service Ports and if connections are not made do the connections. In figure connections are already made.

EyeQ CDD Integration



Remove inconsistent connectors as shown below(click on the link that is highlighted error will be removed.)



Once NvM Block references are added cross check the length of the blocks with the reference of Core software. Sizes of status blocks of DEM in NvM and admin blocks, primary blocks of dem need to be checked. If any warnings are present integrator should fix them. S37 of NVM need to adapted according to layout(Application Software).

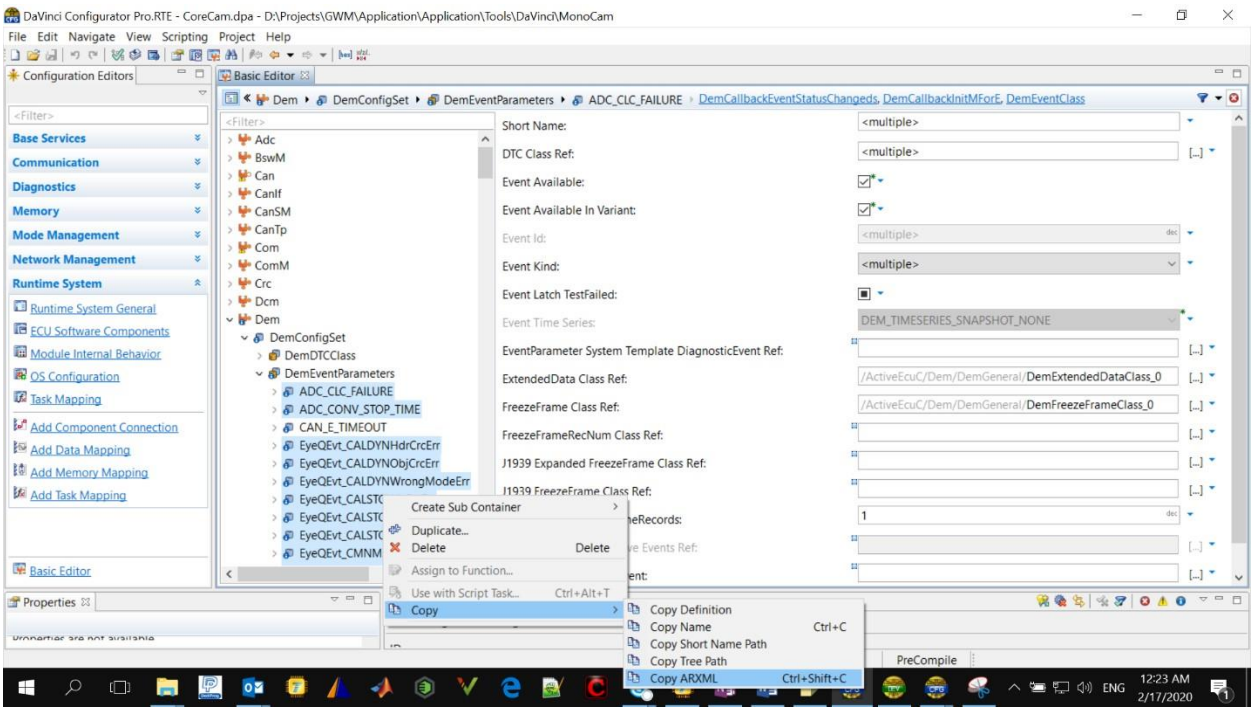
NOTE : If the arxmls(component descriptions) are over written only the effected blocks need to be confgured. If component is newly added all NvM blocks need to be configured.

EyeQ CDD Integration

10.3 DEM Integration

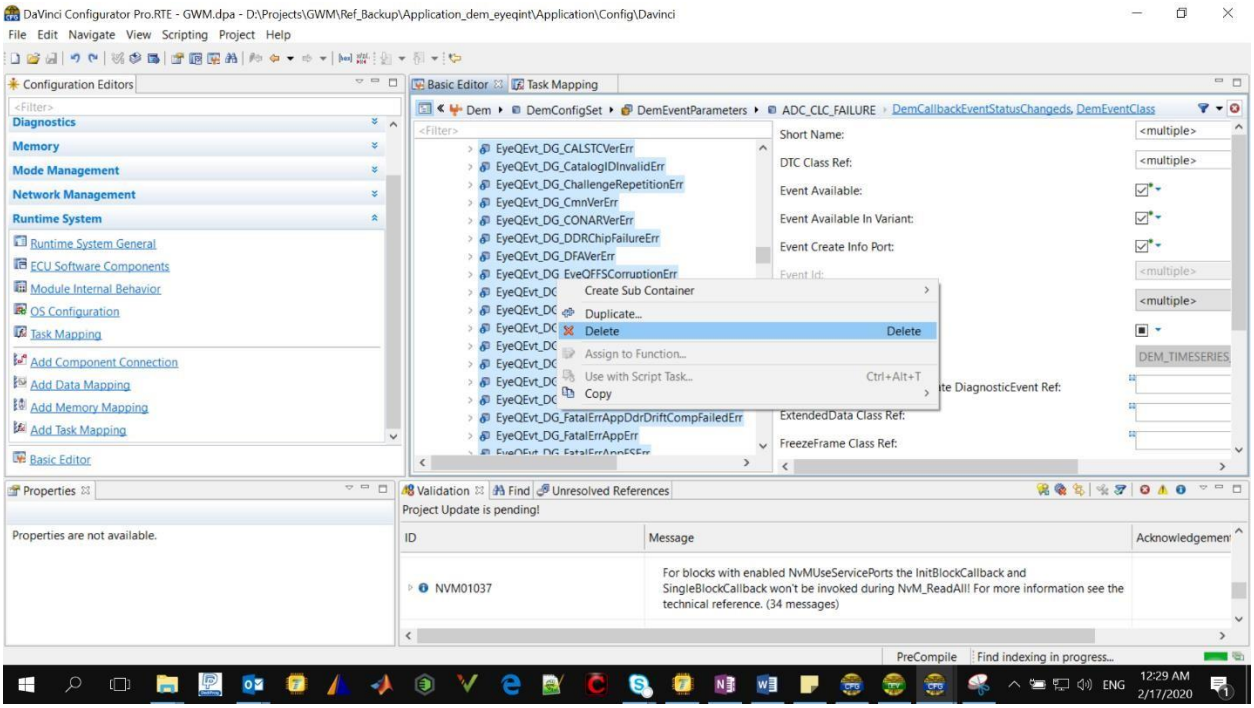
In DEM Integration all the EyeQCDD specific DEM events(Core project) need to be configured in Application project. Adapt the package paths in DEM event configurations from application project. Resolve the configuration errors if exists. Once configuration is completed connect the DEM ports(platfom specific).

Copy Core DEM Events from Core project to application project

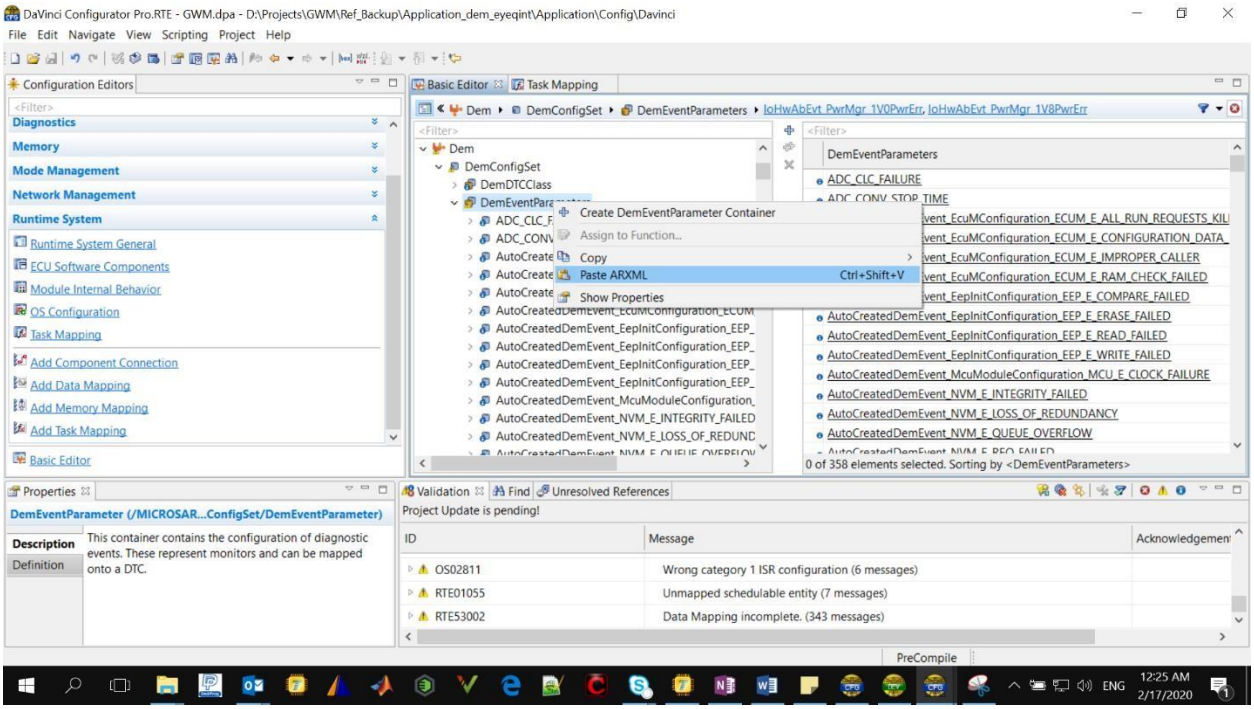


Delete existing Core DEM Events form Application project. For the first time integration this step is not applicable.

EyeQ CDD Integration

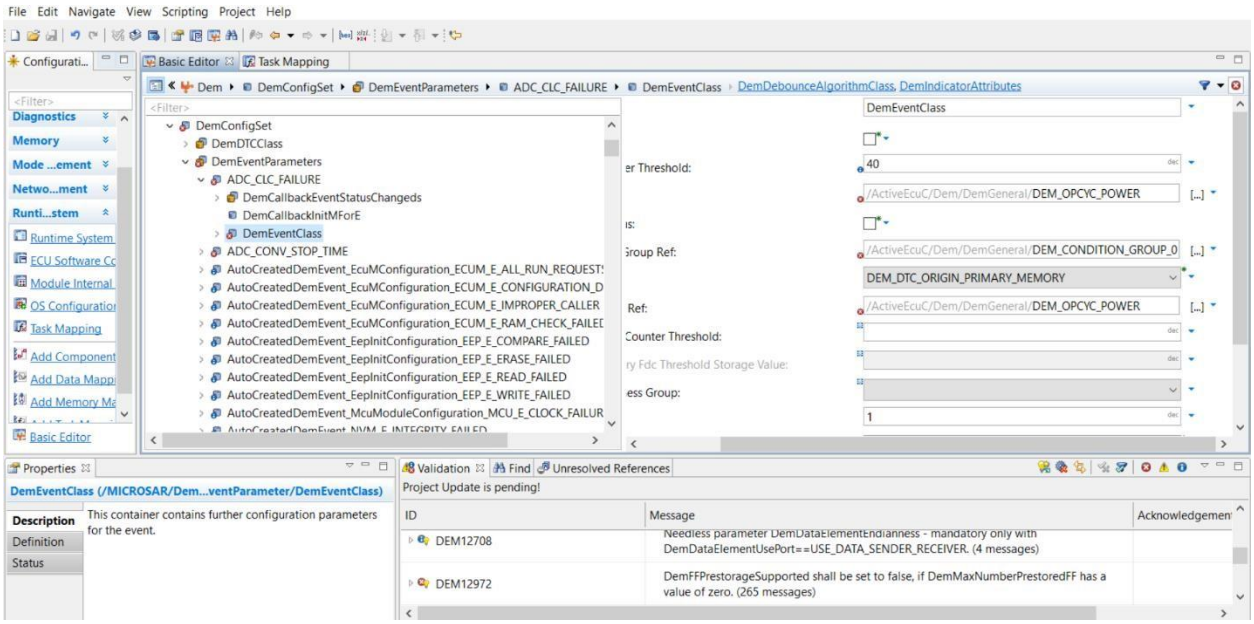


Right click on DemEventParameters Paste in Application Project

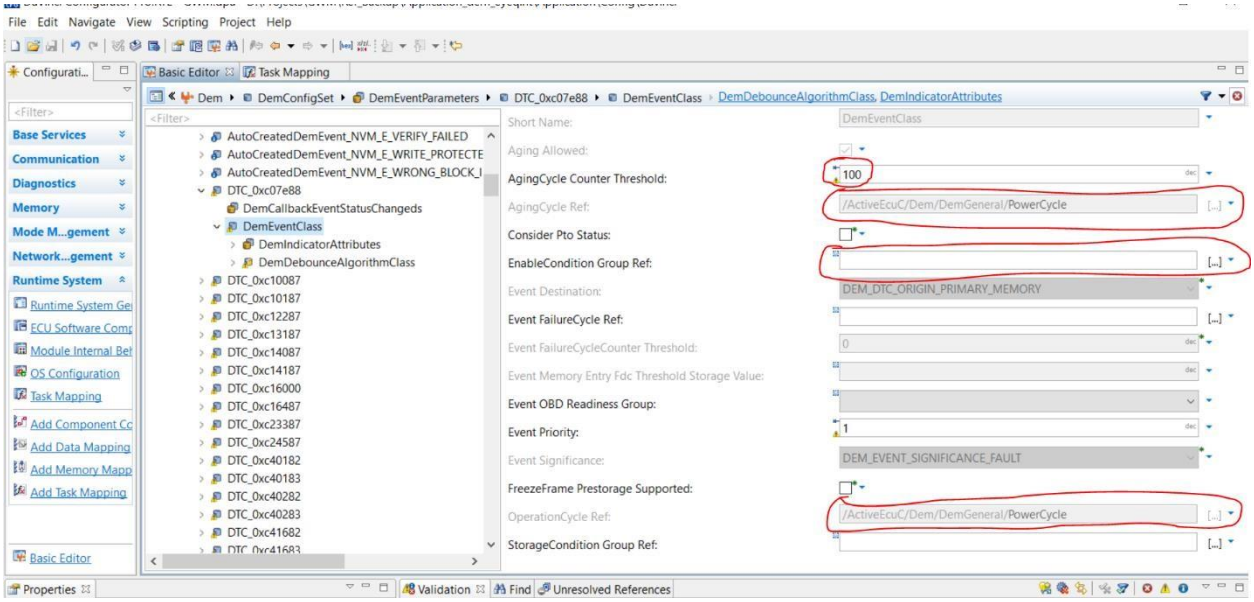


Adapt DEM configuration from Application software

EyeQ CDD Integration



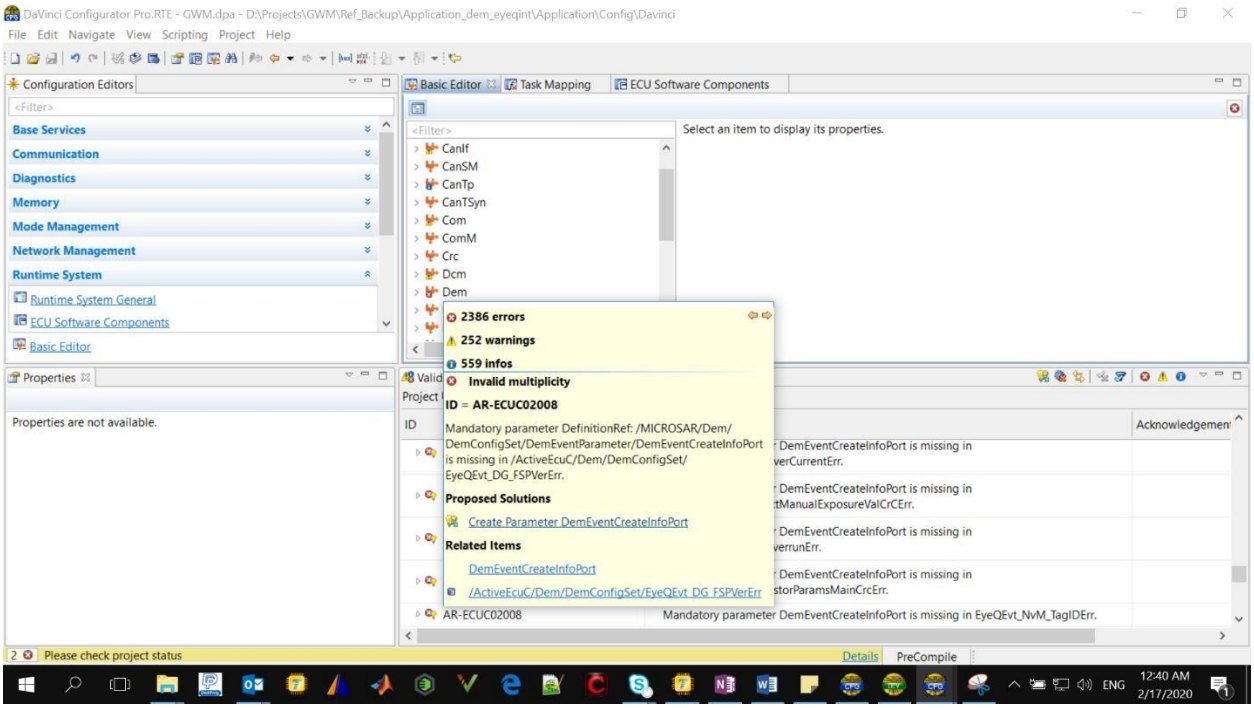
Take one reference DEM event configuration from application adapt highlighted changes into Core DEM events.



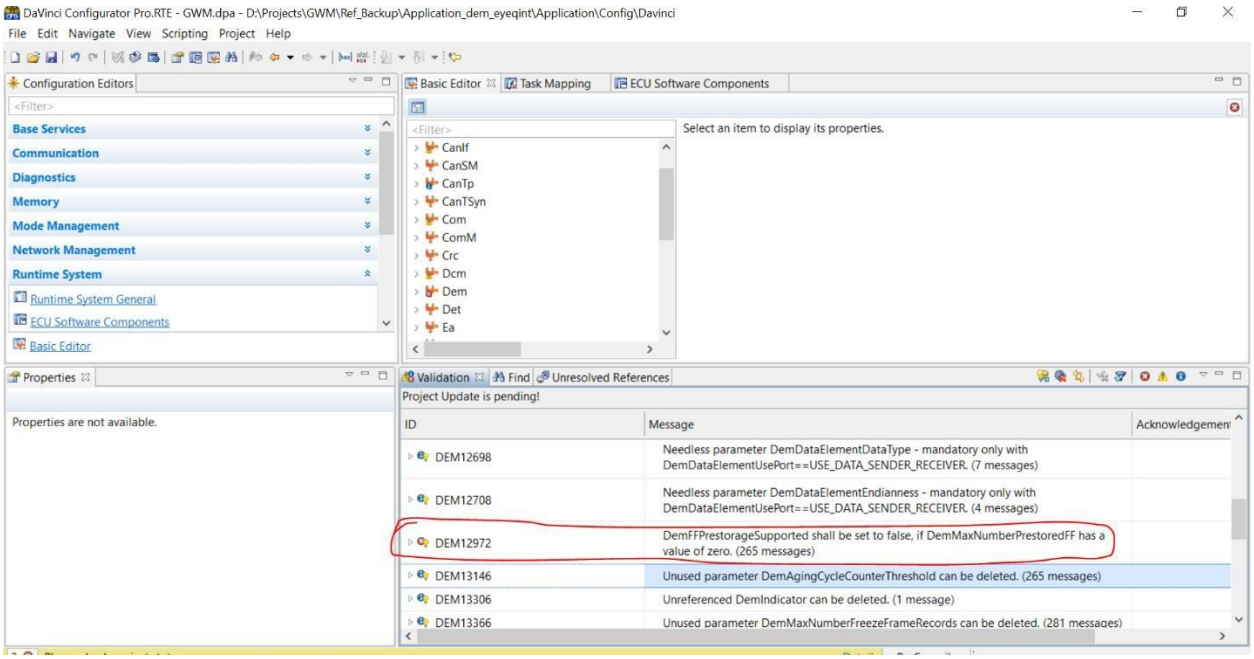
Repeat the steps for all DEM events copied from Core software.

Create a DEM port as shown. Click on Create Parameter DemEventCreateInfoPort(for all DEM Events).

EyeQ CDD Integration

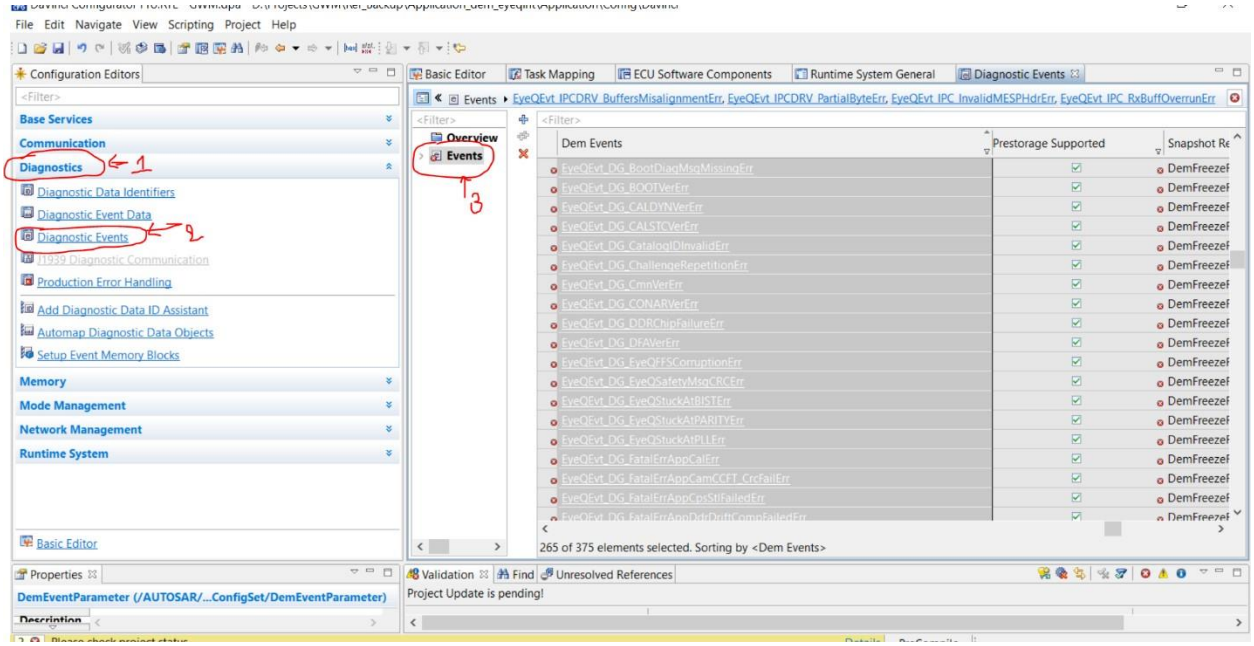


Check other errors related to DEM and resolve
the errors Ex:

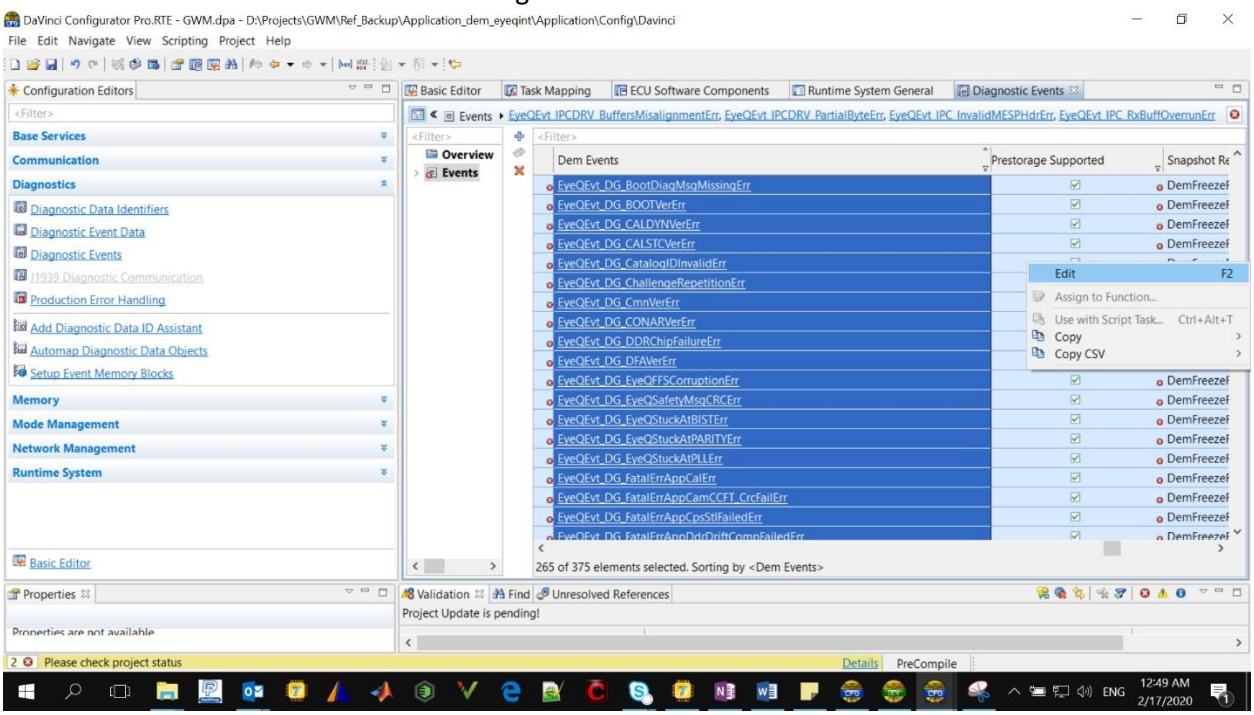


Click on Diagnostics and then select Dem Events as shown

EyeQ CDD Integration



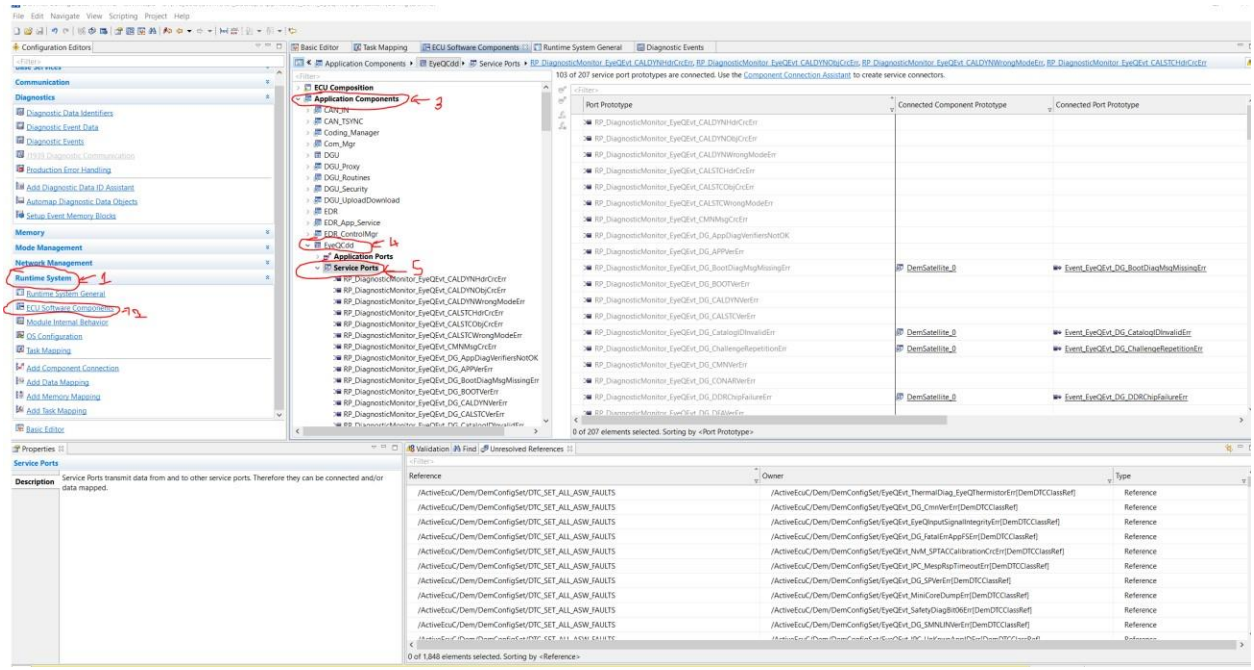
Find the Error field and select events and right click and select Edit.



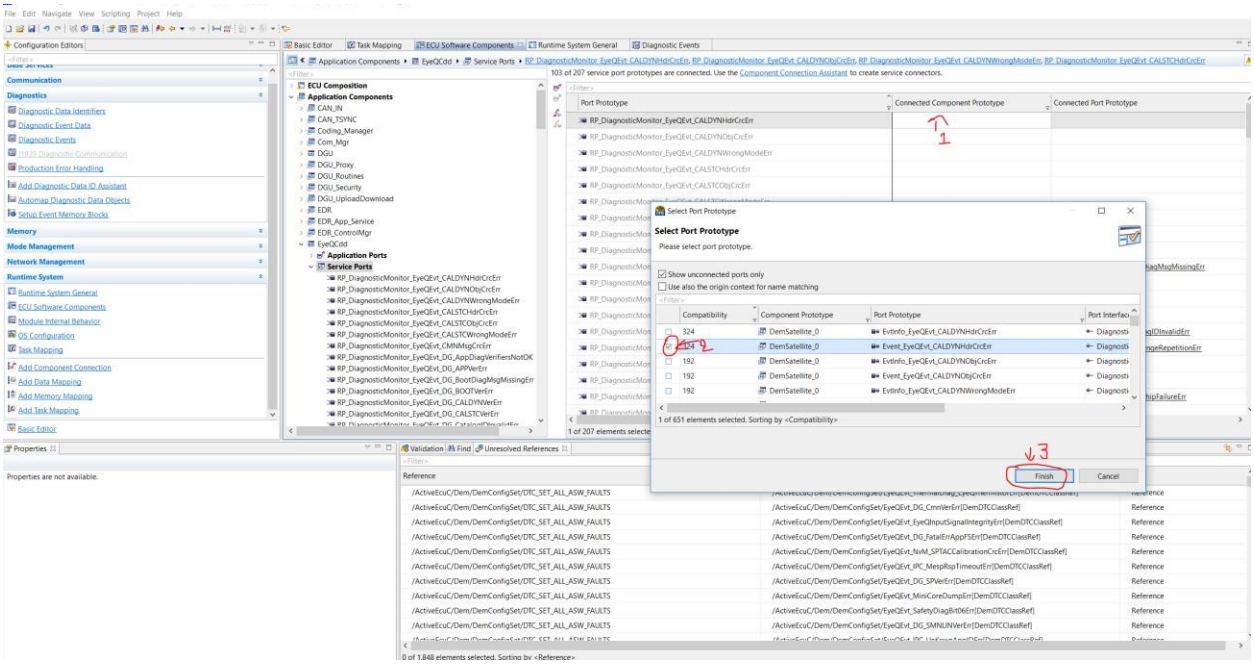
Resolve all the configuration errors.

Navigate through Runtime System->ECU Software Components->Application Components->EyeQCDD>Service Ports

EyeQ CDD Integration

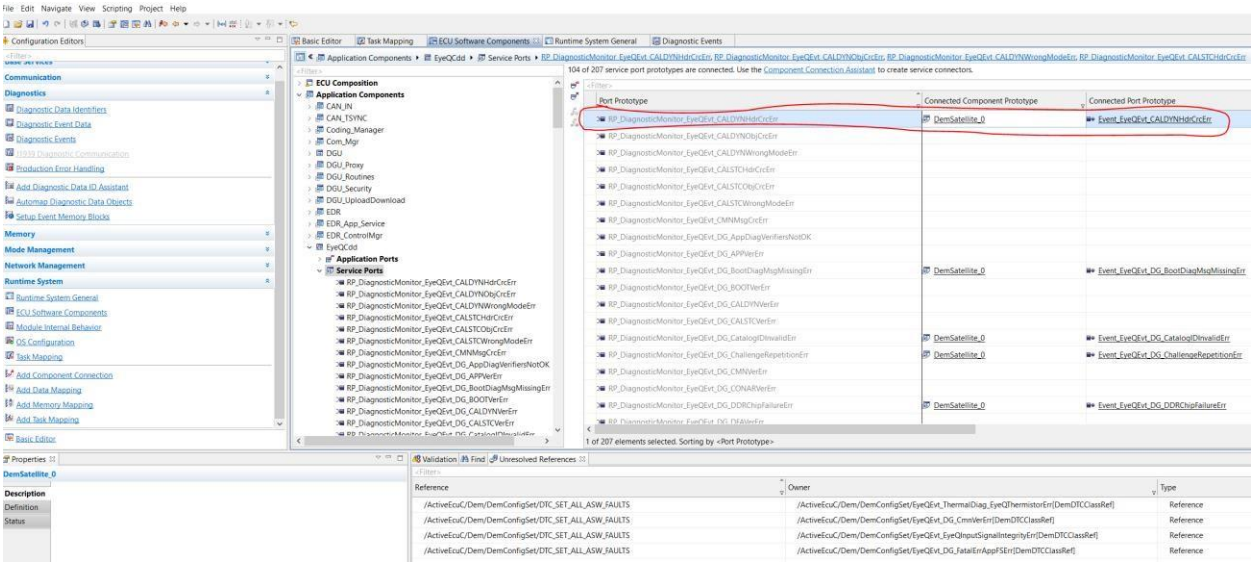


Double click on empty field in connected component prototype column and the select the destination port from opened window and then click on the check box and then click on finish.



Once port is connected connection status will be updated as shown in below figure.

EyeQ CDD Integration



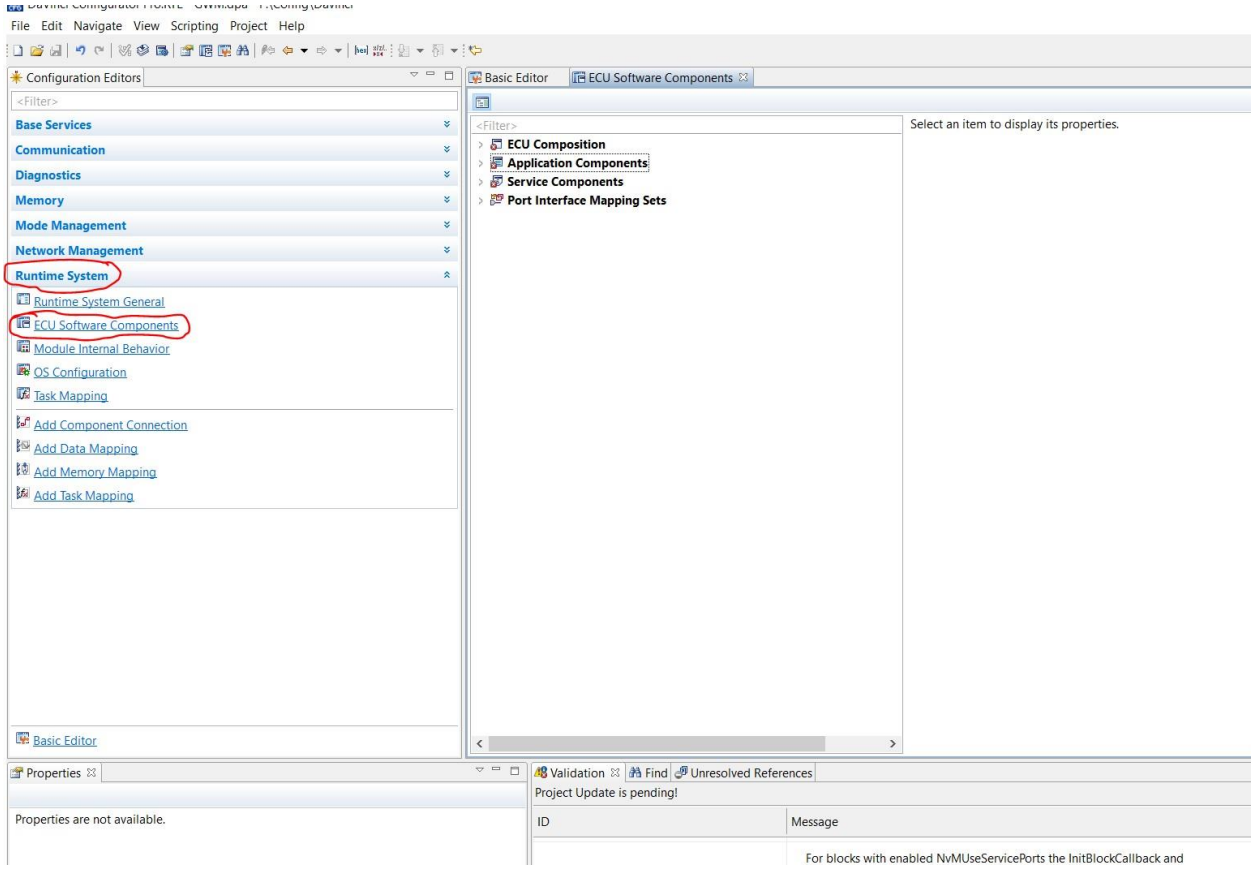
Repeat the same steps for all unconnected DEM ports.

EyeQ CDD Integration

10.4 Integration of Proxy modules

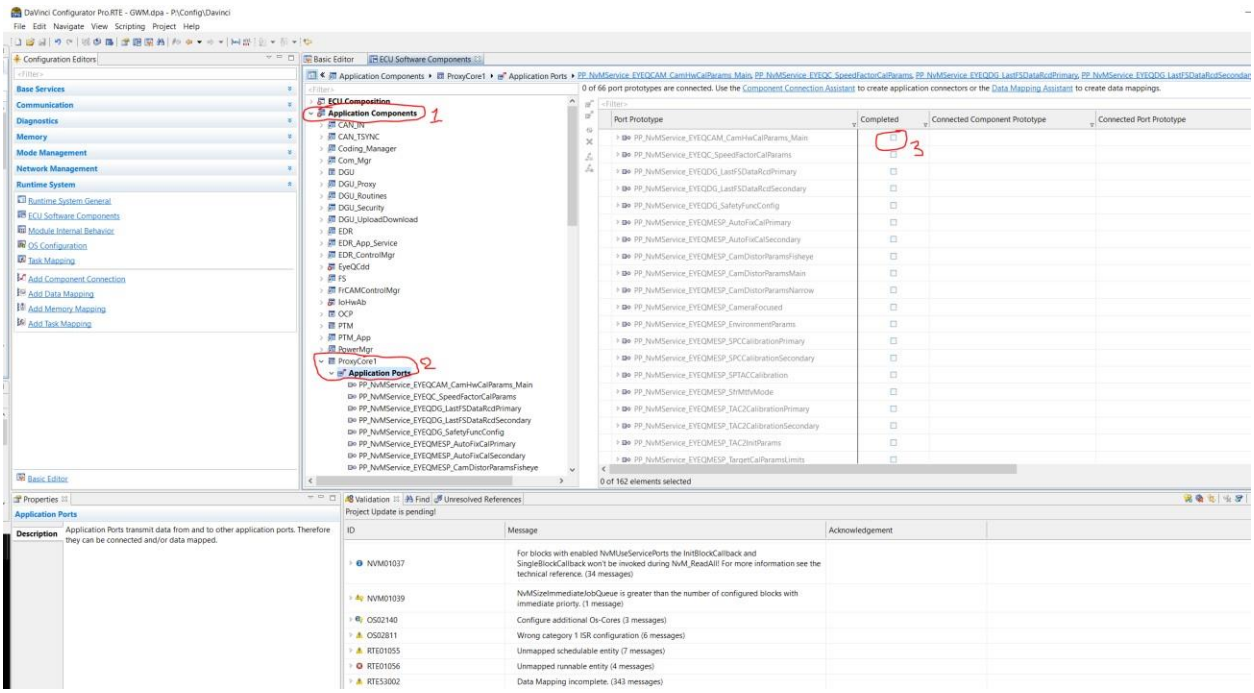
ProxyCore1(or ProxyCore0 depends on architecture) component has application port interfaces and service port interfaces. Application port interfaces need to be connected to destination ports in EyeQCDD component. Service ports need to be connected to NvM and DEM modules(Platform specific).

Click on Runtime system and then click on Ecu Software components.

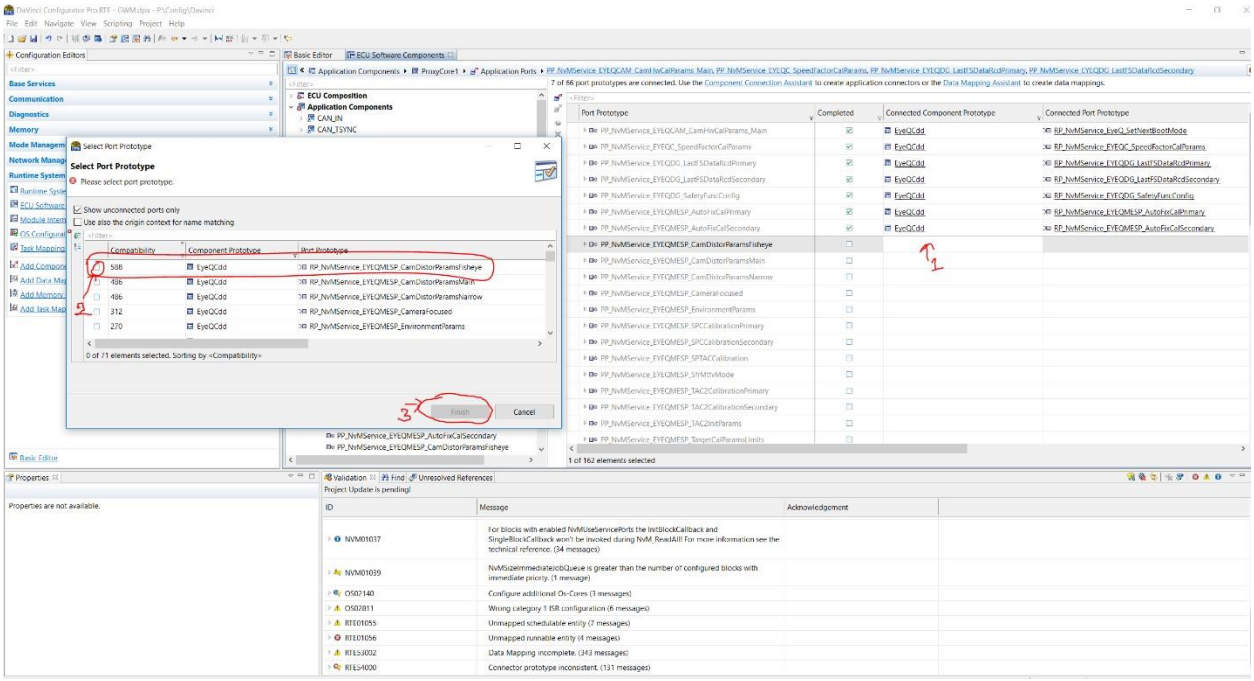


Click on Application components and then click on proxycore1 and the application ports and check for the connection status.

EyeQ CDD Integration

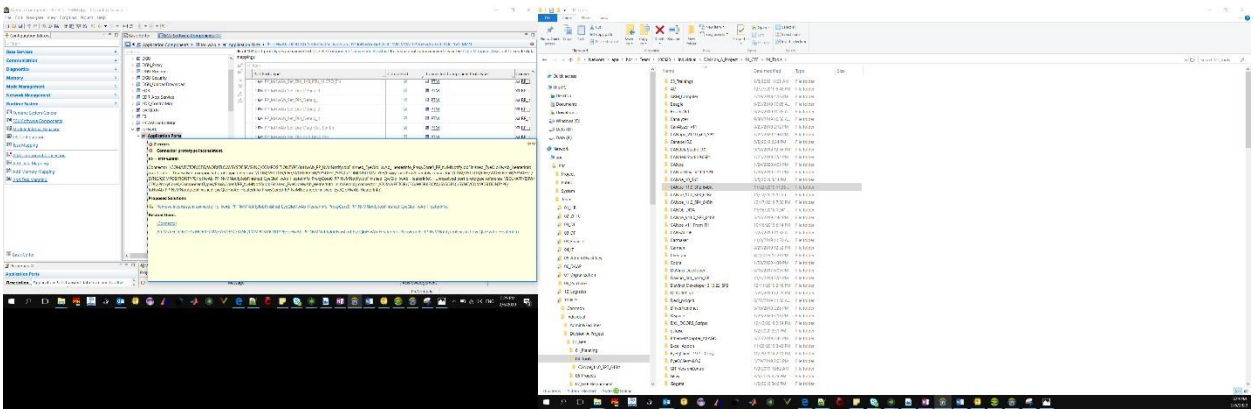


Compare with the port connections provided in core project and do the needful connections. Double click on Connected component prototype and select the port to connect and click on finish.



Remove inconsistent connectors as shown below(click on the link that is highlighted error will be removed.)

EyeQ CDD Integration

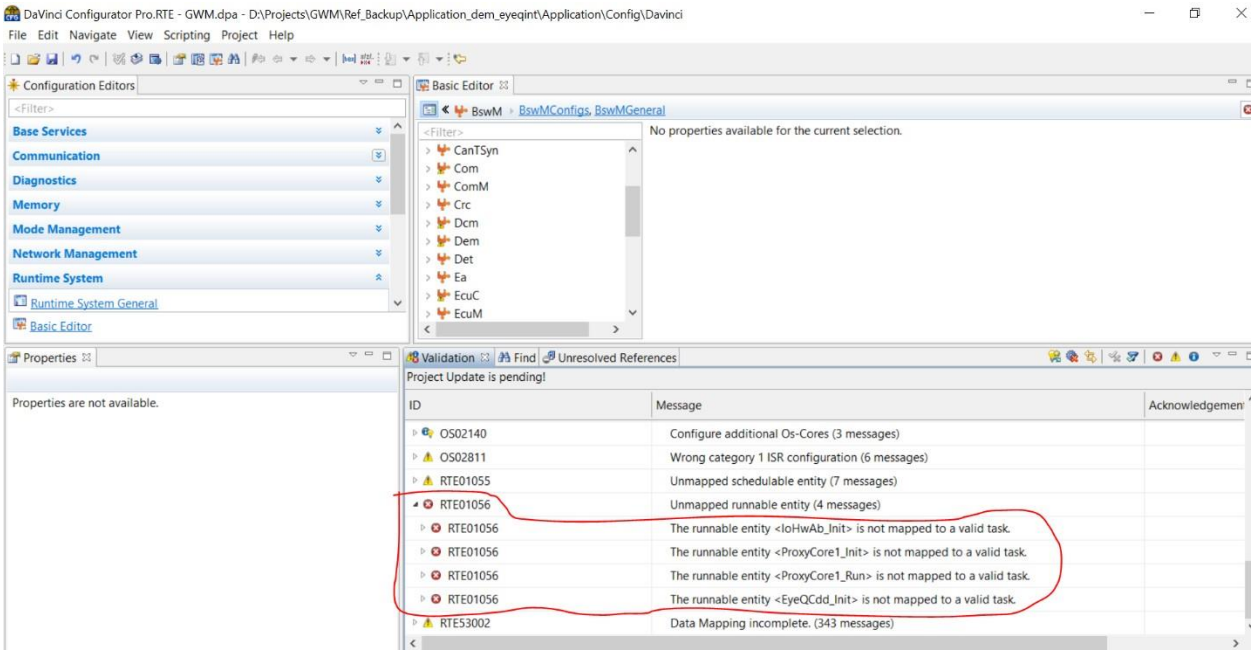


10.5 Os Configuration

In Os configuration the unmapped runnable entities should be mapped to Os tasks.

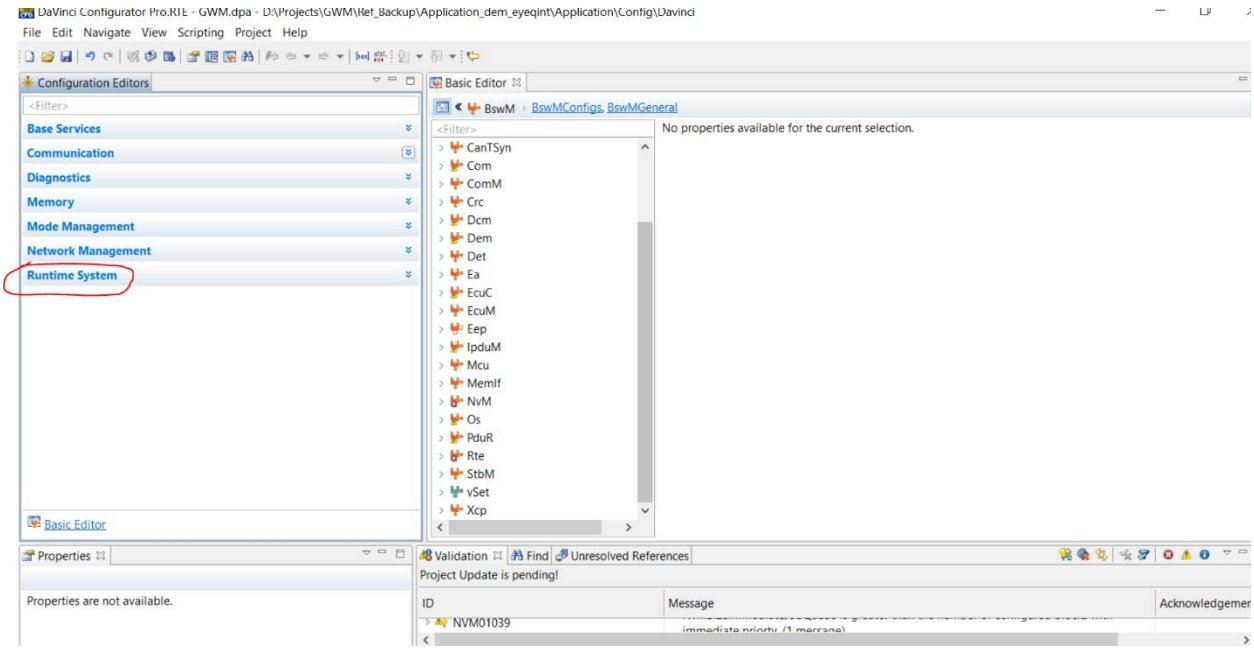
Trusted function and Irq configurations related to EyeQ CDD need to be adapted from Core project. If there are no Os tasks present in Os configuration, create Os tasks and then provide the task reference to events generated by RTE according to component descriptions. Provide an appropriate Os Application reference to the created task.

Check for unmapped runnable`e entities:

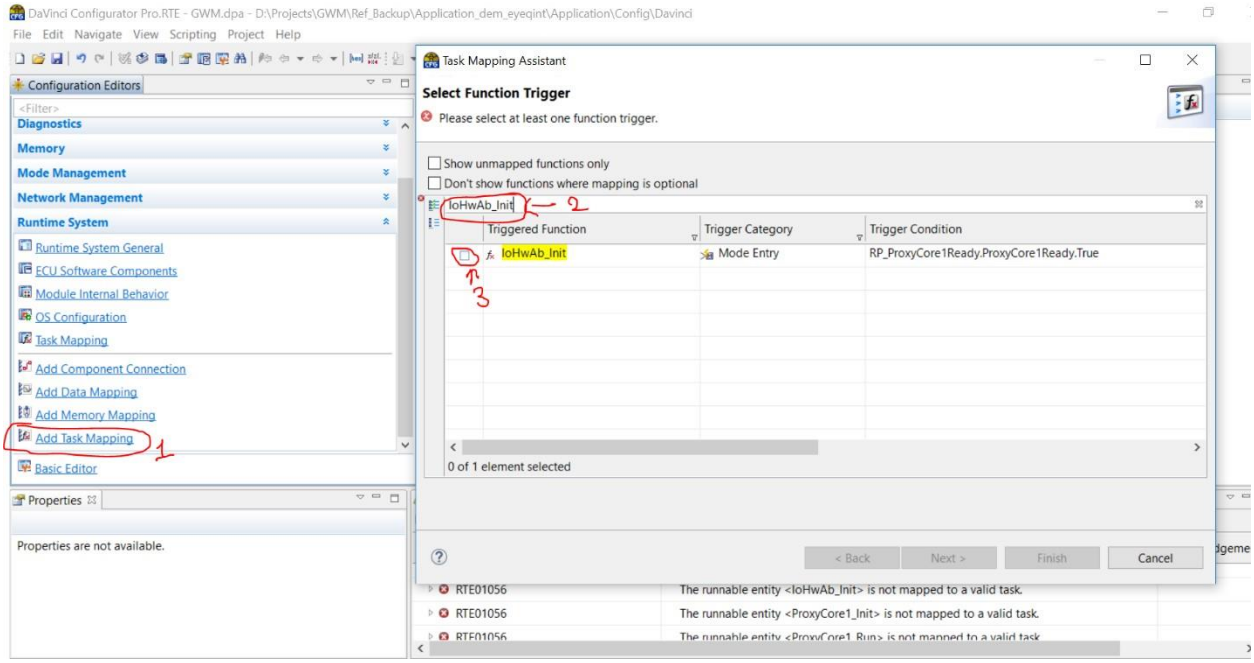


Click on runtime system

EyeQ CDD Integration

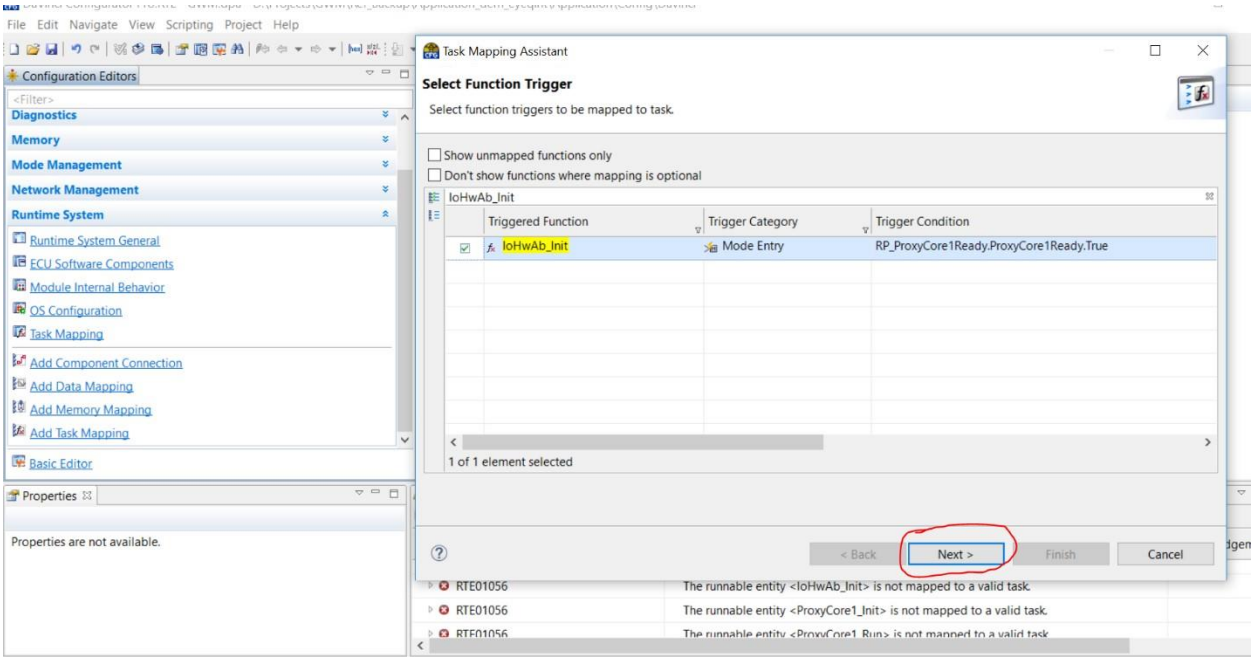


Now select Task Mapping and search for the runnable as shown and click on the check box.

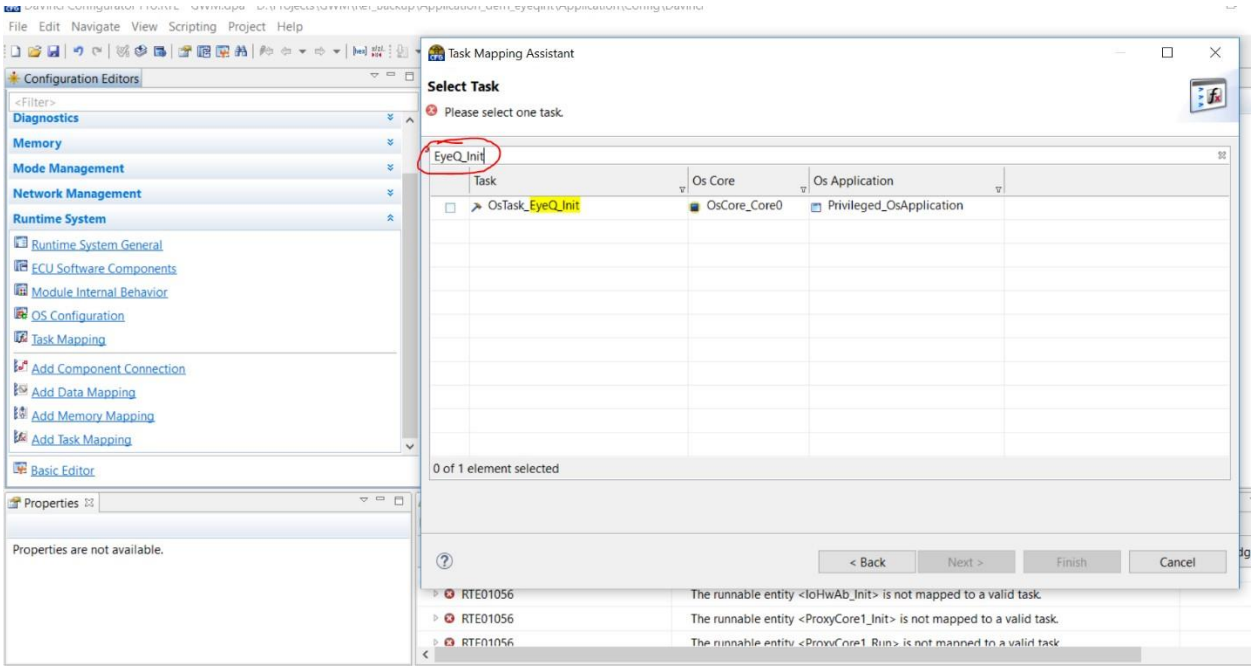


Click on Next:

EyeQ CDD Integration

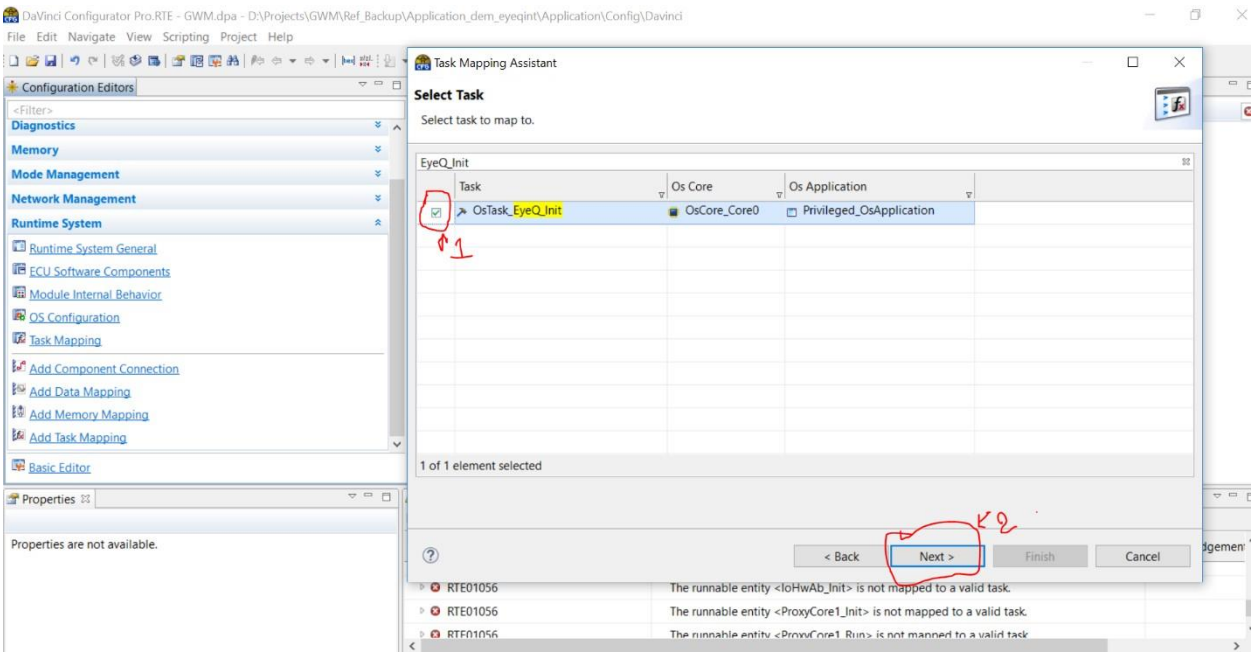


Search for the Os task to map the runnable entity

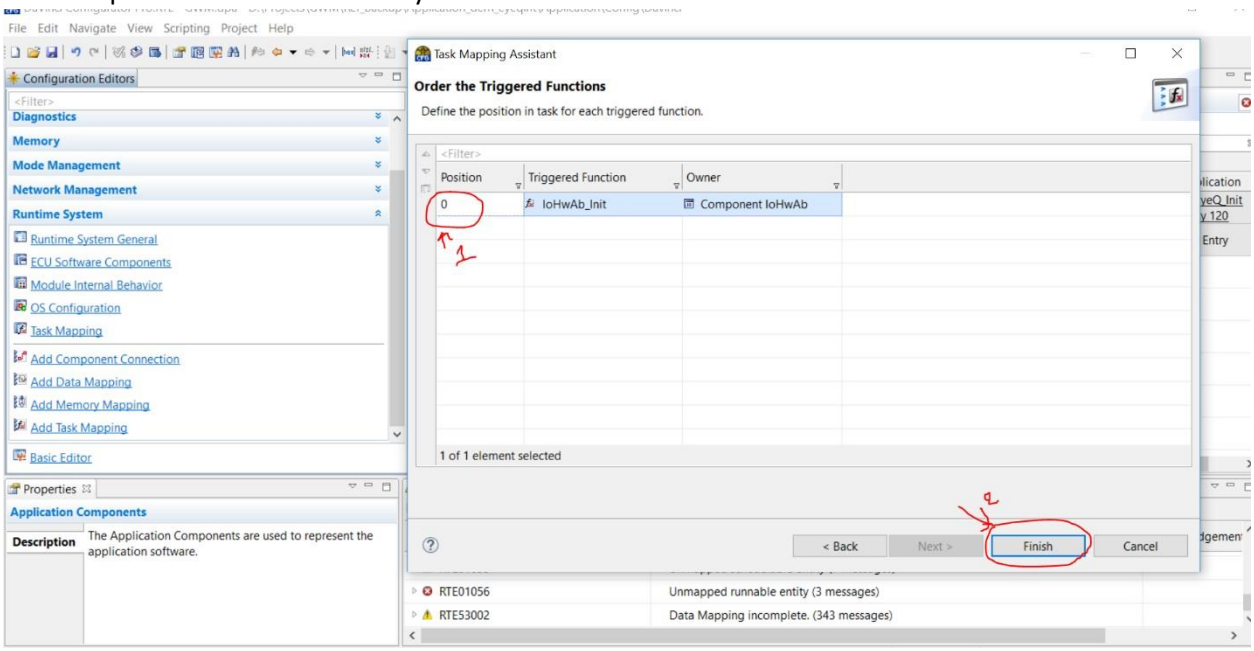


Click on the check box and click next

EyeQ CDD Integration



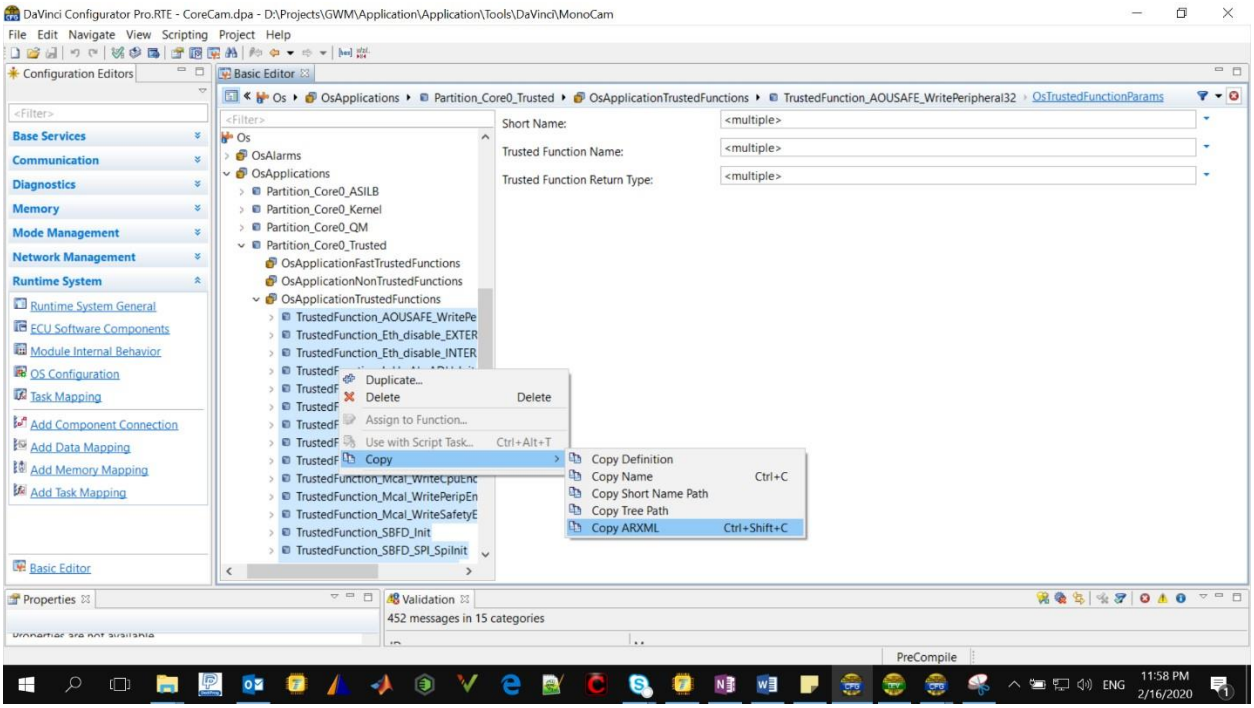
Give the position of runnable entity in the task and click on Finish



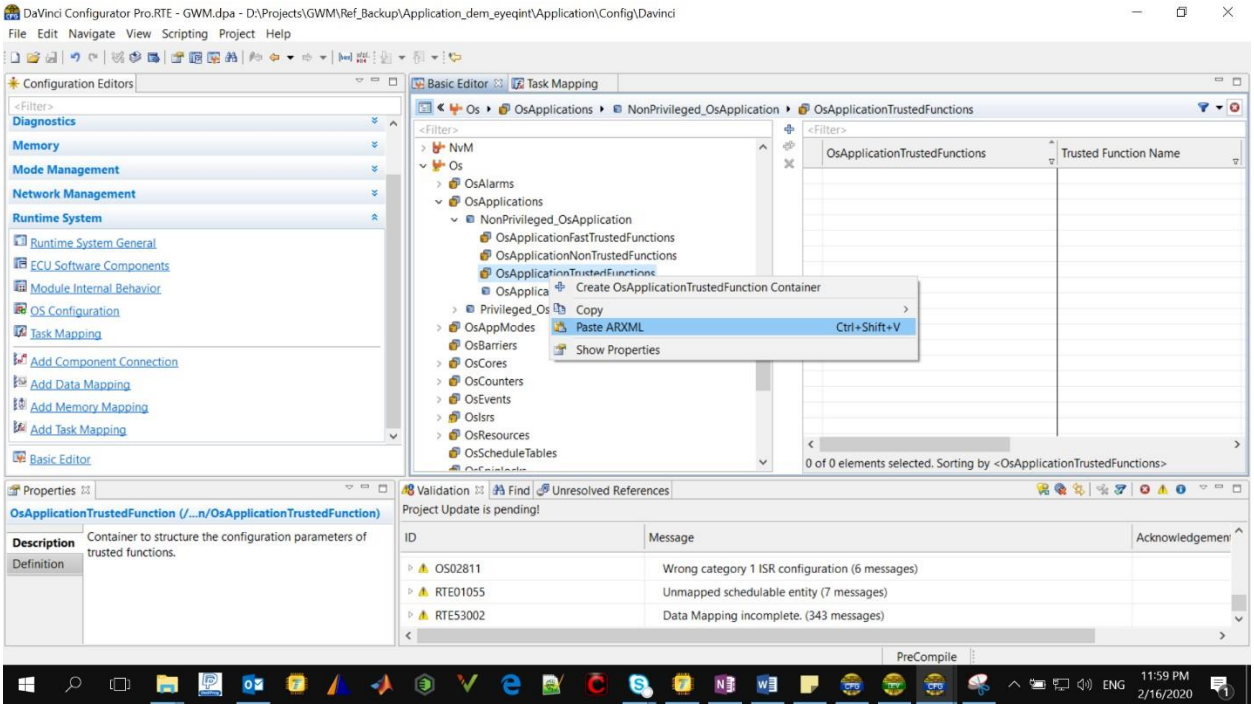
Repeat the same steps for remaining unmapped runnable entities and validate.

Adapt Os trusted function configuration from Core software. Copy the trusted function configuration from core project to application project.

EyeQ CDD Integration



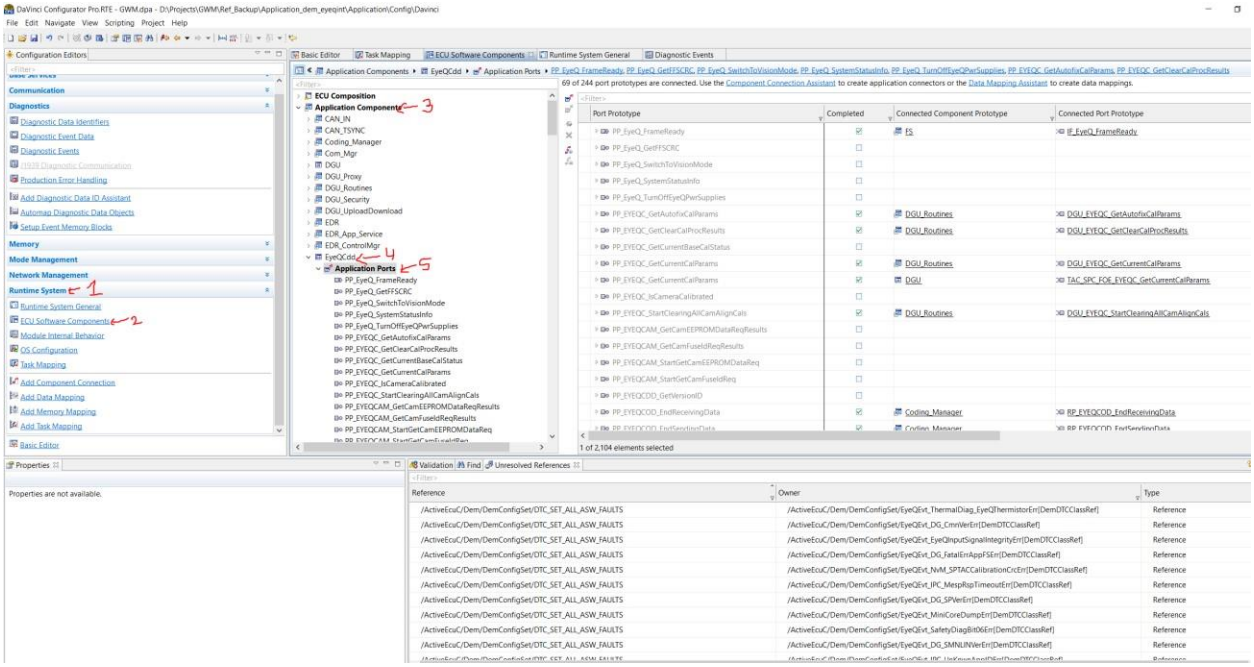
Paste the copied configuration into appropriate Os application in application project.



10.6 CDD Component Integration(Port Connections)

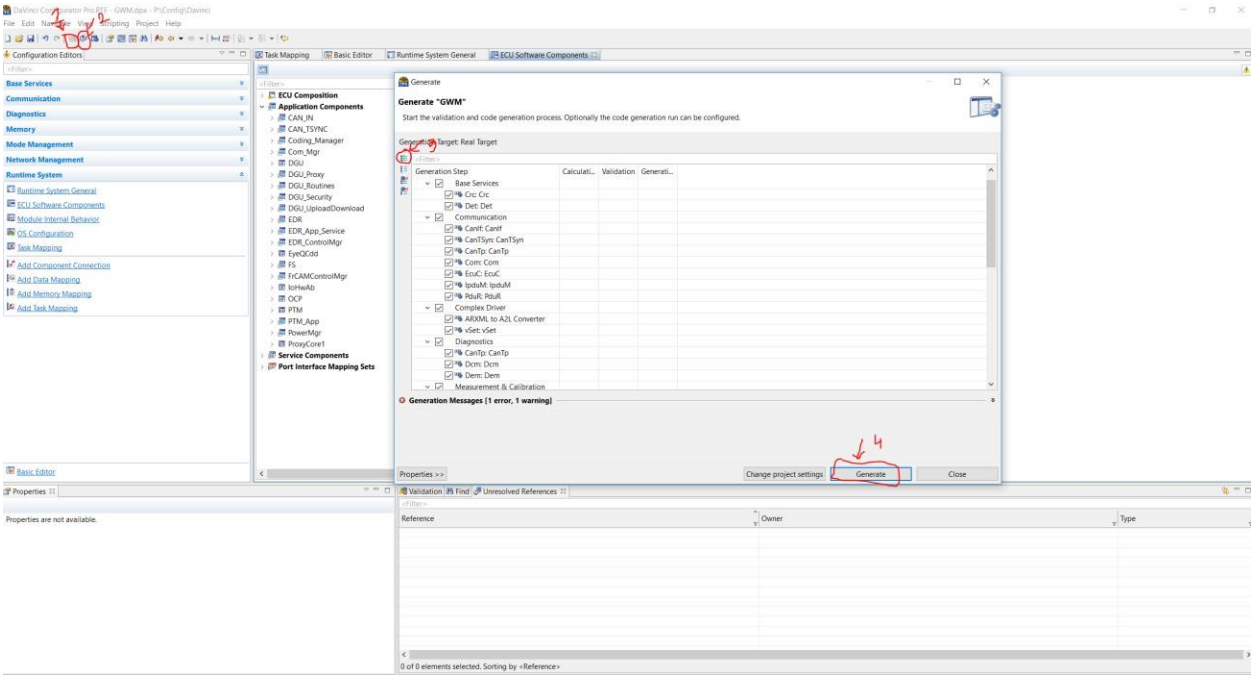
Navigate through Runtime System->ECU Software Components->Application Components->EyeQCDD->Application Ports and do the required connections(Project specific).

EyeQ CDD Integration



10.7 Source Code Generation and Compilation

After resolving all configurator errors click on validate button on the screen. If there are no error's click on Generate button, then select all modules and then generate the files.,



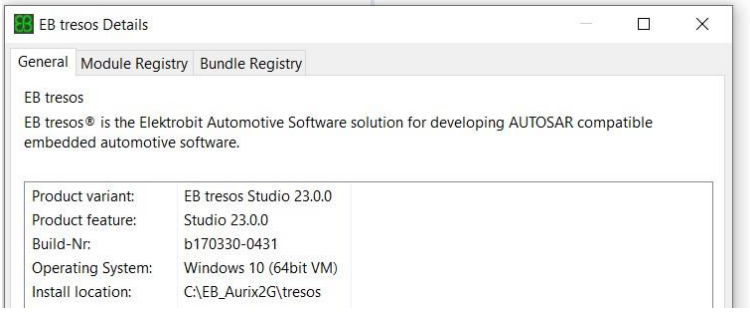
EyeQ CDD Integration

Once the files are generated copy into the destination folder in application software **Source Code**(Project Specific) folder and then go for compilation procedure(Project Specific).

11. MCAL Integration XDM changes and Dynamic code:

Prerequisites:

a) Eb Tresos software should be installed below are details of version



b) MCAL plugins for respective hardware should be installed ,should check plugins from core release and install respective plugins. Details can be seen in core release documents folder with name "Aurix 2G Tresos and MCAL Plugin installation guide".

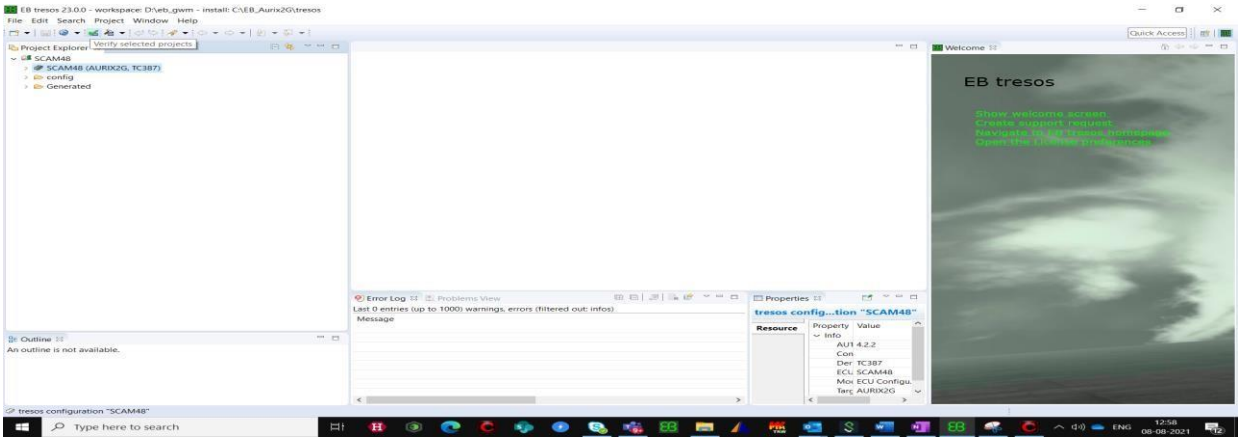
EyeQ CDD Integration

Path: Application\Documents\UserGuide\ Aurix 2G Tresos and MCAL Plugin installation guide core document path.

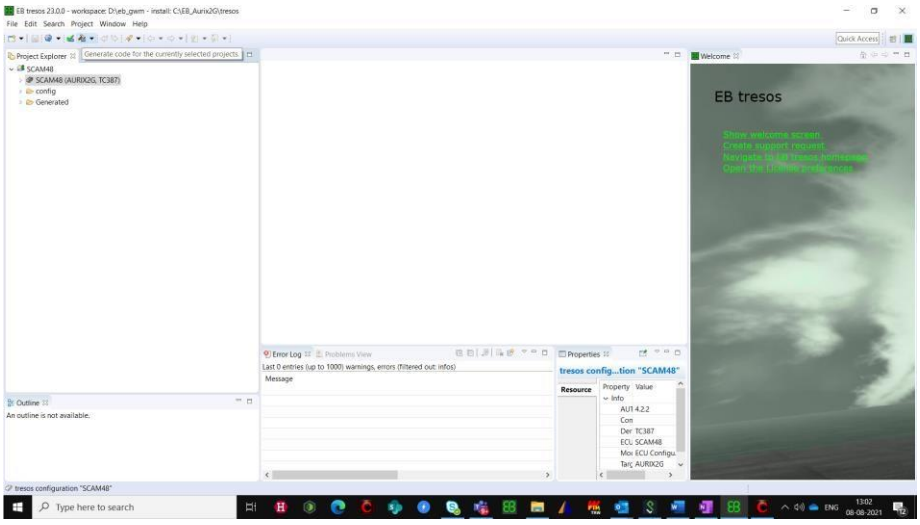
11.1 XDM integration and Dynamic code generation:

When there is new release from core we need to compare MCAL XDM's of core release from core path and application MCAL XDM's. There might be application project specific changes done in MCAL XDM's like Port,DMA,IRQ etc ,this created confusion while comparison. To avoid this we need to compare XDM's of core latest release which we are going to integrate with XDM's of core release which we integrated earlier. This clearly gives new changes from core MCAL. Merge Core XDM's which are changed in new release to application care need to be taken while merging in order not loose application MCAL changes which are project specific.

After XDM changes are done open EB tresos import project and click on verify selection project to validate.

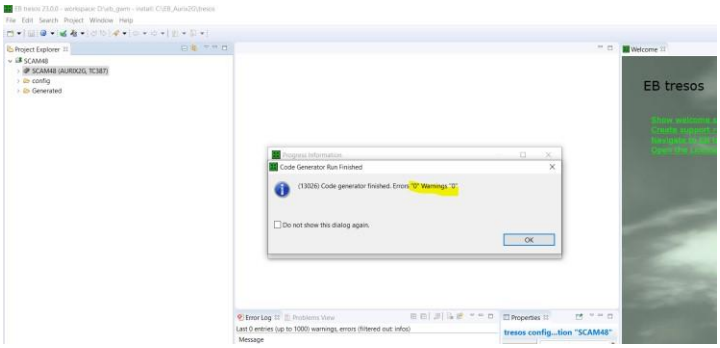


If no errors proceed for code generation as below by clicking on generate code for the current selected projects.

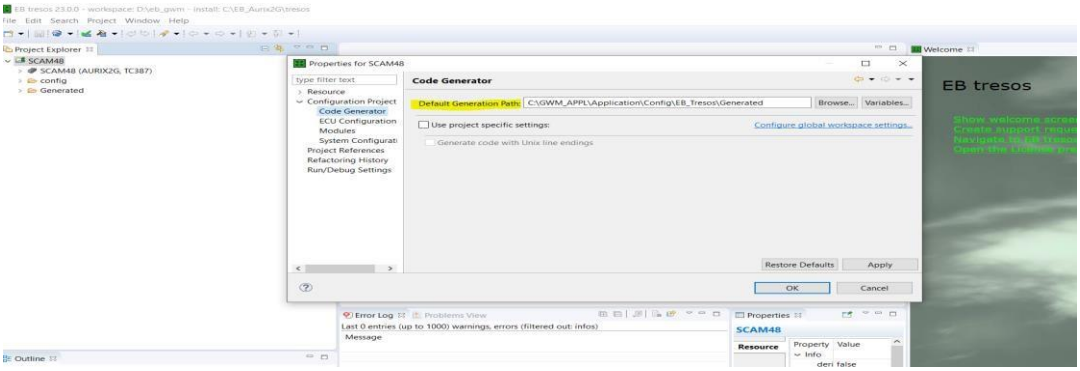


Once code is generated below window pops out.

EyeQ CDD Integration



Code is generated at path given in project options as below.



Now compare code in generated path from EB with application MCAL path ” Application\SourceCode\BSW\McalGendata” and merge the code. Also we need to merge code in “Application\Config\EB_Tresos\generated”

11.2 Things to be taken care while check-in MCAL changes :

1)MCAL XDM path: Application\Config\EB_Tresos\config should be check-in

2)File in path “Application\Config\EB_Tresos\.prefs\ preferences.xdm” contains details of Eb configuration information like modules enabled version, code generation path etc should be check-in.

3)Generated code in:

a)” Application\SourceCode\BSW\McalGendata”

b)

“Application\Config\EB_Tresos\generat

ed” should be check-in.

EyeQ CDD Integration

4)Static MCAL changes which we discussed earlier in Static code merging section should be check-in.

12. NVM default values updates:

In GWM and variants NVM defaults values are generally given in NVM .cfg files. Core do not fallow this, core use FEE they maintain default values in ROM. When there are any change in default values of NVM block parameters and parameters names this should also be updated in NVM .cfg files of application project to which we are integrating.

Below are ways to find differences in default values in core in latest release. Below are files where all NVM blocks with ROM defaults are defined.

Core_Application/SourceCode/CDD/EyeQ_CDD/EyeQ_NVM/src/EyeQ_NvM.c

Application\SourceCode\IoHwAbs\IoHwAb_NvM\src\IoHwAb_NvM.c

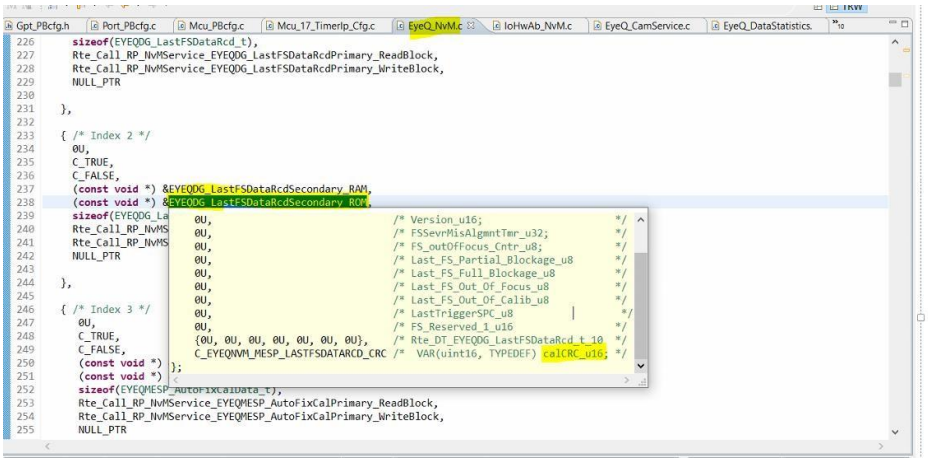
There are two structure each for EyeQ blocks and one for IoHwAB blocks defined as below .In EyeQ_NvM.c and IoHwAb_NvM.c files there is structure member `/* Rom Block location */  const void* RomBlock_p; .`

EyeQ CDD Integration

```
/* Structure to hold static block configuration da
typedef struct
{
    /* NvM TagId */
    uint16_t TagID_u16;
    /* TagId  Error */
     uint8_t IsSafeStateReqdForTagIDErr_u8;
    /* CRC Error */
    uint8_t IsSafeStateReqdForCRCErr_u8;
    /* RamBlock location*/
    const void* RamBlock_p;
    /* Rom Block location */
    const void* RomBlock_p;
    /* Size of the RamBlock */
    uint32_t RamBlockSize_u32;
    /* Function to read NvData from lower layer */
    ReadBlock_t ReadBlock_pf;
    /* Function to write NvData to lower layer */
    WriteBlock_t WriteBlock_pf;
    /* Function to set the DEM event */
    SetCRCErrorEvent_t SetCRCErrorEvent_pf;

}EyeONvM_BlockConfiguration_t;
```

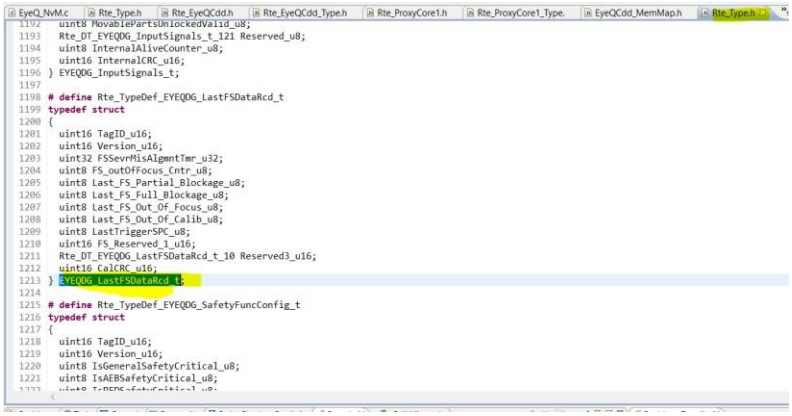
```
/* Structure to hold static block configuration data */
typedef struct
{
    /* NvM TagId */
    uint16_t TagID_u16;
    /* TagId  Error */
     uint8_t IsSafeStateReqdForTagIDErr_u8;
    /* CRC Error */
    uint8_t IsSafeStateReqdForCRCErr_u8;
    /* RamBlock location*/
    const void* RamBlock_p;
    /* Rom Block location */
    const void* RomBlock_p;
    /* Size of the RamBlock */
    uint32_t RamBlockSize_u32;
    /* Function to read NvData from lower layer */
    ReadBlock_t ReadBlock_pf;
    /* Function to write NvData to lower layer */
    WriteBlock_t WriteBlock_pf;
    /* Function to set the DEM event */
    SetCRCErrorEvent_t SetCRCErrorEvent_pf;

}IoHwAbNvM_BlockConfiguration_t;
```

Example to find changes in default values:In below screen shot we can find EYEQDG_LastFSDataRcdSecondary blocks ROM member which defines default values of member of blocks ,the only way to find if any update in core is compare with earlier release of core default value.



Similarly for IoHwAb blocks we need to find any changes in ROM member default values and updating NVM .cfg files in application .We can also find any changes in members of structure elements names by comparing Application\SourceCode\BSW\Generated\MonoCam\Rte_Type.h

EyeQ CDD Integration



13. SConstruct and linker changes

13.1 SConstruct integration:

Gwm and variants use SConstruct as build system unlike core which use make build system. We need to identify changes like file name changes in core,new files added or files deleted and also compiler options etc changes in make of core and adapt them SConstuct of application . Care should be taken and we need to discuss with architect regarding any compiler options changes between core before changing and adapting in application complier options .

Path core Make files: Core_Application/Build/Make

Path GWM application SConstuct file: Application\Build\ SConstruct

13.2 Linker changes:

For core related linker changes we need to fallow document

EyeQ CDD Integration

DAS_Core_Software_EyeQ_CDD_IntegrationGuide and compare linker files of core with previous release at core_Application\Build\Make like CoreApplicationLinker.lsl, tc3xx, traptabx etc.

Before adapting linker changes in application related linker files detail discussion with architect should be done and impact of changes if any on application should be analyzed.

For placement of core buffers and file related in particular section we need to strictly fallow DAS_Core_Software_EyeQ_CDD_IntegrationGuide document in core release.

Path: Application\Documents\IntegrationGuide\CDD\
DAS_Core_Software_EyeQ_CDD_IntegrationGuide

14.General Issues, debugging steps and validating EyeQ integration.

In this chapter we will discuss about validating after EyeQ integration, general issues and debugging steps.

14.1 Validating EyeQ integration:

After EyeQ integration build is successful we need to flash respective compatible ME files using MFT, flash vision file of NVM(in GWM we will move EyeQ into vision by NVM vision file ,we can also move EyeQ into vision using PTM and debugger) .

Below are variables to be checked :

1) EYEQDG_CurrentMainState_u8: This variable gives EyeQ states if it is in vision or some other state. Below are some of states of EyeQ. If everything is as expected THEN EYEQDG_CurrentMainState_u8 should be 2 that is vision .

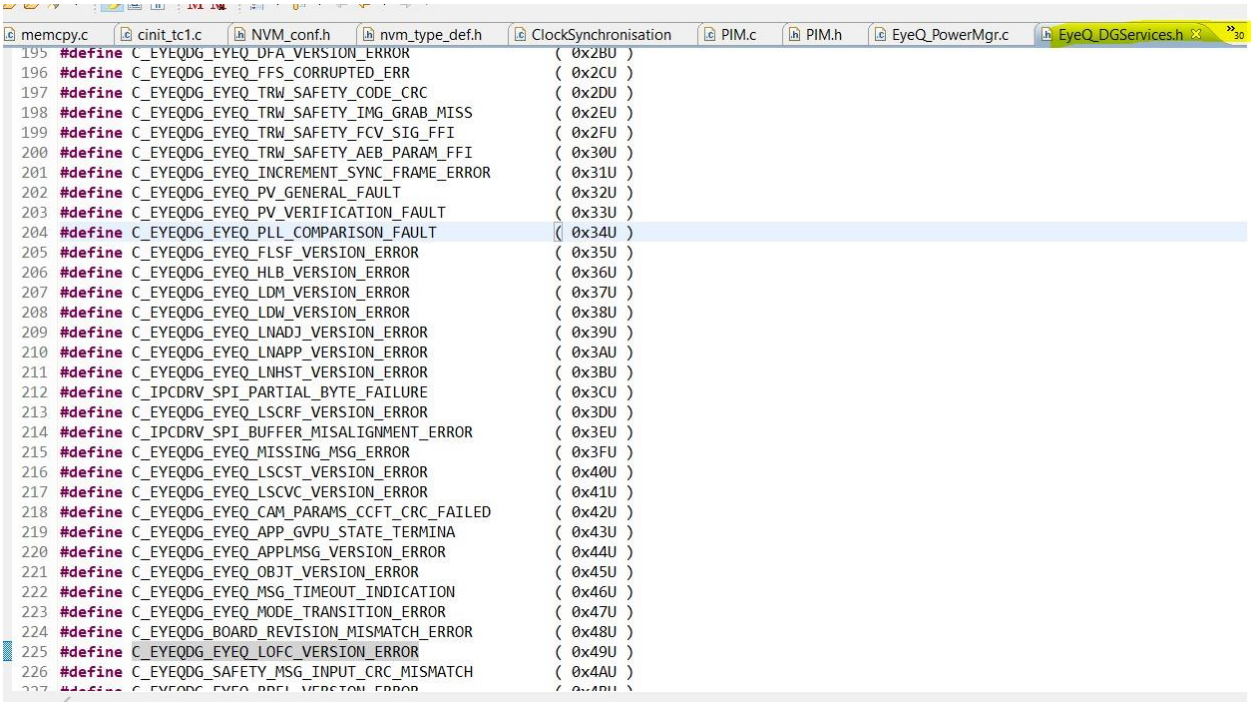
EyeQ CDD Integration

- #define C_EYEQBS_EYEQ_MAIN_STATE_UNKNOWN (0x00U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_PENDING (0x01U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_BOOT (0x03U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_PENDING_DV (0x85U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_RUNNING_APP_BURN (0xAAU)
- #define C_EYEQBS_EYEQ_MAIN_STATE_PENDING_VISION (0x92U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_VISION (0x02U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_PENDING_SPC (0x82U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_RUNNING_SPC (0x22U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_PENDING_SPTAC (0x81U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_RUNNING_SPTAC (0x21U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_PENDING_TAC (0x80U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_RUNNING_TAC (0x20U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_PENDING_TAC2 (0xABU)
- #define C_EYEQBS_EYEQ_MAIN_STATE_RUNNING_TAC2 (0xACU)
- #define C_EYEQBS_EYEQ_MAIN_STATE_PENDING_SFR (0x83U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_RUNNING_SFR (0xB0U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_PENDING_SP (0x89U)
- #define C_EYEQBS_EYEQ_MAIN_STATE_RUNNING_SP (0xAEU)
- #define C_EYEQBS_EYEQ_MAIN_STATE_RUNNING_STEREO (0xAFU) • #define C_EYEQBS_EYEQ_STATE_UNAVAILBLE (0xFFU)

2) EYEQP_CurrentErrorStatus_u32 :This variable value should be '0' if there are no errors reported in EyeQ. If any errors is logged to know the cause we need to refer file
EyeQ_DGServices.h at

EyeQ CDD Integration

Application/Core_Comp/CDD/EyeQ_CDD/EyeQ_DG/include/EyeQ_DGServices.h this file gives detail of error which is logged as below.



3) EyeQLastResetSource_u8: This variable gives last error source which has reset EyeQ . We can refer EyeQDGServices.h to find details. Ideally value of variable should be 0XFF.

4) EyeQ_RestartAttemptCnt_u16: This variable gives number of resets attempts by EyeQ if any error is reported and if reset strategy is available for respective error. For reset strategy refer

Application\Core_Comp\CDD\EyeQ_CDD\EyeQ_MGR\src\EyeQ_PowerMgr.c which has reset table with details. Ideally the value of this varible should be 0.

5) EYEQBS_Versions_s: This structure will give information of ME flashed, boot version of ME, MEST version etc as below example.

EyeQ CDD Integration

```
EYEQBS_Versions_s = (
  • VerRet_u32 = 0x0,
  • VersLen_u32 = 0x5,
  • BootMgrVer_u32 = 0x08070200,
  • BootLdrVer_u32 = 0x08070200,
  • MESTVer_u32 = 0x140A0502,
  • ProtocolVer_u32 = 0x140A0C08,
  • EtcVer_u32 = 0x00050002)
```

6) EYEQDG_RxMsgTracker_s: This structure gives details of various messages received by EyeQ CDD using IPC communication this helps in debugging any timeout issue etc.

7) EYEQP_EyeQResetTracker_as: This structure gives details of any resets logged by EyeQ .To refer particular error we need to refer reset table(`ResetTable_s`) in

Application\Core_Comp\CDD\EyeQ_CDD\EyeQ_MGR\src\EyeQ_PowerMgr.c

```
284
285 static const EyeQResetTable_t ResetTable_s[C_EYEQP_NUM_OF_EYEQ_RESET_EVENTS] =
286 {
287   /*-------------------------------------------------------------------------------------------------------------------------------
288                                          /* FaultSystemState_u8 = C_EYEQSYSS_SYSTEM_SAFE_STATE */
289   /*-------------------------------------------------------------------------------------------------------------------------------
290                                         /*    ErrEvent_u8                                , ErrClass_u8       , ErrLimit_u16
291                                         /*----*Configure DG message based from here*----*/
292   /* 0:  C_EYEQDG_EYEQ_CONAR_VERSION_ERROR   */#if ( C_SUPPORT_OK == C_EYEQ_DG_CONAR_SUPPORT )
293                                           { C_EYEQDG_EYEQ_CONAR_VERSION_ERROR           , C_EYEQDG_RESET_OK , C_EYEQ_5_RESTART_A
294                                         #else
295                                           { C_EYEQDG_EYEQ_CONAR_VERSION_ERROR           , C_EYEQDG_RESET_NOK, C_EYEQ_0_RESTART_A
296                                         #endif
297   /* 1:  C_EYEQDG_EYEQ_DFA_VERSION_ERROR    */ #if ( C_SUPPORT_OK == C_EYEQ_DG_DFA_SUPPORT )
298                                           { C_EYEQDG_EYEQ_DFA_VERSION_ERROR            , C_EYEQDG_RESET_OK , C_EYEQ_5_RESTART_
299                                         #else
300                                           { C_EYEQDG_EYEQ_DFA_VERSION_ERROR            , C_EYEQDG_RESET_NOK, C_EYEQ_0_RESTART_
301                                         #endif
```

8)We need to verify if NVM integrity is proper and valid with help of structure NvM_BlockMngmtArea_at.NvRamErrorStatus_u8 should not be 3,5,6,7 or 8.

```
NvM_BlockMngmtArea_at = (
  [0] = (NvDataIndex_t = 0, NvRamErrorStatus_u8 = 0, NvRamAttributes_u8 = 0),
  [1] = (NvDataIndex_t = 0, NvRamErrorStatus_u8 = 1, NvRamAttributes_u8 = 1),
  [2] = (NvDataIndex_t = 0, NvRamErrorStatus_u8 = 0, NvRamAttributes_u8 = 1),
  [3] = (NvDataIndex_t = 0, NvRamErrorStatus_u8 = 0, NvRamAttributes_u8 = 1),
  [4] = (NvDataIndex_t = 0, NvRamErrorStatus_u8 = 0, NvRamAttributes_u8 = 1),
  [5] = (NvDataIndex_t = 0, NvRamErrorStatus_u8 = 0, NvRamAttributes_u8 = 1),
  [6] = (NvDataIndex_t = 0, NvRamErrorStatus_u8 = 0, NvRamAttributes_u8 = 1),
  [7] = (NvDataIndex_t = 0, NvRamErrorStatus_u8 = 0, NvRamAttributes_u8 = 1),
  [8] = (NvDataIndex_t = 0, NvRamErrorStatus_u8 = 0, NvRamAttributes_u8 = 1),
  [9] = (NvDataIndex_t = 0, NvRamErrorStatus_u8 = 0, NvRamAttributes_u8 = 1),
  [10] = (NvDataIndex_t = 0, NvRamErrorStatus_u8 = 0, NvRamAttributes_u8 = 1),
  [11] = (NvDataIndex_t = 0, NvRamErrorStatus_u8 = 0, NvRamAttributes_u8 = 1),
```

```
#define NVM_REQ_OK                  (0U)
#define NVM_REQ_NOT_OK              (1U)
#define NVM_REQ_PENDING             (2U)
#define NVM_REQ_INTEGRITY_FAILED    (3U)
#define NVM_REQ_BLOCK_SKIPPED       (4U)
#define NVM_REQ_NV_INVALIDATED      (5U)
#define NVM_REQ_CANCELED            (6U)
#define NVM_REQ_REDUNDANCY_FAILED   (7U)
#define NVM_REQ_RESTORED_FROM_ROM   (8U)
```

9) SysTimer: System timer should be running.

10)We need to verify if diagnostic commands are working by using giving commands in cano and also should verify CAN messages log in canoe.

11)If gliwa is enabled We need to t verify any EyeQ task or other task are overloading .

14.2 General Issues, debugging steps:

General issue in Eyeq integration are as fallows along with debugging steps.

1)Safety reset issues .

Safety related resets we need to refer structure AoUSafe_LastMcuResetReason_a, AoUSafe_SafetyModulesTestStatus for source of smu alarm which triggered by which we can debug issue.

2)Bootloader and application shared variables not at same location. To find this issue we need to flash faulty application see if this is reported at bootloader

3)Application user configuration file not properly configures which is compatible with ME. ME supported feature are present Meio.cfg file which is discussed earlier in ME related chapter. Core_Comp\AppConfig\Config_EyeQCdd\include\EyeQ_DGUserServices.h

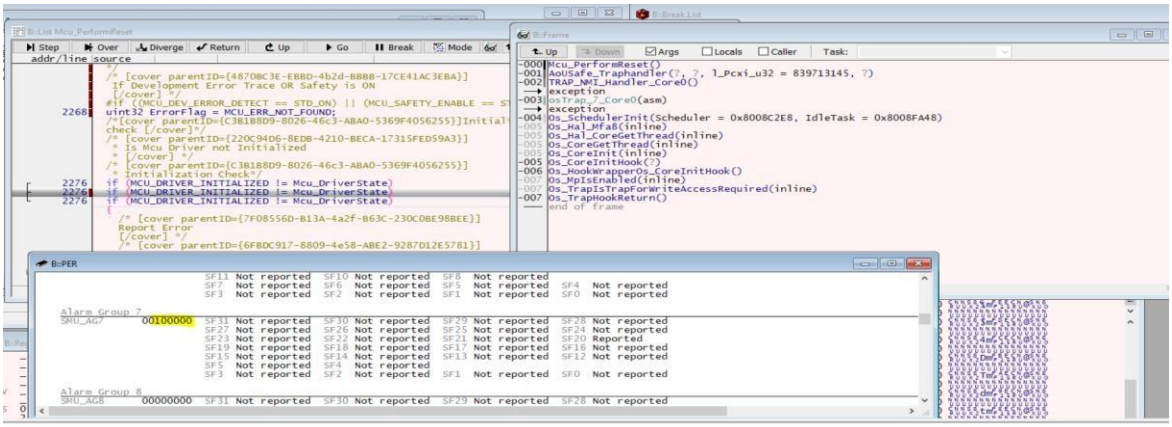
EyeQ CDD Integration

We see #define C_EYEQDG_EYEQ_MSG_TIMEOUT (0xD3U) error if any mismatch.

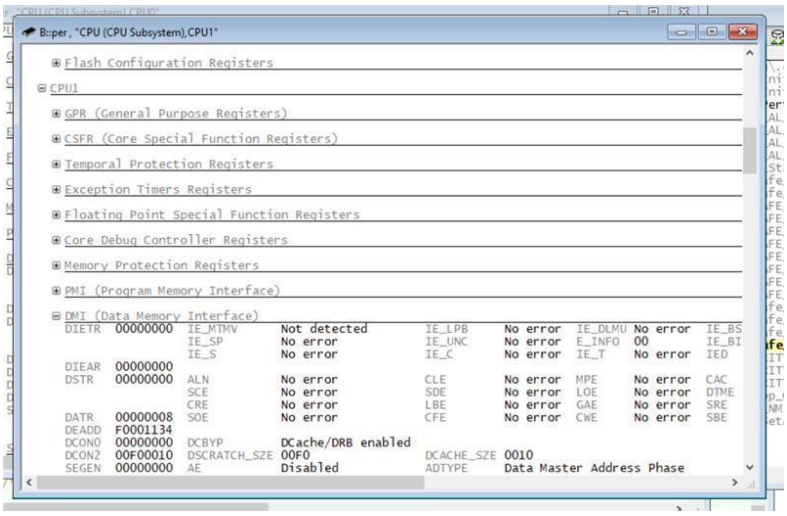
Examples of some of issues faced in G35 jupiter integration which gives details how to debug:

ISSUE 1) STM 1 Access Failure

a) Tried to find stack frame and SMU alarm's, its access related alarm

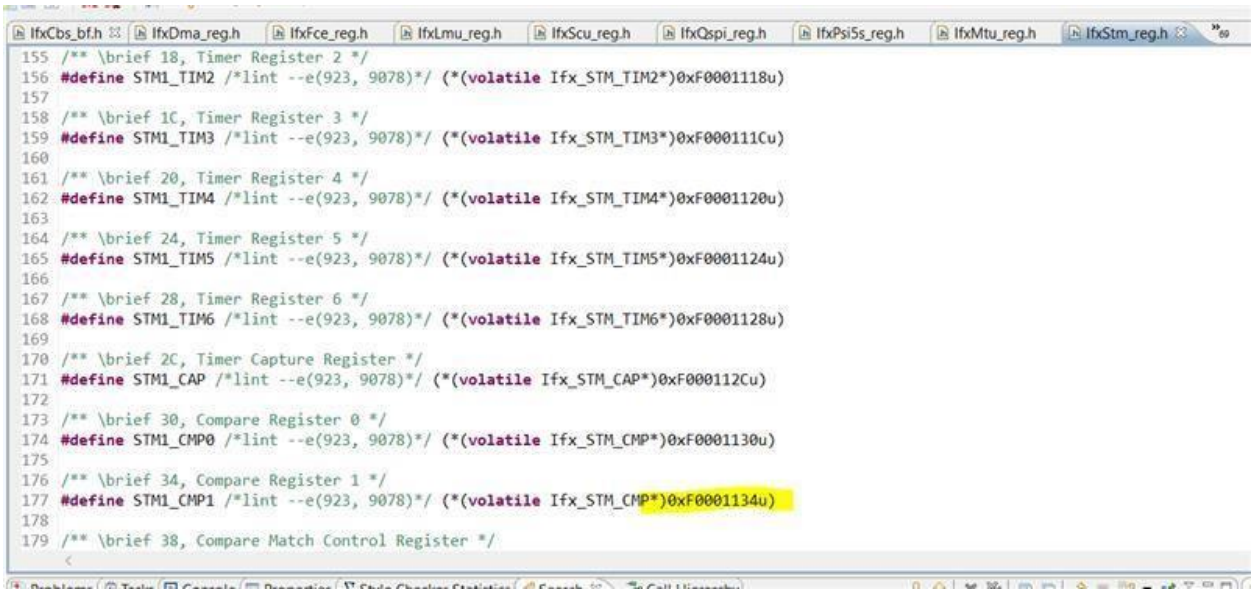


b)Identified source of reset from DEADD register its 0XF0001134 as below

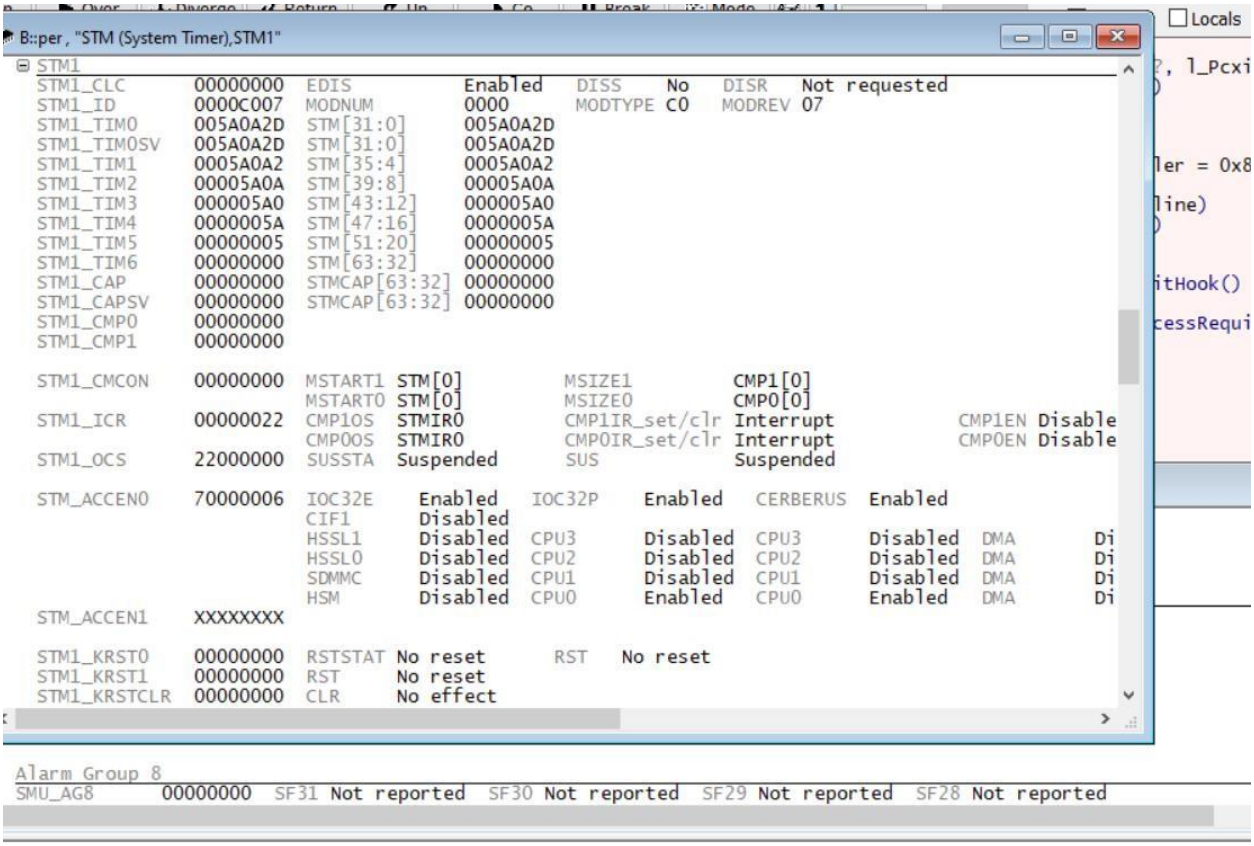


c)From MCAL register address 0XF0001134 belong to STM1 as below

EyeQ CDD Integration

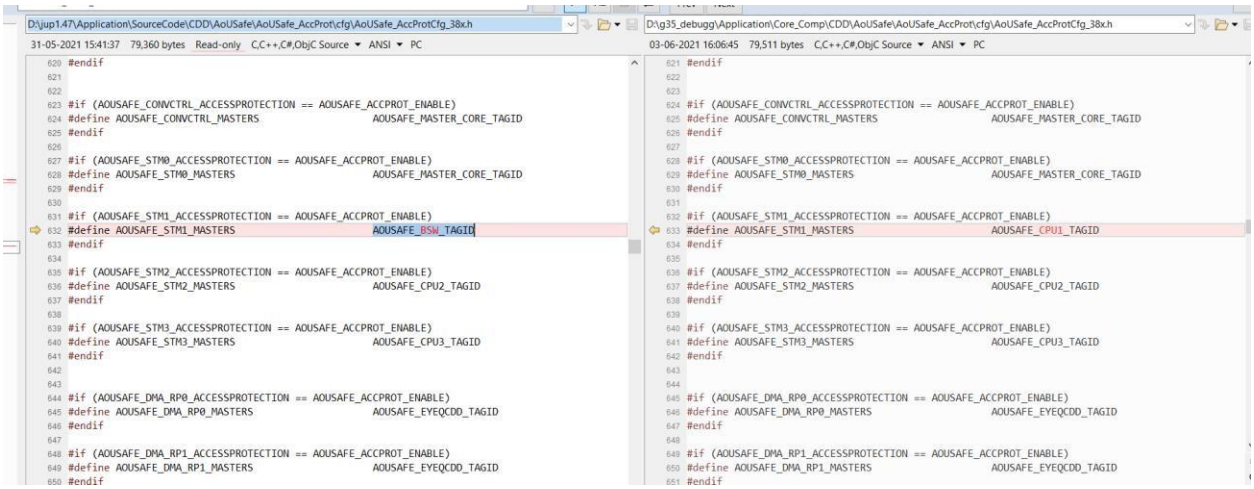


d)In below screen shot STM1_ACCENO is having access to core0 ,but expected is in core 1.

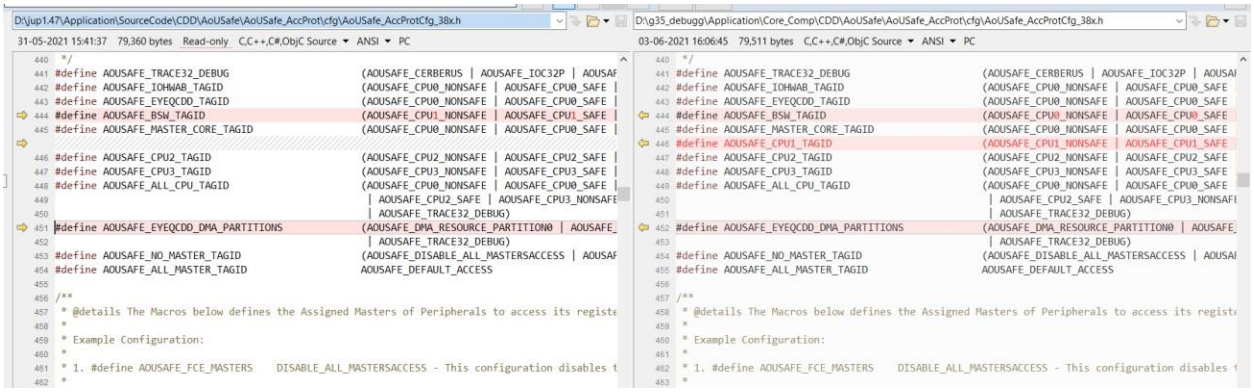


EyeQ CDD Integration

e) Root cause as core as BSW in core1 its correct from them but G35 BSW is in core0 ,so for G35 we need to give CPU1 access by using separate MACRO, as BSW_TAGIG is 0.

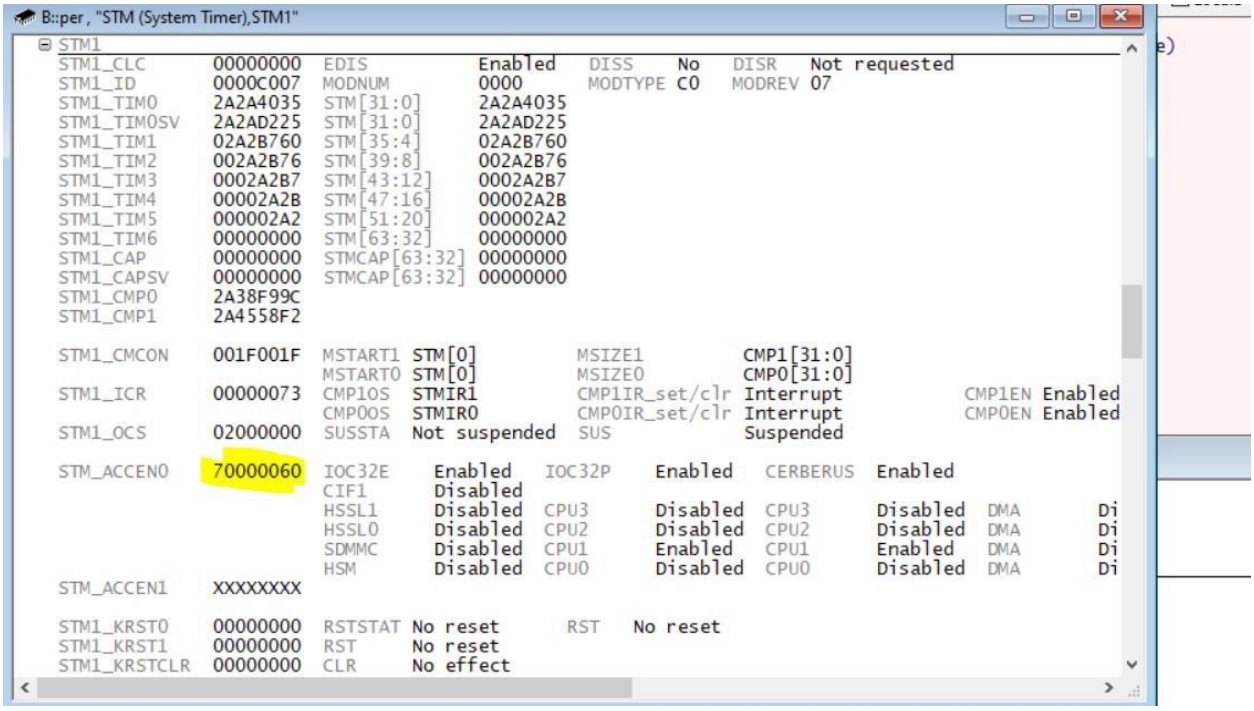


f) Separate macro is added for CPU1 and provided STM access as above with below macro.

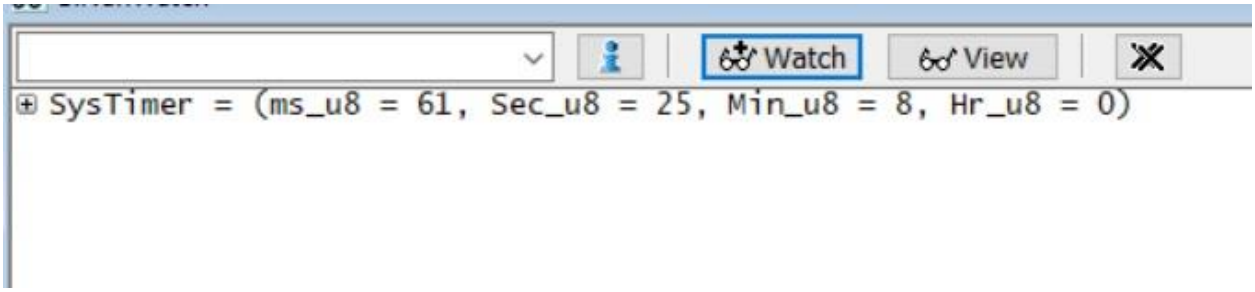


g)STM1 as STM_ACCEN0 with access given to CPU1 now as below

EyeQ CDD Integration



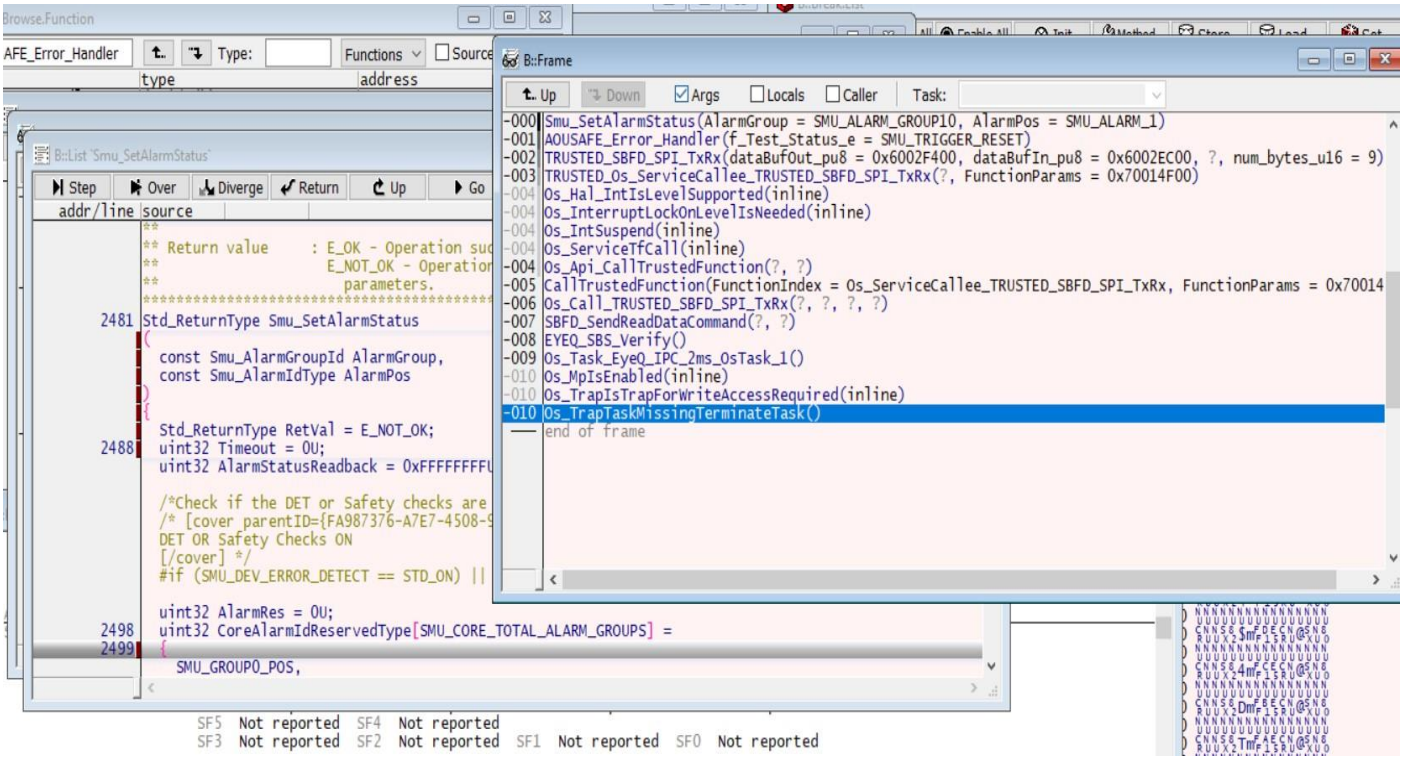
h) Issue is resolved as software is running.



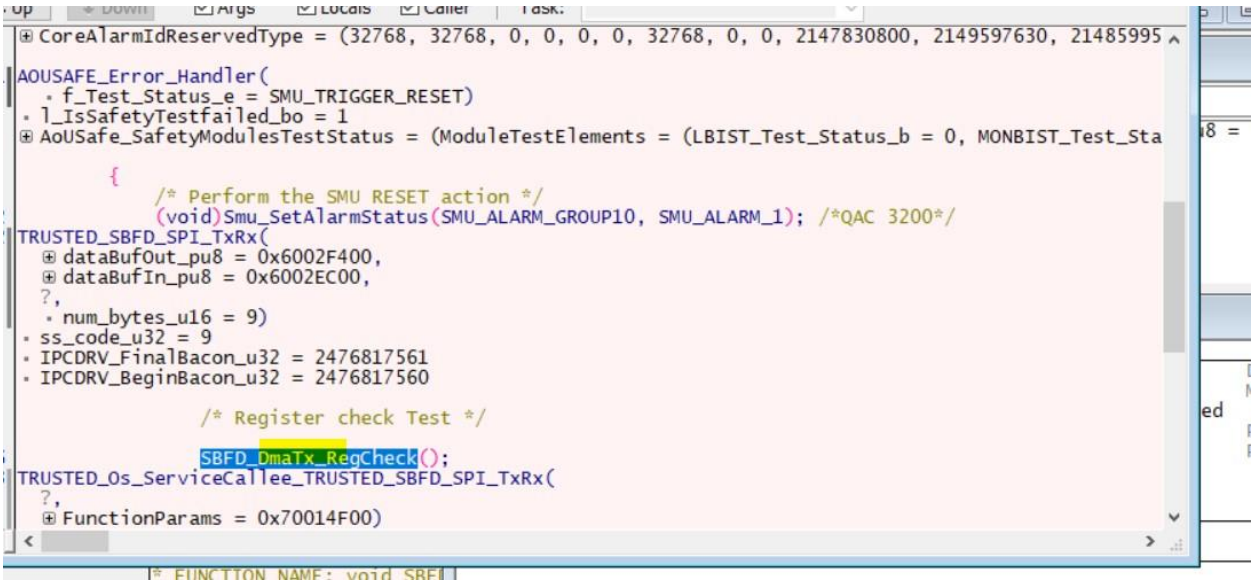
ISSUE 2)DMA related additional access problem as CDD is not integrated as per project Decision.

a) Could see safety alarm set from SBFD_SPI_TX_RX

EyeQ CDD Integration

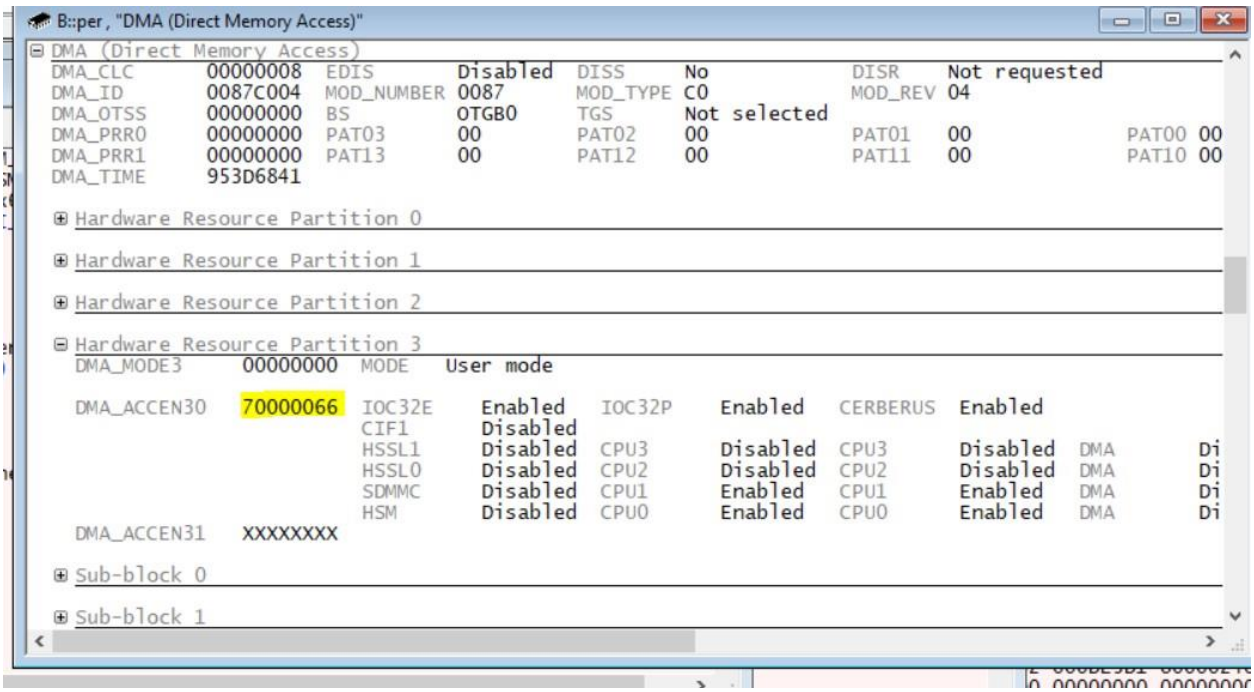


b) IN SDFD_SPI_TX_RX source of safety issue is DMA_TX_REGCHECK function

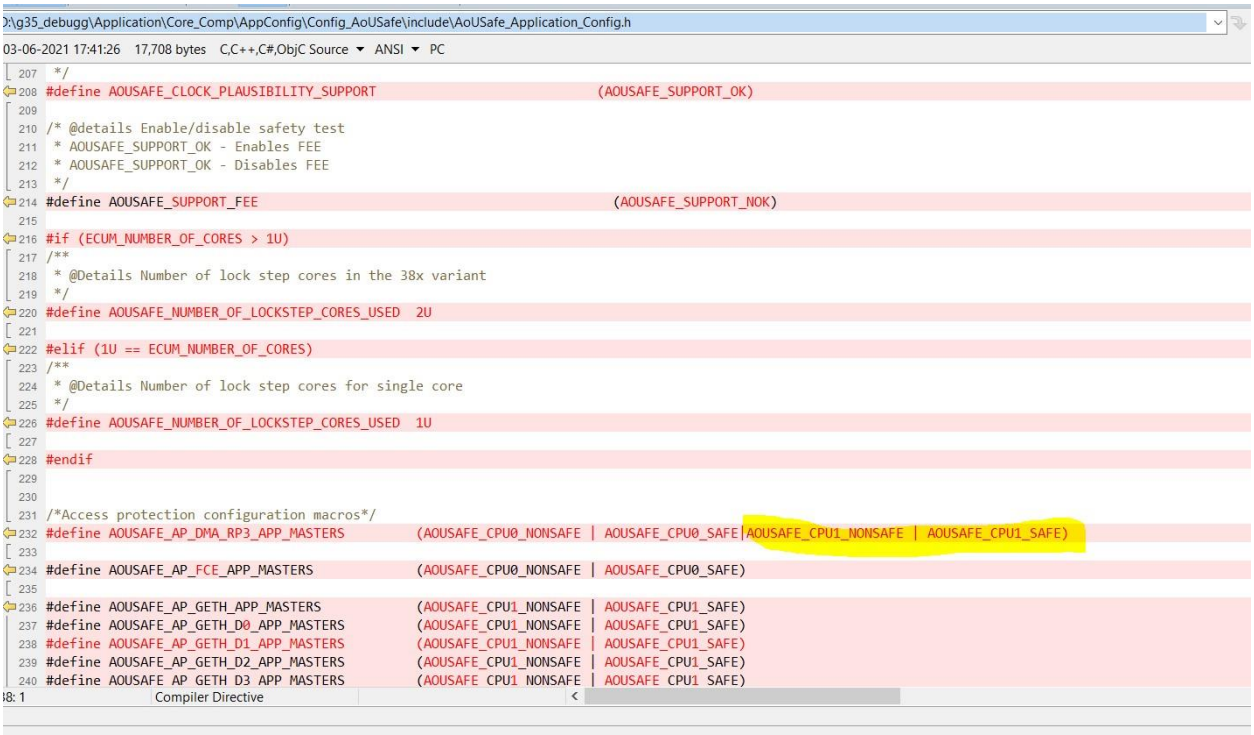


c)Could see DMA belong to partition 3 with access to both CPU0,CPU1.

EyeQ CDD Integration



d)root cause , access in below file with access to CPU1,CPU0.



e)After going through DMA reg check function there its giving access to only 1 core need to remove extra access as below. Issue introduced as we did not integrate CDD as per project decision. In latest code of CDD this issue will not be there as REG check of DMA is present .

EyeQ CDD Integration

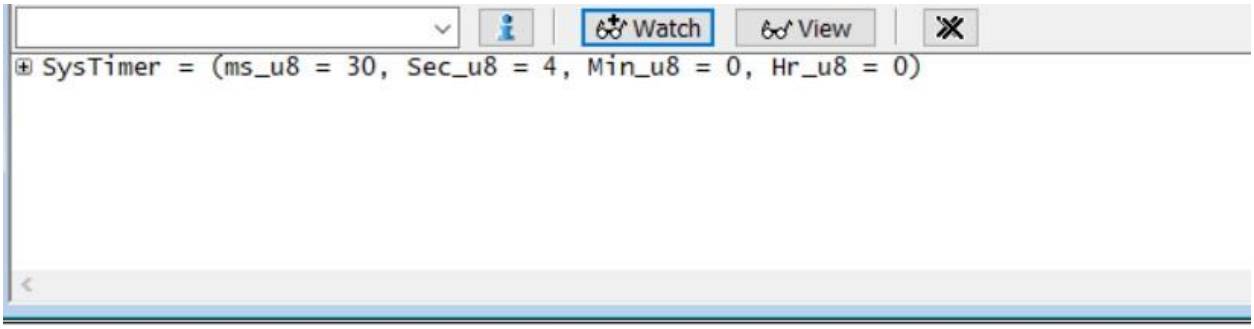
```
...35_debugg\Application\Core_Comp\AppConfig\Config_AoUSafe\include\AoUSafe_Application_Config.h
06-2021 17:58:41   17,667 bytes   C,C++,C#,ObjC Source   ANSI   PC
228 #endif
229
230
231 /*Access protection configuration macros*/
232 #define AOUSAFE_AP_DMA_RP3_APP_MASTERS          (AOUSAFE_CPU0_NONSAFE | AOUSAFE_CPU0_SAFE)
233
234 #define AOUSAFE_AP_FCE_APP_MASTERS              (AOUSAFE_CPU0_NONSAFE | AOUSAFE_CPU0_SAFE)
235
236 #define AOUSAFE_AP_GETH_APP_MASTERS             (AOUSAFE_CPU1_NONSAFE | AOUSAFE_CPU1_SAFE)
237 #define AOUSAFE_AP_GETH_D0_APP_MASTERS          (AOUSAFE_CPU1_NONSAFE | AOUSAFE_CPU1_SAFE)
238 #define AOUSAFE_AP_GETH_D1_APP_MASTERS          (AOUSAFE_CPU1_NONSAFE | AOUSAFE_CPU1_SAFE)
239 #define AOUSAFE_AP_GETH_D2_APP_MASTERS          (AOUSAFE_CPU1_NONSAFE | AOUSAFE_CPU1_SAFE)
240 #define AOUSAFE_AP_GETH_D3_APP_MASTERS          (AOUSAFE_CPU1_NONSAFE | AOUSAFE_CPU1_SAFE)
241
242 #define AOUSAFE_AP_CAN0_APP_MASTERS              (AOUSAFE_CPU0_NONSAFE | AOUSAFE_CPU0_SAFE|AOUSAFE_CPU1_NONSAFE | AOUSAFE_CPU1_SAFE)
243 #define AOUSAFE_AP_CAN0_CTR_APP_MASTERS          (AOUSAFE_CPU0_NONSAFE | AOUSAFE_CPU0_SAFE|AOUSAFE_CPU1_NONSAFE | AOUSAFE_CPU1_SAFE)
244
245
246 #define AOUSAFE_AP_CAN1_APP_MASTERS              (AOUSAFE_CPU0_NONSAFE | AOUSAFE_CPU0_SAFE|AOUSAFE_CPU1_NONSAFE | AOUSAFE_CPU1_SAFE)
247 #define AOUSAFE_AP_CAN1_CTR_APP_MASTERS          (AOUSAFE_CPU0_NONSAFE | AOUSAFE_CPU0_SAFE|AOUSAFE_CPU1_NONSAFE | AOUSAFE_CPU1_SAFE)
248
249 #define AOUSAFE_AP_CAN2_APP_MASTERS              (AOUSAFE_CPU0_NONSAFE | AOUSAFE_CPU0_SAFE|AOUSAFE_CPU1_NONSAFE | AOUSAFE_CPU1_SAFE)
250 #define AOUSAFE_AP_CAN2_CTR_APP_MASTERS          (AOUSAFE_CPU0_NONSAFE | AOUSAFE_CPU0_SAFE|AOUSAFE_CPU1_NONSAFE | AOUSAFE_CPU1_SAFE)
251
252 #define AOUSAFE_AP_DMU_APP_MASTERS               (AOUSAFE_CPU0_NONSAFE | AOUSAFE_CPU0_SAFE|AOUSAFE_CPU1_NONSAFE | AOUSAFE_CPU1_SAFE)
253
254 #define AOUSAFE_AP_QSPI3_APP_MASTERS             (AOUSAFE_CPU0_NONSAFE | AOUSAFE_CPU0_SAFE|AOUSAFE_DMA_RESOURCE_PARTITION3)
255 #define AOUSAFE_AP_QSPI4_APP_MASTERS             (AOUSAFE_CPU0_NONSAFE | AOUSAFE_CPU0_SAFE)
256
257
258
259 /*****************************************************************************************************************************************************
260 /*                                                                  DATATYPES                                                                        *
```

f) After modification we could see ACCESS is given to only CPU0 as below

```
B::per, "DMA (Direct Memory Access)"
DMA (Direct Memory Access)
  DMA_CLC       00000008  EDIS        Disabled  DISS      No              DISR      Not requested
  DMA_ID        0087C004  MOD_NUMBER  0087      MOD_TYPE  C0              MOD_REV   04
  DMA_OTSS      00000000  BS          OTGB0     TGS       Not selected
  DMA_PRR0      00000000  PAT03       00        PAT02     00              PAT01     00              PAT00 00
  DMA_PRR1      00000000  PAT13       00        PAT12     00              PAT11     00              PAT10 00
  DMA_TIME      29356B68

  ⊞ Hardware Resource Partition 0

  ⊞ Hardware Resource Partition 1

  ⊞ Hardware Resource Partition 2

  ⊟ Hardware Resource Partition 3
    DMA_MODE3     00000000  MODE    User mode

    DMA_ACCEN30   70000006  IOC32E     Enabled   IOC32P     Enabled   CERBERUS   Enabled
                            CIF1       Disabled
                            HSSL1      Disabled  CPU3       Disabled  CPU3       Disabled  DMA       Di
                            HSSL0      Disabled  CPU2       Disabled  CPU2       Disabled  DMA       Di
                            SDMMC      Disabled  CPU1       Disabled  CPU1       Disabled  DMA       Di
                            HSM        Disabled  CPU0       Enabled   CPU0       Enabled   DMA       Di
    DMA_ACCEN31   XXXXXXXX

    ⊞ Sub-block 0

    ⊞ Sub-block 1
```

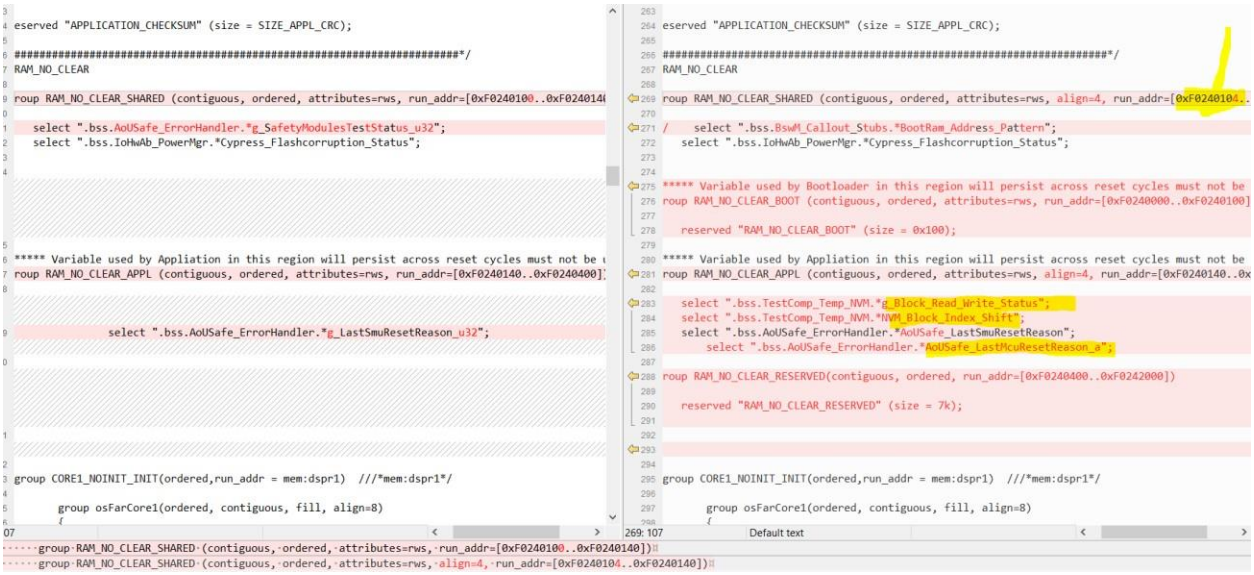
g) Above change fixed issue , software is running as expected.

EyeQ CDD Integration



Issue 3)Shared variables between application and bootloader location should be identical.

These shared variables helps in knowing any resets of SMU,MCU.We have modified address in application to match bootloader otherwise these variables interpretation will be wrong we will not be able to log issues in bootloader .



This can be checked by introducing faulty application by setting an alarm and tracing it in bootloader using SMU alarm API and then we can validate