32 BIT BOOTLOADER DOCUMENTATION

Octillion Power Systems India Private Limited

302, Sector 10, Sector 17/19, MIDC, Bhosari,

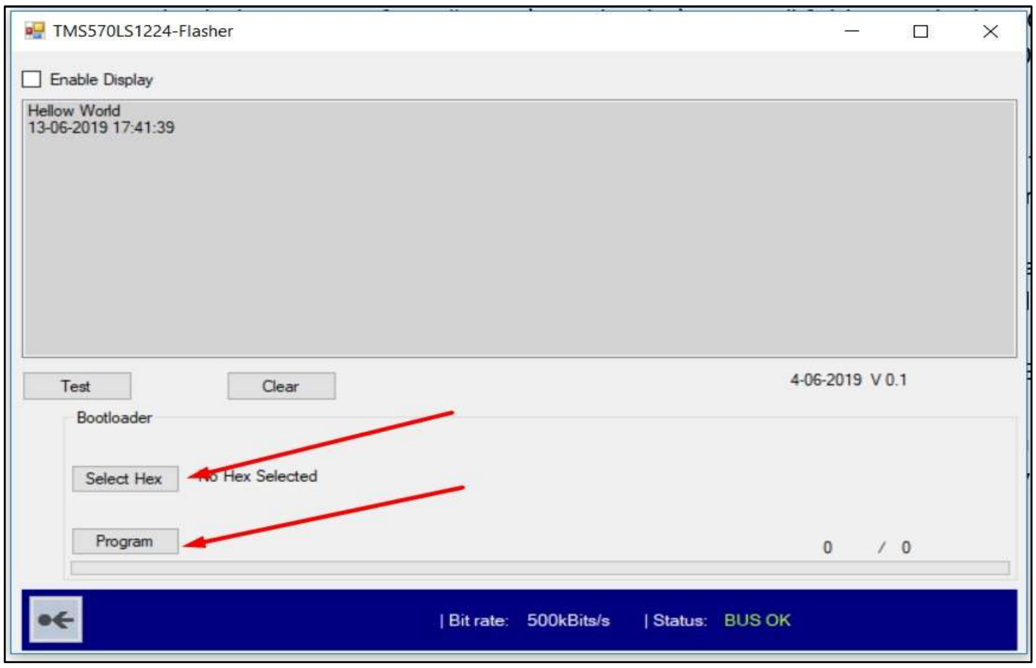
Pimpri – Chinchwad, Maharastra-411026.

www.octillion.us

Content

A.] Steps to follow to Testing the existing Bootloader v3:

1. Download the Bootloader folder from the GitHub. And unzip the file.
2. Open both the projects from “32Bit \Bootloader\Boot v3” folder inside the CCS environment. Upload the Bootloader project named “Boot v3 Bootloader” to the TMS570 Launchpad and then Debug the project named “Boot v3 Application”. On TMS570 Launchpad Led Blink would start working.
   [This will simulate the normal application running condition. For further testing the flashing activity make sure the “Memory Browser” which can be found in “View” menu inside the CCS is opened. Turn the “Continuous Refresh” on of “Memory Browser” tab.]
3. Make sure the “PCAN, transreceiver and TMS570 Launchpad” setup is connected and now open the GUI and make sure the GUI version is v0.1 and file is named as “32bit-Bootloader-GUI-V0.1”. The GUI executable can be found at “32Bit \GUI\32bit-Bootloader-GUI-installer-V0.1”. A PCAN initialized window will appear then click OK button which will be on the window.
4. After that GUI will open. Click the “Select Hex” button and browse to the “Boot v3 Application Int.hex” file in the folder “32Bit \Bootloader\Boot v3 \Boot v3 Application Int\Debug” and press “open” after selecting the hex file.


TMS570LS1224-Flasher

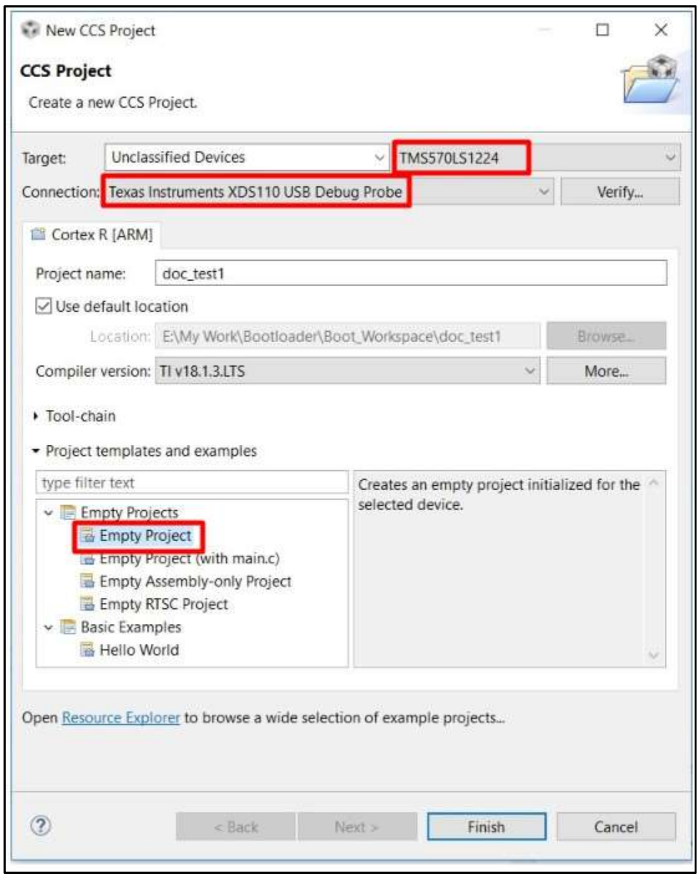
Enable Display

Hellow World
13-06-2019 17:41:39

5. After selecting the hex file. Click Program button on the GUI and the flashing activity will start. The start address of application is “0x14000”. To verify the flashing activity, open debug window of CCS and go to the start address in “Memory Browser” and check whether the addresses are been programmed or not.
6. Wait until full file is been flashed and “Programming Successful” has been displayed. After that the application program of Led blink must start working normally.

B.] Steps to follow to Make new Bootloader Project from Blank Page:

1. Open CCS and create a new project in the CCS. Make sure the red marked options are selected.



New CCS Project

CCS Project

Create a new CCS Project.

Target: Unclassified Devices — TMS570LS1224

Connection: Texas Instruments XDS110 USB Debug Probe

Cortex R [ARM]

Project name: doc_test1

☑ Use default location

Location: E:\My Work\Bootloader\Boot_Workspace\doc_test1

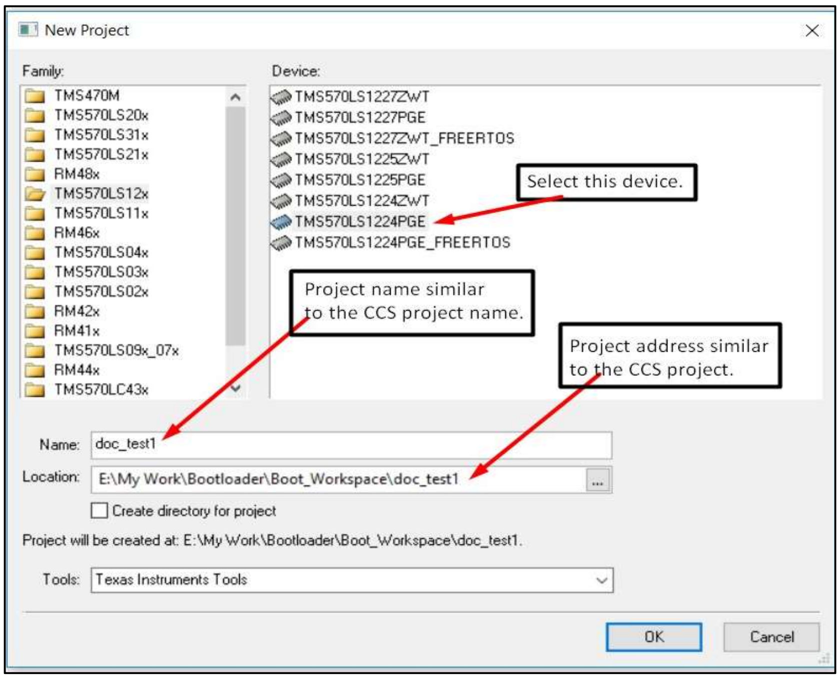
Compiler version: TI v18.1.3.LTS

Project templates and examples: Empty Projects > Empty Project

Creates an empty project initialized for the selected device.

Open Resource Explorer to browse a wide selection of example projects...

2. Open HalcoGen and create a new project and keep the name of HalcoGen project name similar to that of CCS for simplicity. Make sure the indicated things are completed.



New Project

Family: TMS570LS12x

Device: TMS570LS1224PGE

Select this device.

Project name similar to the CCS project name.

Project address similar to the CCS project.

Name: doc_test1

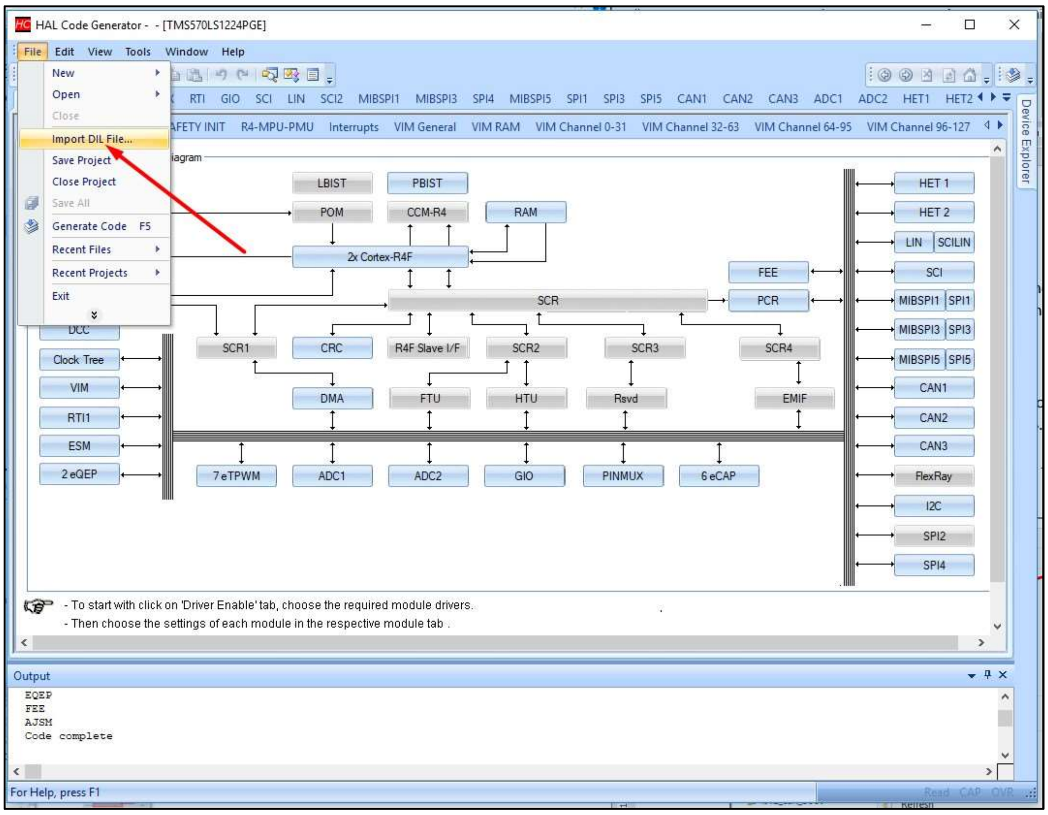
Location: E:\My Work\Bootloader\Boot_Workspace\doc_test1

☐ Create directory for project

Project will be created at: E:\My Work\Bootloader\Boot_Workspace\doc_test1.

Tools: Texas Instruments Tools

3. Now you can make any required changes in the HalcoGen project. For now, you can import the dil file named "Boot v3 Bootloader.dil" from "32Bit\Bootloader\Default Files" folder. This would add the HalcoGen settings used for "Boot v3 Bootloader" project.

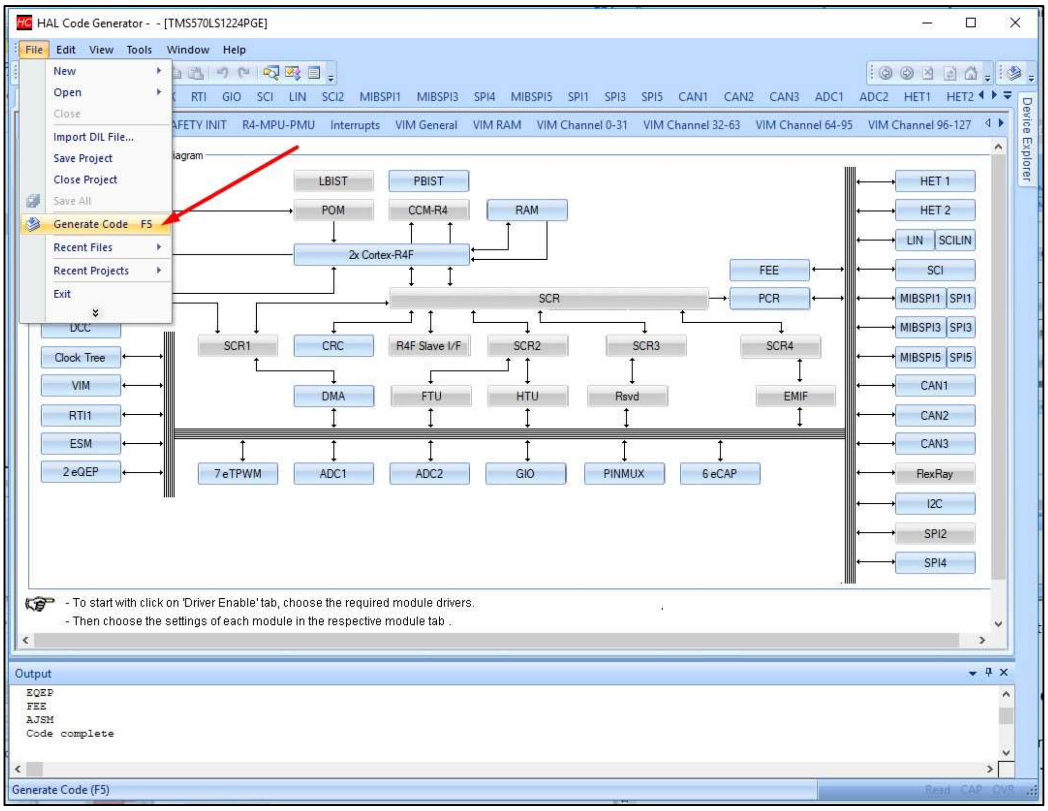


- To start with click on 'Driver Enable' tab, choose the required module drivers.
- Then choose the settings of each module in the respective module tab .

```
Output
EQEP
FEE
AJSM
Code complete
```

For Help, press F1

4. Generate code by going to "File-> Generate Code" or Pressing F5.

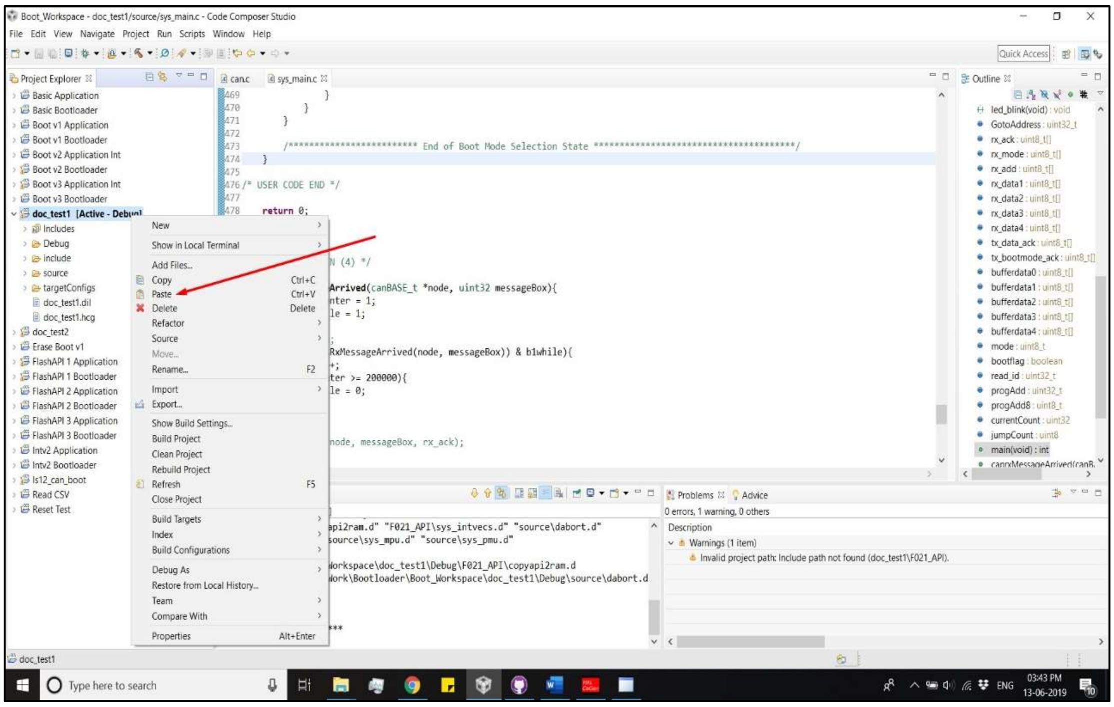


- To start with click on 'Driver Enable' tab, choose the required module drivers.
- Then choose the settings of each module in the respective module tab .

```
Output
EQEP
FEE
AJSM
Code complete
```

Generate Code (F5)

5. After code is generated. We a can shift to CCS.
6. Copy the “F021_API” folder from “32Bit\Bootloader\Default Files” folder.
7. Now in CCS, move towards “Project Explorer” tab and right click on the current project and Paste the folder inside the current project.



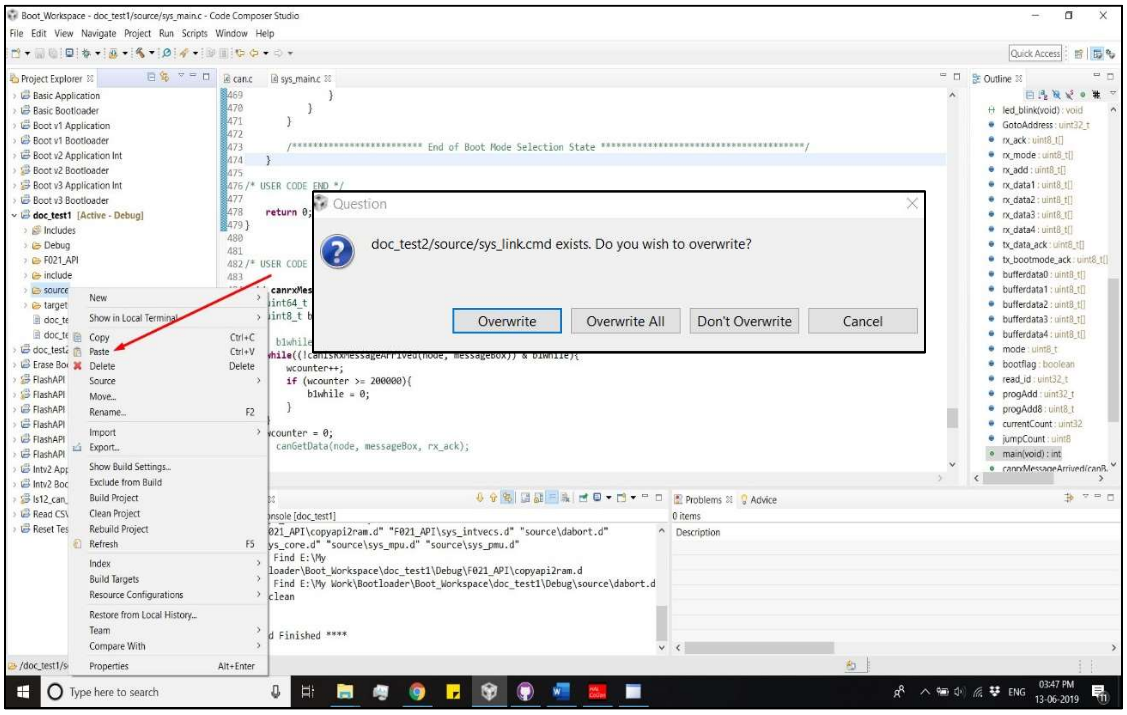
```
            }
        }
    }

    /************************ End of Boot Mode Selection State **************************************/
}

/* USER CODE END */

    return 0;
...
N (4) */
...Arrived(canBASE_t *node, uint32 messageBox){
...nter = 1;
...le = 1;
...RxMessageArrived(node, messageBox)) & b1while){
...+;
...ter >= 200000){
...le = 0;
...node, messageBox, rx_ack);
```

```
...api2ram.d" "F021_API\sys_intvecs.d" "source\dabort.d"
...source\sys_mpu.d" "source\sys_pmu.d"
...Workspace\doc_test1\Debug\F021_API\copyapi2ram.d
...Work\Bootloader\Boot_Workspace\doc_test1\Debug\source\dabort.d
```

0 errors, 1 warning, 0 others

Description
Warnings (1 item)
Invalid project path: Include path not found (doc_test1\F021_API).

8. After that copy the “sys_link.cmd” file from “32Bit\Bootloader\Default Files” folder.
9. Now in CCS, move towards “Project Explorer” tab inside the current project “source folder can be found. Right click on the source folder and Paste the file inside the source folder. A Question window will appear then click on Overwrite and the existing “sys_link.cmd” file will be replaced by new.



doc_test2/source/sys_link.cmd exists. Do you wish to overwrite?

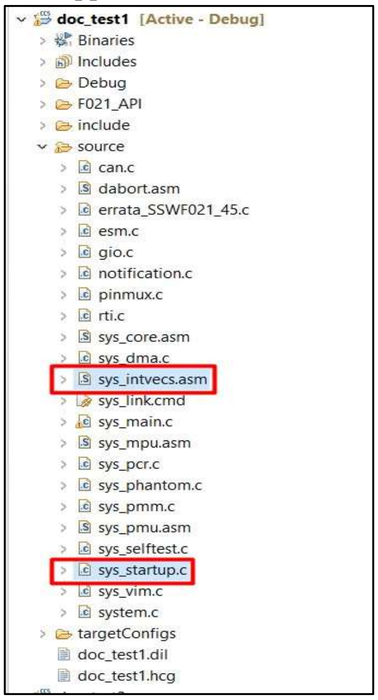
```
            }
        }
    }

    /************************ End of Boot Mode Selection State **************************************/
}

/* USER CODE END */

    return 0;
}


/* USER CODE
...
    b1while
while((!canIsRxMessageArrived(node, messageBox)) & b1while){
    wcounter++;
    if (wcounter >= 200000){
        b1while = 0;
    }
}
wcounter = 0;
canGetData(node, messageBox, rx_ack);
```

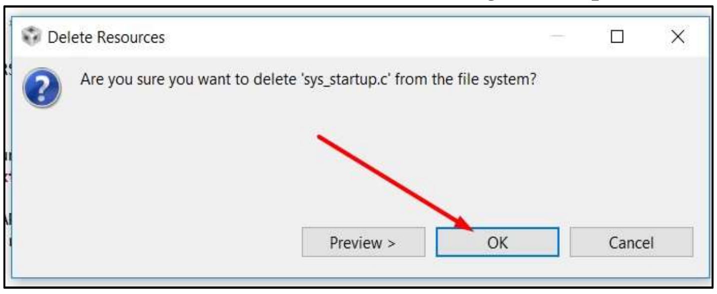
```
Console [doc_test1]
...021_API\copyapi2ram.d" "F021_API\sys_intvecs.d" "source\dabort.d"
...ys_core.d" "source\sys_mpu.d" "source\sys_pmu.d"
 Find E:\My
...loader\Boot_Workspace\doc_test1\Debug\F021_API\copyapi2ram.d
 Find E:\My Work\Bootloader\Boot_Workspace\doc_test1\Debug\source\dabort.d
clean

...d Finished ****
```

0 items

Description

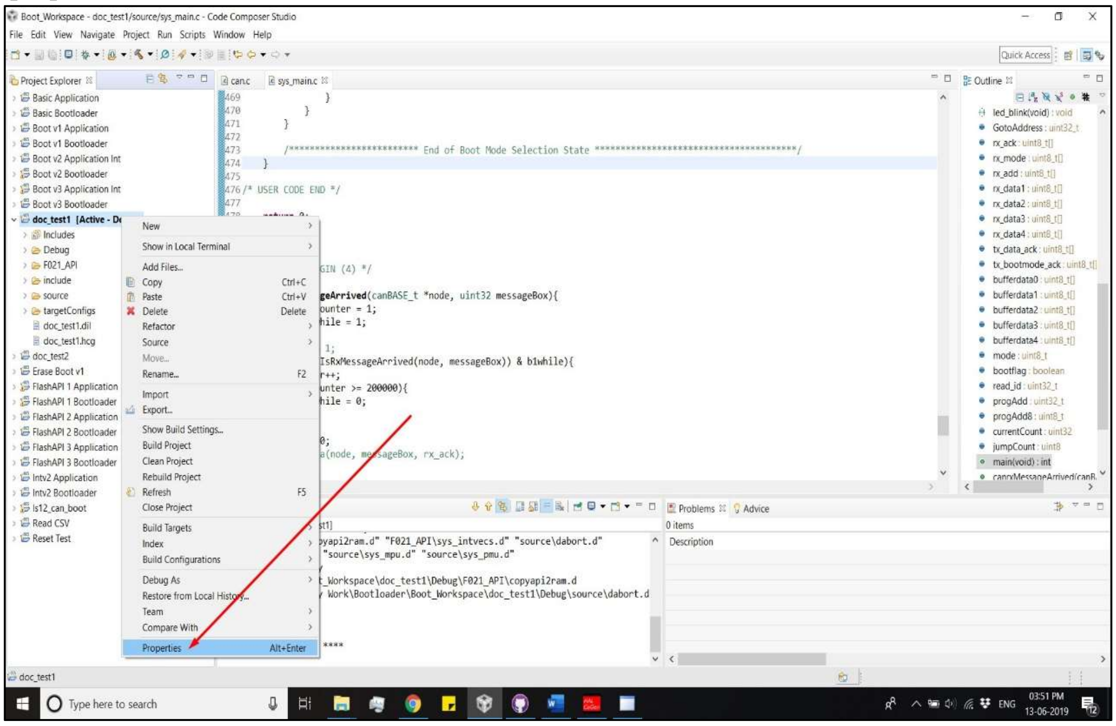
10. Delete two files from the source folder named “sys_intvecs.asm” & “sys_startup.c” because these files are included inside the “F021_API” folder. The delete option can be found by right clicking on the specific file. After selecting the delete option the a “Delete resources” window will appear then click OK button to delete the file. Delete both the files using similar procedure.





Are you sure you want to delete 'sys_startup.c' from the file system?

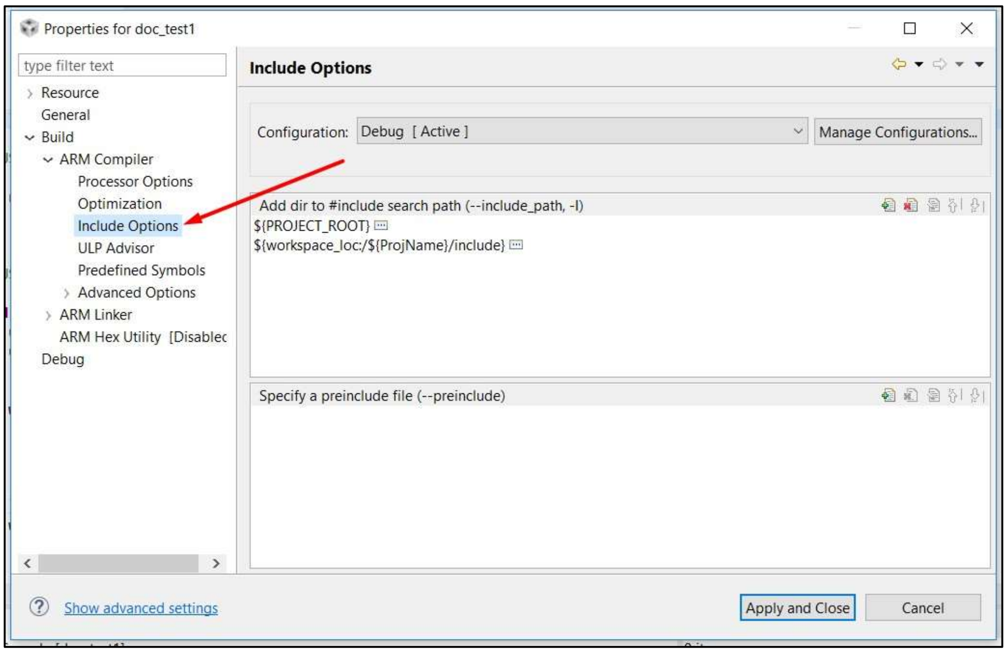
11. After that inside CCS in Project Explore Tab right click on the current project and select properties.



```
                }
            }
        }

        /************************ End of Boot Mode Selection State ****************************************/
    }

/* USER CODE END */

GIN (4) */

geArrived(canBASE_t *node, uint32 messageBox){
ounter = 1;
hile = 1;

 1;
IsRxMessageArrived(node, messageBox)) & b1while){
r++;
unter >= 200000){
hile = 0;

0;
a(node, messageBox, rx_ack);
```

```
st1]
pyapi2ram.d" "F021_API\sys_intvecs.d" "source\dabort.d"
 "source\sys_mpu.d" "source\sys_pmu.d"

t_Workspace\doc_test1\Debug\F021_API\copyapi2ram.d
y Work\Bootloader\Boot_Workspace\doc_test1\Debug\source\dabort.d

****
```

12. A properties window appears. Their inside “Build -> ARM Complier” select “Include options”. A include options configuration window must appear as shown in the below screenshot.

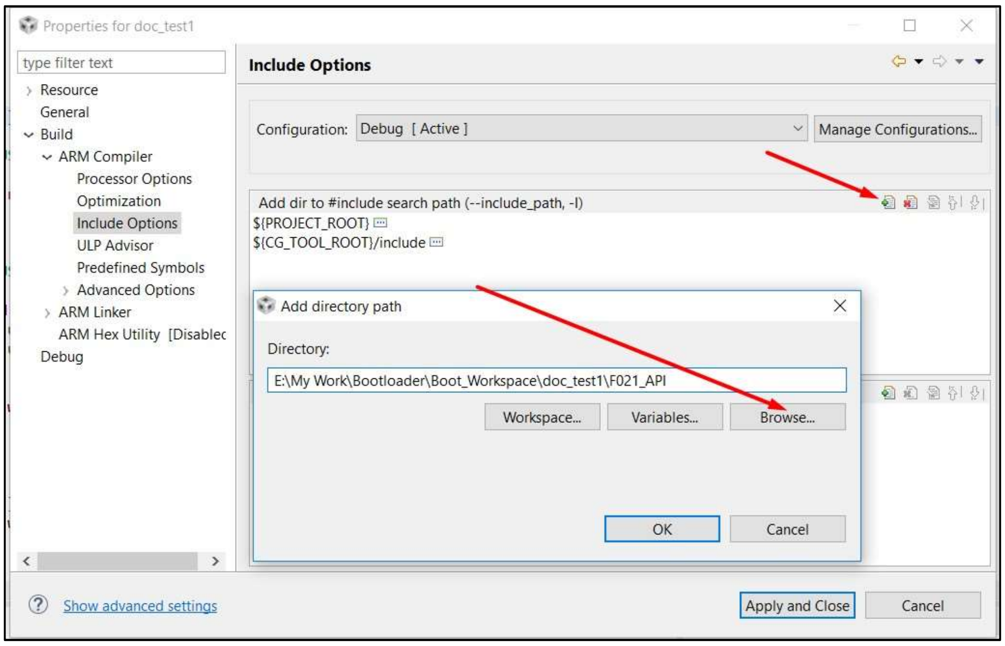


Configuration: Debug [Active]

Add dir to #include search path (--include_path, -I)
${PROJECT_ROOT}
${workspace_loc:/${ProjName}/include}

Specify a preinclude file (--preinclude)

13. Then, click the “Add” button as shown in the below screenshot and after that a “Add directory path” window will appear. Their select “Browse” button and browse towards and select the “F021_API” folder you pasted inside your current project and click “OK”.



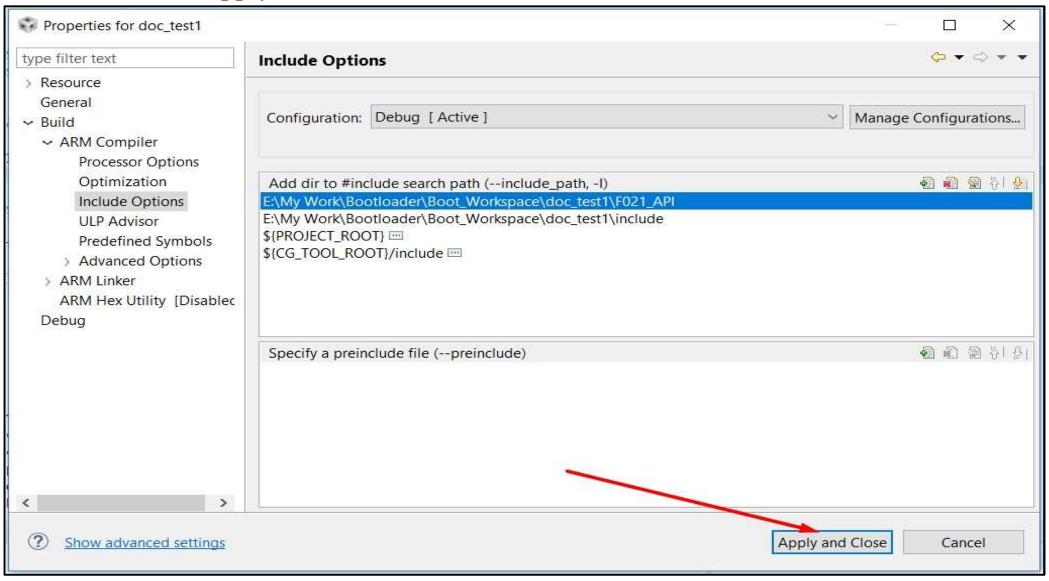
Configuration: Debug [Active]

Add dir to #include search path (--include_path, -I)
${PROJECT_ROOT}
${CG_TOOL_ROOT}/include

Directory: E:\My Work\Bootloader\Boot_Workspace\doc_test1\F021_API

14. Similar to “F021_API” folder also include the path of “include” folder created by HalcoGen inside your current project folder. Click the “Add” button and after that a “Add directory path” window will appear. Their select “Browse” button and browse towards and select the “include” folder which is inside your current project and click “OK”.

15. After that Click “Apply and Close”.



Properties for doc_test1 — Include Options

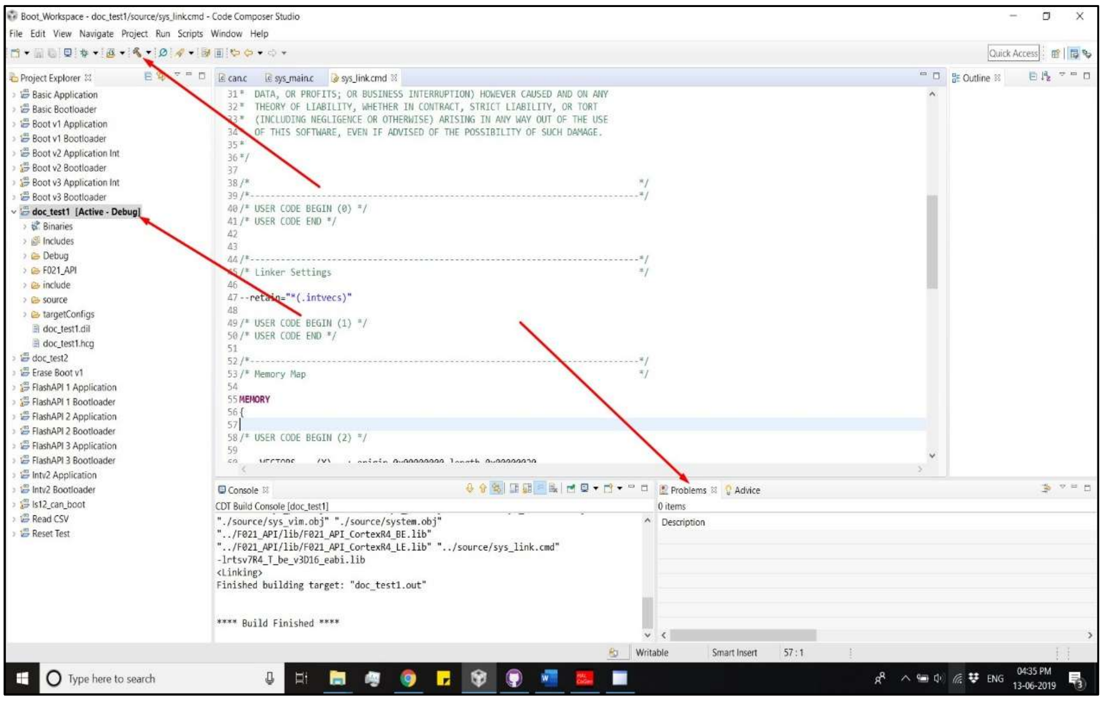
Configuration: Debug [Active]

Add dir to #include search path (--include_path, -I)

```
E:\My Work\Bootloader\Boot_Workspace\doc_test1\F021_API
E:\My Work\Bootloader\Boot_Workspace\doc_test1\include
${PROJECT_ROOT}
${CG_TOOL_ROOT}/include
```

Specify a preinclude file (--preinclude)

16. Now after selecting the current project you can Compile the project by clicking the hammer icon shown in the below screenshot. After compiling make sure that there are no error showing on the problems tab. There might be some warnings showing but their must not be any error. If there is any error recheck from step 6 till step 15.



Boot_Workspace - doc_test1/source/sys_link.cmd - Code Composer Studio

```
*  DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
*  THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
*  (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
*  OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*
*/


/*                                                                    */
/*--------------------------------------------------------------------*/
/* USER CODE BEGIN (0) */
/* USER CODE END */


/*--------------------------------------------------------------------*/
/* Linker Settings                                                    */

--retain="*(.intvecs)"

/* USER CODE BEGIN (1) */
/* USER CODE END */

/*--------------------------------------------------------------------*/
/* Memory Map                                                         */

MEMORY
{

/* USER CODE BEGIN (2) */
```

CDT Build Console [doc_test1]

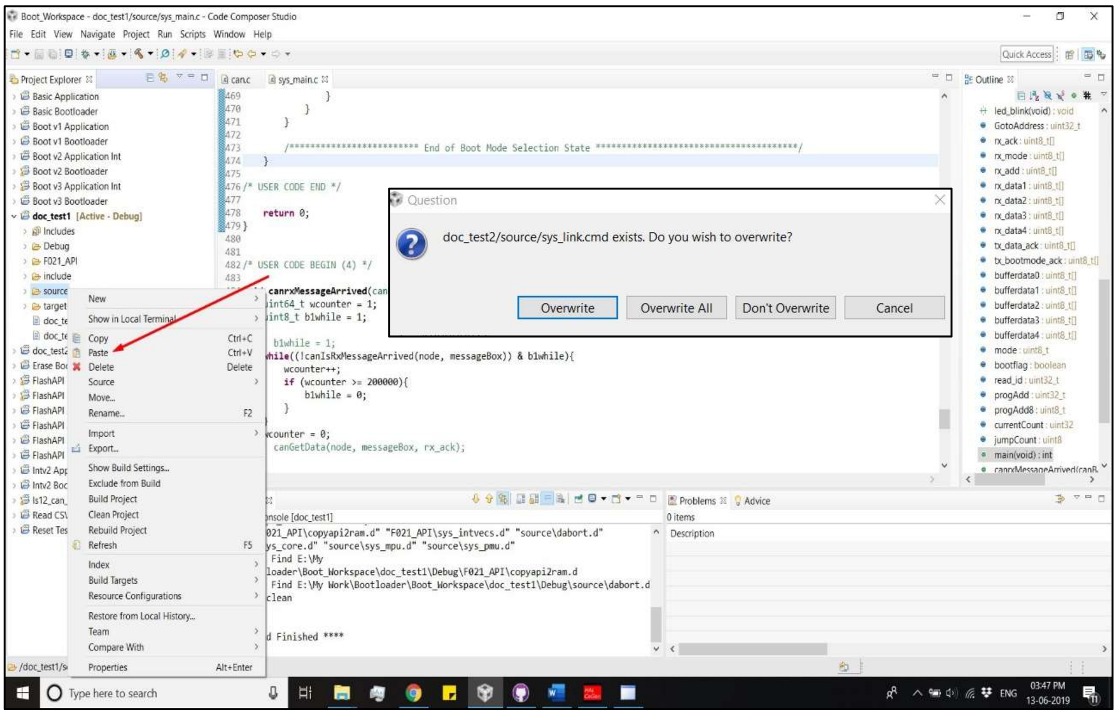
```
"./source/sys_vim.obj" "./source/system.obj"
"../F021_API/lib/F021_API_CortexR4_BE.lib"
"../F021_API/lib/F021_API_CortexR4_LE.lib" "../source/sys_link.cmd"
-lrtsv7R4_T_be_v3D16_eabi.lib
<Linking>
Finished building target: "doc_test1.out"


**** Build Finished ****
```

Problems: 0 items

17. If project is compiled properly the you can now use this project for developing your bootloader application.
18. For purely testing purpose you can copy the “sys_main.c” file from “32Bit\Bootloader\Default Files\Extra” folder and paste it to source folder as explained in step 9. Override the existing file. Then save the project and Compile the project as shown in step 16. Now this project is similar to “Boot v3 Bootloader” project and you can flash this project instead of “Boot v3 Bootloader” as a Bootloader for testing with “32bit-Bootloader-GUI-installer-V0.1” GUI.

C.] Steps to follow for changed HalcoGen setting in Bootloader Proj:

1. Open the same project in HalcoGen by opening the ".hcg" file of the project in HalcoGen.
2. Make the required changes and Generate the code for the project by going to "File-> Generate Code" or Pressing F5.
3. Open CCS, move towards "Project Explorer" tab inside the current project "source folder can be found. Right click on the source folder and Paste the file inside the source folder. A Question window will appear then click on Overwrite and the existing "sys_link.cmd" file will be replaced by new.

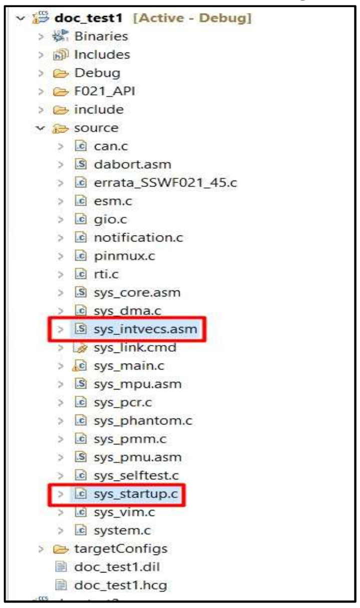


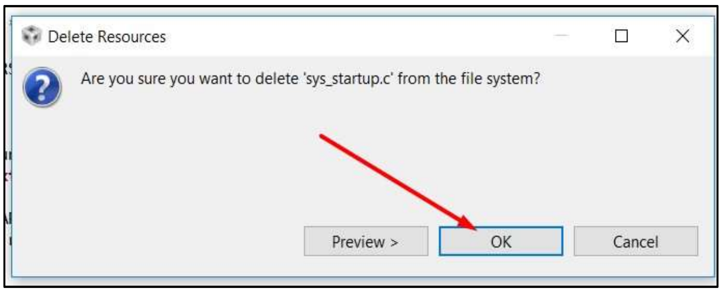
doc_test2/source/sys_link.cmd exists. Do you wish to overwrite?

```
                }
            }
        }

        /************************ End of Boot Mode Selection State ************************************/
    }

/* USER CODE END */

    return 0;
}


/* USER CODE BEGIN (4) */

    canrxMessageArrived(can
    int64_t wcounter = 1;
    int8_t b1while = 1;

    b1while = 1;
    while((!canIsRxMessageArrived(node, messageBox)) & b1while){
        wcounter++;
        if (wcounter >= 200000){
            b1while = 0;
        }
    }
    wcounter = 0;
    canGetData(node, messageBox, rx_ack);
```

```
Console [doc_test1]
021_API\copyapi2ram.d" "F021_API\sys_intvecs.d" "source\dabort.d"
ys_core.d" "source\sys_mpu.d" "source\sys_pmu.d"
 Find E:\My
loader\Boot_Workspace\doc_test1\Debug\F021_API\copyapi2ram.d
 Find E:\My Work\Bootloader\Boot_Workspace\doc_test1\Debug\source\dabort.d
clean


d Finished ****
```

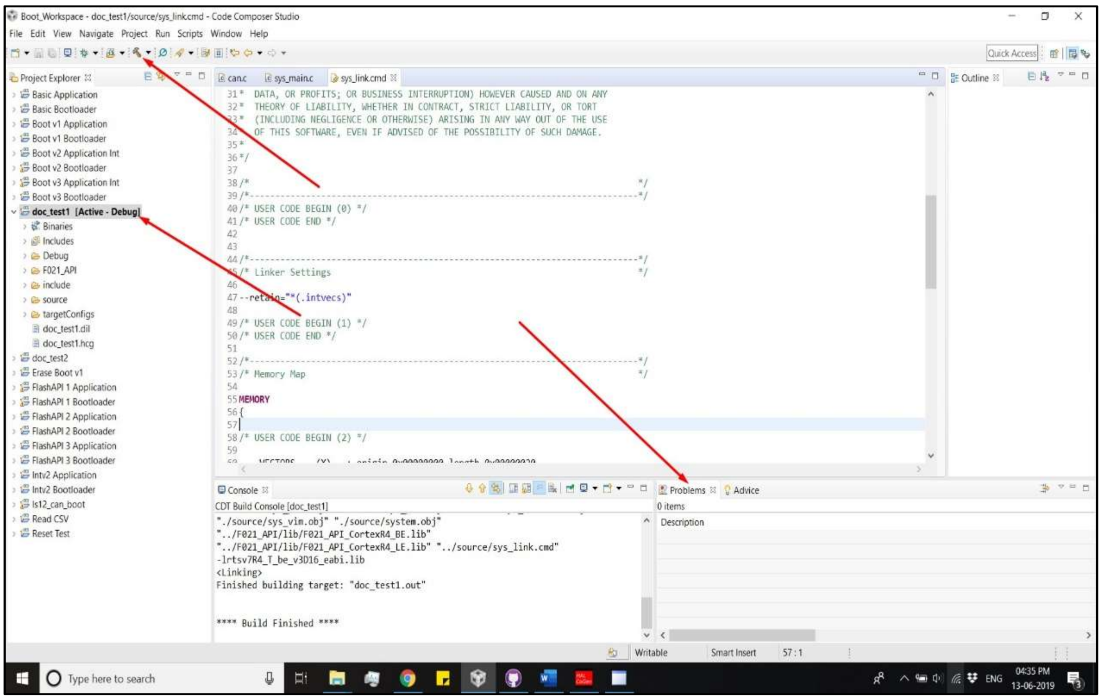
4. After that delete two files from the source folder named "sys_intvecs.asm" & "sys_startup.c". The delete option can be found by right clicking on the specific file. After selecting the delete option the a "Delete resources" window will appear then click OK button to delete the file. Delete both the files using similar procedure.





Are you sure you want to delete 'sys_startup.c' from the file system?

5. Now after selecting the current project you can Compile the project by clicking the hammer icon shown in the below screenshot. After compiling make sure that there are no error showing on the problems tab. There might be some warnings showing but their must not be any error.



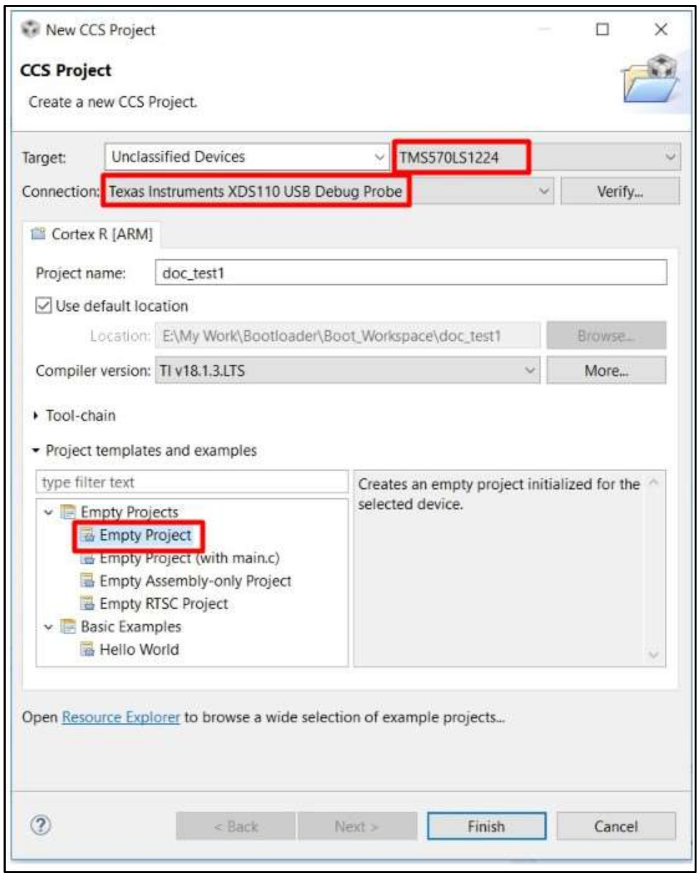
```
*  DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
*  THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
*  (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
*  OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*
*/

/*                                                                    */
/*----------------------------------------------------------------------*/
/* USER CODE BEGIN (0) */
/* USER CODE END */


/*----------------------------------------------------------------------*/
/* Linker Settings                                                      */

--retain="*(.intvecs)"

/* USER CODE BEGIN (1) */
/* USER CODE END */

/*----------------------------------------------------------------------*/
/* Memory Map                                                           */

MEMORY
{

/* USER CODE BEGIN (2) */

```

```
"./source/sys_vim.obj" "./source/system.obj"
"../F021_API/lib/F021_API_CortexR4_BE.lib"
"../F021_API/lib/F021_API_CortexR4_LE.lib" "../source/sys_link.cmd"
-lrtsv7R4_T_be_v3D16_eabi.lib
<Linking>
Finished building target: "doc_test1.out"


**** Build Finished ****
```

6. If project is compiled properly then you can make further changes in C code inside CCS.

D.] Steps for setting up the Application project & Generate Hex file:

1. Open CCS and create a new project in the CCS. Make sure the red marked options are selected.



New CCS Project

CCS Project

Create a new CCS Project.

Target: Unclassified Devices | TMS570LS1224

Connection: Texas Instruments XDS110 USB Debug Probe

Cortex R [ARM]

Project name: doc_test1

☑ Use default location

Location: E:\My Work\Bootloader\Boot_Workspace\doc_test1

Compiler version: TI v18.1.3.LTS

Tool-chain

Project templates and examples

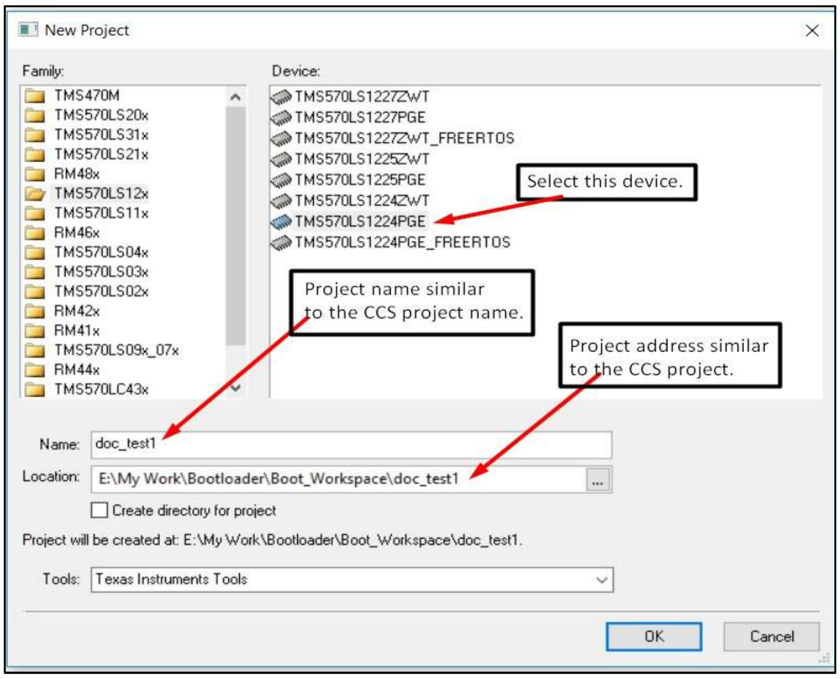
Empty Projects: Empty Project; Empty Project (with main.c); Empty Assembly-only Project; Empty RTSC Project

Basic Examples: Hello World

Creates an empty project initialized for the selected device.

Open Resource Explorer to browse a wide selection of example projects...

2. Open HalcoGen and create a new project and keep the name of HalcoGen project name similar to that of CCS for simplicity. Make sure the indicated things are completed.



New Project

Family: TMS470M, TMS570LS20x, TMS570LS31x, TMS570LS21x, RM48x, TMS570LS12x, TMS570LS11x, RM46x, TMS570LS04x, TMS570LS03x, TMS570LS02x, RM42x, RM41x, TMS570LS09x_07x, RM44x, TMS570LC43x

Device: TMS570LS1227ZWT, TMS570LS1227PGE, TMS570LS1227ZWT_FREERTOS, TMS570LS1225ZWT, TMS570LS1225PGE, TMS570LS1224ZWT, TMS570LS1224PGE, TMS570LS1224PGE_FREERTOS

Name: doc_test1

Location: E:\My Work\Bootloader\Boot_Workspace\doc_test1

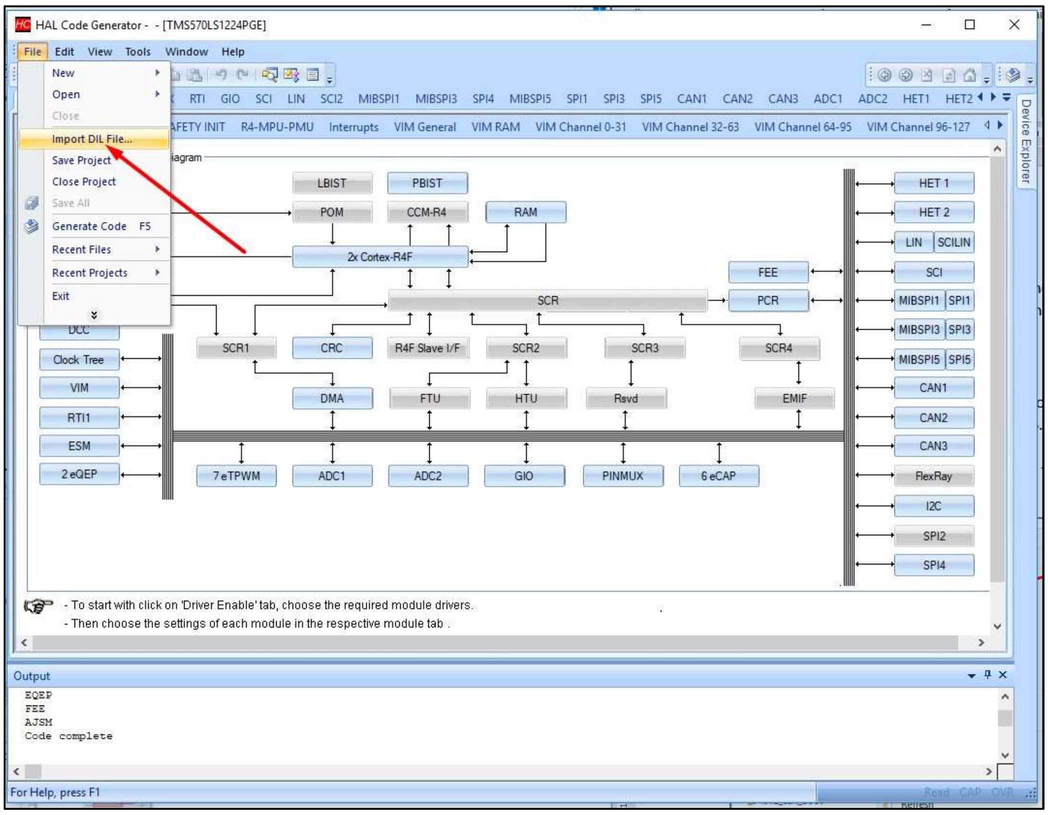
☐ Create directory for project

Project will be created at: E:\My Work\Bootloader\Boot_Workspace\doc_test1.

Tools: Texas Instruments Tools

Select this device.
Project name similar to the CCS project name.
Project address similar to the CCS project.
OK
Cancel

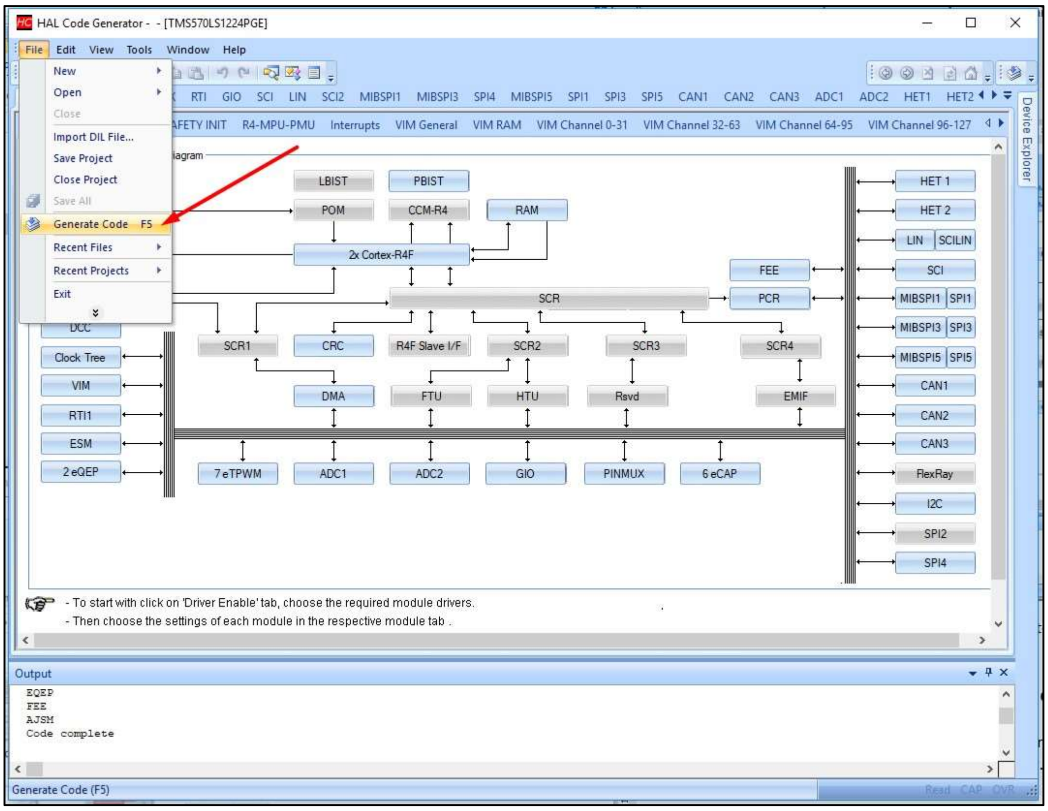
3. Now you can make any required changes in the HalcoGen project. For now, you can import the dil file named "Boot v3 Bootloader.dil" from "32Bit\Bootloader\Default Files" folder. This would add the HalcoGen settings used for "Boot v3 Bootloader" project.



- To start with click on 'Driver Enable' tab, choose the required module drivers.
- Then choose the settings of each module in the respective module tab .

```
EQEP
FEE
AJSM
Code complete
```

4. Generate code by going to "File-> Generate Code" or Pressing F5.



- To start with click on 'Driver Enable' tab, choose the required module drivers.
- Then choose the settings of each module in the respective module tab .

```
EQEP
FEE
AJSM
Code complete
```

5. After code is generated. We a can shift to CCS.
6. Open the "sys_link.cmd" file which is inside the source folder of current application project and replace the following lines

   VECTORS (X): origin=0x00000000 length=0x00000020

   FLASH0 (RX): origin=0x00000020 length=0x0013FEE0

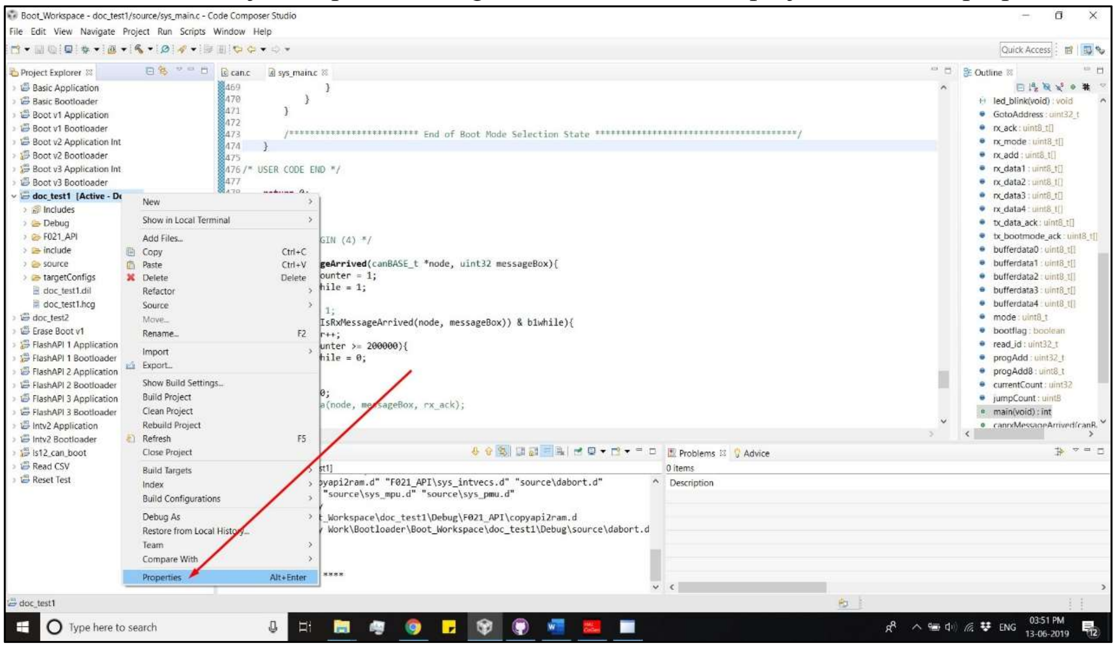
   to

   VECTORS (X): origin=0x00014000 length=0x00000020

   FLASH0 (RX): origin=0x00014020 length=0x0014FEE0

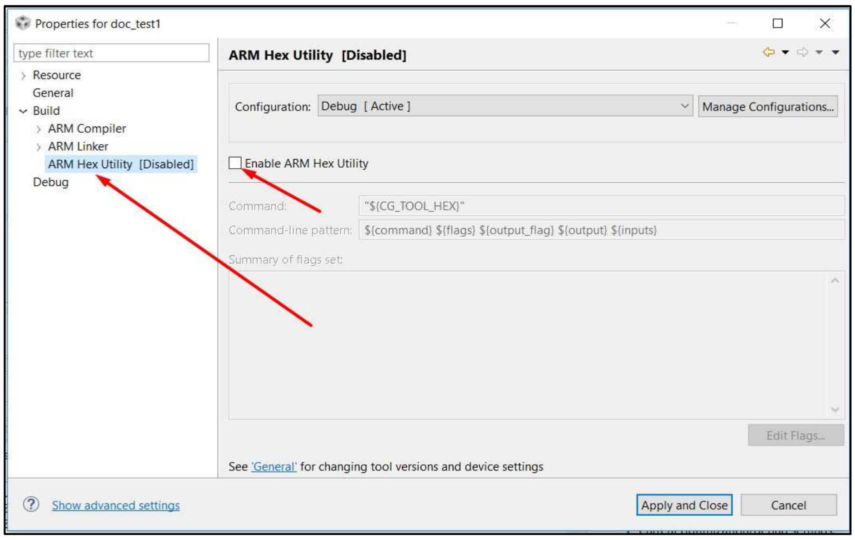
   Here, VECTORS 0x14000 is start address & FLASH0 0x14020 is start address + 0x20. These addresses can be changed according to change in start address.

7. After that inside Project Explore Tab right click on the current project and select properties.



```
      }
    }
  }
  /************************ End of Boot Mode Selection State ***************************************/
}
/* USER CODE END */
```

8. A properties window appears. Their inside "Build" select "ARM Hex Utility". A ARM Hex Utility window will appear as shown in the below screenshot. There check the "Enable ARM Hex Utility" Box.



Configuration: Debug [Active]

☐ Enable ARM Hex Utility

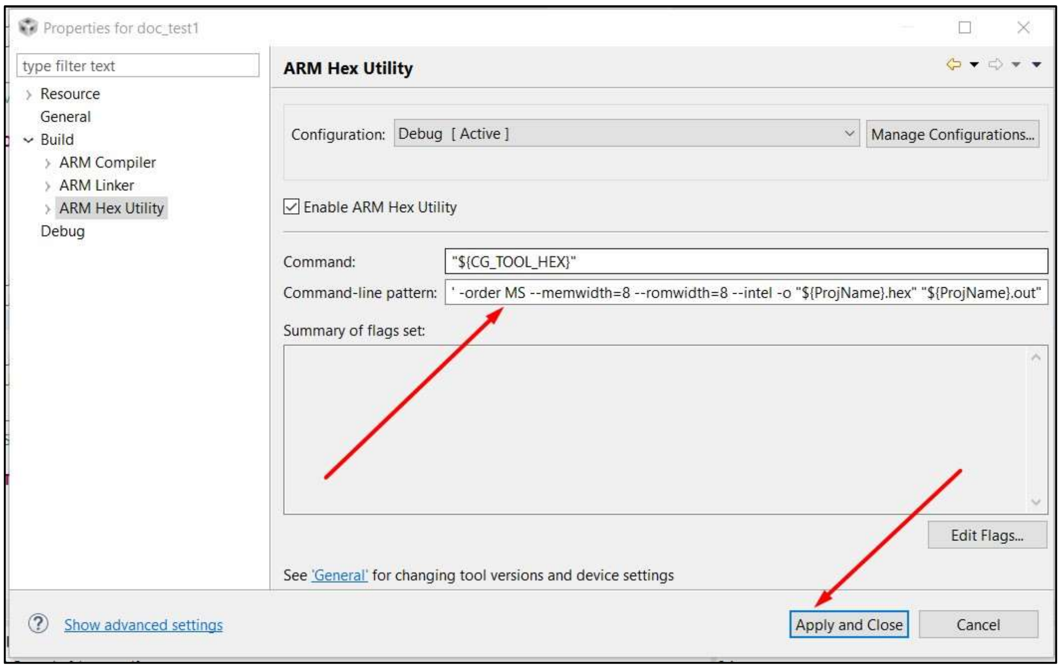
Command: "${CG_TOOL_HEX}"

Command-line pattern: ${command} ${flags} ${output_flag} ${output} ${inputs}

Summary of flags set:

See 'General' for changing tool versions and device settings

9. After that, change the text inside "Command-line pattern" box from
   ${command} ${flags} ${output_flag} ${output} ${inputs}
   with
   "${CG_TOOL_HEX}" -order MS --memwidth=8 --romwidth=8 --intel -o "${ProjName}.hex" "${ProjName}.out"



Properties for doc_test1

ARM Hex Utility

Configuration: Debug [Active]

☑ Enable ARM Hex Utility

Command: "${CG_TOOL_HEX}"

Command-line pattern: ' -order MS --memwidth=8 --romwidth=8 --intel -o "${ProjName}.hex" "${ProjName}.out"

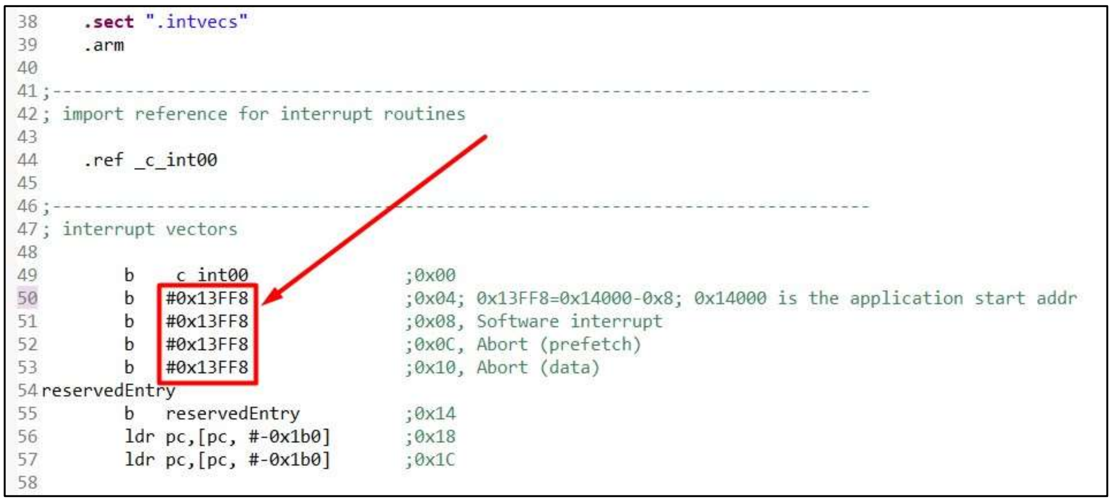
Summary of flags set:

See 'General' for changing tool versions and device settings

   After that click "Apply and Close".
10. Now after Compiling the project a hex file will be generate inside "Debug" folder of the current project. The name of the file will be "current project name".hex
11. After completing all the steps, programming the application code can be carried out. After completing the application code build the project and use the hex file generated in "Debug" folder for flashing the MCU using GUI.

E.] Steps to change the Jump address of Application:

1. Open both, the Bootloader and Application project inside the CCS environment.
2. From Bootloader project open the “sys_intvecs.asm” file inside the “F021_API” folder.



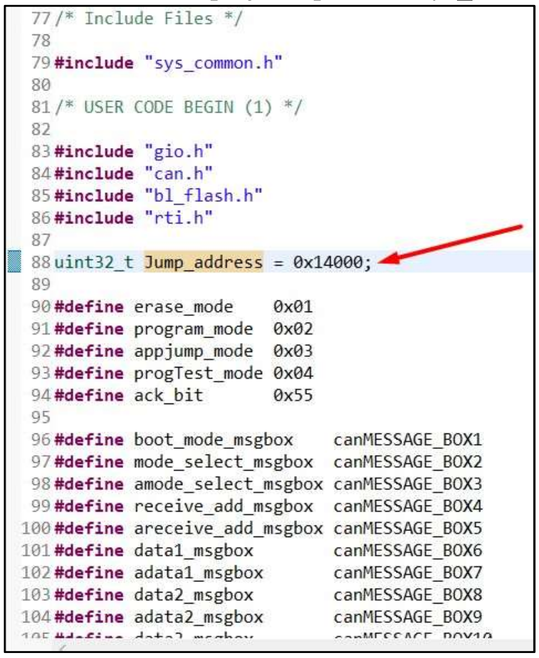
```
    .sect ".intvecs"
    .arm

;-------------------------------------------------------------------------------
; import reference for interrupt routines

    .ref _c_int00

;-------------------------------------------------------------------------------
; interrupt vectors

        b   _c_int00                ;0x00
        b   #0x13FF8                ;0x04; 0x13FF8=0x14000-0x8; 0x14000 is the application start addr
        b   #0x13FF8                ;0x08, Software interrupt
        b   #0x13FF8                ;0x0C, Abort (prefetch)
        b   #0x13FF8                ;0x10, Abort (data)
reservedEntry
        b   reservedEntry           ;0x14
        ldr pc,[pc, #-0x1b0]        ;0x18
        ldr pc,[pc, #-0x1b0]        ;0x1C
```

All the four numbers inside the box are depending upon the starting address of application.

Here, Number = “Starting address” – 0x08.

Replace all the four numbers according to the above formula.

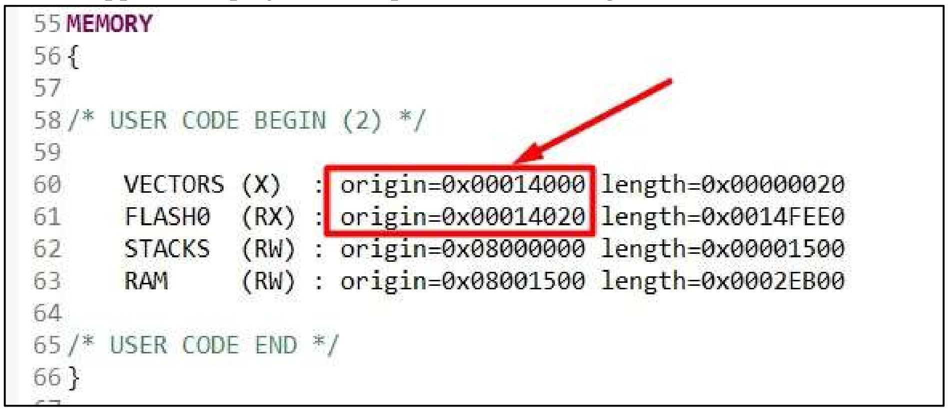
3. From Bootloader project open the “sys_main.c” file inside the “source” folder.



```
/* Include Files */

#include "sys_common.h"

/* USER CODE BEGIN (1) */

#include "gio.h"
#include "can.h"
#include "bl_flash.h"
#include "rti.h"

uint32_t Jump_address = 0x14000;

#define erase_mode     0x01
#define program_mode   0x02
#define appjump_mode   0x03
#define progTest_mode  0x04
#define ack_bit        0x55

#define boot_mode_msgbox     canMESSAGE_BOX1
#define mode_select_msgbox   canMESSAGE_BOX2
#define amode_select_msgbox  canMESSAGE_BOX3
#define receive_add_msgbox   canMESSAGE_BOX4
#define areceive_add_msgbox  canMESSAGE_BOX5
#define data1_msgbox         canMESSAGE_BOX6
#define adata1_msgbox        canMESSAGE_BOX7
#define data2_msgbox         canMESSAGE_BOX8
#define adata2_msgbox        canMESSAGE_BOX9
```

Change the Jump_Address variable value to the new address.

4. From Application project open the “sys_link.cmd” file which is inside the source folder of current application project and replace the following two address inside the box.



```
MEMORY
{

/* USER CODE BEGIN (2) */

    VECTORS (X)  : origin=0x00014000 length=0x00000020
    FLASH0  (RX) : origin=0x00014020 length=0x0014FEE0
    STACKS  (RW) : origin=0x08000000 length=0x00001500
    RAM     (RW) : origin=0x08001500 length=0x0002EB00

/* USER CODE END */
}
```

Here, VECTORS (X): 0x14000 is start address & FLASH0 (RX): 0x14020 is start address + 0x20. These addresses can be changed according to change in start address.

Where, VECTORS (X): “Start address” & FLASH0 (RX): “Start address” + 0x20.

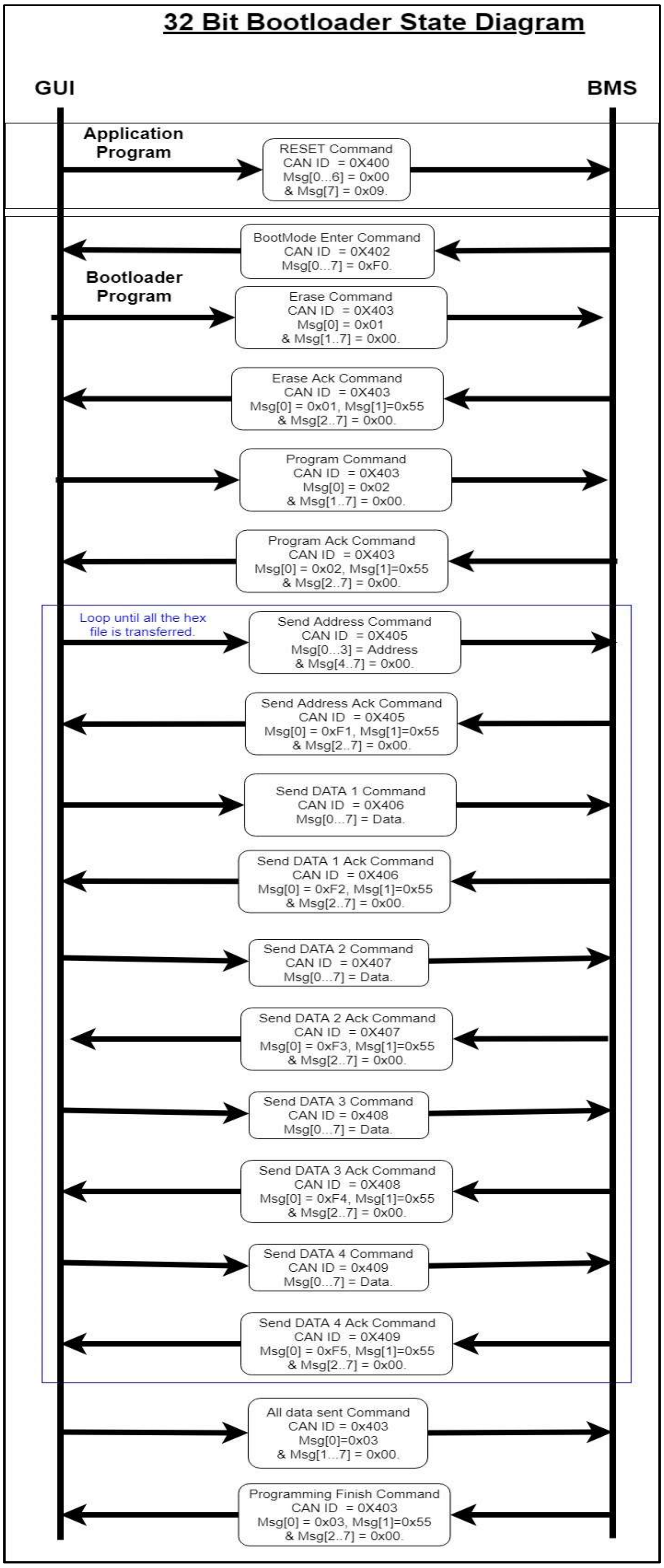
5. Save all the changes. Making above changes will change your application start address.

F.] Important Notes:

1. The base example referred was "CAN Bus Bootloader for TMS570LS12x MCU" the Texas document is named as SPNA186. "F021_API" files were taken from this (http://www.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=spna186&fileType=zip) example project.

2. The only change made to the library was in "bl_flash.c" file inside "**Fapi_BlockProgram**" function line number 191 and the function call was changed from
   *Fapi_issueProgrammingCommand((uint32_t *)dst, (uint8_t *)src, (uint32_t) bytes, 0, 0, Fapi_AutoEccGeneration);*
   to
   *Fapi_issueProgrammingCommand((uint32_t *)dst, (uint8_t *)src, (uint32_t) bytes, 0, 0, Fapi_DataOnly);*
   As the same address was not get programmed twice. We required this feature because for blank space we are programming 0xFF initially and once the data is received for the same address, we are programming the data.

G.] 32 Bit Bootloader and GUI Communication State diagram:


32 Bit Bootloader State Diagram
GUI
BMS
Application Program
RESET Command CAN ID = 0X400 Msg[0...6] = 0x00 & Msg[7] = 0x09.
BootMode Enter Command CAN ID = 0X402 Msg[0...7] = 0xF0.
Bootloader Program
Erase Command CAN ID = 0X403 Msg[0] = 0x01 & Msg[1..7] = 0x00.
Erase Ack Command CAN ID = 0X403 Msg[0] = 0x01, Msg[1]=0x55 & Msg[2..7] = 0x00.
Program Command CAN ID = 0X403 Msg[0] = 0x02 & Msg[1..7] = 0x00.
Program Ack Command CAN ID = 0X403 Msg[0] = 0x02, Msg[1]=0x55 & Msg[2..7] = 0x00.
Loop until all the hex file is transferred.
Send Address Command CAN ID = 0X405 Msg[0...3] = Address & Msg[4..7] = 0x00.
Send Address Ack Command CAN ID = 0X405 Msg[0] = 0xF1, Msg[1]=0x55 & Msg[2..7] = 0x00.
Send DATA 1 Command CAN ID = 0X406 Msg[0...7] = Data.
Send DATA 1 Ack Command CAN ID = 0X406 Msg[0] = 0xF2, Msg[1]=0x55 & Msg[2..7] = 0x00.
Send DATA 2 Command CAN ID = 0X407 Msg[0...7] = Data.
Send DATA 2 Ack Command CAN ID = 0X407 Msg[0] = 0xF3, Msg[1]=0x55 & Msg[2..7] = 0x00.
Send DATA 3 Command CAN ID = 0x408 Msg[0...7] = Data.
Send DATA 3 Ack Command CAN ID = 0X408 Msg[0] = 0xF4, Msg[1]=0x55 & Msg[2..7] = 0x00.
Send DATA 4 Command CAN ID = 0x409 Msg[0...7] = Data.
Send DATA 4 Ack Command CAN ID = 0X409 Msg[0] = 0xF5, Msg[1]=0x55 & Msg[2..7] = 0x00.
All data sent Command CAN ID = 0x403 Msg[0]=0x03 & Msg[1...7] = 0x00.
Programming Finish Command CAN ID = 0X403 Msg[0] = 0x03, Msg[1]=0x55 & Msg[2..7] = 0x00.