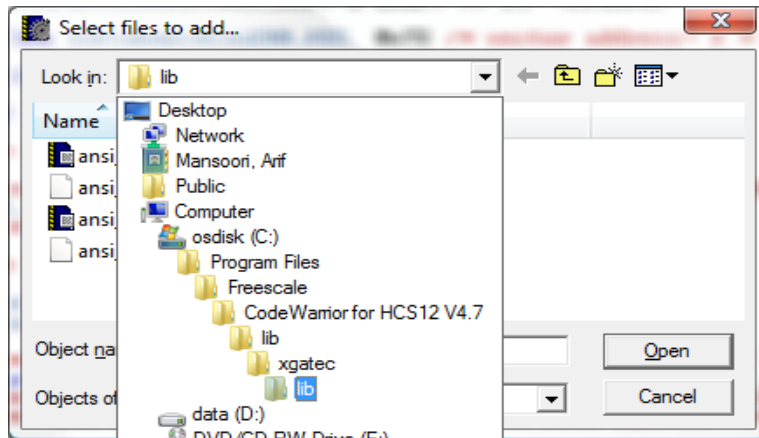


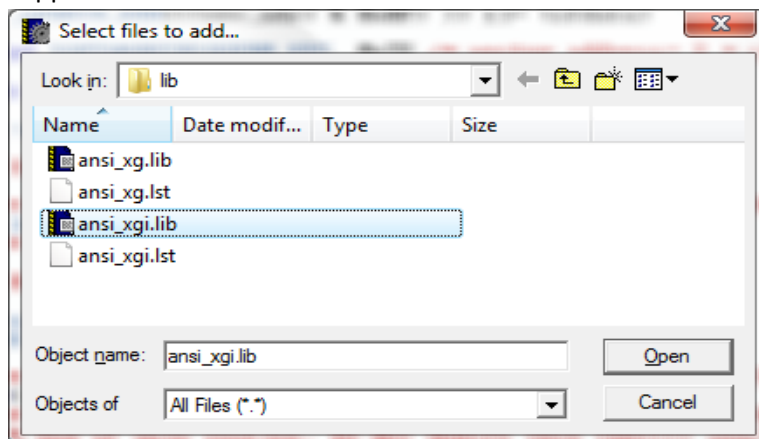
To add XGATE support to projects using Processor Expert for HCS12X

Perform the following steps

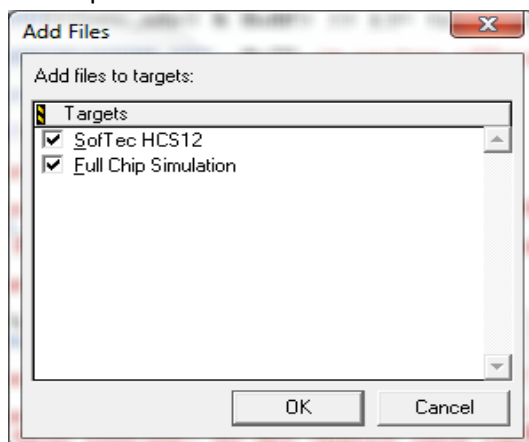
1. Click on Files. Right click on Libraries, then click on Add Files.



From this directory select ansi_xgi.lib for non-floating support and ansi_xg.lib for floating support.





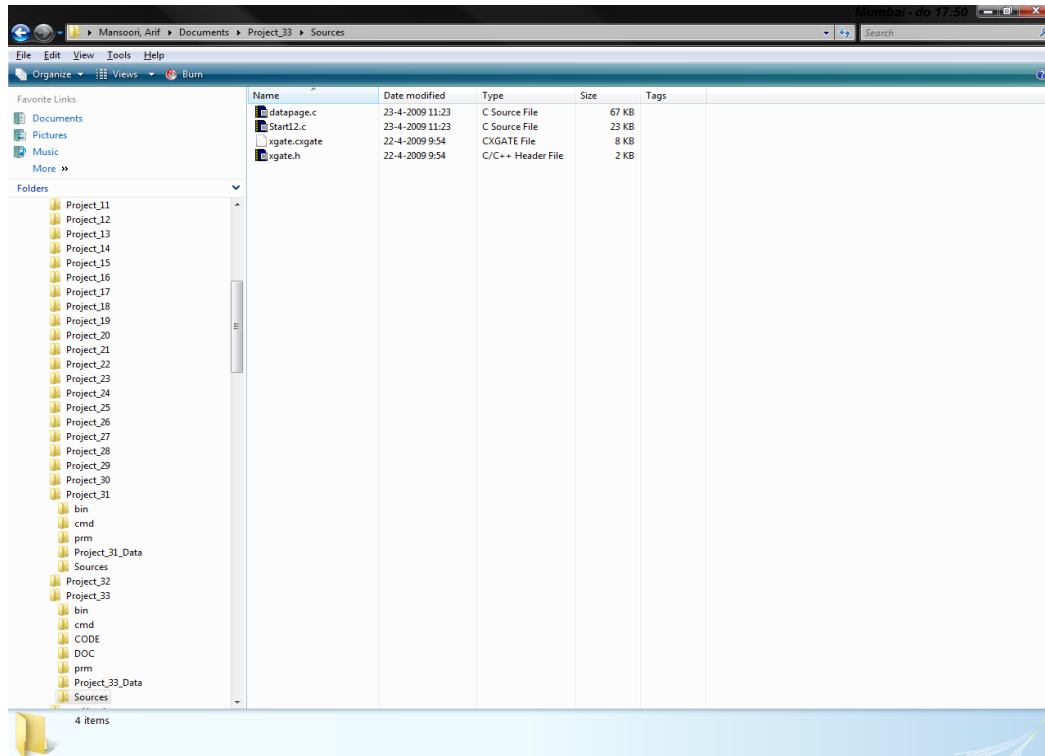
Click Open.



Click OK. The files are attached here for quick reference.

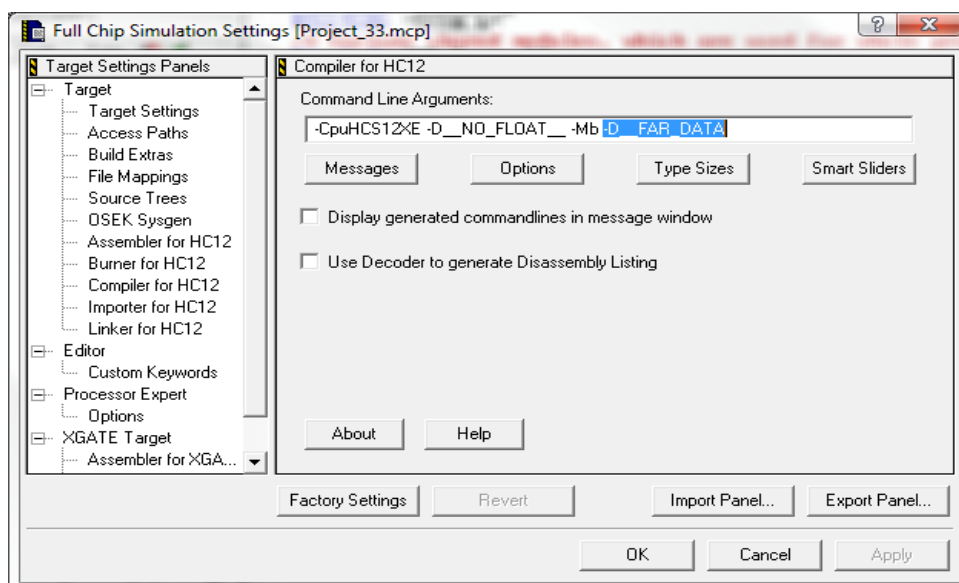


2. Copy the following two files   into the sources subdirectory in your project folder for e.g.



Then import these two files into the project. Use the same procedural tasks as in step 1. Instead of Libraries use the appropriate folder.

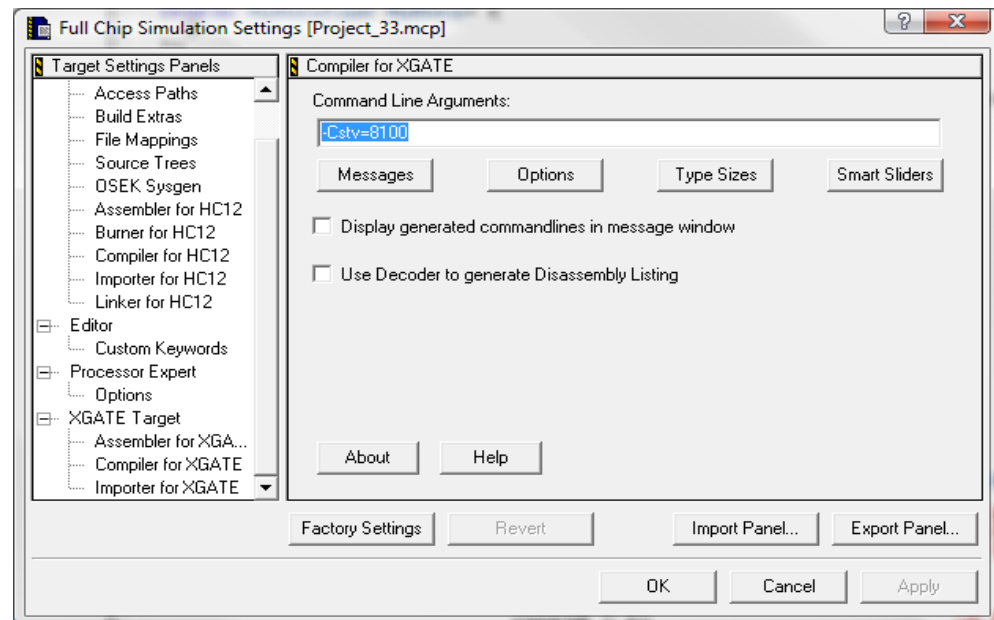
3. Press ALT+F7, compiler for HC12, add **-D__FAR_DATA** to the command line arguments. Do the same for all target settings (Full Chip Simulation or SofTec HCS12).



In Compiler for XGATE, add `-Cstv=8100` to the command line arguments. Do the same for all target settings (Full Chip Simulation or SofTec HCS12). This corresponds to the prm line,

```
RAM_XGATE_STK = READ_WRITE 0xF81000 TO 0xF810FF; /* The stack is set by the XGATE compiler option -Cstv=8100 */
```

in the project prm file.



4. `#include <string.h>`
`#include "xgate.h"` in your main file.

Use the attached prm files to replace the prm file created using processor expert. Depending on the target (Full Chip Simulation or SofTec HCS12) replace the project prm accordingly. Precede the `VECTOR 0 _Startup` statement in the prm file by `//`.

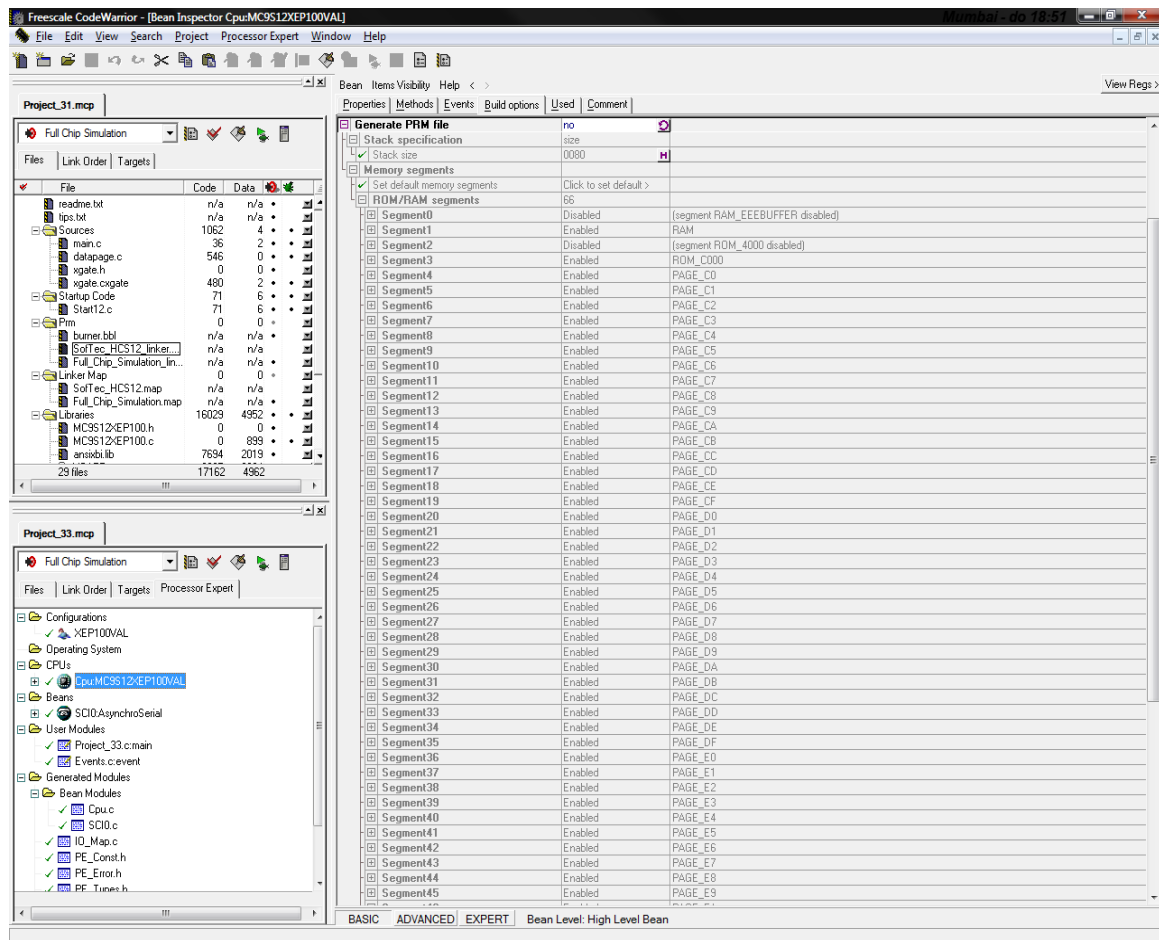


SofTec_HCS12_linker.prm



Full_Chip_Simulation_linker.prm

Disable prm generation in CPU bean inspector. Right click on CPU name and select CPU Inspector. Select 'no' for Generate PRM file.



5. Insert the following piece of code in the project main file just before the definition of the main function,

/ this variable definition is to demonstrate how to share data between XGATE and S12X */*

```
#pragma DATA_SEG SHARED_DATA
volatile int shared_counter; /* volatile because both cores are accessing it. */
#pragma DATA_SEG DEFAULT
```

```
#define ROUTE_INTERRUPT(vec_adr, cfdata) \
INT_CFADDR= (vec_adr) & 0xF0;           \
INT_CFDATA_ARR[((vec_adr) & 0x0F) >> 1]= (cfdata)
```

```
#define SOFTWARETRIGGER0_VEC 0x72 /* vector address= 2 * channel id */
static void SetupXGATE(void);
```

Insert following code after the end of the main function,

```
static void SetupXGATE(void)
{
    /* initialize the XGATE vector block and set the XGVBR register to its start address */
    XGVBR = (unsigned int)(void* __far)(XGATE_VectorTable - XGATE_VECTOR_OFFSET);

    /* switch software trigger 0 interrupt to XGATE */
}
```

```

ROUTE_INTERRUPT(SOFTWARETRIGGER0_VEC, 0x81); /* RQST=1 and PRIO=1 */

/* enable XGATE mode and interrupts */
XGMCTL= 0xFBC1; /* XGE | XGFRZ | XGIE */

/* force execution of software trigger 0 handler */
XGSWT= 0x0101;
}

```

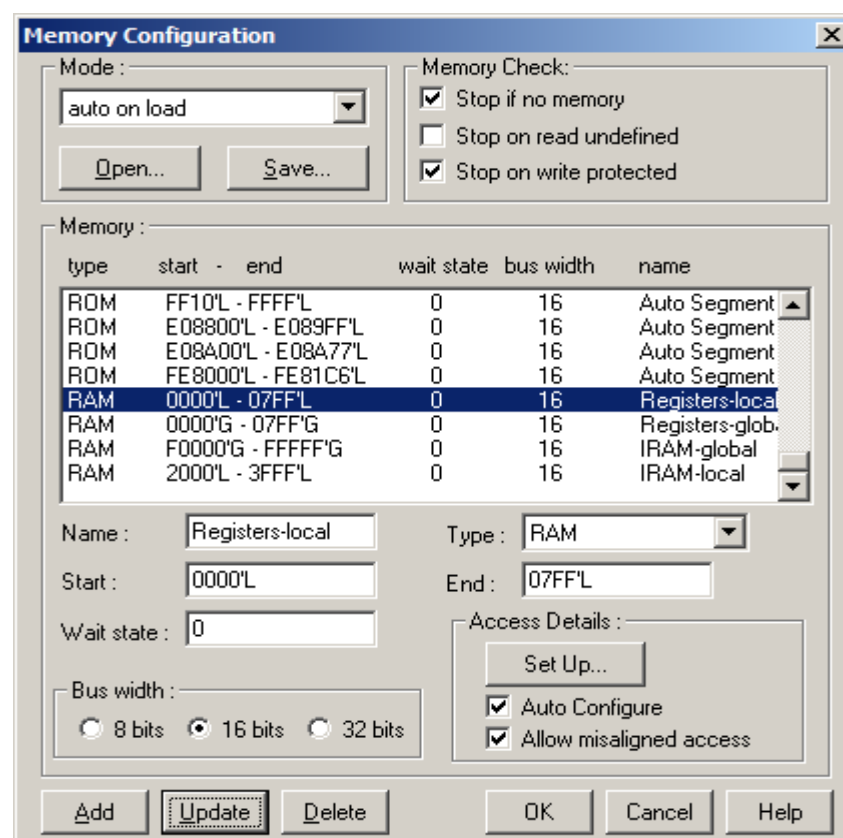
The function SetupXGATE must be called from within the main function immediately after the low level initialization function is called. Enable all interrupts following this.

- Press F5 to call debugger, in debugger menu "Component", "Open", "Source" to open a new source file window, in the new window, right click mouse and select "show core", "xgate". This window is used to show your current xgate code running state. This same procedure applies to other components like assembly, memory, registers, etc.
- If you use simulator then it does not contains defined memory map. It looks like problem of the Code Warrior. I have reported it to the CodeWarrior expert group. In the picture below you can see 4 memory spaces I have added to the Memory Configuration to be able to simulate. In this way you can add entire space – see attached memory map or compare with debugging memory map in the debugger when you use Softec target.

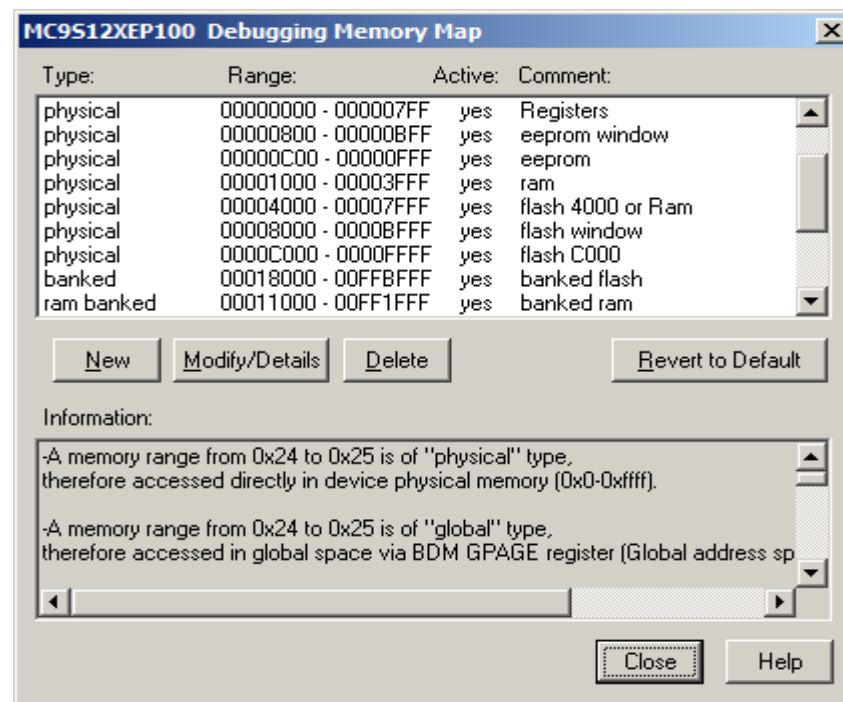


Default.mem)

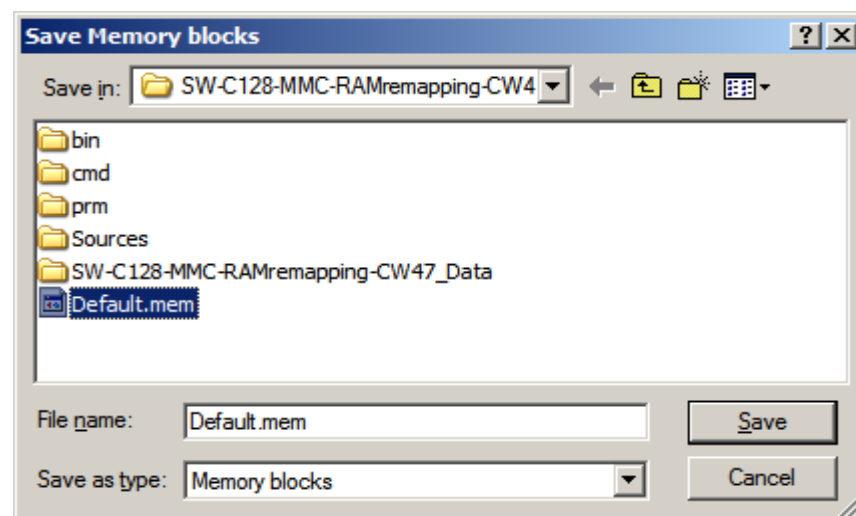
Memory configuration in simulator (file attached for quick reference



Memory configuration in Softec target:



If you change configuration do not forget to save it.



- How to save the current debugger window configuration: go to debugger menu "File", "save configuration". When you open debugger next time, the layout of the debugger windows will be same.
- Enhance your code to configure other interrupts that can be handled by XGATE. Use `ROUTE_INTERRUPT(vector number, 0x81)` function to route interrupts to XGATE.
- Add an ISR (for e.g. myfunction) in file `xgate.cxgate` with the template,


```
interrupt void myfunction(int dataptr)
{
    User code...
}
```

11. In the vector table for XGATE (XGATE_VectorTable) in file xgate.cxgate replace the function name for the associated channel id with 'myfunction' and optionally a data pointer type casted to an int.
12. Repeat steps 8 – 10 for other interrupts routed to XGATE and their associated ISRs.