

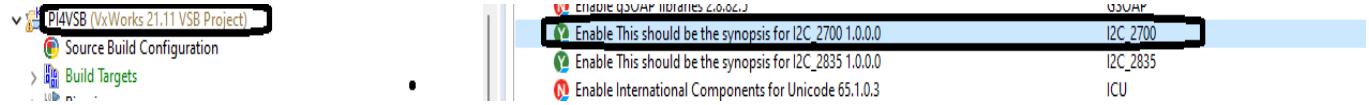


Déployer un pilote

Sous vxworks Raspberry

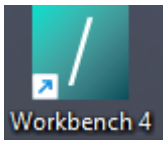
PI4

La première étape consiste à intégrer notre pilote dans le source build de la PI4 (PI4VSB)

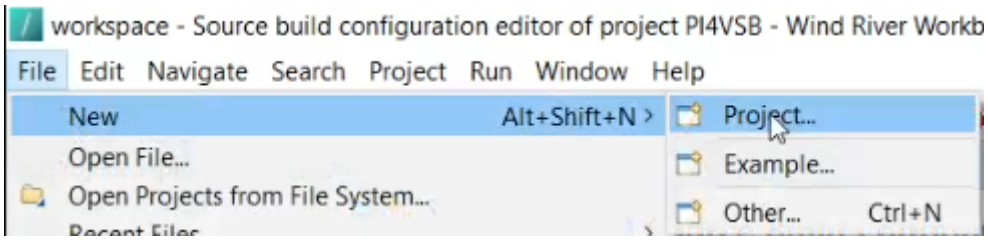


1) Création d'un projet sous Workbench

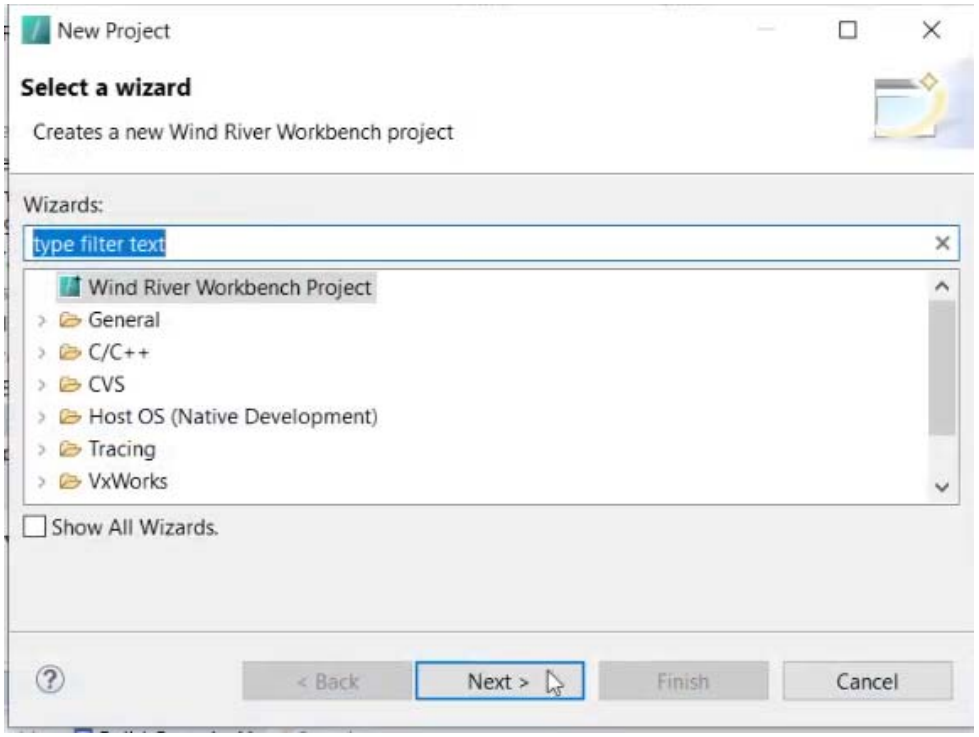
Lancez le programme workbench

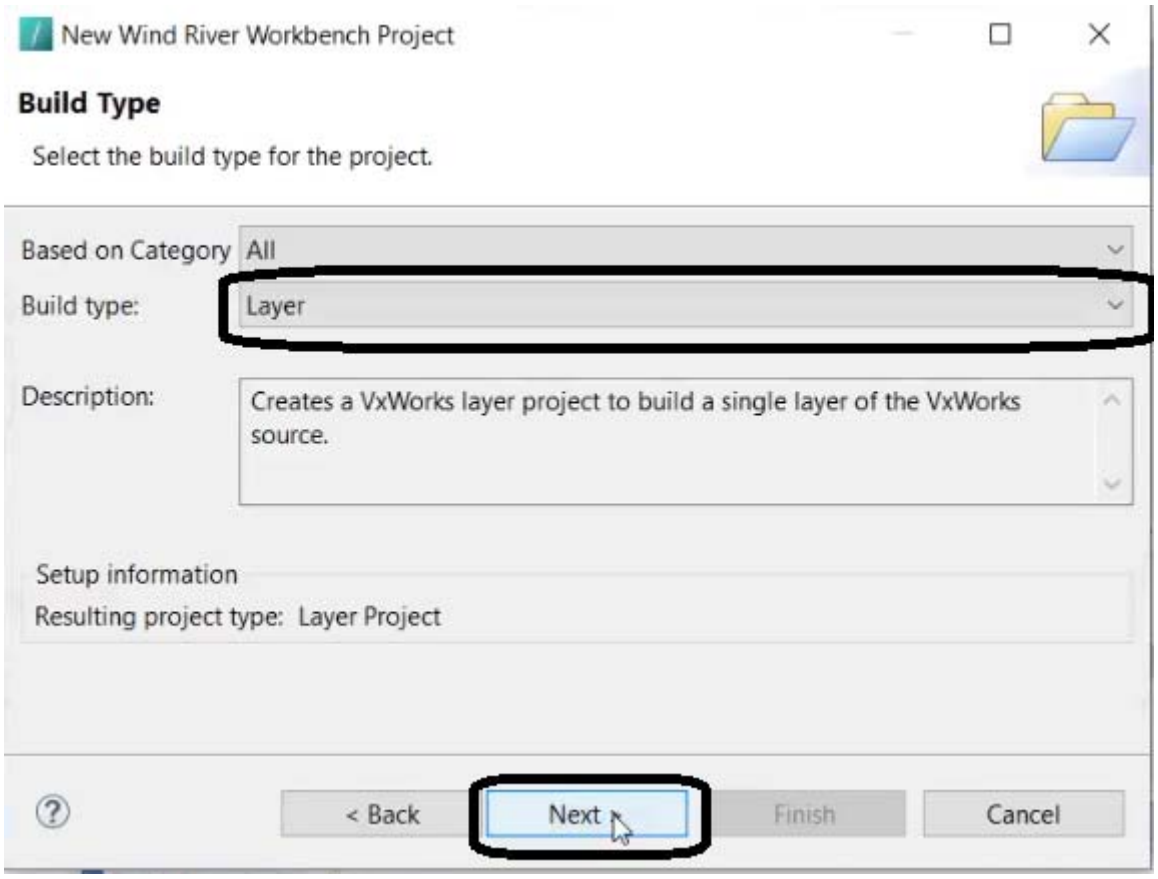


Il va falloir intégrer une nouvelle couche dans le PI4VSB
Créez un nouveau projet du type Layer

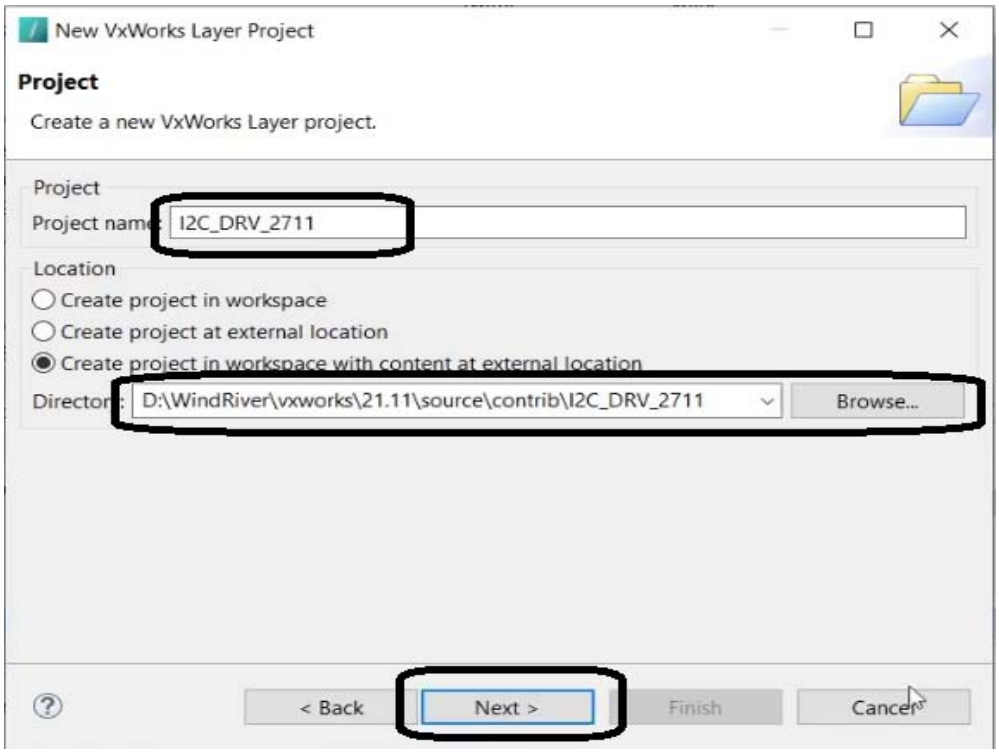


Puis cliquez sur Next

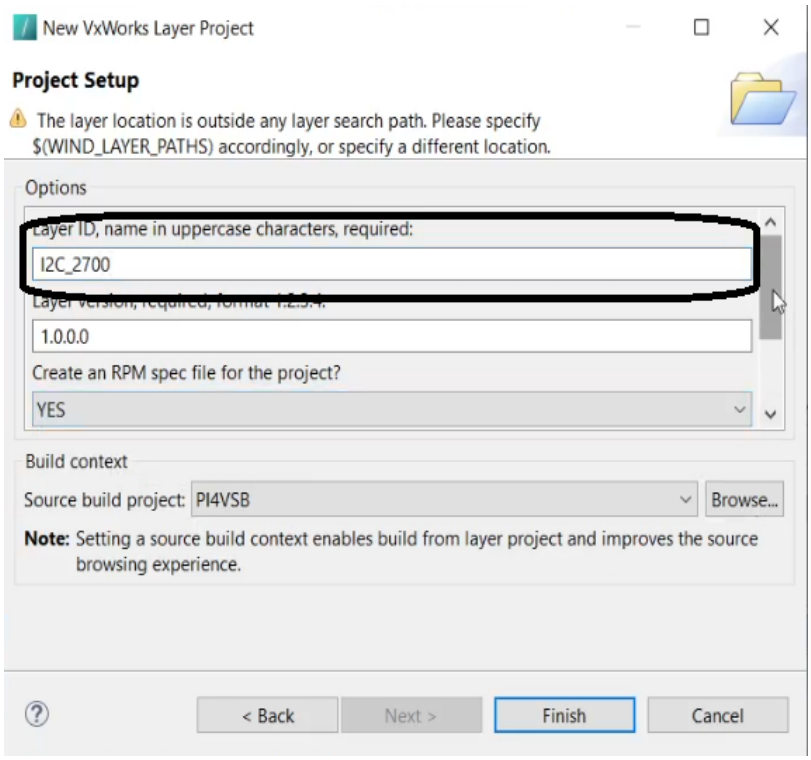




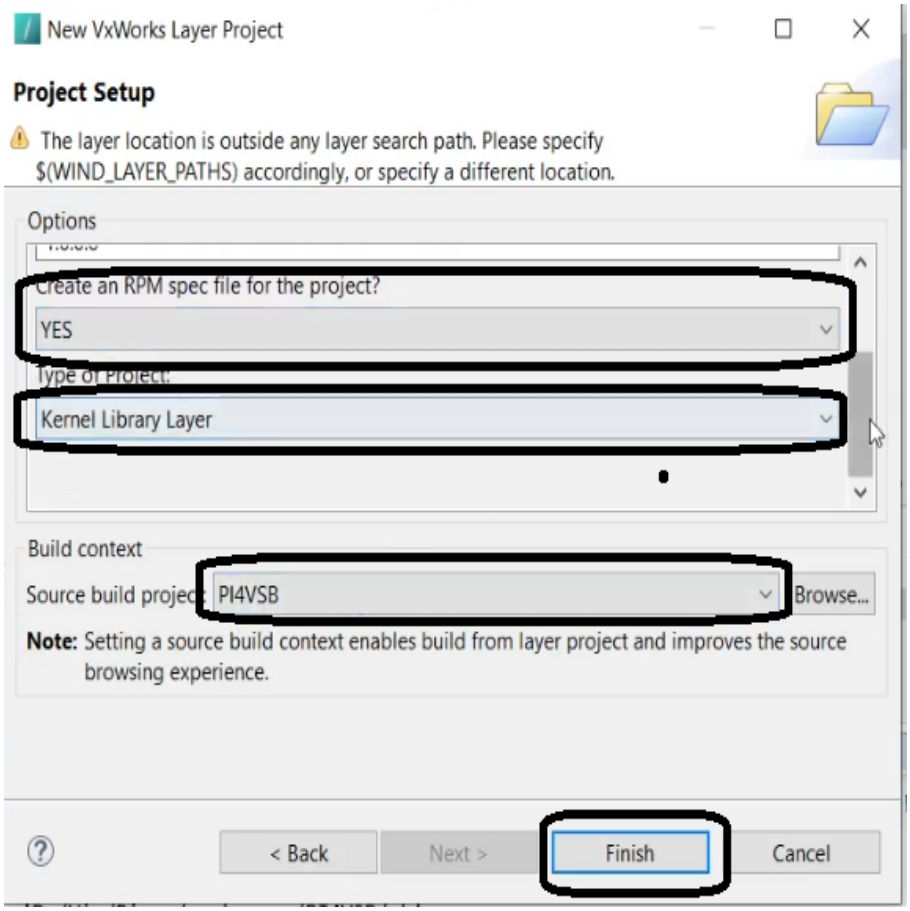
Donnez un nom à votre projet et sélectionner le chemin ou se trouve le répertoire de votre futur projet



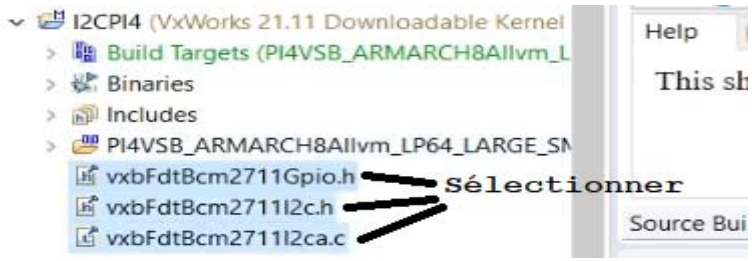
Donnez un nom à la couche pour PI4VSB



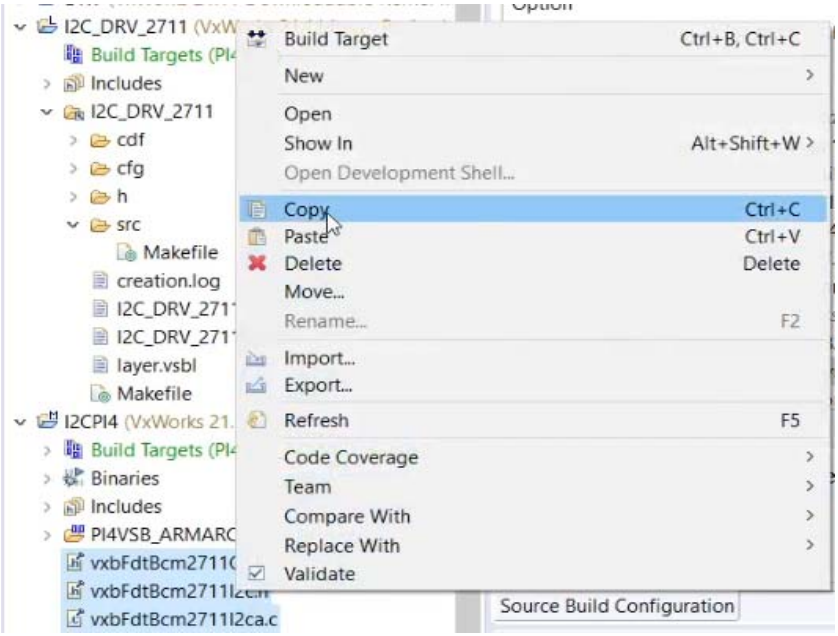
Sélectionner les éléments suivants :



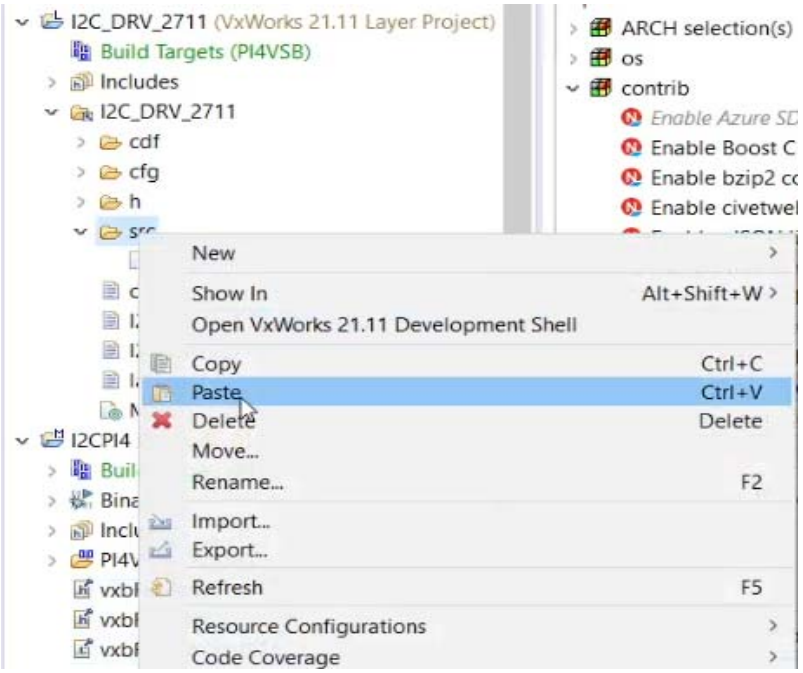
Déplacer les codes sources de votre répertoire de travail vers le répertoire du projet :

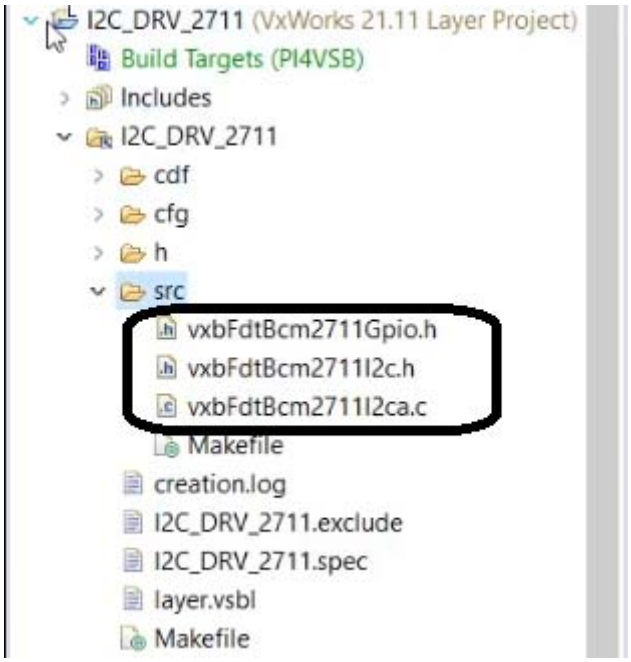


Faire un click droit sur le projet I2CPI4 puis copy

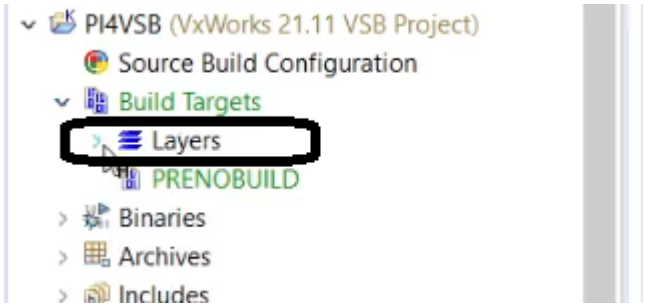


Puis coller ces fichiers dans le répertoire src du projet I2C_DRV_2711

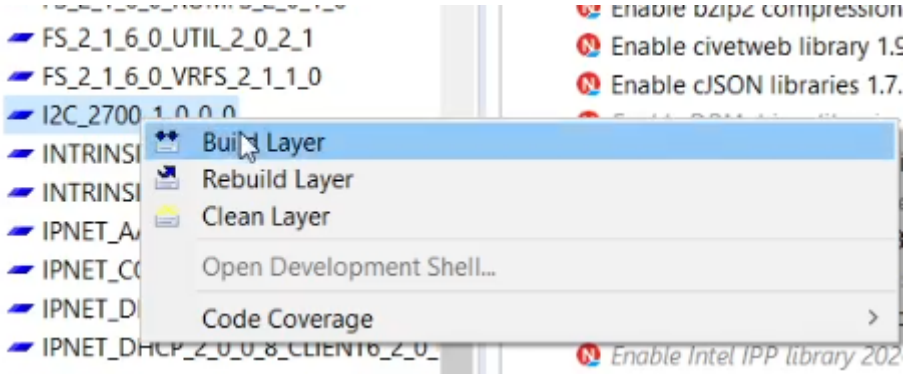




Dans PI4VSB dérouler le menu layers :



Sélectionner notre couche et build layer

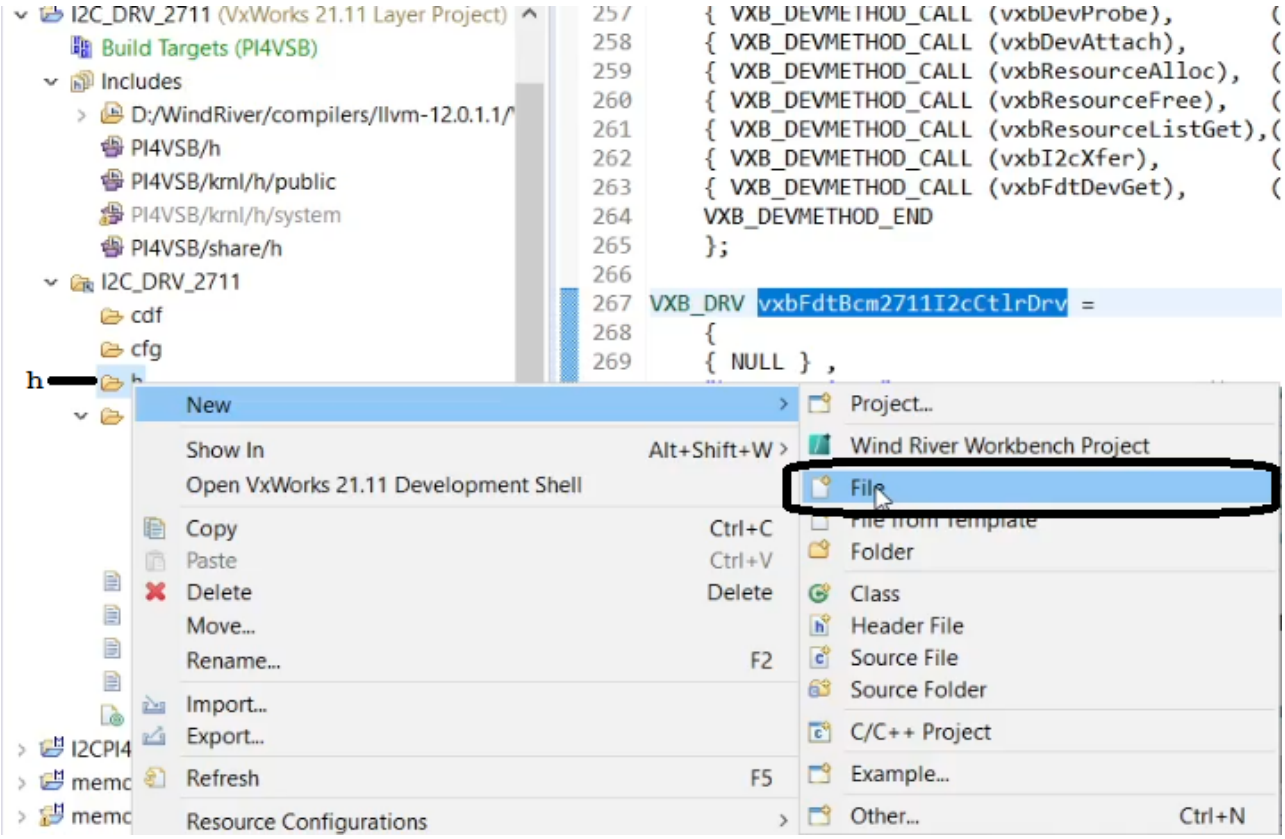


Laisser la compilation se terminer

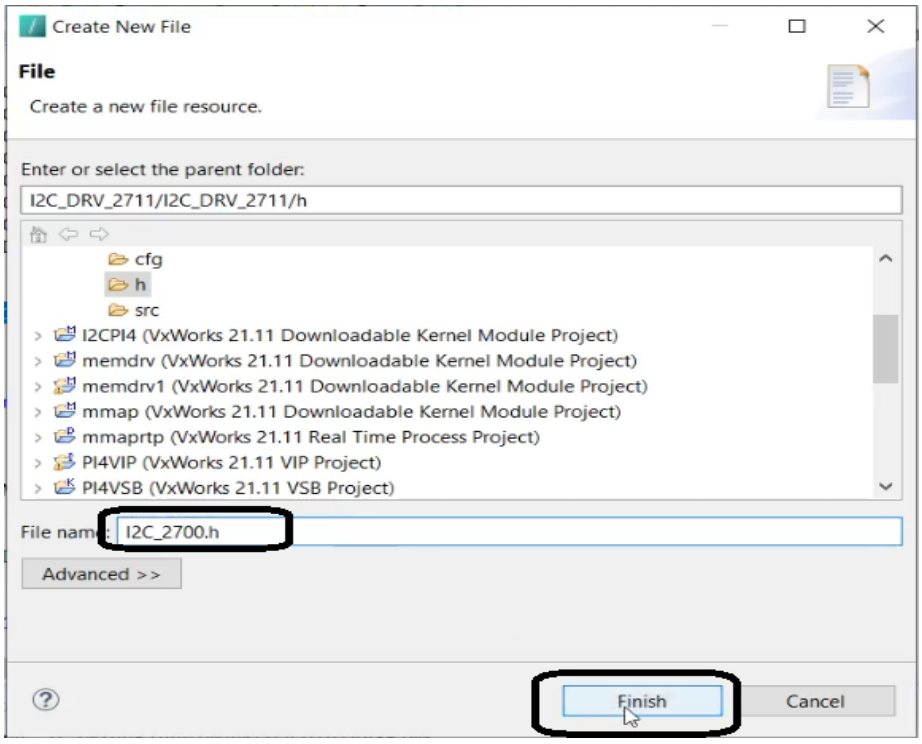
```
rm -f D:/WindRiver/workspace/PI4VSB/mk/cleaned.d
POSTBUILD done
make[1]: Leaving directory 'D:/WindRiver/workspace/PI4VSB/mk'
rm -f D:/WindRiver/workspace/PI4VSB/mk/vsb_build_lock
done all
Doubleclick here to scan Project 'PI4VSB' for errors.
Build Finished in Project 'PI4VSB':   2022-02-24 16:11:03   (Elapsed Time: 00:25)
```

La seconde étape est intégré le pilote dans le Image project de Vxworks (VIP)

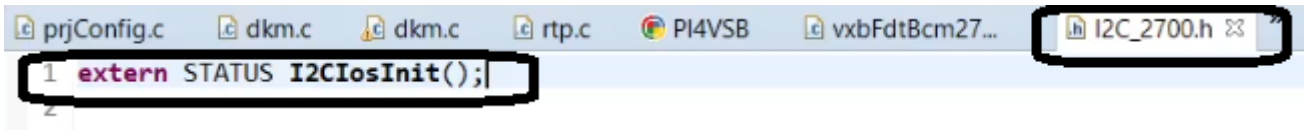
Dans les fichiers d'entête de notre projet créez un nouveau fichier :



Le nom du fichier doit correspondre au nom de la couche choisie :



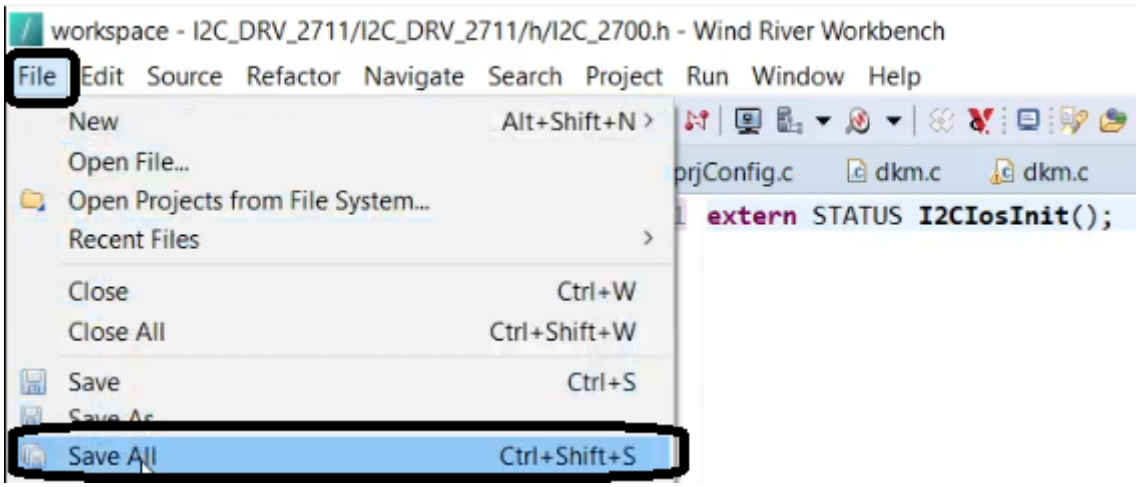
Complétez ce fichier avec le lien vers la fonction externe



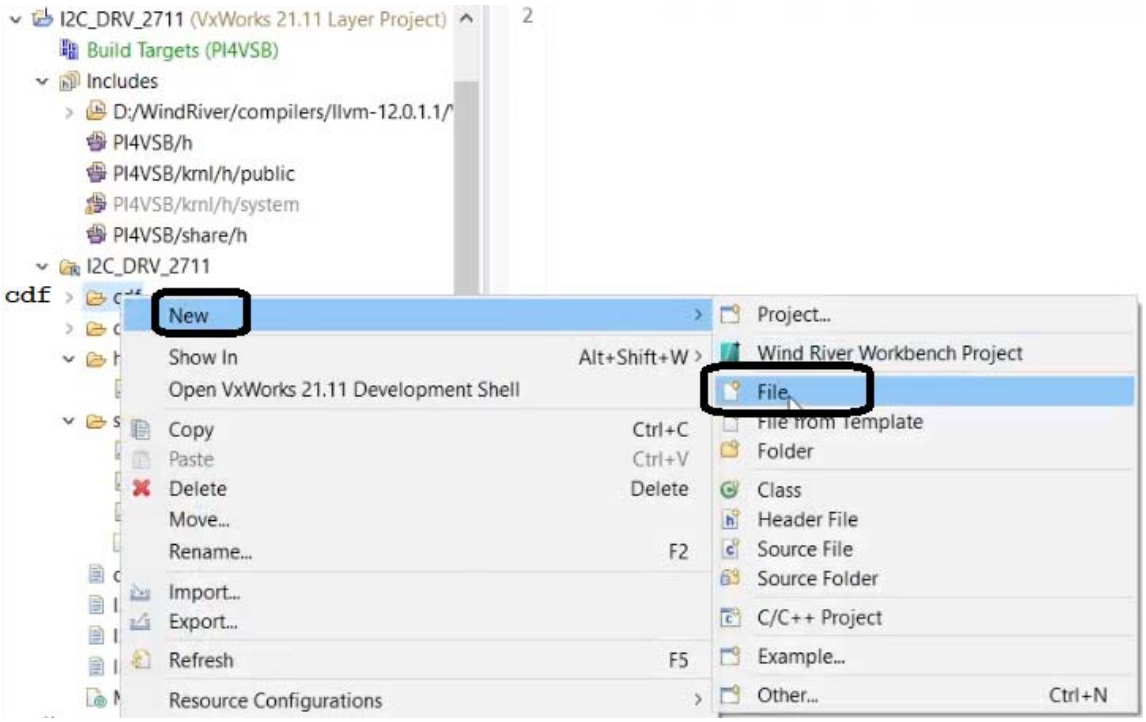
La fonction I2CIOSInit met en place le pilote pour les entrées sorties :

```
STATUS I2CIosInit() {
/* Add the I/O driver */
      I2CIosDrvNum = iosDrvInstall_(
(DRV_CREATE_PTR) NULL,
(DRV_REMOVE_PTR) NULL,
(DRV_OPEN_PTR) I2CIosOpen,
(DRV_CLOSE_PTR) I2CIosClose,
(DRV_READ_PTR) I2CIosRead,
(DRV_WRITE_PTR) I2CIosWrite,
(DRV_IOCTL_PTR) I2CIosIoctl
);
if (I2CIosDrvNum == ERROR) {//numero de driver
return ERROR;
}
else
      return OK;
      }
```

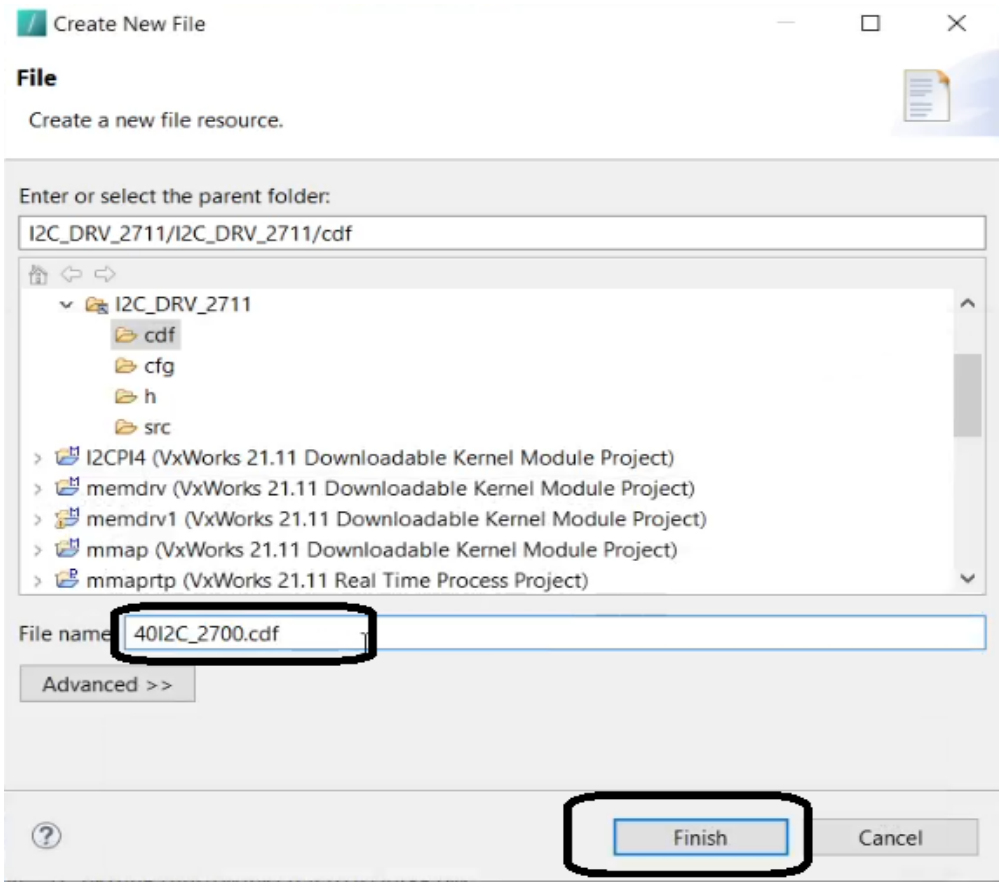
Sauvegardez votre fichier I2C_2700.h



Dans le répertoire cdf de notre projet créez un nouveau fichier :



Donnez lui un nom 40I2C_2700.cdf (40 pour les pilotes de périphérique):

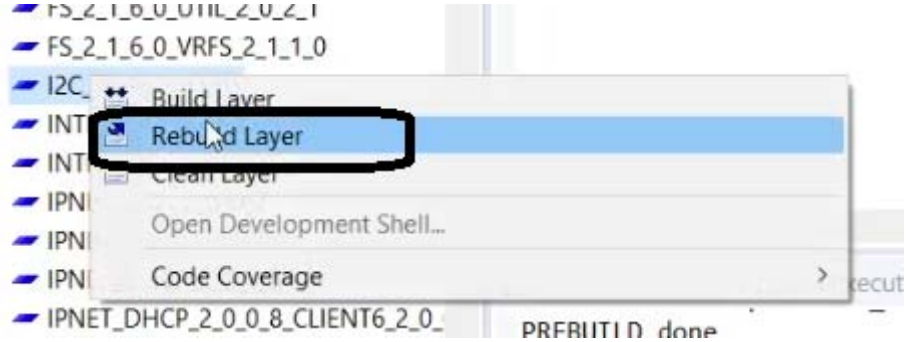


Complétez ce fichier avec les bonnes correspondances pour notre projet:

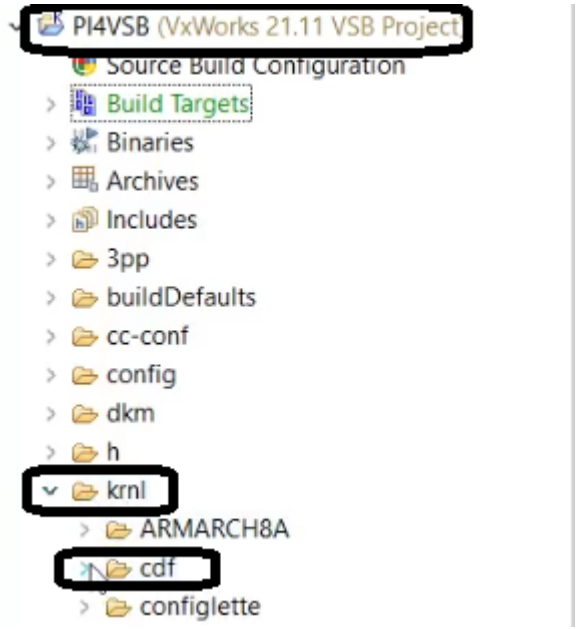
```
Component    DRV_I2C_2711 {
NAME         drivers I2C platform PI4
SYNOPSIS     Support for PI4
_CHILDREN    FOLDER_DRIVERS
INIT_RTN     I2CIosInit(); /*port driver *///-> point de départ du pilote pour fichier periph
_INIT_ORDER  usrRoot
INIT_AFTER   usrIosCoreInit
INIT_BEFORE  INCLUDE_VXBUS
HDR_FILES    I2C_2700.h
MODULES      vxbFdtBcm2711I2ca.o            //→ nom du fichier source objet à compiler
LINK_SYMS    vxbFdtBcm2711I2cCtlrDrv
REQUIRES     INCLUDE_VXBUS \
             INCLUDE_PCI_BUS \
             INCLUDE_IO_BASIC
}
```

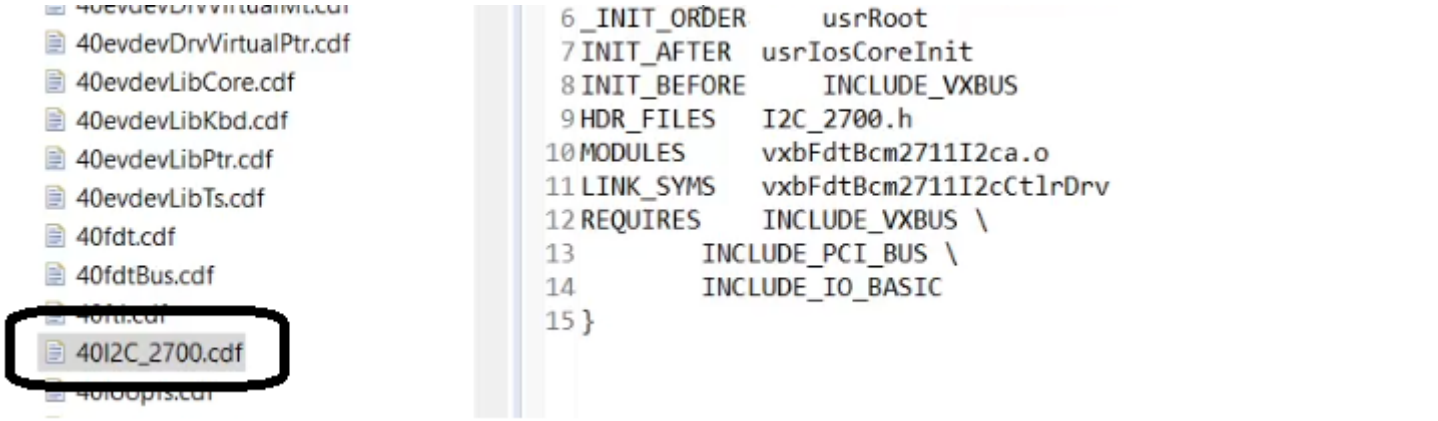
Puis sauvegardez le :

Reconstruire à nouveau la couche :

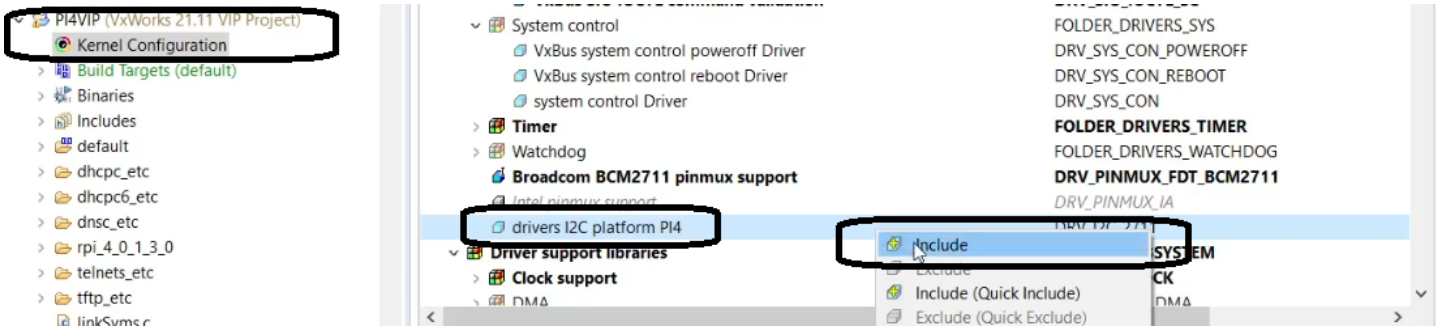


Attendre la fin de la compilation
Dans le projet PI4VSB répertoire krnl cdf vérifiez la présence de notre fichier 40I2C_2700.cdf

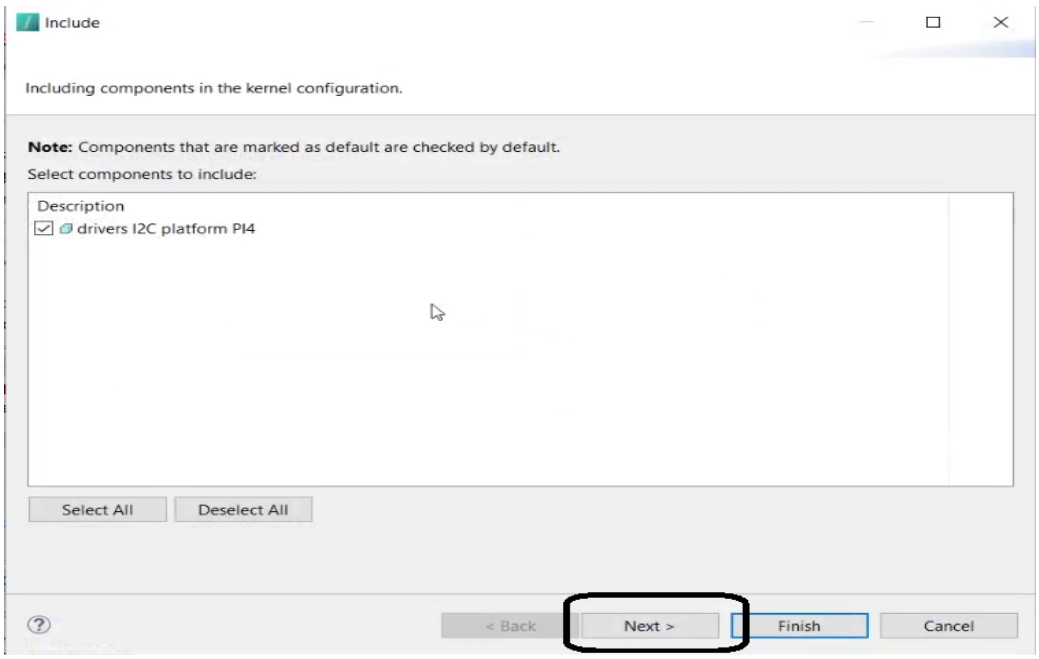


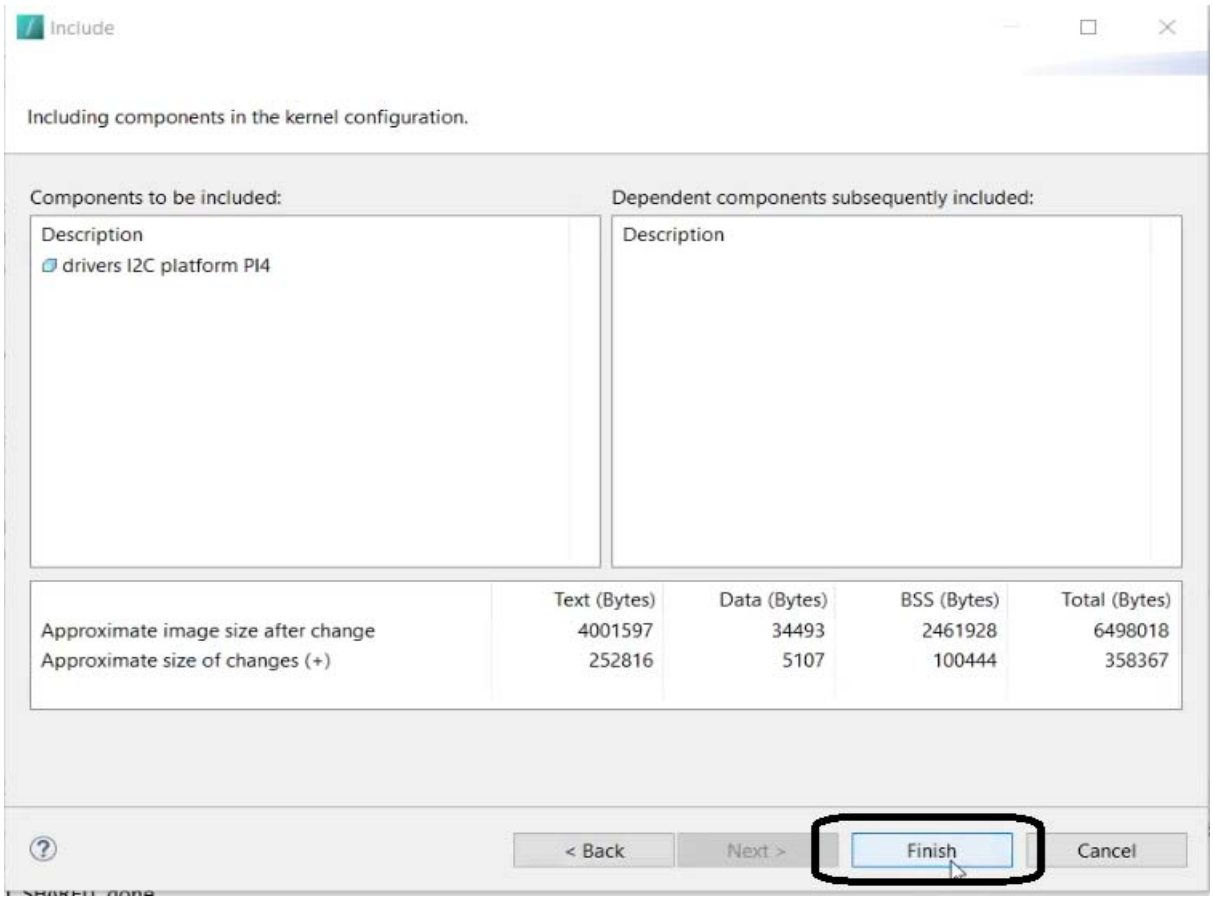


Dans PI4VIP Device drivers vérifiez la prise en compte de notre pilote :



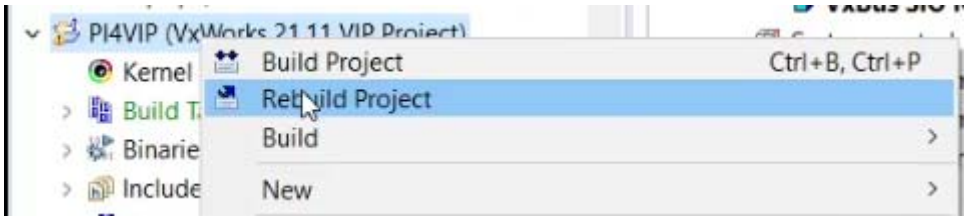
Inclure ce pilote dans le noyau Vxworks :







Relancez la compilation du noyau :



Attendre la fin de la compilation :

```
vxprj vip recalculate ../PI4VIP.wpj
VX_CPU_FAMILY=arm64 mkimage -O vxworks -T kernel -C none -n vxworks -A arm64 -a 0x00100000 -e 0x00100000 -d vxWorks.bin uVxWorks
Image Name:   vxworks
Created:      Thu Feb 24 16:40:54 2022
Image Type:   AArch64 VxWorks Kernel Image (uncompressed)
Data Size:    8566496 Bytes = 8365.72 KiB = 8.17 MiB
Load Address: 00100000
Entry Point:  00100000
rm rpi-4b.dts.i versionTag
make[1]: Leaving directory 'D:/WindRiver/workspace/PI4VIP/default'
make: built targets of D:/WindRiver/workspace/PI4VIP
Build Finished in Project 'PI4VIP':   2022-02-24 16:40:54   (Elapsed Time: 00:24)
```

Le noyau est prêt à prendre notre nouveau pilote

Pour le temporisateur qui ne possède pas de fichier de périphérique on a :

```
extern STATUS vxbBcm2711TimerDrvAdd();
```

```
Component    TIMER_I2C_2711 {
NAME         drivers SYSTIMER platform PI4
SYNOPSIS     Support for PI4
_CHILDREN    FOLDER_DRIVERS
INIT_RTN     vxbBcm2711TimerDrvAdd(); /*port driver *// point de départ du pilote
_INIT_ORDER  usrRoot
INIT_AFTER   usrIosCoreInit
INIT_BEFORE  INCLUDE_VXBUS
HDR_FILES    DRIVERS_2711.h
MODULES      TIMER.o            //→ nom du fichier source objet à compiler
LINK_SYMS    vxbFdtBcmSytTimerDrva
REQUIRES     INCLUDE_VXBUS \
             INCLUDE_PCI_BUS \
             INCLUDE_IO_BASIC
}
```

```
STATUS vxbBcm2711TimerDrvAdd() {

return vxbDrvAdd(&vxbFdtBcmSytTimerDrva);
}

VXB_DRV vxbFdtBcmSytTimerDrva =
    {
    { NULL } ,
    "bcm2837-system-timera",                          /* Name */
    "Broadcom System Timer FDT driver",              /* Description */
    VXB_BUSID_FDT,                                   /* Class */
    0,                                               /* Flags */
    0,                                               /* Reference count */
    &fdtBcmSytTimerMethodList[0]                     /* Method table */
    };

VXB_DRV_DEF(vxbFdtBcmSytTimerDrva)
```