1. Tìm ra stub function -> local function
2. Tìm ra globlal variable (kiểm tra các giá trị input và output của các hàm nhỏ xem có lấy ở global không)
3. Tìm ra input parameter và return value

//Trong r\_impdrv\_api.c

e\_impdrv\_errorcode\_t R\_IMPDRV\_AttrSetClBrkAddr(

    impdrv\_ctrl\_handle\_t            handle,

    const st\_impdrv\_core\_info\_t     \*const p\_core\_info,

    const uintptr\_t                 cl\_brk\_addr

)

có hàm

impdrv\_api\_chk\_core\_info (\*p\_core\_info) //nếu error code = OK  
impdrv\_api\_cnv\_ctrl\_handle (handle, &p\_impdrv\_ctl)//nếu error code= OK, gán handle vào p\_impdrv\_ctl

impdrv\_genctl\_set\_cl\_brk\_addr (



            p\_impdrv\_ctl,

            p\_core\_info,

            (const uint32\_t)cl\_brk\_addr

        ); //hàm xử lý chính

impdrv\_api\_cnv\_error\_code (ercd) //chuyển error code về giá trị khác nếu ko OK

3 hàm kia chỉ đơn giản nhận input từ hàm **R\_IMPDRV\_AttrSetClBrkAddr** và trả về error code, ta tập trung vào hàm

**impdrv\_genctl\_set\_cl\_brk\_addr** :

Trong đây có :

impdrv\_genctl\_prologue

impdrv\_genctl\_chk\_core\_info

impdrv\_cmnctl\_mutex\_lock  
 impdrv\_osdep\_mutex\_lock //lock mutex  
 impdrv\_cmnctl\_test\_checksum // kiểm tra checksum  
 impdrv\_cmnctl\_reg\_test\_checksum //kiểm tra checksum chứa biến global g\_hwrsc\_mng\_table  
 impdrv\_osdep\_mutex\_unlock //unlock mutex

trong đây sử dụng hàm có truyền vào biến global variable **g\_impdrv\_cmn\_ctl\_ins**, đây là struct chứa **mutex\_handle** và **mutex\_time\_period**

gọi hàm **R\_OSAL\_ThsyncMutexLockForTimePeriod** , đây là stub function, sẽ đưa vào trong file excel

impdrv\_genctl\_epilogue

impdrv\_cmnctl\_mutex\_unlock //kiểm tra trước khi unlock mutex

impdrv\_osdep\_mutex\_unlock //unlock mutex

gọi hàm **R\_OSAL\_ThsyncMutexUnlock**, đây là stub function, sẽ đưa vào trong file excel

impdrv\_cmnctl\_get\_corectl\_func (p\_core\_info->core\_type) //lấy function trong array

trong đây có định nghĩa 1 func pointer

typedef struct

{

    bool (\*p\_impdrvCorectlIsValidCore)(

        const uint32\_t  core\_num

    );  /\*\*< Function table for distinguishing each core. Valid core.   \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlCheckState)(

        const st\_impdrv\_corectl\_t   \*const p\_corectl,

        const uint32\_t              core\_num,

        const e\_impdrv\_state\_t      state

    );  /\*\*< Function table for distinguishing each core. Check state.  \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlSetState)(

        st\_impdrv\_corectl\_t         \*const p\_corectl,

        const e\_impdrv\_state\_t      state

    );  /\*\*< Function table for distinguishing each core. Set state.    \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlInitStart)(

        st\_impdrv\_coretypectl\_t     \*const p\_coretypectl,

        const uint32\_t              core\_num

    );  /\*\*< Function table for distinguishing each core. Init start.   \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlInitEnd)(

        st\_impdrv\_coretypectl\_t     \*const p\_coretypectl,

        const uint32\_t              core\_num

    );  /\*\*< Function table for distinguishing each core. Init end.     \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlStart)(

        st\_impdrv\_coretypectl\_t     \*const p\_coretypectl,

        const uint32\_t              core\_num

    );  /\*\*< Function table for distinguishing each core. Start core control. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlStop)(

        st\_impdrv\_coretypectl\_t     \*const p\_coretypectl,

        const uint32\_t              core\_num

    );  /\*\*< Function table for distinguishing each core. Stop core control. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlAttInit)(

        st\_impdrv\_corectl\_t         \*const p\_corectl,

        const uint32\_t              core\_num

    );  /\*\*< Function table for distinguishing each core. Attribute initialize. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlSetMemInit)(

        st\_impdrv\_corectl\_t     \*const p\_corectl,

        const uint32\_t          core\_num,

        const e\_impdrv\_param\_t  enable

    );  /\*\*< Function table for distinguishing each core. Set core memory initialize.   \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlSetCoreMap)(

        st\_impdrv\_corectl\_t     \*const p\_corectl,

        const uint32\_t          core\_num,

        const uint8\_t           syncc\_val[IMPDRV\_COREMAP\_MAXID]

    );  /\*\*< Function table for distinguishing each core. Set sync core map.    \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlSetCl)(

        st\_impdrv\_corectl\_t     \*const p\_corectl,

        const uint32\_t          core\_num,

        const uint32\_t          claddr\_phys

    );  /\*\*< Function table for distinguishing each core. Set CL address.   \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlSetIrqMask)(

        st\_impdrv\_corectl\_t     \*const p\_corectl,

        const uint32\_t          core\_num,

        const bool              irq\_mask[IMPDRV\_IRQMASK\_MAX]

    );  /\*\*< Function table for distinguishing each core. Set IRQ mask setting. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlExecute)(

        st\_impdrv\_corectl\_t     \*const p\_corectl,

        const uint32\_t          core\_num,

        const p\_impdrv\_cbfunc\_t callback\_func,

        void                    \*const p\_callback\_args

    );  /\*\*< Function table for distinguishing each core. Execute core control. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlIntHandler)(

        st\_impdrv\_corectl\_t     \*const p\_corectl,

        const uint32\_t          core\_num

    );  /\*\*< Function table for distinguishing each core. int handler imp control. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlResumeExe)(

        st\_impdrv\_corectl\_t     \*const p\_corectl,

        const uint32\_t          core\_num

    );  /\*\*< Function table for distinguishing each core. Resume core control. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlPmSetPolicy)(

        st\_impdrv\_coretypectl\_t        \*const p\_coretypectl,

        const uint32\_t                  core\_num,

        const e\_impdrv\_pm\_policy\_t      policy

    );  /\*\*< Function table for distinguishing each core. Pm set policy core control. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlPmGetPolicy)(

        st\_impdrv\_coretypectl\_t         \*const p\_coretypectl,

        const uint32\_t                  core\_num,

        e\_impdrv\_pm\_policy\_t            \*const p\_policy

    );  /\*\*< Function table for distinguishing each core. Pm get policy core control. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlSetDsp)(

        st\_impdrv\_corectl\_t         \*p\_corectl,

        const uint32\_t              core\_num,

        const st\_impdrv\_dsp\_data\_t  \*const p\_dsp\_app,

        const st\_impdrv\_dsp\_data\_t  \*const p\_dsp\_fw,

        const st\_impdrv\_dsp\_data\_t  \*const p\_dsp\_data,

        const st\_impdrv\_dsp\_data\_t  \*const p\_dsp\_dtcm

    );  /\*\*< Function table for distinguishing each core. Pm get policy core control. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlBusIfCheck)(

        st\_impdrv\_coretypectl\_t         \*const p\_coretypectl,

        const uint32\_t                  core\_num,

        const st\_impdrv\_chk\_resource\_t  \*const p\_chk\_resource

    );  /\*\*< Function table for distinguishing each core. Interface bus check of core control. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlConfRegCheck)(

        const st\_impdrv\_coretypectl\_t   \*const p\_coretypectl,

        const uint32\_t                  core\_num,

        const st\_impdrv\_chk\_param\_t     \*const p\_chk\_param,

        const uint32\_t                  param\_num

    );  /\*\*< Function table for distinguishing each core. Configuration register check of core control. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlSetClBrkAddr)(

        st\_impdrv\_corectl\_t     \*const p\_corectl,

        const uint32\_t          core\_num,

        const uint32\_t          cl\_brk\_addr

    );  /\*\*< Function table for distinguishing each core. Set CL break address.   \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlSetGosubCond)(

        st\_impdrv\_corectl\_t         \*const p\_corectl,

        const uint32\_t              core\_num,

        const e\_impdrv\_gosub\_cond\_t condition

    );  /\*\*< Function table for distinguishing each core. Set Conditional GOSUB instruction attribute.  \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlPmGetState)(

        st\_impdrv\_coretypectl\_t     \*const p\_coretypectl,

        const uint32\_t              core\_num,

        e\_impdrv\_pm\_state\_t         \*const p\_pmstate

    );  /\*\*< Function table for distinguishing each core. Pm get state core control. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlQuit)(

        const uint32\_t              core\_num

    );  /\*\*< Function table for distinguishing each core. Quit core control. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlSyncStart)(

        st\_impdrv\_coretypectl\_t coretype\_ctl[IMPDRV\_CORE\_TYPE\_MAX],

        const uint32\_t          core\_num

    );  /\*\*< Function table for distinguishing each core. Starts controlling the core in sync. \*/

    e\_impdrv\_errorcode\_t (\*p\_impdrvCorectlSyncStop)(

        st\_impdrv\_coretypectl\_t coretype\_ctl[IMPDRV\_CORE\_TYPE\_MAX],

        const uint32\_t          core\_num

    );  /\*\*< Function table for distinguishing each core. Stop controlling the core in sync. \*/

} st\_impdrv\_corectl\_func\_t;

Ở đây gồm 26 function pointer (chứa địa chỉ của func) trong struct này

Bên cạnh đó hàm này thực thi truyền vào 26 function pointer vào trong struc **func,** ví dụ khi truyền core\_type là IMP thì sẽ nhận được 26 function sau vào struct **func**

/\*\* IMPDRV\_CORE\_TYPE\_IMP \*/

{   impdrv\_impctl\_is\_valid\_core,    impdrv\_impctl\_check\_state,      impdrv\_impctl\_set\_state,

    impdrv\_impctl\_init\_start,       impdrv\_impctl\_init\_end,         impdrv\_impctl\_start,

    impdrv\_impctl\_stop,             impdrv\_impctl\_attr\_init,        impdrv\_impctl\_set\_mem\_init,

    impdrv\_impctl\_set\_core\_map,     impdrv\_impctl\_set\_cl,           impdrv\_impctl\_set\_irq\_mask,

    impdrv\_impctl\_execute,          impdrv\_impctl\_int\_handler,      impdrv\_impctl\_resume\_exe,

impdrv\_impctl\_pm\_set\_policy,    impdrv\_impctl\_pm\_get\_policy,    NULL,

    NULL,                           NULL,                           impdrv\_impctl\_set\_cl\_brk\_addr,

    impdrv\_impctl\_set\_cond\_gosub,   impdrv\_impctl\_pm\_get\_state,     impdrv\_impctl\_quit,

    NULL,                           NULL

},

Sau khi sao chép xong thì hàm sẽ lấy đúng vị trí **p\_impdrvCorectlSetClBrkAddr** để thực thi (như ta thấy trong struct thì hàm này nằm ở vị trí 19 tính từ 0)

ercd = p\_funcs->p\_impdrvCorectlPmSetPolicy(…)

tức nó đang thực thi (chỉ khi array chép vào ko NULL)

impdrv\_impctl\_set\_cl\_brk\_addr //set CL brake address

bao gồm:

impdrv\_cnnctl\_is\_valid\_core //kiểm tra số core có hợp lệ rồi return

impdrv\_impctl\_check\_state //kiểm tra core state (phải là ready hoặc interrupt)

impdrv\_impctl\_is\_valid\_core //kiểm tra core number (nhỏ hơn 2 -> true)

Nếu error code OK thì gán **cl\_brk\_addr** (truyền vào lúc đầu) vào **p\_corectl->param[IMPDRV\_IMP\_EXEPARAM\_CLBRK]** :output parameter

XỬ LÝ CÁC GIÁ TRỊ

Sau khi **handle** ko **INVALID** và **p\_core\_info** ko NULL thì tiếp tục

if ((IMPDRV\_CLBRK\_ADDR\_INVALID != cl\_brk\_addr)

                && (0U != (cl\_brk\_addr & IMPDRV\_CHK\_CL\_MEM\_ALIGN)))

1. Dòng này cho thấy Nó không được bằng 0xFFFFFFFFU (4294967295 trong hệ thập phân).
2. Hai bit cuối cùng của nó không được là 00 (tức là, nó không chia hết cho 4).

Nếu cả 2 đk này thỏa thì lỗi -> **cl\_brk\_addr** phải chia hết cho 4

if ((IMPDRV\_CORE\_TYPE\_IMP != core\_info.core\_type)

        && (IMPDRV\_CORE\_TYPE\_IMP\_SLIM != core\_info.core\_type)

        && (IMPDRV\_CORE\_TYPE\_OCV != core\_info.core\_type)

        && (IMPDRV\_CORE\_TYPE\_DMAC != core\_info.core\_type)

        && (IMPDRV\_CORE\_TYPE\_DMAC\_SLIM != core\_info.core\_type)

        && (IMPDRV\_CORE\_TYPE\_PSCEXE != core\_info.core\_type)

        && (IMPDRV\_CORE\_TYPE\_CNN != core\_info.core\_type)

        && (IMPDRV\_CORE\_TYPE\_DSP != core\_info.core\_type))

    {

        ercd = IMPDRV\_EC\_NG\_PARAM;

    }

else if (IMPDRV\_CORE\_NUM\_MAX <= core\_info.core\_num)

    {

        ercd = IMPDRV\_EC\_NG\_PARAM;

    }

**Core\_type** phải nằm trong những phần này, ko thì lỗi. Đúng **core\_type** thì **core\_num** phải nhỏ hơn 8

Tới đây error code = OK thì gán **handle** vào **&p\_impdrv\_ctl** (ko NULL)

        ercd = impdrv\_genctl\_set\_cl\_brk\_addr(

            p\_impdrv\_ctl,

            p\_core\_info,

            (const uint32\_t)cl\_brk\_addr

        );

Khi thực hiện hàm này, hàm chắc chắn sẽ thực hiện lock mutex (do các giá trị đã VALID)

    /\*\* Mutex lock and prologue \*/

    ercd = impdrv\_genctl\_prologue(p\_impdrv\_ctl, p\_core\_info);

ercd = impdrv\_cmnctl\_mutex\_lock(&p\_impdrv\_ctl->common\_ctl);(v4m)

ercd = impdrv\_cmnctl\_test\_checksum();

ercd = impdrv\_cmnctl\_reg\_test\_checksum(); //kiểm tra với g\_hwrsc\_mng\_table (biến global, cho vào excel

p\_funcs = impdrv\_cmnctl\_get\_corectl\_func(p\_core\_info->core\_type);(v4m)

kế đó core\_type sẽ được cho vào so sánh để lấy array function, lúc này có 2 kq là vị trí 19 sẽ là NULL hoặc **impdrv\_impctl\_set\_cl\_brk\_addr**:

* Các **core\_type** valid để ko NULL:
  + IMPDRV\_CORE\_TYPE\_IMP
  + IMPDRV\_CORE\_TYPE\_OCV
  + IMPDRV\_CORE\_TYPE\_DMAC
  + IMPDRV\_CORE\_TYPE\_DMAC\_SLIM
  + IMPDRV\_CORE\_TYPE\_PSCEXE
  + IMPDRV\_CORE\_TYPE\_CNN

Nếu vị trí đó ko NULL thì mới tiến tiếp thực thi hàm đã được truyền vào : **impdrv\_impctl\_set\_cl\_brk\_addr (v4m)**

e\_impdrv\_errorcode\_t impdrv\_impctl\_set\_cl\_brk\_addr( //v4m

    st\_impdrv\_corectl\_t     \*const p\_corectl,// &(p\_impdrv\_ctl->coretype\_ctl[p\_core\_info->core\_type].core\_ctl[p\_core\_info->core\_num]),

    const uint32\_t          core\_num, // p\_core\_info->core\_num,  
 const uint32\_t          cl\_brk\_addr

)

&(p\_impdrv\_ctl->coretype\_ctl[p\_core\_info->core\_type].core\_ctl[p\_core\_info->core\_num]:

A computer screen shot of a program code

Description automatically generated

Tới đây thì chạy thẳng vào:

        /\*\* Check core number \*/

        is\_valid\_core = impdrv\_impctl\_is\_valid\_core(core\_num);

Lúc này **core\_num** phải nhỏ hơn 2

    /\*\* Check core state \*/

    if (IMPDRV\_EC\_OK == ercd)

    {

        ercd = impdrv\_impctl\_check\_state(p\_corectl, core\_num, IMPDRV\_STATE\_READY);

        if (IMPDRV\_EC\_NG\_SEQSTATE == ercd)

        {

            ercd = impdrv\_impctl\_check\_state(p\_corectl, core\_num, IMPDRV\_STATE\_INT);

        }

    }

Cần phải debug bằng gdb

Tới đây error code vẫn OK thì gán **cl\_brk\_addr** vào **p\_corectl->param[IMPDRV\_IMP\_EXEPARAM\_CLBRK]**

e\_impdrv\_errorcode\_t impdrv\_genctl\_epilogue(

    st\_impdrv\_ctl\_t             \*const p\_impdrv\_ctl,

    const e\_impdrv\_errorcode\_t  ercd

)

Tới đây chạy thẳng vào để unlock mutex

retcd = impdrv\_cmnctl\_mutex\_unlock(&p\_impdrv\_ctl->common\_ctl);

và release

lRetOsal = R\_OSAL\_ThsyncMutexUnlock(p\_mutex\_handle->handle);

Nếu tới đây error code = OK thì trả về OK, ko thì vào đây xử lý:

        if (IMPDRV\_EC\_OK != ercd)

        {

            /\*\* Convert error code from private error \*/

            ercd = impdrv\_api\_cnv\_error\_code(ercd);

        }

Chuyển Argument Null và Parameter Error thành System Error, còn lại giữ nguyên

    /\*\* Convert error code \*/

    if ((IMPDRV\_EC\_NG\_ARGNULL == retcd) || (IMPDRV\_EC\_NG\_PARAM == retcd))

    {

        retcd = IMPDRV\_EC\_NG\_SYSTEMERROR;

    }

//chưa sử dụng hình này

A screenshot of a computer

Description automatically generated

Sau khi chuẩn bị excel file thì vào thư mục spec sẽ có file **IMPDRV.ini** trỏ vào **input PATH** vào file excel và **output path** vào thư mục **ut\_framework**

InputPath = "C:\\Users\\long.trinh-tien\\Documents\\Git\\training\_ut\_it\_Long\\spec\\IMPD\_X30\_Unit\_Test\_SpecificationST.xlsx"

; Ex: ${Workspace}/ut\_framework

OutputPath = "C:\\Users\\long.trinh-tien\\Documents\\Git\\training\_ut\_it\_Long\\rcar-env\\renesas\\driver\\soc\\platform\\imp\\test\\unit\_test\\application\\ut\_framework"

* Ở mục Target of Testing, comment hết các mục không cần test, giữ lại function cần test (chọn nhiều bằng cách chọn con trỏ ở vị trí ban đầu, kéo xuống dưới cùng, ấn Alt+shift+Click trái vào dòng cuối, ấn “;” để comment hàng loạt)

A screenshot of a computer

Description automatically generated

Sau đó vào thư mục UTS for C++ (check python version và openpyxl) để trỏ vào file IMPDRV.ini mà ta config lúc nãy

A screenshot of a computer

Description automatically generated

Sau đó copy **make.exe** to **rcar-env\renesas\driver\soc\platform\imp\test\unit\_test\application\ut\_framework .**Ngay tại đây ta sửa DEVICE cho phù hợp với target (chưa đầy đủ)

Vào **rcar-env\renesas\driver\soc\platform\imp\test\unit\_test\application\ut\_framework\ut\imp\include\test\_case.h**

Để comment các test\_case không dùng tới trong struct :

static const struct TestInfo TEST\_UT\_Info[] =

{