Document Title	**DCX BLTC Software Low Level Design (LLD)**
Number	
Revision	**Z1**
Revision Date	**11/06/2008**
Author(s)	**Gamma Chen**

Internal
Level 2

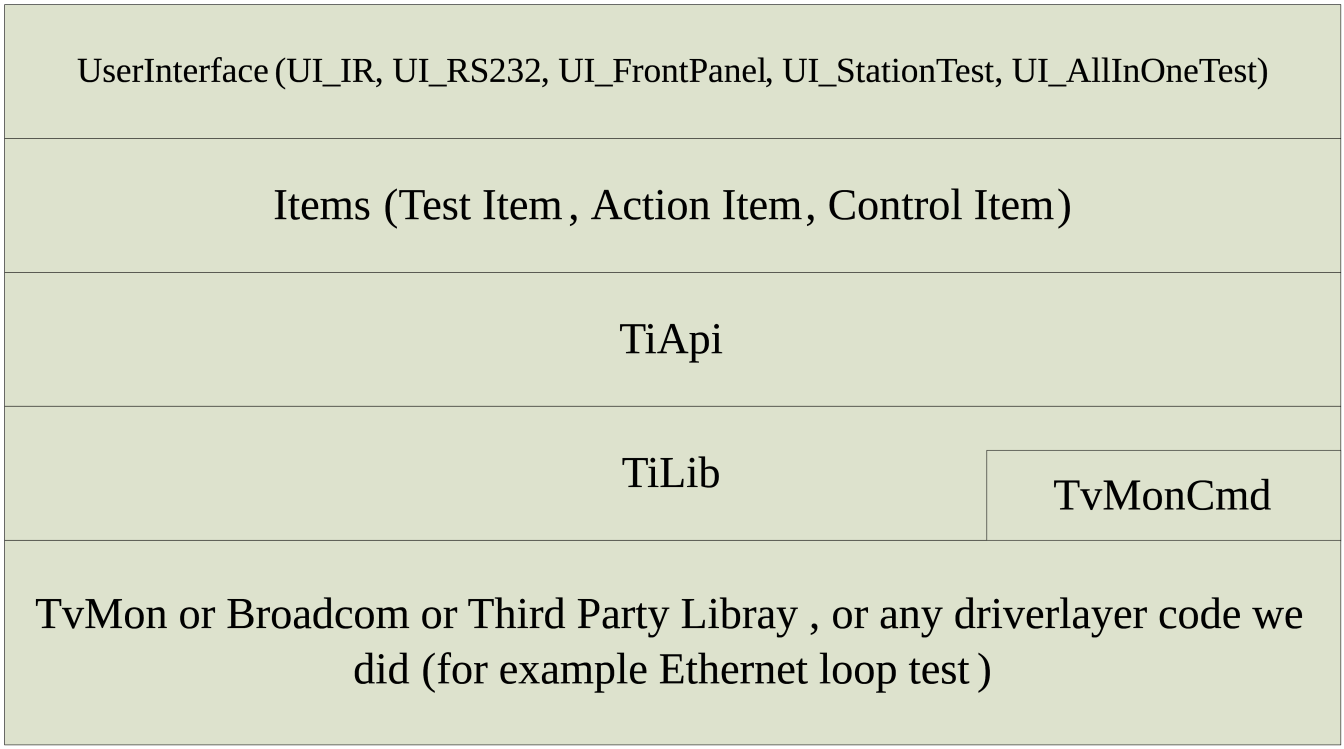
Draft

Revision History – see Workflow History for approvers and approval dates, and Notice for release dates

Rev	Notice #	Description	Author	Revision Date

TABLE OF CONTENTS

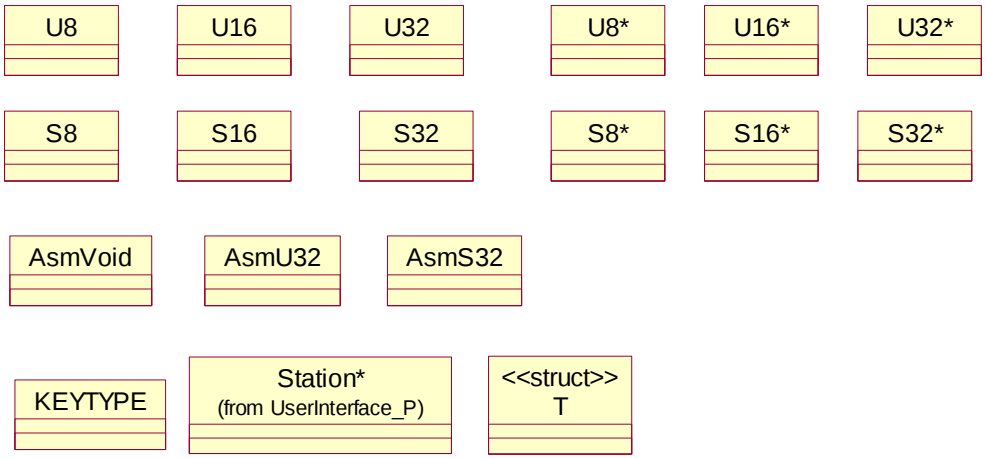
Follow are the Software Structure we mentioned in HLD. In LLD, we will explain class diagram and Sequence Diagram according the Software Layer we use in BLTC design.



1. CLASS DIAGRAM

1.1. BLTC Basic Type

Broadcom used to using their basic type (U8, U16, …). BLTC should avoid to define the same type name.



BltcBasicType.h contents as follow,

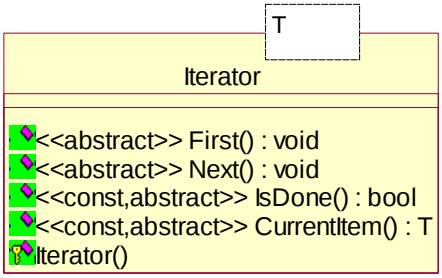
```
#include "gitypes.h"        // "gitypes.h" include above type and const.

// Follows for Code Generation Function of QIP7xxx BLTC can work in Rational Ross (UML tool).
#define AsmVoid        asm void
#define AsmU32         asm U32
#define AsmS32         asm S32
```

Rational Ross be chosen as OOA (Object Oriented Analysis) tool. I made AsmVoid define since “asm void” will make the Rational Ross dead in Code Generation Function or Reverse Engineer Function.

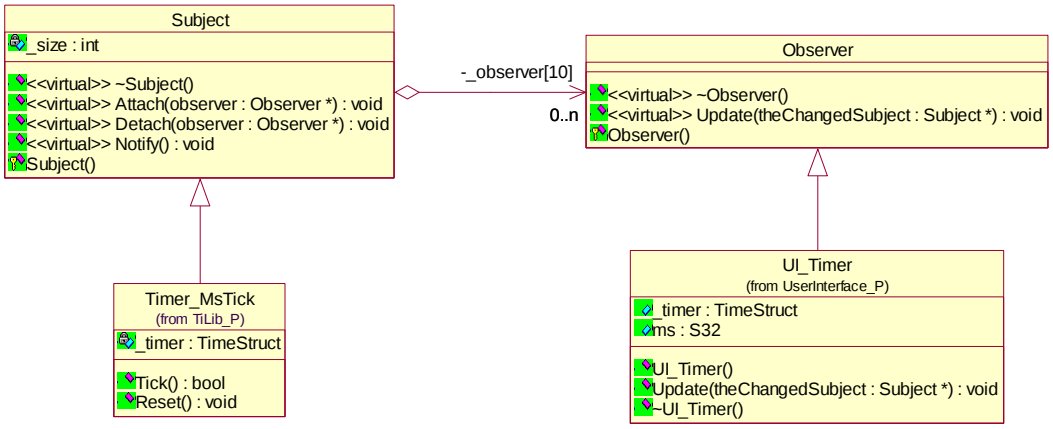
1.2. Pattern

A couple of pattern used in QIP7xxx BLTC design (Refer book, Design Patterns, published by Addison Wesley in 1995). They are:

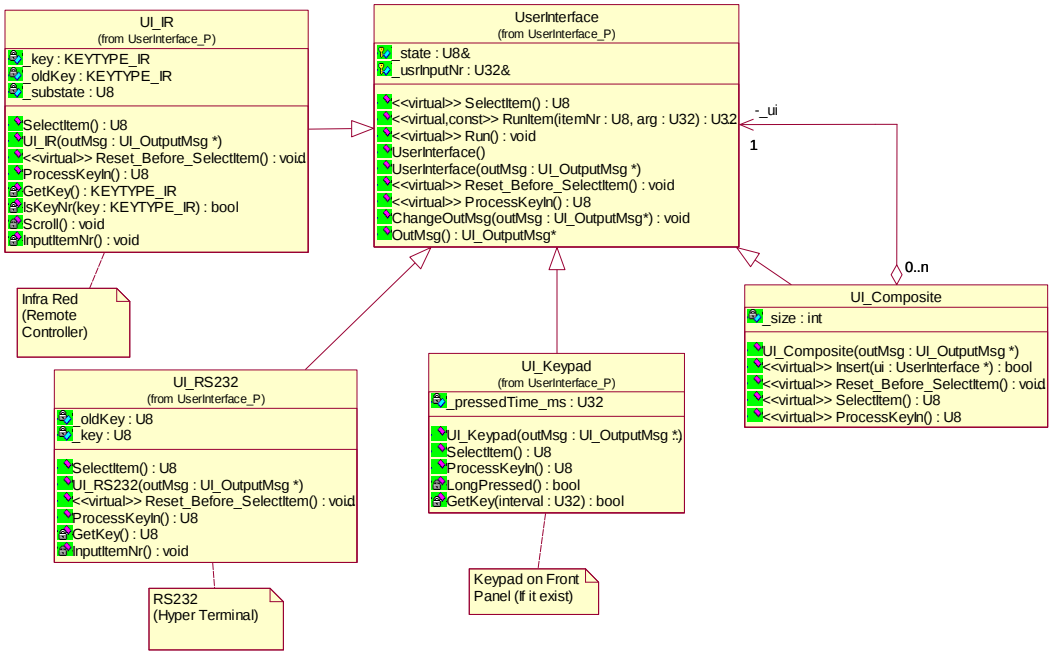
Iterator pattern

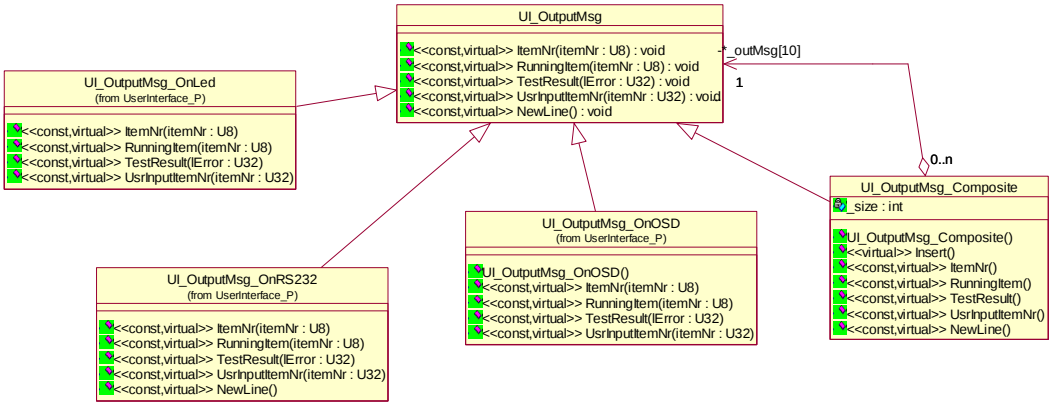


Observer pattern

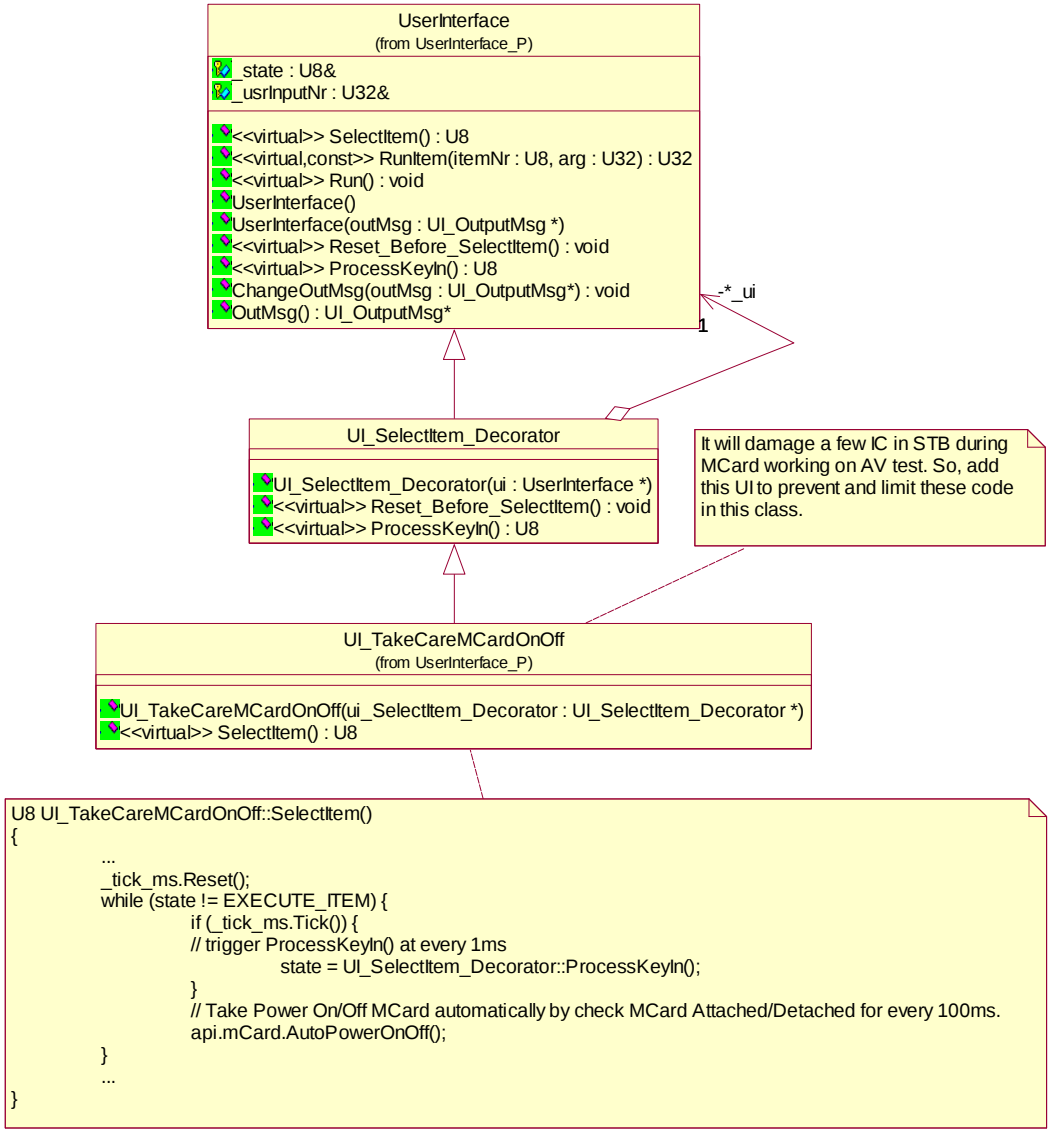


Composite pattern





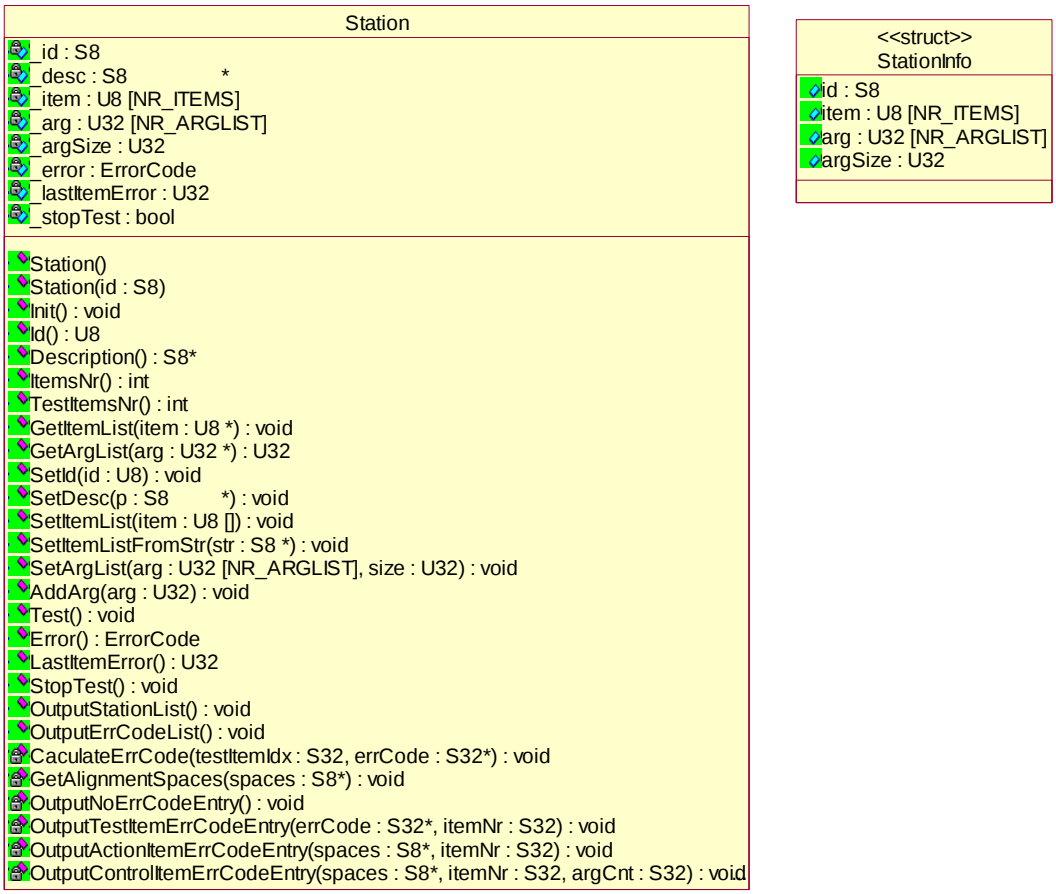
Decorator pattern

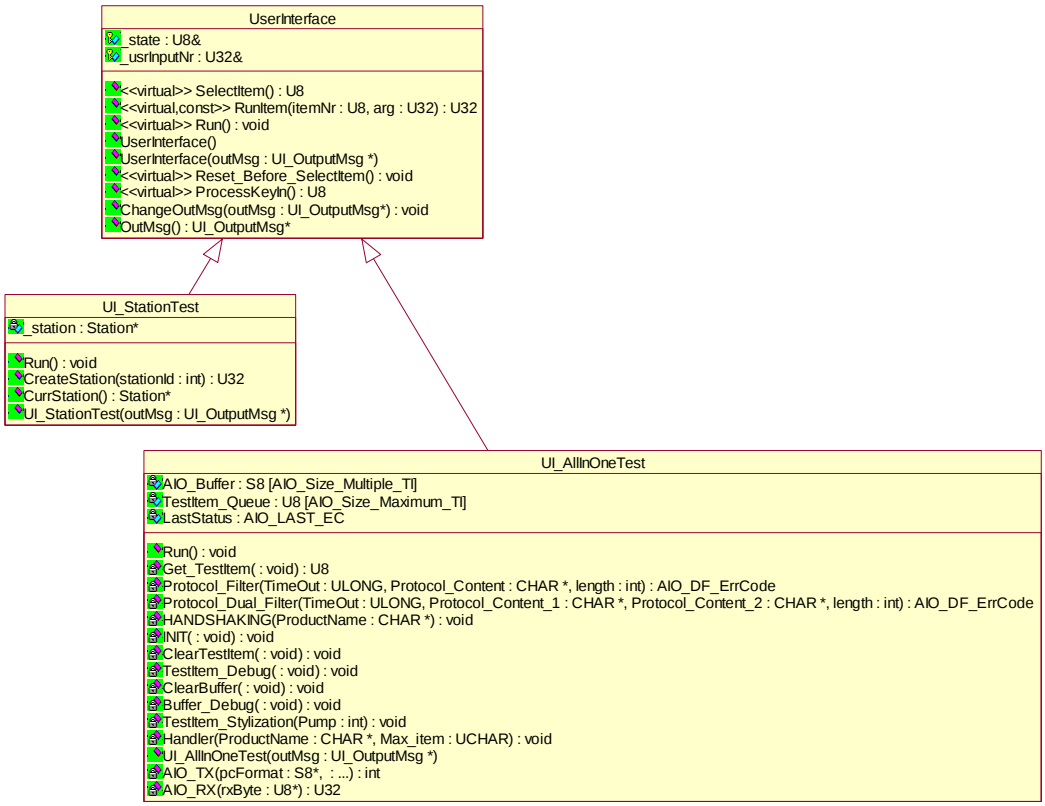


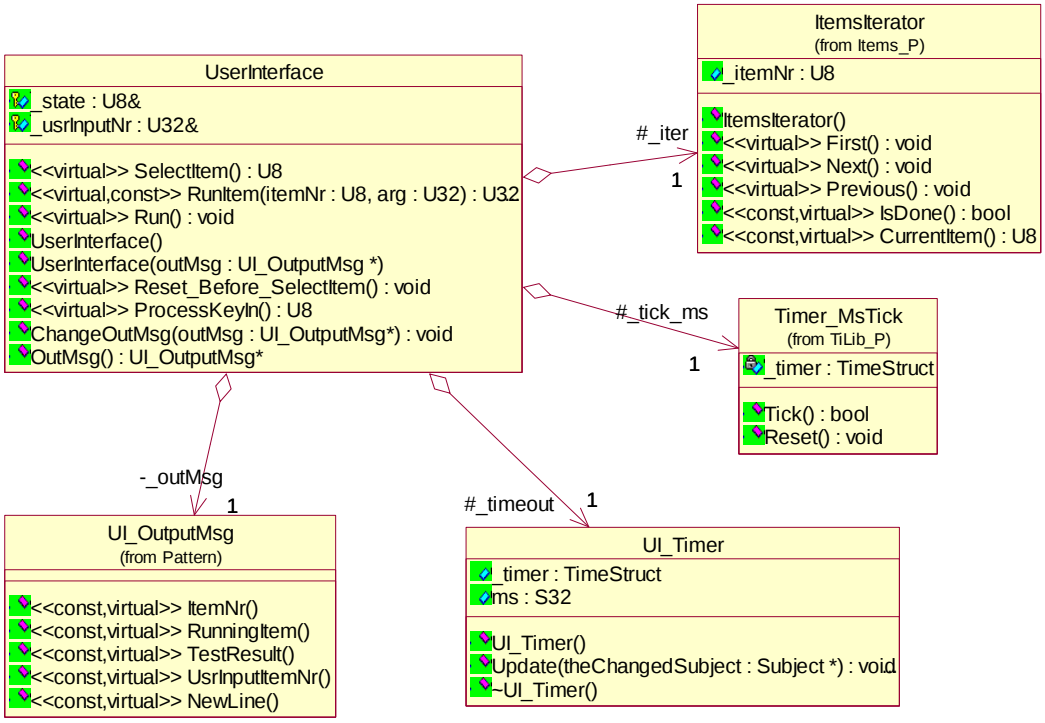
1.3. User Interface

For lab, select test item by Remote Controller (IR: Infra Red) or Front Panel (Keypad) or RS232, run single test item for debug (UI_IR, UI_RS232, UI_FrontPanel user interface), (refer Decorator Pattern as above).

In production-line, we using Station Test, UI_StationTest, currently, and have a plan All In One Test, UI_AllInOne, in future.




UserInterface
_state : U8&
_usrInputNr : U32&
<<virtual>> SelectItem() : U8
<<virtual,const>> RunItem(itemNr : U8, arg : U32) : U32
<<virtual>> Run() : void
UserInterface()
UserInterface(outMsg : UI_OutputMsg *)
<<virtual>> Reset_Before_SelectItem() : void
<<virtual>> ProcessKeyIn() : U8
ChangeOutMsg(outMsg : UI_OutputMsg*) : void
OutMsg() : UI_OutputMsg*
UI_StationTest
_station : Station*
Run() : void
CreateStation(stationId : int) : U32
CurrStation() : Station*
UI_StationTest(outMsg : UI_OutputMsg *)
UI_AllInOneTest
AIO_Buffer : S8 [AIO_Size_Multiple_TI]
TestItem_Queue : U8 [AIO_Size_Maximum_TI]
LastStatus : AIO_LAST_EC
Run() : void
Get_TestItem(: void) : U8
Protocol_Filter(TimeOut : ULONG, Protocol_Content : CHAR *, length : int) : AIO_DF_ErrCode
Protocol_Dual_Filter(TimeOut : ULONG, Protocol_Content_1 : CHAR *, Protocol_Content_2 : CHAR *, length : int) : AIO_DF_ErrCode
HANDSHAKING(ProductName : CHAR *) : void
INIT(: void) : void
ClearTestItem(: void) : void
TestItem_Debug(: void) : void
ClearBuffer(: void) : void
Buffer_Debug(: void) : void
TestItem_Stylization(Pump : int) : void
Handler(ProductName : CHAR *, Max_item : UCHAR) : void
UI_AllInOneTest(outMsg : UI_OutputMsg *)
AIO_TX(pcFormat : S8*, : ...) : int
AIO_RX(rxByte : U8*) : U32


UserInterface
_state : U8&
_usrInputNr : U32&
<<virtual>> SelectItem() : U8
<<virtual,const>> RunItem(itemNr : U8, arg : U32) : U32
<<virtual>> Run() : void
UserInterface()
UserInterface(outMsg : UI_OutputMsg *)
<<virtual>> Reset_Before_SelectItem() : void
<<virtual>> ProcessKeyIn() : U8
ChangeOutMsg(outMsg : UI_OutputMsg*) : void
OutMsg() : UI_OutputMsg*
ItemsIterator
(from Items_P)
_itemNr : U8
ItemsIterator()
<<virtual>> First() : void
<<virtual>> Next() : void
<<virtual>> Previous() : void
<<const,virtual>> IsDone() : bool
<<const,virtual>> CurrentItem() : U8
#_iter
1
#_tick_ms
1
Timer_MsTick
(from TiLib_P)
_timer : TimeStruct
Tick() : bool
Reset() : void
-_outMsg
1
UI_OutputMsg
(from Pattern)
<<const,virtual>> ItemNr()
<<const,virtual>> RunningItem()
<<const,virtual>> TestResult()
<<const,virtual>> UsrInputItemNr()
<<const,virtual>> NewLine()
#_timeout
1
UI_Timer
_timer : TimeStruct
ms : S32
UI_Timer()
Update(theChangedSubject : Subject *) : void
~UI_Timer()

1.4. Items

Give a number for each test item, it’s purpose is to communicate with PE clearly.

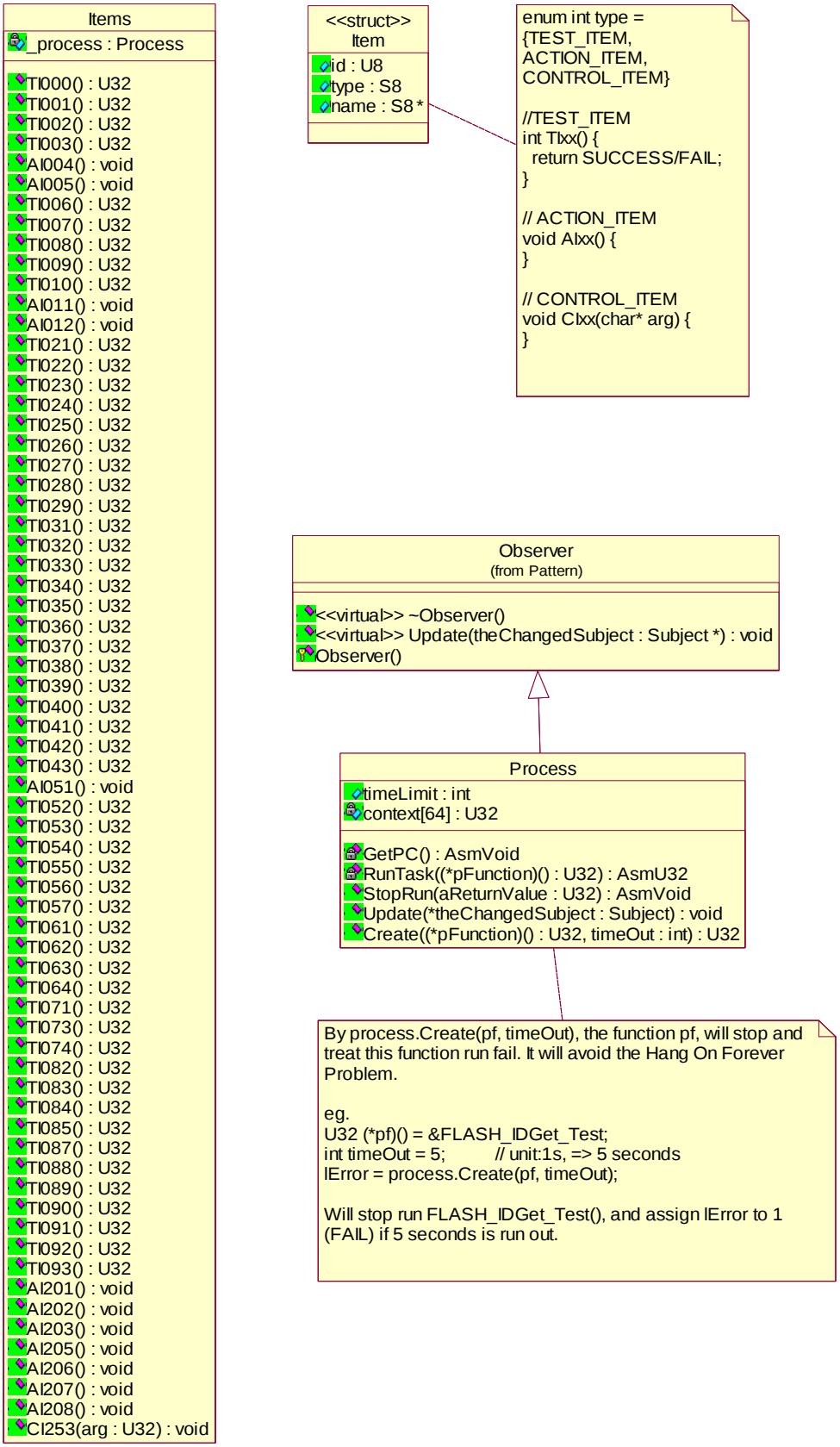
For example,

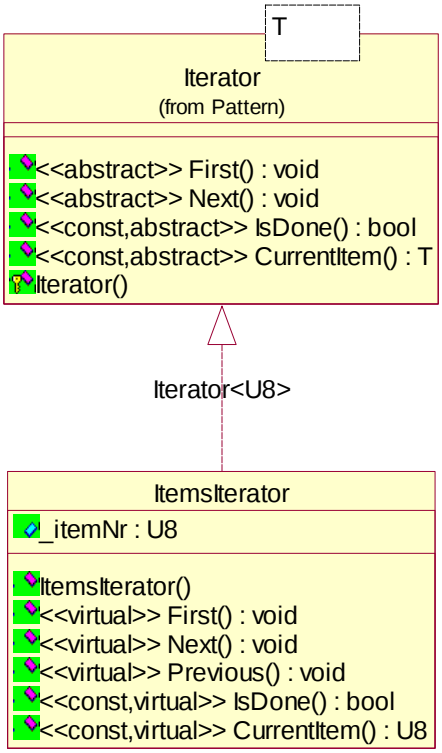
TI000: "Display station information on RS232, setting:(38400,8,O,1,n)"

TI103: “Tuner 1(64QAM, channel=8, pid=8c0 8c0 8c1) A/V test, and 1394 Tx Test"

TI105: “Tuner 1(256QAM, channel=6, pid=310 310 311) A/V test, and 1394 Tx Test"

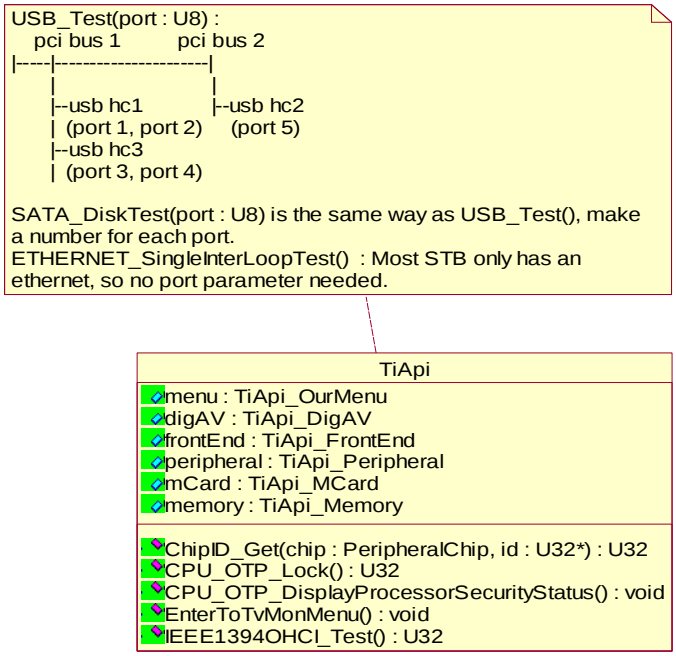
When operator report to PE, and PE told us the test item number has trouble, eg.103, then we know exactly the TI103: “Tuner1(64QAM)…” has trouble.

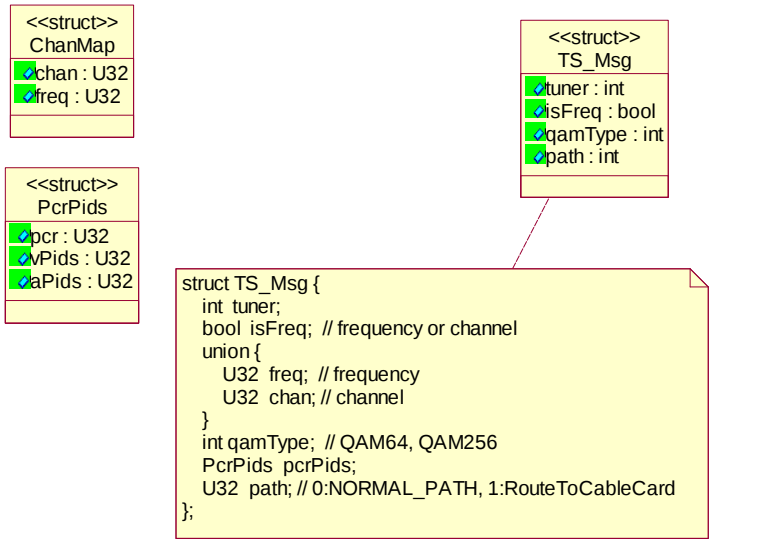

Items
_process : Process
TI000() : U32
TI001() : U32
TI002() : U32
TI003() : U32
AI004() : void
AI005() : void
TI006() : U32
TI007() : U32
TI008() : U32
TI009() : U32
TI010() : U32
AI011() : void
AI012() : void
TI021() : U32
TI022() : U32
TI023() : U32
TI024() : U32
TI025() : U32
TI026() : U32
TI027() : U32
TI028() : U32
TI029() : U32
TI031() : U32
TI032() : U32
TI033() : U32
TI034() : U32
TI035() : U32
TI036() : U32
TI037() : U32
TI038() : U32
TI039() : U32
TI040() : U32
TI041() : U32
TI042() : U32
TI043() : U32
AI051() : void
TI052() : U32
TI053() : U32
TI054() : U32
TI055() : U32
TI056() : U32
TI057() : U32
TI061() : U32
TI062() : U32
TI063() : U32
TI064() : U32
TI071() : U32
TI073() : U32
TI074() : U32
TI082() : U32
TI083() : U32
TI084() : U32
TI085() : U32
TI087() : U32
TI088() : U32
TI089() : U32
TI090() : U32
TI091() : U32
TI092() : U32
TI093() : U32
AI201() : void
AI202() : void
AI203() : void
AI205() : void
AI206() : void
AI207() : void
AI208() : void
CI253(arg : U32) : void
<<struct>>
Item
id : U8
type : S8
name : S8 *
enum int type =
{TEST_ITEM,
ACTION_ITEM,
CONTROL_ITEM}
//TEST_ITEM
int TIxx() {
return SUCCESS/FAIL;
}
// ACTION_ITEM
void AIxx() {
}
// CONTROL_ITEM
void CIxx(char* arg) {
}
Observer
(from Pattern)
<<virtual>> ~Observer()
<<virtual>> Update(theChangedSubject : Subject *) : void
Observer()
Process
timeLimit : int
context[64] : U32
GetPC() : AsmVoid
RunTask((*pFunction)() : U32) : AsmU32
StopRun(aReturnValue : U32) : AsmVoid
Update(*theChangedSubject : Subject) : void
Create((*pFunction)() : U32, timeOut : int) : U32
By process.Create(pf, timeOut), the function pf, will stop and treat this function run fail. It will avoid the Hang On Forever Problem.
eg.
U32 (*pf)() = &FLASH_IDGet_Test;
int timeOut = 5; // unit:1s, => 5 seconds
lError = process.Create(pf, timeOut);
Will stop run FLASH_IDGet_Test(), and assign lError to 1 (FAIL) if 5 seconds is run out.

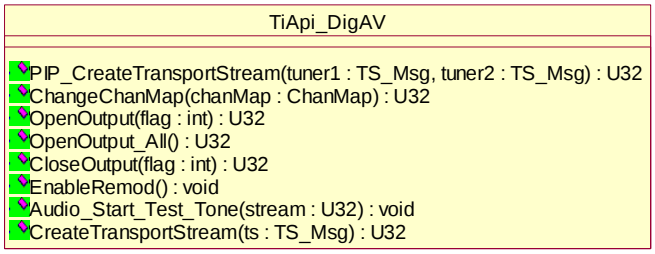


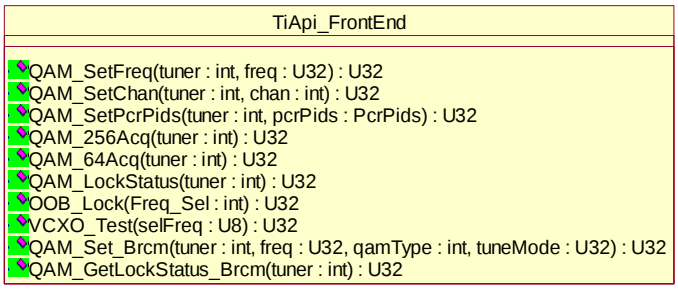
1.5. TiApi

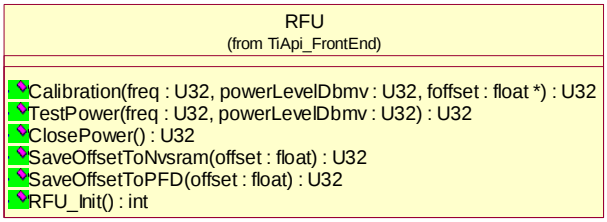
The layer be called by layer items or above.

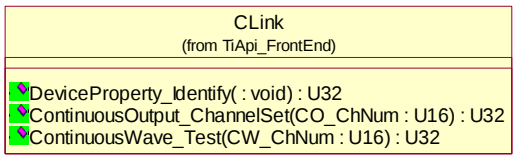


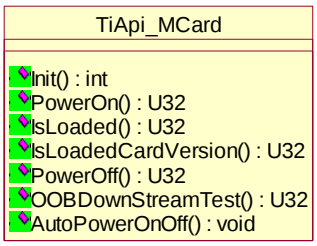

<<struct>>
ChanMap
chan : U32
freq : U32
<<struct>>
PcrPids
pcr : U32
vPids : U32
aPids : U32
<<struct>>
TS_Msg
tuner : int
isFreq : bool
qamType : int
path : int
struct TS_Msg {
int tuner;
bool isFreq; // frequency or channel
union {
U32 freq; // frequency
U32 chan; // channel
}
int qamType; // QAM64, QAM256
PcrPids pcrPids;
U32 path; // 0:NORMAL_PATH, 1:RouteToCableCard
};

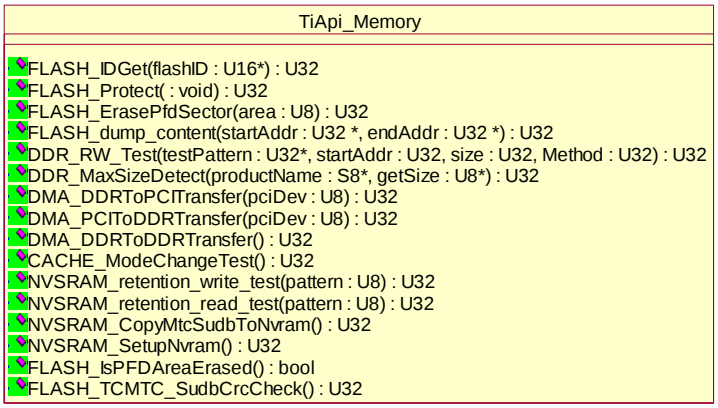

TiApi_DigAV
PIP_CreateTransportStream(tuner1 : TS_Msg, tuner2 : TS_Msg) : U32
ChangeChanMap(chanMap : ChanMap) : U32
OpenOutput(flag : int) : U32
OpenOutput_All() : U32
CloseOutput(flag : int) : U32
EnableRemod() : void
Audio_Start_Test_Tone(stream : U32) : void
CreateTransportStream(ts : TS_Msg) : U32

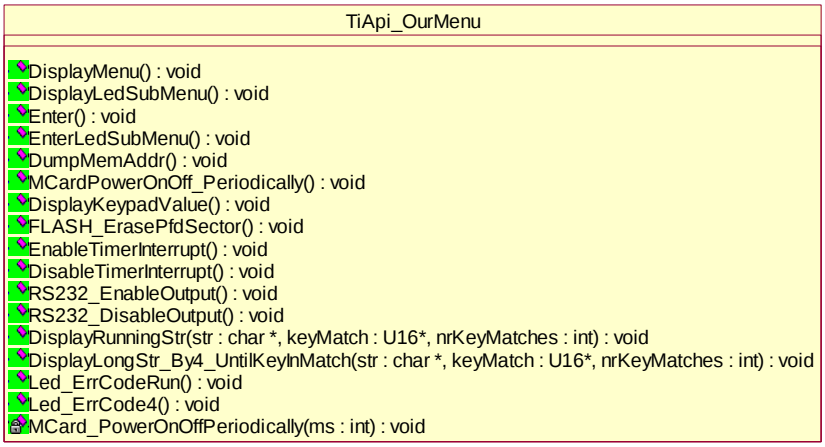

TiApi_FrontEnd
QAM_SetFreq(tuner : int, freq : U32) : U32
QAM_SetChan(tuner : int, chan : int) : U32
QAM_SetPcrPids(tuner : int, pcrPids : PcrPids) : U32
QAM_256Acq(tuner : int) : U32
QAM_64Acq(tuner : int) : U32
QAM_LockStatus(tuner : int) : U32
OOB_Lock(Freq_Sel : int) : U32
VCXO_Test(selFreq : U8) : U32
QAM_Set_Brcm(tuner : int, freq : U32, qamType : int, tuneMode : U32) : U32
QAM_GetLockStatus_Brcm(tuner : int) : U32

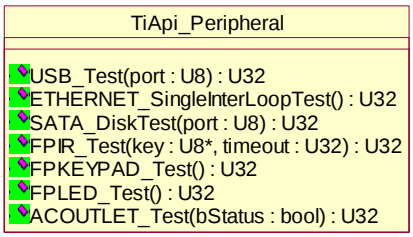

RFU
(from TiApi_FrontEnd)
Calibration(freq : U32, powerLevelDbmv : U32, foffset : float *) : U32
TestPower(freq : U32, powerLevelDbmv : U32) : U32
ClosePower() : U32
SaveOffsetToNvsram(offset : float) : U32
SaveOffsetToPFD(offset : float) : U32
RFU_Init() : int


CLink
(from TiApi_FrontEnd)
DeviceProperty_Identify(: void) : U32
ContinuousOutput_ChannelSet(CO_ChNum : U16) : U32
ContinuousWave_Test(CW_ChNum : U16) : U32



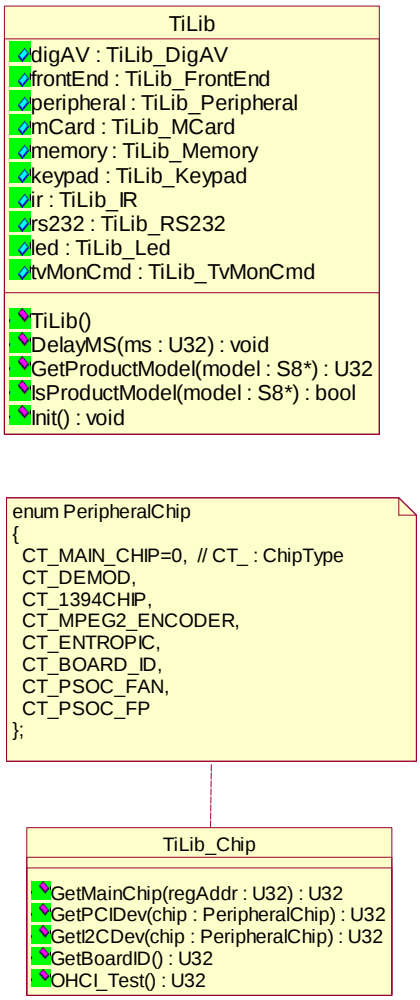


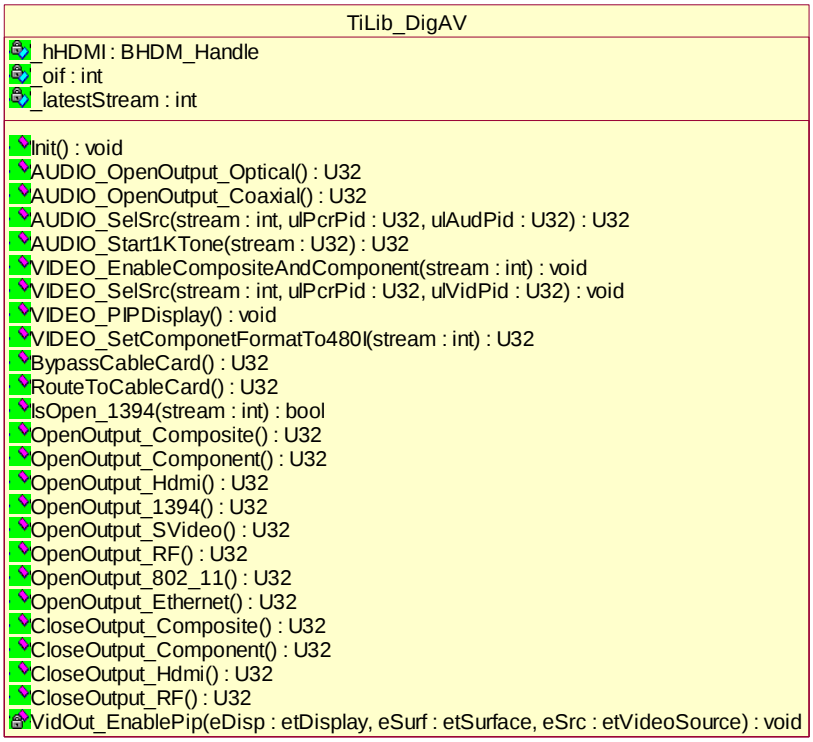




1.6. TiLib

The layer be called by layer TiApi or above.


TiLib
digAV : TiLib_DigAV
frontEnd : TiLib_FrontEnd
peripheral : TiLib_Peripheral
mCard : TiLib_MCard
memory : TiLib_Memory
keypad : TiLib_Keypad
ir : TiLib_IR
rs232 : TiLib_RS232
led : TiLib_Led
tvMonCmd : TiLib_TvMonCmd
TiLib()
DelayMS(ms : U32) : void
GetProductModel(model : S8*) : U32
IsProductModel(model : S8*) : bool
Init() : void
enum PeripheralChip
{
CT_MAIN_CHIP=0, // CT_ : ChipType
CT_DEMOD,
CT_1394CHIP,
CT_MPEG2_ENCODER,
CT_ENTROPIC,
CT_BOARD_ID,
CT_PSOC_FAN,
CT_PSOC_FP
};
TiLib_Chip
GetMainChip(regAddr : U32) : U32
GetPCIDev(chip : PeripheralChip) : U32
GetI2CDev(chip : PeripheralChip) : U32
GetBoardID() : U32
OHCI_Test() : U32


TiLib_DigAV
_hHDMI : BHDM_Handle
_oif : int
_latestStream : int
Init() : void
AUDIO_OpenOutput_Optical() : U32
AUDIO_OpenOutput_Coaxial() : U32
AUDIO_SelSrc(stream : int, ulPcrPid : U32, ulAudPid : U32) : U32
AUDIO_Start1KTone(stream : U32) : U32
VIDEO_EnableCompositeAndComponent(stream : int) : void
VIDEO_SelSrc(stream : int, ulPcrPid : U32, ulVidPid : U32) : void
VIDEO_PIPDisplay() : void
VIDEO_SetComponetFormatTo480I(stream : int) : U32
BypassCableCard() : U32
RouteToCableCard() : U32
IsOpen_1394(stream : int) : bool
OpenOutput_Composite() : U32
OpenOutput_Component() : U32
OpenOutput_Hdmi() : U32
OpenOutput_1394() : U32
OpenOutput_SVideo() : U32
OpenOutput_RF() : U32
OpenOutput_802_11() : U32
OpenOutput_Ethernet() : U32
CloseOutput_Composite() : U32
CloseOutput_Component() : U32
CloseOutput_Hdmi() : U32
CloseOutput_RF() : U32
VidOut_EnablePip(eDisp : etDisplay, eSurf : etSurface, eSrc : etVideoSource) : void

```
enum VCXO_Frequency_Set
{
  VCXO_27_H=0,
  VCXO_27_M,
  VCXO_27_L,
  VCXO_25_GPIO_Output
};
```

TiLib_FrontEnd

- INB_Parameter : INB_INIT_ARG
- OOB_Parameter : OOB_INIT_ARG

- OOB_Frequency_Set(Freq_Sel : eOOB_Freq_Sel) : eTiLib_OOB_Error
- OOB_ACQ(: void) : eTiLib_OOB_Error
- OOB_LockStatus_Get(: void) : eTiLib_OOB_Error
- INB_INIT(: void) : U32
- QAM_SymboyRate_Set(DS_Port : eINB_Channel_Sel, DeMod_Sel : eINB_DeMod_Sel) : U32
- QAM_IF_AGC_Control_Set(Type_IF_AGC : int) : U32
- OOB_INIT(: void) : U32
- OOB_SymboyRate_Set(: void) : U32

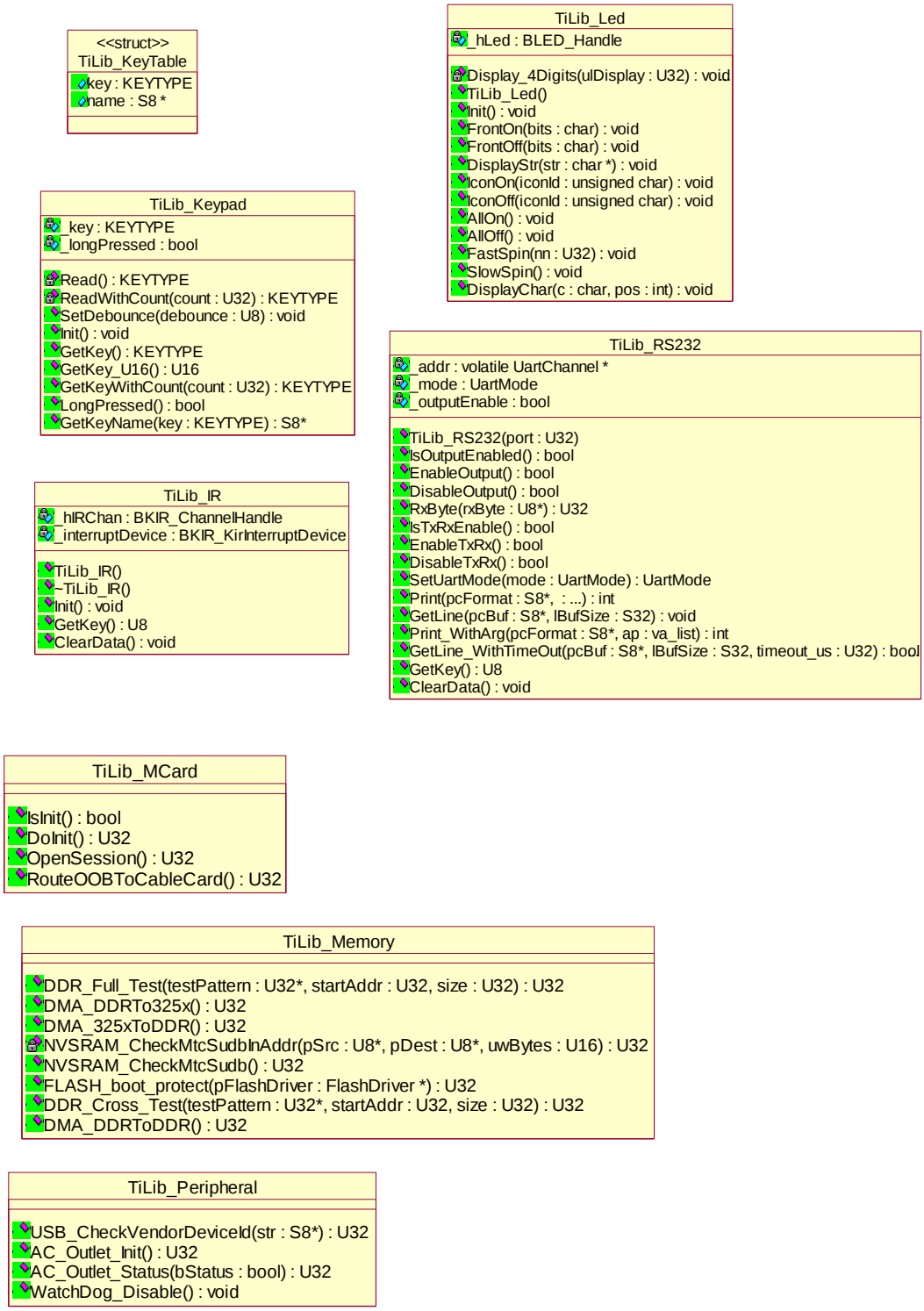
RFU
(from TiLib_FrontEnd)

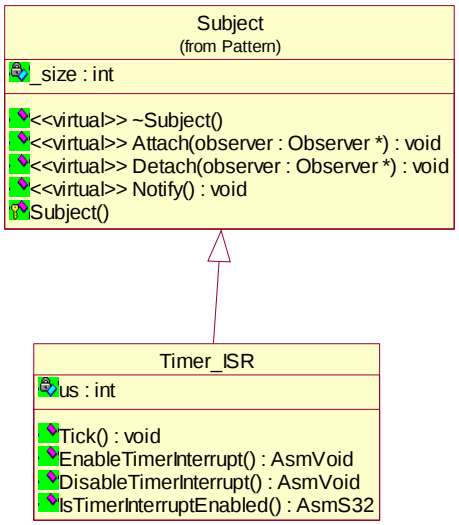
- SetFrequency(ulSetFreqKhz : U32) : eRfUpstreamError_t
- SetPowerLevel(fPowerDbmv : U32) : eRfUpstreamError_t
- GetPowerLevel(fPowerDbmv : U32 &) : eRfUpstreamError_t
- Calibrate(offset : float *) : eRfUpstreamError_t
- ReadPowerFromPowerMeter(freq : U32, powerLevelDbm : U32*) : int
- CheckPower(freq : U32, powerLevelDbm : U32) : int

CLink
(from TiLib_FrontEnd)

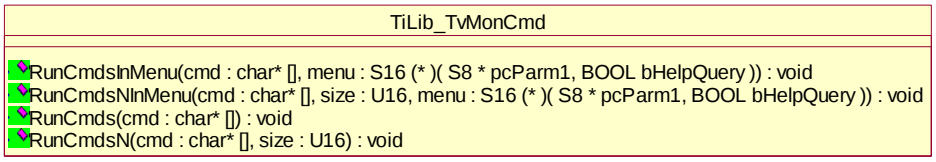
- CLink_Parameter : CLink_INIT_ARG

- CLink_Device_Identify(PCI_Id : U16 &, Chip_Version : U16 &) : eTiLib_CLink_Error
- CLink_CO_Channel_Tune(CO_ChNum : U16) : eTiLib_CLink_Error
- CLink_CW_Channel_Tune(CW_ChNum : U16) : eTiLib_CLink_Error
- CLink_INIT(: void) : U32

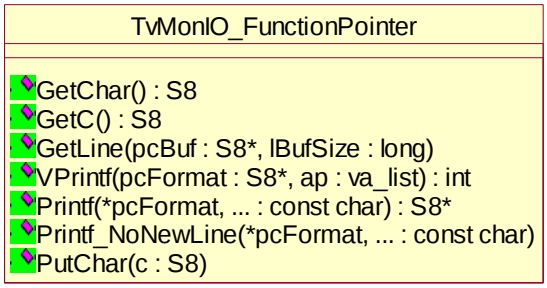

<<struct>>
TiLib_KeyTable
key : KEYTYPE
name : S8 *
TiLib_Keypad
_key : KEYTYPE
_longPressed : bool
Read() : KEYTYPE
ReadWithCount(count : U32) : KEYTYPE
SetDebounce(debounce : U8) : void
Init() : void
GetKey() : KEYTYPE
GetKey_U16() : U16
GetKeyWithCount(count : U32) : KEYTYPE
LongPressed() : bool
GetKeyName(key : KEYTYPE) : S8*
TiLib_IR
_hIRChan : BKIR_ChannelHandle
_interruptDevice : BKIR_KirInterruptDevice
TiLib_IR()
~TiLib_IR()
Init() : void
GetKey() : U8
ClearData() : void
TiLib_Led
_hLed : BLED_Handle
Display_4Digits(ulDisplay : U32) : void
TiLib_Led()
Init() : void
FrontOn(bits : char) : void
FrontOff(bits : char) : void
DisplayStr(str : char *) : void
IconOn(iconId : unsigned char) : void
IconOff(iconId : unsigned char) : void
AllOn() : void
AllOff() : void
FastSpin(nn : U32) : void
SlowSpin() : void
DisplayChar(c : char, pos : int) : void
TiLib_RS232
_addr : volatile UartChannel *
_mode : UartMode
_outputEnable : bool
TiLib_RS232(port : U32)
IsOutputEnabled() : bool
EnableOutput() : bool
DisableOutput() : bool
RxByte(rxByte : U8*) : U32
IsTxRxEnable() : bool
EnableTxRx() : bool
DisableTxRx() : bool
SetUartMode(mode : UartMode) : UartMode
Print(pcFormat : S8*, : ...) : int
GetLine(pcBuf : S8*, lBufSize : S32) : void
Print_WithArg(pcFormat : S8*, ap : va_list) : int
GetLine_WithTimeOut(pcBuf : S8*, lBufSize : S32, timeout_us : U32) : bool
GetKey() : U8
ClearData() : void
TiLib_MCard
IsInit() : bool
DoInit() : U32
OpenSession() : U32
RouteOOBToCableCard() : U32
TiLib_Memory
DDR_Full_Test(testPattern : U32*, startAddr : U32, size : U32) : U32
DMA_DDRTo325x() : U32
DMA_325xToDDR() : U32
NVSRAM_CheckMtcSudbInAddr(pSrc : U8*, pDest : U8*, uwBytes : U16) : U32
NVSRAM_CheckMtcSudb() : U32
FLASH_boot_protect(pFlashDriver : FlashDriver *) : U32
DDR_Cross_Test(testPattern : U32*, startAddr : U32, size : U32) : U32
DMA_DDRToDDR() : U32
TiLib_Peripheral
USB_CheckVendorDeviceId(str : S8*) : U32
AC_Outlet_Init() : U32
AC_Outlet_Status(bStatus : bool) : U32
WatchDog_Disable() : void

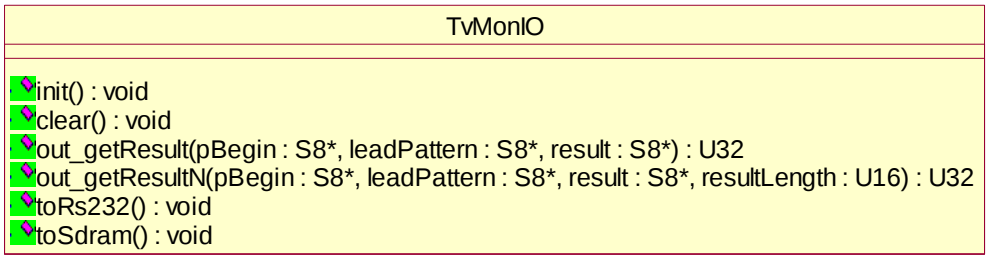


1.7. TvMonCmd



1.8. TvMonIO Function Pointer and TvMonIO (in Broadcom Hal Layer)



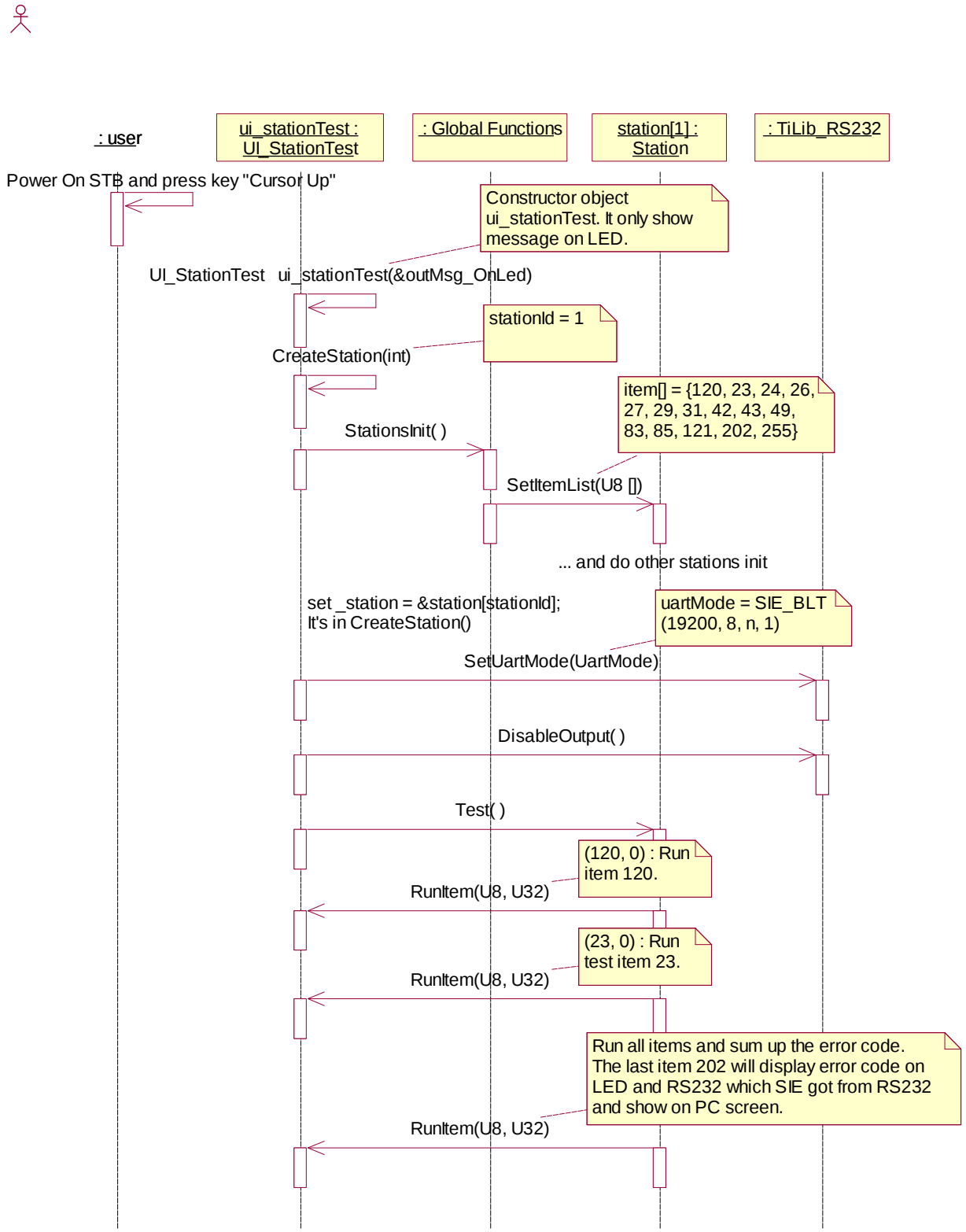


2. SEQUENCE DIAGRAM

This section using sequence diagram to describes how objects co-work to finish the system functions. UI_DebugTest and UI_StationTest are described in 6.1 User Interface. Some test items are described in 6.2 Items.

2.1. User Interface

UI_StationTest:

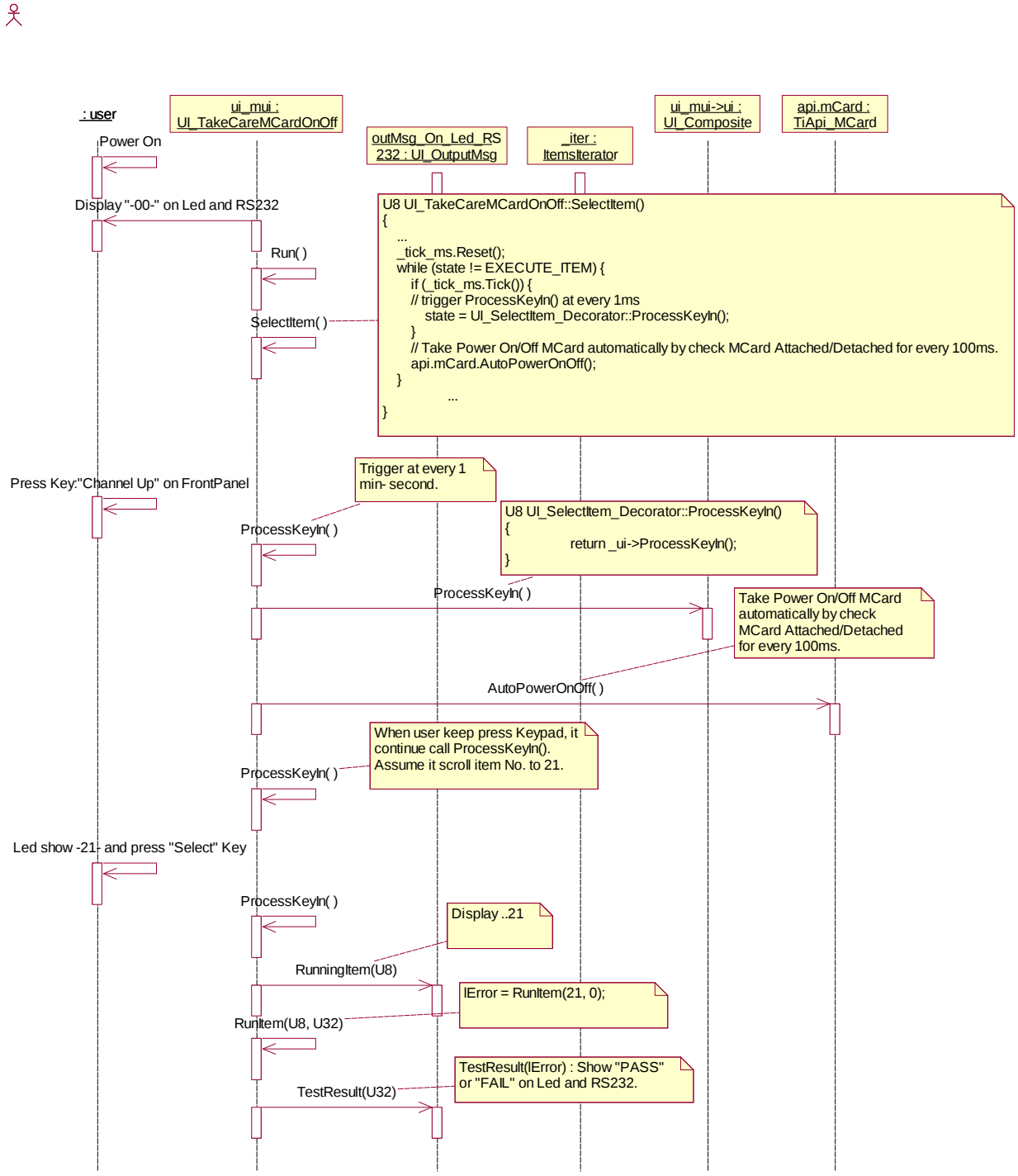

: user
ui_stationTest : UI_StationTest
: Global Functions
station[1] : Station
: TiLib_RS232
Power On STB and press key "Cursor Up"
Constructor object ui_stationTest. It only show message on LED.
UI_StationTest ui_stationTest(&outMsg_OnLed)
stationId = 1
CreateStation(int)
item[] = {120, 23, 24, 26, 27, 29, 31, 42, 43, 49, 83, 85, 121, 202, 255}
StationsInit( )
SetItemList(U8 [])
... and do other stations init
set _station = &station[stationId];
It's in CreateStation()
uartMode = SIE_BLT (19200, 8, n, 1)
SetUartMode(UartMode)
DisableOutput( )
Test( )
(120, 0) : Run item 120.
RunItem(U8, U32)
(23, 0) : Run test item 23.
RunItem(U8, U32)
Run all items and sum up the error code. The last item 202 will display error code on LED and RS232 which SIE got from RS232 and show on PC screen.
RunItem(U8, U32)

UI_DebugTest:

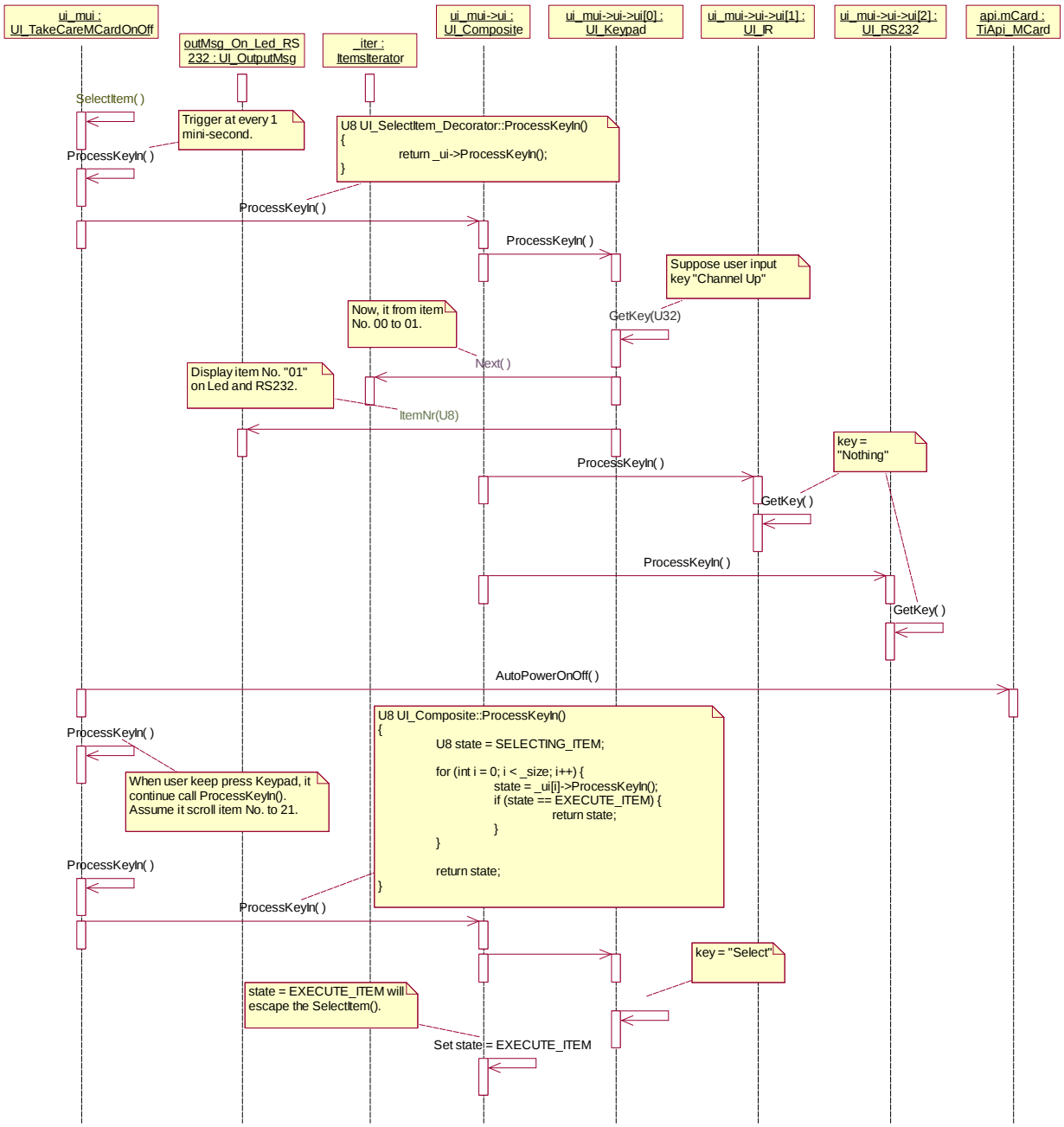
It will damage a few IC in STB during MCard working on AV test. So, add this UI_TakeCareMCardOnOff to prevent and limit these code in this class. The ui_mui is an object as type of class UI_TakeCareMCardOnOff, and include object ui_mui->ui which include 3 UI objects, UI_FrontPanel, UI_IR and UI_RS232.

SelectItem() will Scroll item No. when user press "Channel Up" or "Channel Down", and escape while loop when press key "Select".

The sequence diagram as follow,



The SelectItem() part as follow,



2.2. Items

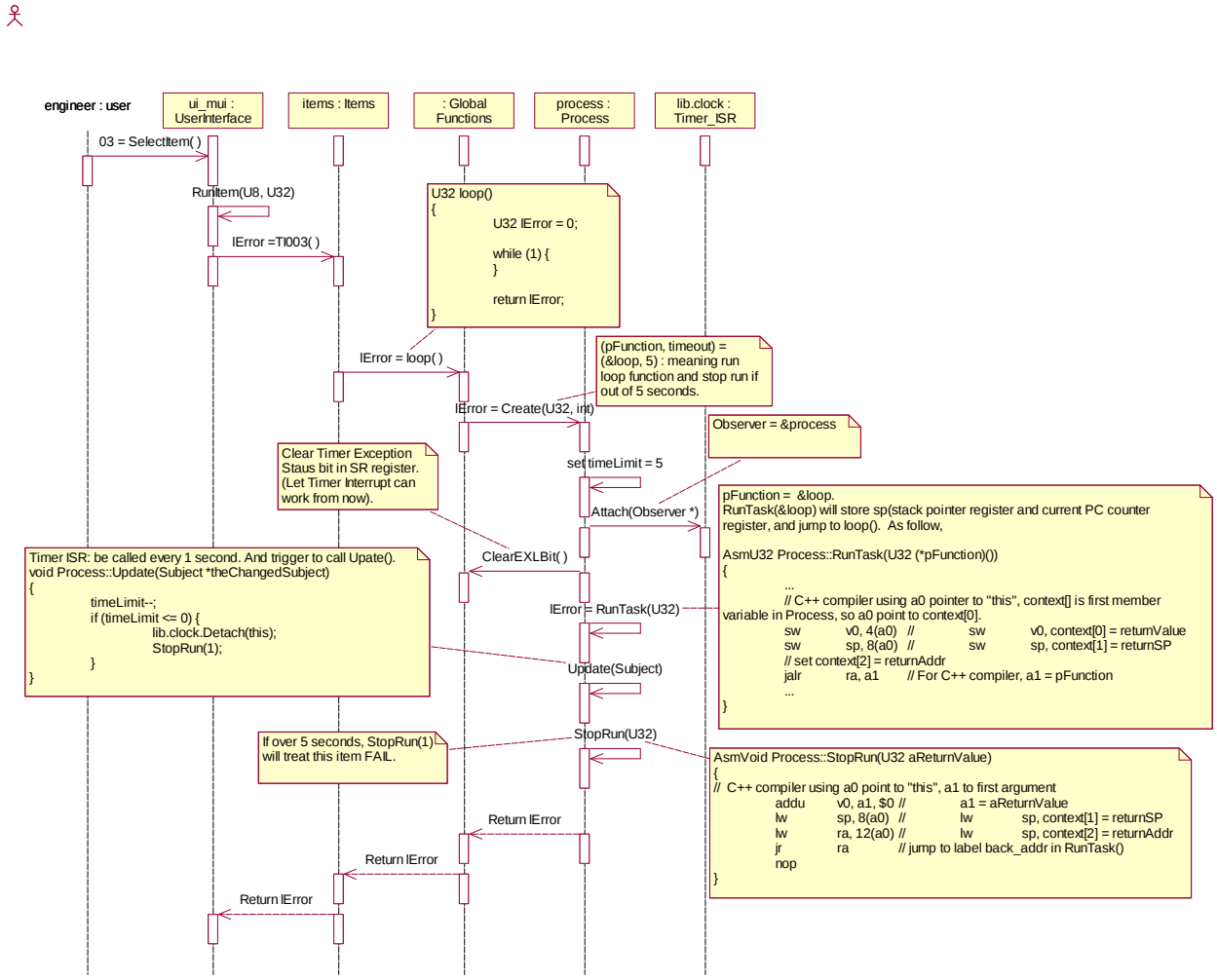
Run Item with Time Out:

I implement the time out mechanism and used in every test items. BLTC give every test item a time limit to run. If it over, then treat this test item is FAIL. The reasons for time out mechanism as follow,

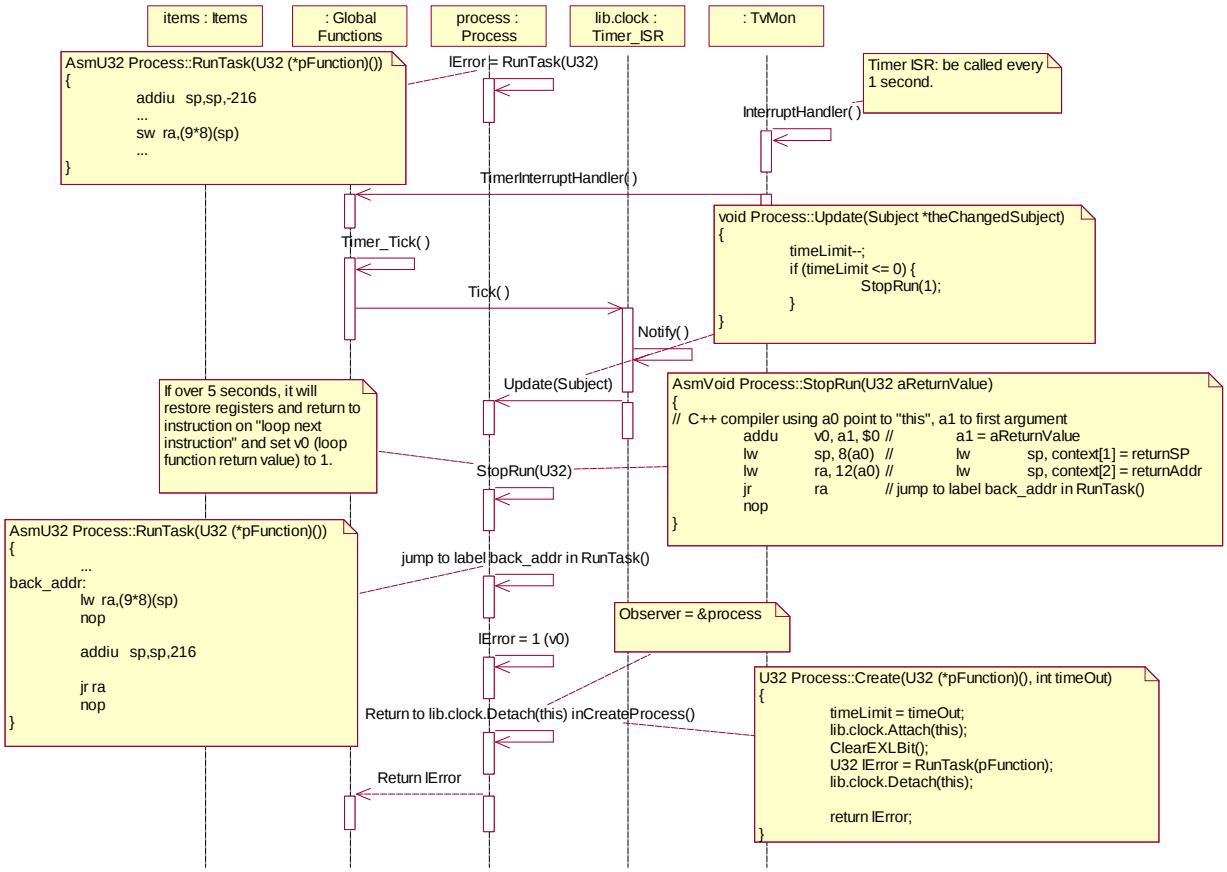
1. Sometimes especially in EPR or PPR stage (still in implement new test item), BLTC will hang on the test items, for example, the Ethernet Loop Test Item, SATA Test Item, and Flash Vendor ID Test Item will have chance to hang on forever (I

have this experience in DCT34xx and DCH P5 and QIP7216 EPR, and I know it came from the TvMon Layer in some special test order). When it happen, the whole station test cannot go on, and we have no idea to go on. The only way to run the EPR is by skip the whole station test and ask operator to run single item test, one by one. With this mechanism, I don't have to worry about this situation any more (it's a nightmare).

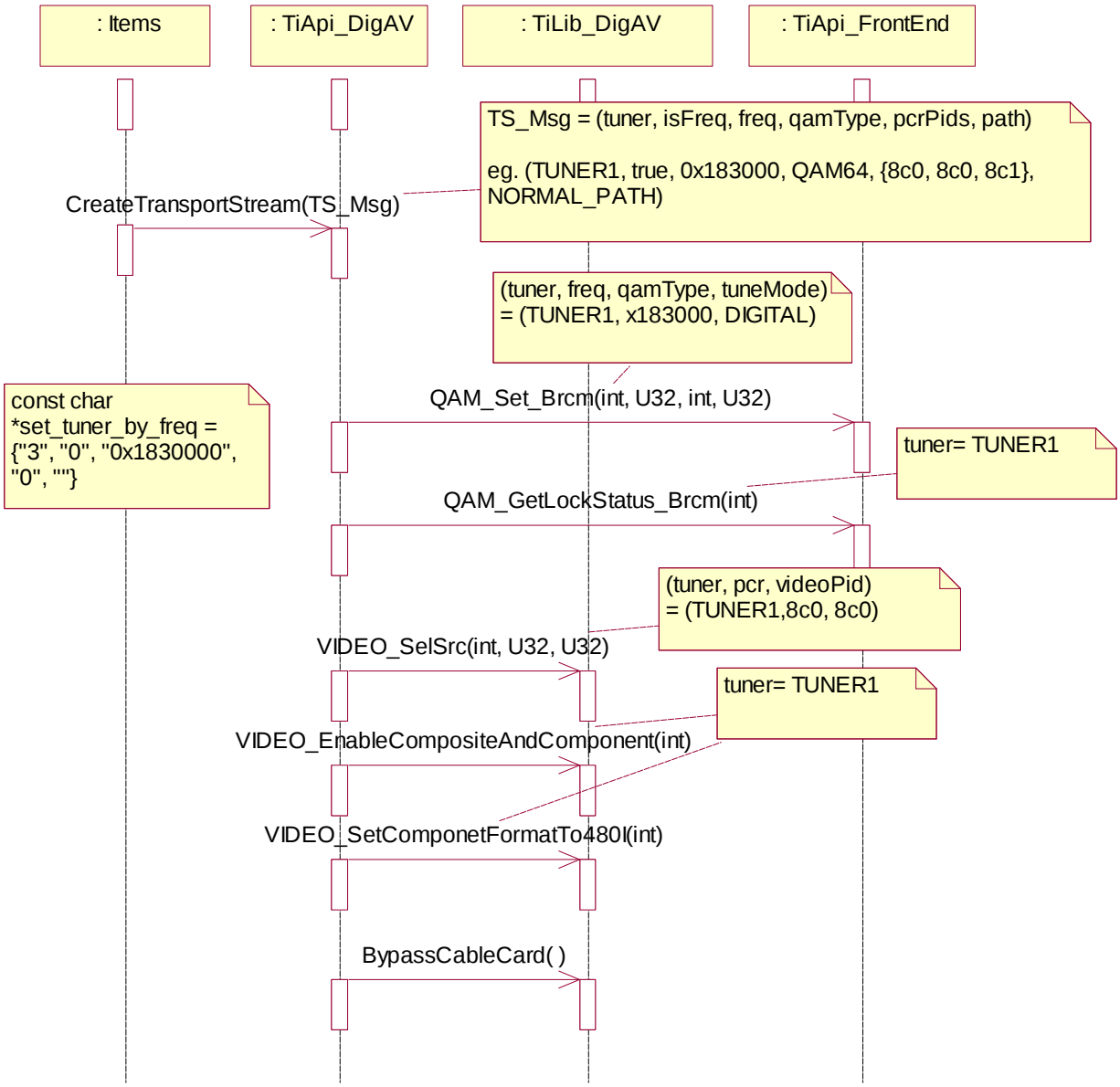
2. BLTC abandoned the solution that stop test as soon as it meet FAIL in run any test item. It run all test items of station and display error code. It's better for rework (find all bad part at one time).



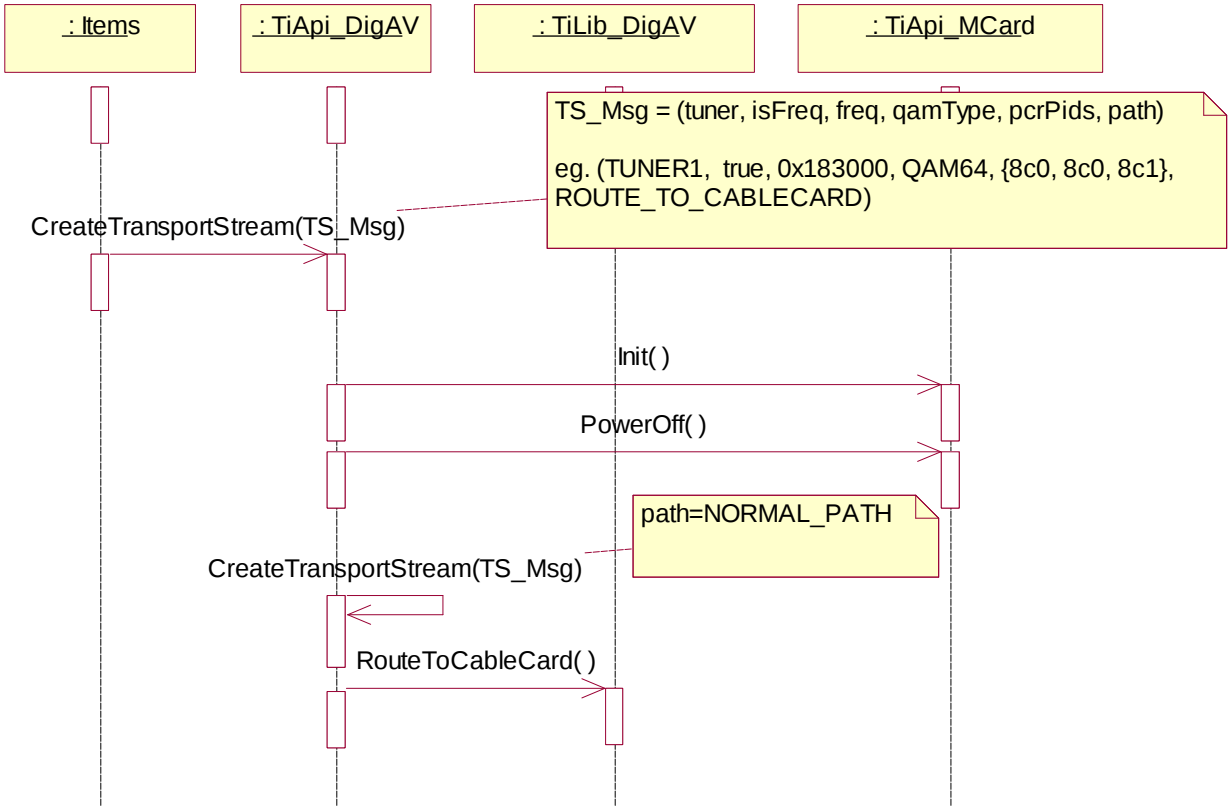
Run Item with Time Out::Process::Update() part,


items : Items
: Global Functions
process : Process
lib.clock : Timer_ISR
: TvMon
AsmU32 Process::RunTask(U32 (*pFunction)())
{
addiu sp,sp,-216
...
sw ra,(9*8)(sp)
...
}
IError = RunTask(U32)
Timer ISR: be called every 1 second.
InterruptHandler( )
TimerInterruptHandler( )
void Process::Update(Subject *theChangedSubject)
{
timeLimit--;
if (timeLimit <= 0) {
StopRun(1);
}
}
Timer_Tick( )
Tick( )
Notify( )
Update(Subject)
AsmVoid Process::StopRun(U32 aReturnValue)
{
// C++ compiler using a0 point to "this", a1 to first argument
addu v0, a1, $0 // a1 = aReturnValue
lw sp, 8(a0) // lw sp, context[1] = returnSP
lw ra, 12(a0) // lw sp, context[2] = returnAddr
jr ra // jump to label back_addr in RunTask()
nop
}
If over 5 seconds, it will restore registers and return to instruction on "loop next instruction" and set v0 (loop function return value) to 1.
StopRun(U32)
AsmU32 Process::RunTask(U32 (*pFunction)())
{
...
back_addr:
lw ra,(9*8)(sp)
nop
addiu sp,sp,216
jr ra
nop
}
jump to label back_addr in RunTask()
Observer = &process
IError = 1 (v0)
U32 Process::Create(U32 (*pFunction)(), int timeOut)
{
timeLimit = timeOut;
lib.clock.Attach(this);
ClearEXLBit();
U32 IError = RunTask(pFunction);
lib.clock.Detach(this);
return IError;
}
Return to lib.clock.Detach(this) inCreateProcess()
Return IError

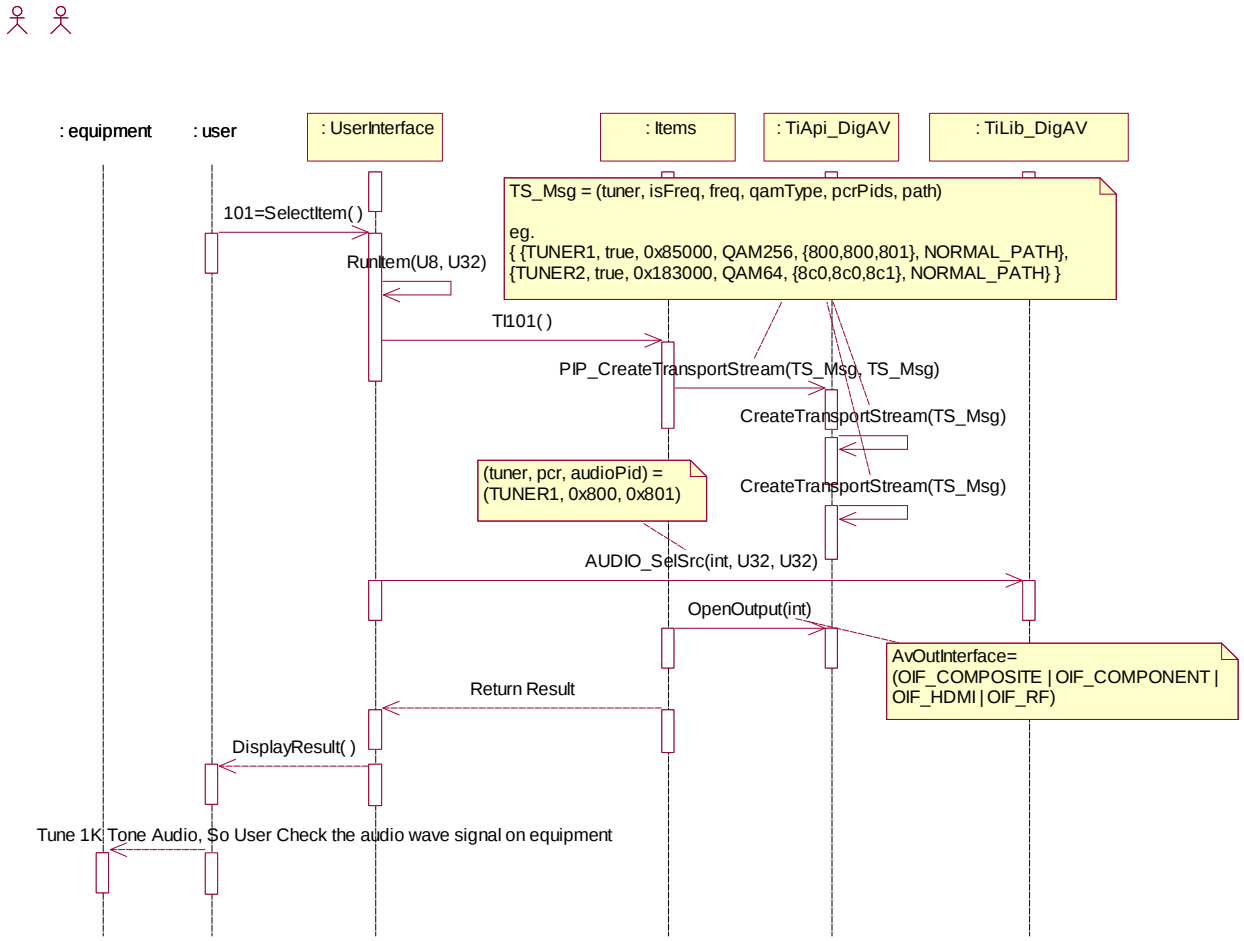
DigAV_CreateTransportStream(NORMAL_PATH)



DigAV_CreateTransportStream(ROUTE_TO_CABLECARD)



Digital AV PIP Test (Normal Path)



Digital AV Test (CableCard and 1394)