Document Title	**QIP7K BLTC Software High Level Design (HLD)**
Number	
Revision	**Z1**
Revision Date	**06/02/2008**
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

1. INTRODUCTION

1.1. Goal

- Define the software layer and API for each layer in BLTC.
- Expect them can work on different hardware platform, Broadcom software, Horsham + Broadcom software or any third party software.
- When change Broadcom software, only the layer which above on Broadcom software need changed.
- When User Interface changed, only the UI layer need to change.

1.2. Goal as BLTC in different stage

- In EPR
  - Test coverage as more as you can, test time is not an issue.
  - Support hardware debug test items as more as you can.
- In PPR,
  - Test coverage as more as you can, test time is not an issue, but must consider the test time in MP.
  - Support hardware debug test items as more as you can.
- In MP,
  - In required of test coverage condition, line rate as high as possible.

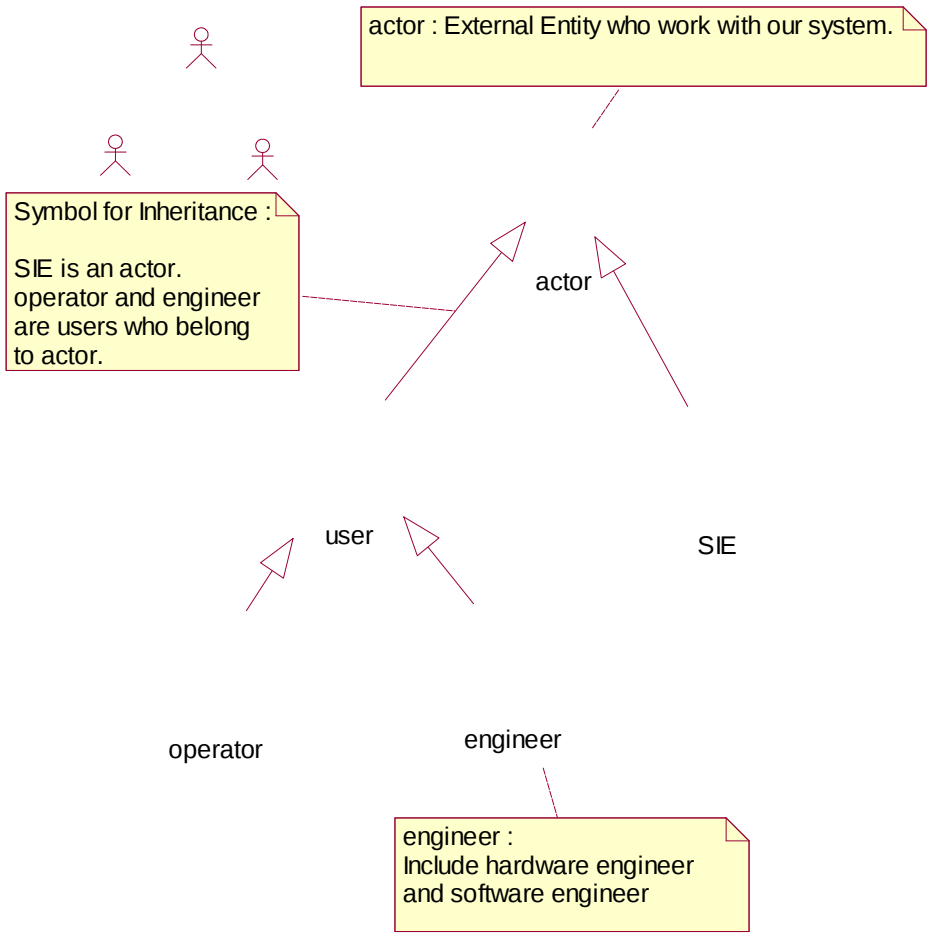
1.3. Definitions, Acronyms and Abbreviations

Acronym	Description
MTC	Manufacturing Test Code
BLTC	Board Level Test Code
OSD	On-Screen Display
STB	Set Top Box
IR	Infra-Red
SIE	System Integration Engineer
UI	User Interface

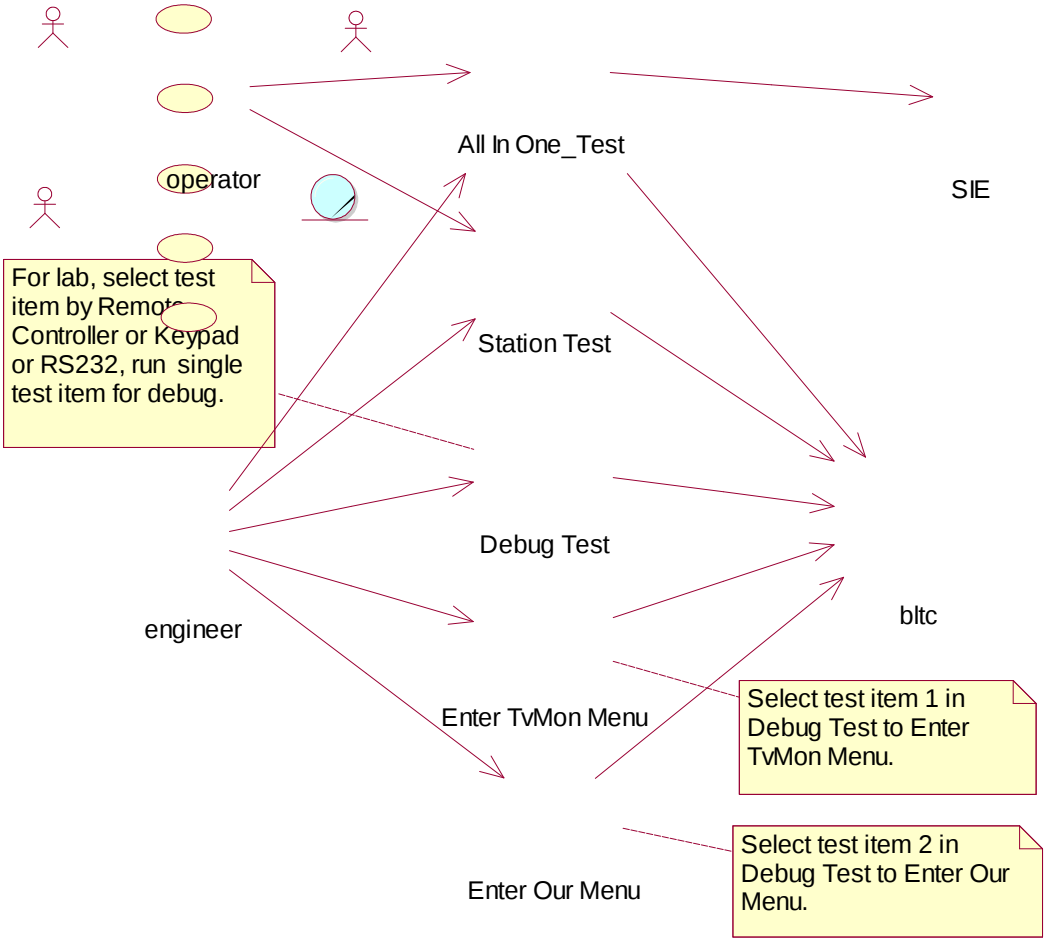
Table 1-1: Definitions, Acronyms, and Abbreviations

2. SYSTEM WORKING MODEL

2.1. Use Case Diagram — actor introduction



2.2. Use Case Diagram — working model



SIE : Program run on PC which communicate with BLTC which run on STB.

All In One Test: All In One Test. BLTC will do the testing and feed back test result to SIE, then SIE display the test result on PC screen. Operator got the test result from PC screen. According production-line plan, they want to do all test items in one single station. Every station can run all test items, it's more flexible to set up product-line. Currently has 3 stations, it's different with original plan.

Station Test: It's the currently test way in production-line. For example, they are 4 stations in QIP7xxx model. Production-line do Station 1 Test by short key "Cursor Up" of keypad on front panel (it's equal to the key "Cursor Up" be pressed). By the same way, Station 2 Test by key "Cursor Left", Station 3 by "Cursor Down", Station 4 by "Cursor Right". Each station can have 16 test items at most since it shows error code on LED and the LED for most product has only 4 characters (Each character can show 0..9 and A..F, meaning the 16 possibility in 4 test items. So, 4 characters can have 16 test items. For example, "0203" meaning error code is 0x0203 (in hex presentation), meaning error code is (0000 0010 0000 0011) in binary presentation, meaning 10th, 2nd and 1st Test Items is FAIL, others are PASS.

Debug Test: Engineer can run each test item in lab. Repair Engineer also needs this interface to fix problem. Based on past experience, the station test stuck on some test item sometimes in EPR. With this interface we can run Debug Test instead of Station Test for that station which meet the stuck problem in EPR. They can select test item by Remote Controller (Infra-red), Keypad or RS232 (Hyper Terminal).

Enter TvMon Menu: According past experience, the Horsham or Broadcom released software has a Menu (They call it TvMon Menu). You can enter TvMon menu by run item 1 in Debug Test. In TvMon Menu, you can do testing by browsing the menu via Hyper Terminal (RS232 interface), then input command in menu to do testing. It can do more testing for flash test and AV test (you can input different frequency and

pids (Program ID) to display all kinds of different video/audio channels. Hardware engineer can check/fix hardware issue by this menu in EPR or PPR stage.

Enter Our Menu: It's a menu for frequently testing item and special test items for engineer debugging. You can enter Our menu by run item 2 in Debug Test. It's option.

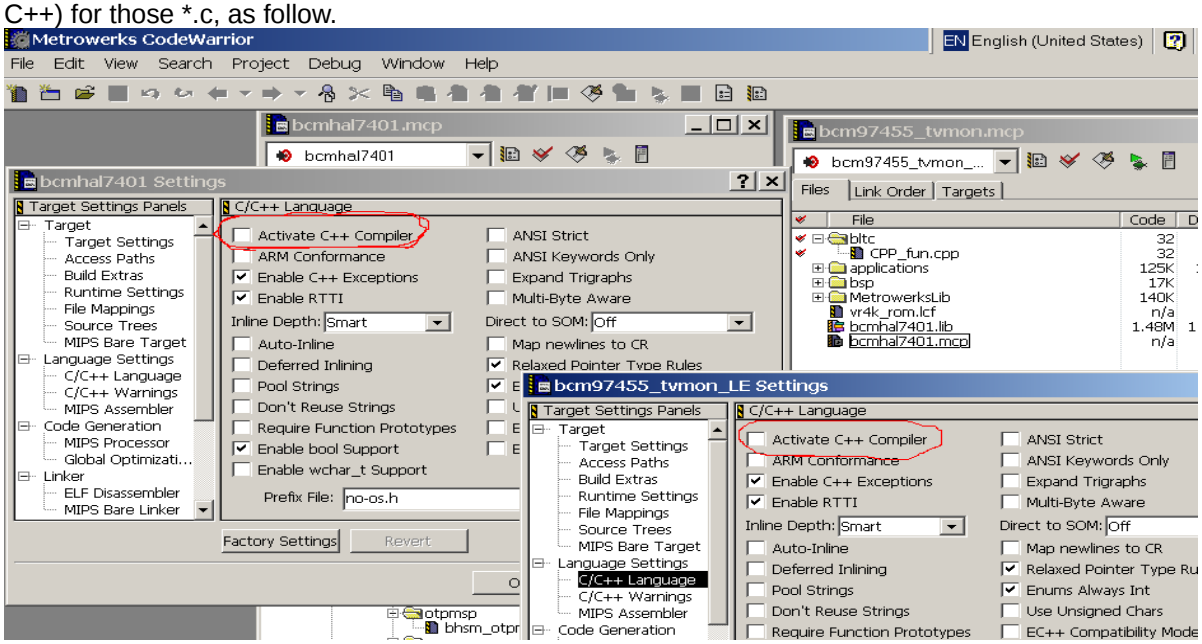
UI_DebugTest_Movie.wmv and UI_StationTest_Movie.wmv files are 2 video films which telling you how to operate it in Debug Test mode and Station Test mode. You can watch these films via Windows Media Player.

3. DEVELOPMENT LANGUAGE SELECT (USING C++ AND C)

Our BLTC team members try to come out a structure for BLTC design in different product model. Base on past experience, it's very sure, the BLTC software based on C/C++ language. But should we using C++ or not. After discussing, we choose to use C++. Follows are our analysis and conclusion.

3.1. Provider software analysis

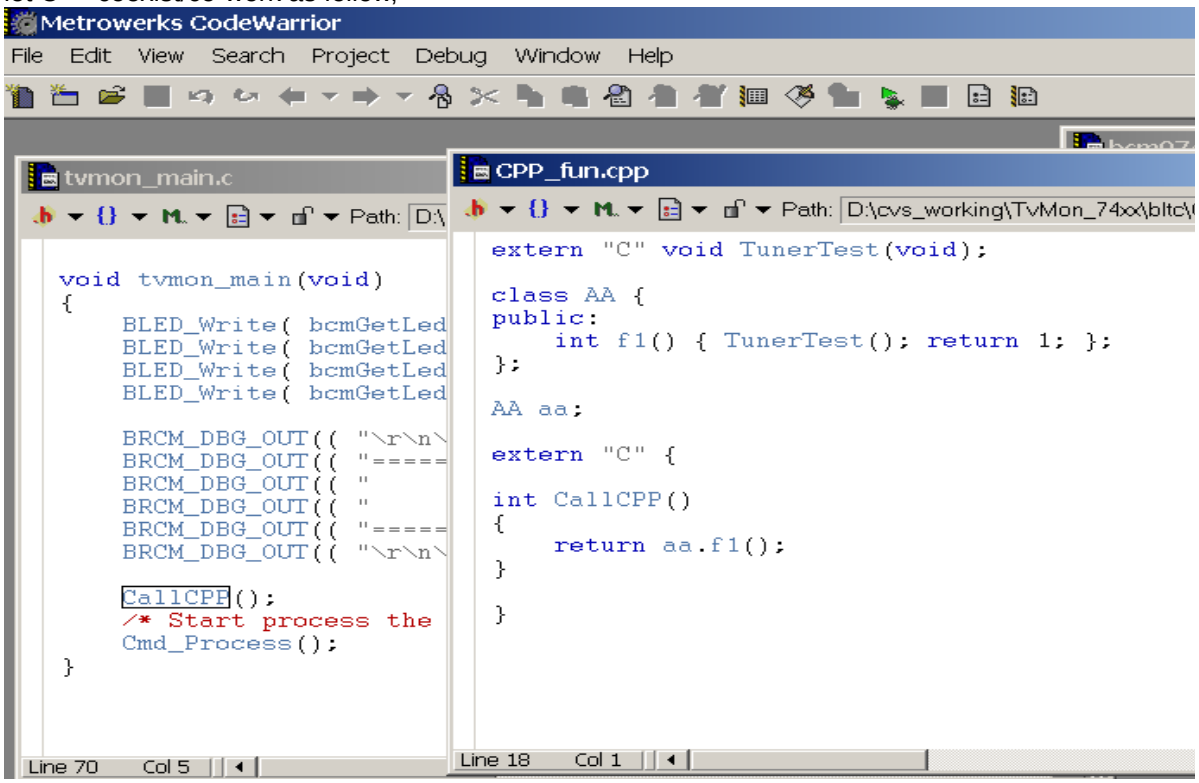
For Broadcom release software, TvMon_74xx (Code Warrior Project), only activate C compiler option (no C++) for those *.c, as follow.



According DCH_P5 experience, QIP7xxx and DCX34xx are same, it active C++ option for Application project (Horsham site did), and active C option for Library source code (Broadcom did).

3.2. Solution for C++ and C link together

We can keep the same setting (no matter C++ or C option is activated by Provider software project), and let C++ coexist/co-work as follow,



If we active C++ for the whole project, bcm97455_tvmon.mcp, will have 50 errors. They came from following,

1. C++ need “extern int funxxx()”.
2. C allow implicit pointer conversion while C++ do not,
   Eg.

   A a;
   int *pi;

   pi = &a; // error in C++, correct is pi = (int*)&a;

3. Argument type checking in function call.

If provider has many software version need to release in future then compiling will produce many errors when you active C++ compiler options. In this situation, our BLTC will keep the original setting and let C++ code coexists with provider’s software as above. If active C++ compiler options won’t cause many errors we can consider opening the C++ compiler option for the whole project.

The C++ and C coexists approach not limited in Code Warrior Compiler. It’s C++ standard. We can apply it to any C++/C compiler tool.

In QIP7xxx and DCX34xx, opening the C++ compiler option for the whole project will cause many errors.

3.3. Conclusion

1. If active C++ for those *.c will incurs many errors, and need much effort to remove those compiler errors, then we keep the project setting (no active C++), and add *.cpp to coexist with *.c.
2. If active C++ did not need much effort to remove those compiler errors, then active C++ for whole project.

4. SOFTWARE ARCHITECTURE

4.1. Software Layer

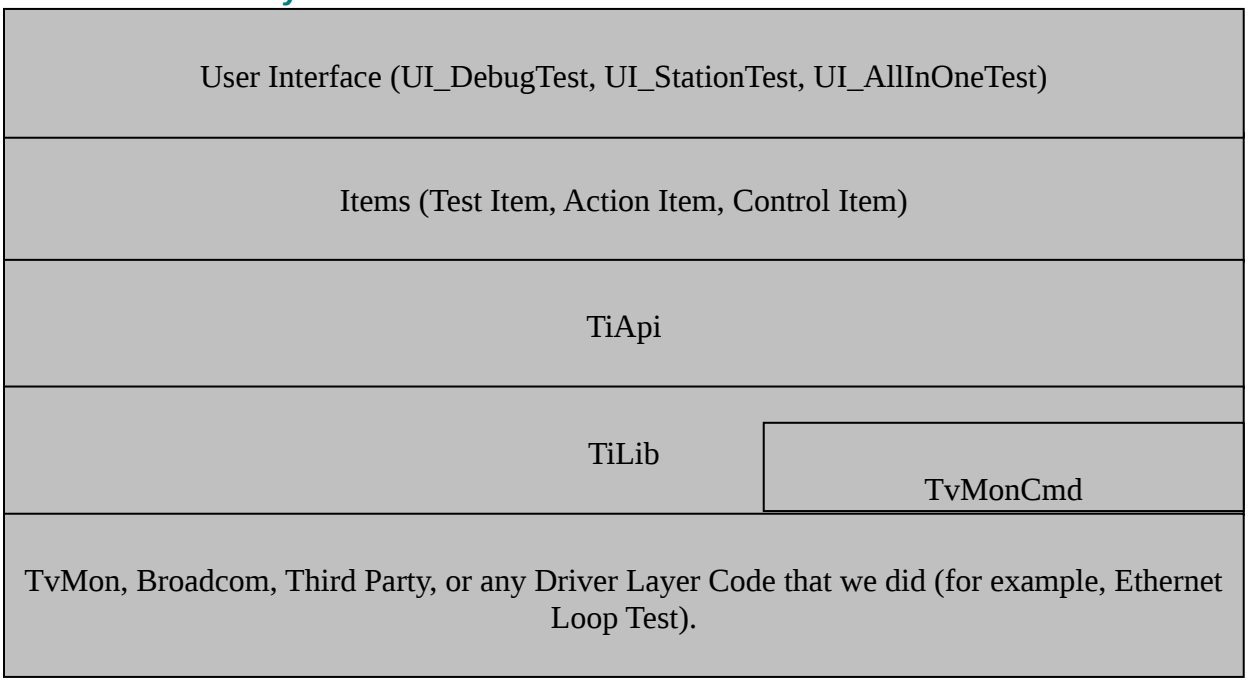


Figure 5-1: software layer

4.2. Flash Memory Map

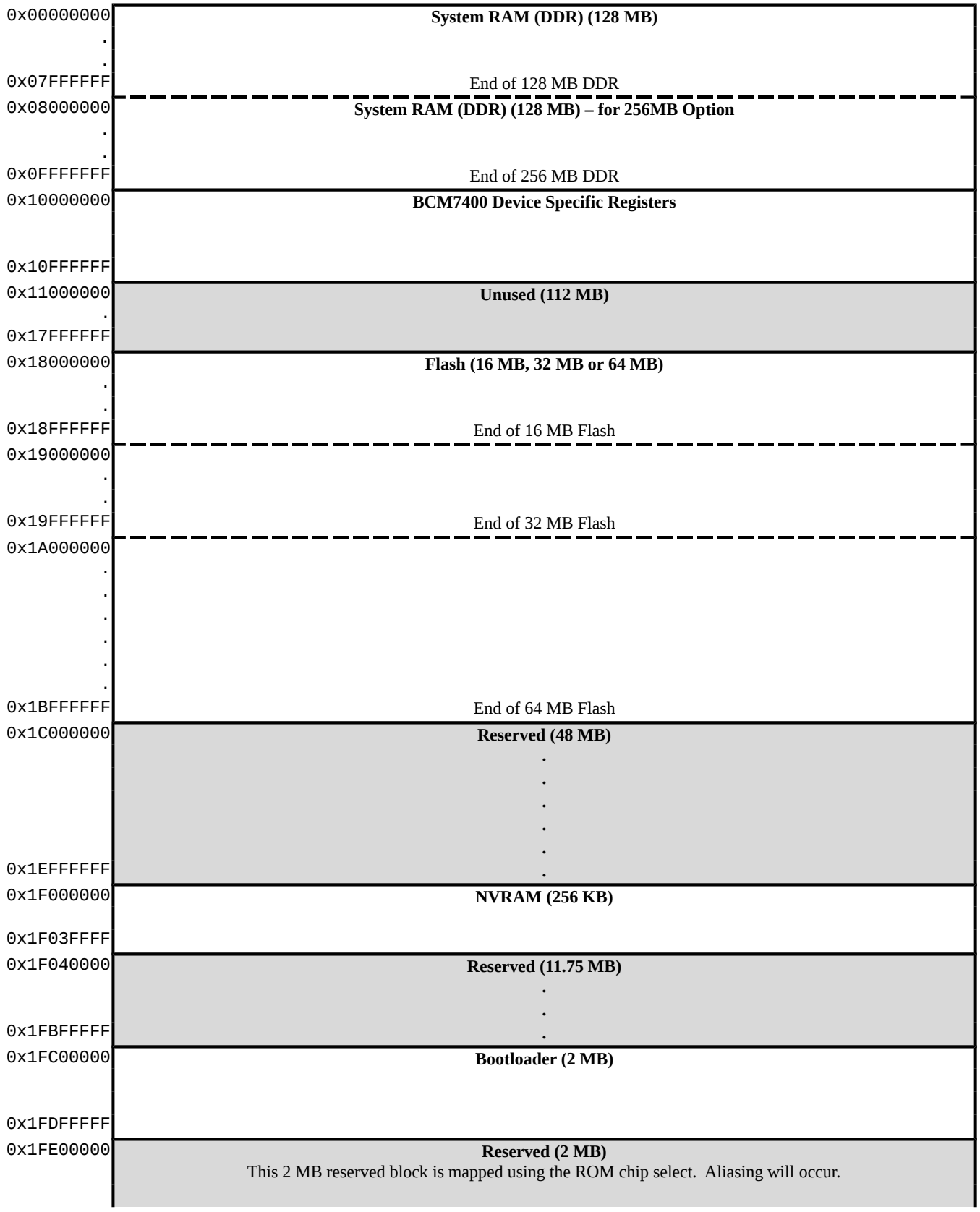
Refer to other document. For example, follows copy from Table 9:QIP7K Flash Memory Map of Motorola Document Number: ENB-5351, Revision:x.3_WIP, QIP7K Production Software Impact.

Offset	P4 USAGE (EPR and Later)	QIP7K USAGE (EPR and Later)	Offset
0x00000000	**Boot Code (128 kBytes)**	**Boot Code (128 kBytes)**	0x00000000
0x0001C000			
0x00020000	**Loader App (128 kBytes)** Executed and authenticated from Boot Menu Supports loading code to flash from OOB or a PC via Ethernet	**Loader App (128 kBytes)** Executed and authenticated from Boot Menu Supports loading code to flash from OOB or a PC via Ethernet	0x00020000 0x0003FFFF
	Reserved (96 kBytes)	**Reserved (96 kBytes)**	0x00040000 0x00057FFF
	Protected Flash Data (32 kBytes)	**Protected Flash Data (32 kBytes)**	0x00058000 0x0005FFFF
	SUDB/ECDS Back-Up (128 kBytes)	**SUDB/ECDS Back-Up (128 kBytes)**	0x00060000 0x0007FFFF
	Reserved (128 kBytes) This sector is reserved to keep P4 and P5 flash memory maps identical. This sector is used for CM data on the DCH P5 products.	**Reserved (128 kBytes)** For future use.	0x00080000 0x0009FFFF
	Platform Object (31.375 MB)	**Platform Object (31.375 MB)**	0x000A0000
0x0FFFFFFF			0x0FFFFFFF

Table 5-1: QIP7K Flash Memory Map – 32MB

4.3. Main Memory Map

Refer to other document. For example, follows copy from Table 8:QIP7K Main Memory Map of Motorola Document Number: ENB-5351, Revision:x.3_WIP, QIP7K Production Software Impact.


0x00000000
System RAM (DDR) (128 MB)
0x07FFFFFF
End of 128 MB DDR
0x08000000
System RAM (DDR) (128 MB) – for 256MB Option
0x0FFFFFFF
End of 256 MB DDR
0x10000000
BCM7400 Device Specific Registers
0x10FFFFFF
0x11000000
Unused (112 MB)
0x17FFFFFF
0x18000000
Flash (16 MB, 32 MB or 64 MB)
0x18FFFFFF
End of 16 MB Flash
0x19000000
0x19FFFFFF
End of 32 MB Flash
0x1A000000
0x1BFFFFFF
End of 64 MB Flash
0x1C000000
Reserved (48 MB)
0x1EFFFFFF
0x1F000000
NVRAM (256 KB)
0x1F03FFFF
0x1F040000
Reserved (11.75 MB)
0x1FBFFFFF
0x1FC00000
Bootloader (2 MB)
0x1FDFFFFF
0x1FE00000
Reserved (2 MB)
This 2 MB reserved block is mapped using the ROM chip select. Aliasing will occur.

0x1FFFFFFF	
0x20000000	**System RAM (DDR) (256 MB) – for 512MB Option**
0x2FFFFFFF	
0x30000000	**Non-KSEG0/KSEG1 Space**
.	.
.	.
.	.
0xB0510000	**BCM7400 PCI SATA Memory Window (TBD kB)**
.	
.	
0xB0520000	**BCM7400 PCI SATA IO Window (24 kB - TBR)**
.	
.	
0xB0525FFF	
0xB0526000	**Non-KSEG0/KSEG1 Space**
.	.
.	.
0xCFFFFFFF	.
0xD0000000	**BCM 7400 PCI Memory 0 (64 MB)** (32 bit word endian swapping) **Memory space for each PCI device is automatically sorted largest to smallest**
.	**and packed together starting at upper bound of the PCI Memory Window.**
.	
0xD7FFFFFF	
0xD8000000	**BCM 7400 PCI Memory 1 (64 MB)**
.	
.	
0xDFFFFFFF	
0xE0000000	**BCM 7400 PCI Memory 2 (64 MB)**
.	
.	
0xE7FFFFFF	
0xE8000000	**BCM 7400 PCI Memory 3 (64 MB)**
.	
.	
0xEFFFFFFF	
0xF0000000	**BCM7400 PCI I/O 0 (6MB)**
.	**I/O space for each PCI device (except PCI-ISA Bridge) is automatically sorted largest to smallest,**
.	**packed together, then allocated in descending order starting at upper bound of the PCI I/O Window.**
.	
0xF05FFFFF	(32 bit word endian swapping)
0xF0600000	**Non-KSEG0/KSEG1 Space**
.	.
0xFFFFFFFF	

Table 5-2: QIP7K Main Memory Map

4.4. Code Tree

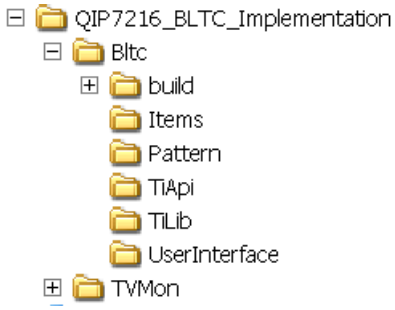


Figure 5-2: Code Tree

- Bltc

  - It all of BLTC code we program.

  - build

    - Include batch file for build BLTC uncompressed file (for Probe download debug used) and compressed file for pack with Boot Loader, MTC (Manufacture Test Code) and TC (Thin Client) into FFI (Factory Flash Image).

    - ReleaseNote document will tell you how to build them.

  - Items

    - Include all our test items.

  - Pattern

    - Include the class pattern we used. Refer book “Design Pattern”.

  - TiApi

    - Include api we designed for items layer. It’s called by items.

  - TiLib

    - Include the lower api we designed. It’s called by TiApi layer or above.

- TvMon

  - Include the files did by Broadcom and Horsham site.

5. USER INTERFACE

As 2.2 Use Case Diagram – Working Model indicated BLTC has 3 kinds of UI, Debug Test, Station Test and All In One Test.

5.1. User Interface Select

As 5.1 indicated, there are a couple of user interface in top layer. QIP7xxx BLTC have 3 Operational Test Mode (3 User Interface), All_In_One Test, Station Test and Debug Test (refer 2.2). EnterTvMonMenu and EnterOurMenu are entered by execute test item 1 and 2 respectively.

When no key be pressed on keypad, it enter the Debug Test Mode. When key “Cursor Right” and key “Power” be pressed on keypad, it enters the All In One Test Mode. When other key mentioned as above, it enter to Station Test Mode.

In Debug Test, there are 3 different kinds of input device in current STB (QIP7xxx has 3 input device, other may has one or more input device). Designed Debug test with composite pattern and decorate pattern structure (ref. Design Pattern Book), we can add or remove those ui_keypad, ui_ir, ui_rs232, and UI_TakeCareMCardOnOff in function, UserInterfaceSelect(), very easy. For example, if you only need ui_rs232 and ui_ir, you can configure it by following,

UI_IR	ui_ir(&outMsg);
UI_RS232	ui_rs232(&outMsg);
UI_Composite	ui_debugTest(&outMsg);

…

ui_ debugTest.Insert(&ui_ir);

ui_ debugTest.Insert(&ui_ rs232);

...

Figure 6-1 is the User Interface Select State Machine Present,

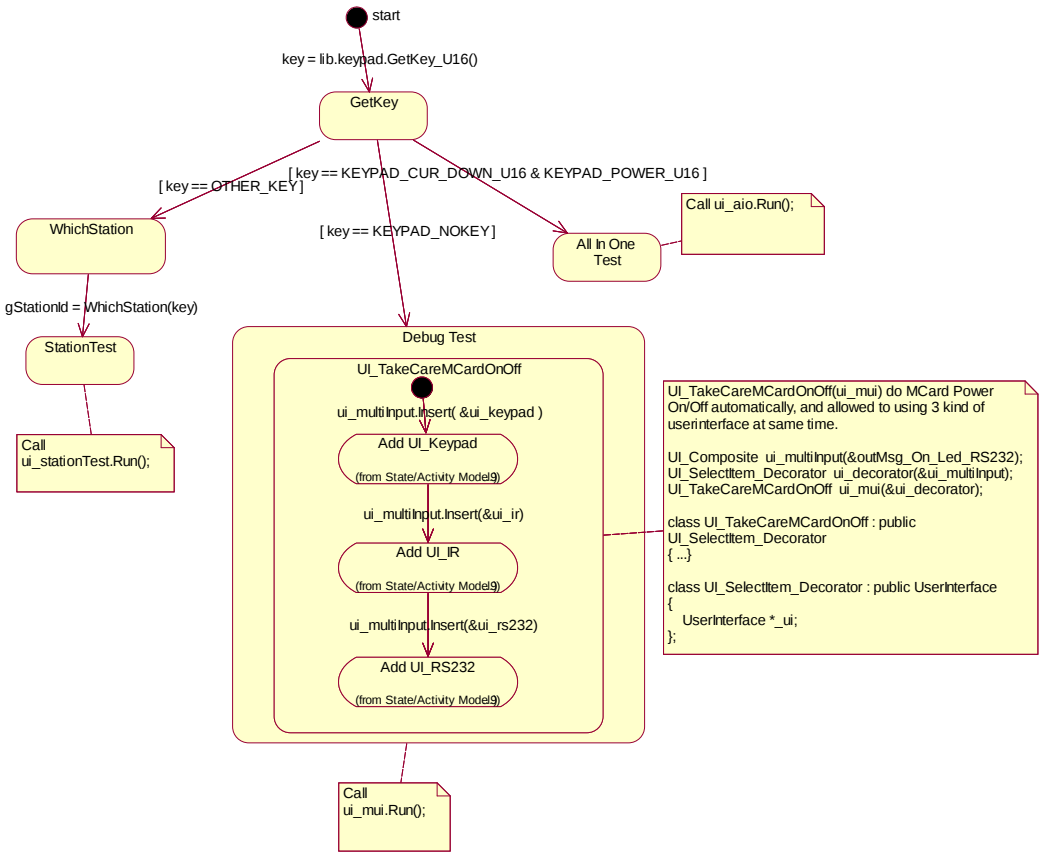


Figure 6-2: Station Machine for User Interface Select

For example, QIP7xxx keypad as follow,

Figure 6-2: QIP7xxx keypad

5.2. Debug Test

When no key be pressed on keypad, it enter the Debug Test Mode as Figure 6-2. It shows “-00-“, and now allow user select test item (scroll test item) by press “Channel Up” or “Channel Down” on keypad and run test item by press “Select”. Or press “Channel Up” or “Channel Down” on IR for Scroll test item and “OK” for run currently test item and press number key (key 0..9) to go to that test item number directly. Or by open a Hyper Terminal setting as (38400, 8, o, 1), and press key “↑” or “↓” to scroll, “0”-”9” to enter that test item directly and “Enter” to run currently test item.

5.3. Station Test

All the BLTC build batch file will call the build_station.bat which’s content as follow,

…

..\createstation.exe stationscr.txt > createstation.output

…

The file, stationsrc.txt, include the stations we used and items included in each station. File createstation.exe will produce output files, error_list.txt station_list.txt and CreateStation.cpp according the input file, stationscr.txt.

error_list.txt which include error code information, station_list which include error code, action items and control items information, CreateStation.cpp which include items and stations information in C++ code format and compile by compiler into ELF file.

The source code for createstation.exe are in Tool\ CreateStationSourceCode. It’s written by C++ and Yacc/Lex. The readme.txt in that directory tell you build procedure and environment setting for generate createstation.exe.

Pick up an example to show how it work,

Follows are the content from stationscr.txt,

// Follows for MODEL = "QIP7100"

MODEL = "QIP7100"

…

STATION 02

DESCRIPTION : "Using Cable Card loaded with Cable Card MTC (CCMTC)"

ITEM 42 87 88 89 93 92 33 34 35 37 74 54 55 201

…

For example, the error code “0203” meaning error code is 0x0203 (hex presentation), meaning error code is (0000 0010 0000 0011) in binary presentation, meaning 10th, 2nd and 1st Test Items is FAIL, other are PASS. If the error code “0203” is showed in station 2 test of QIP7100, it meaning test item 42:Flash Vendor ID & size Check FAIL, and test item 87:Ethernet External loop back test FAIL, and test item 37:NVSRAM data retention test (read back) (pattern = 0x55) FAIL since 0203 = 0001 + 0002 + 0200.

(0001) 42:Flash Vendor ID & size Check

(0002) 87:Ethernet External loop back test

(0200) 37:NVSRAM data retention test (read back) (pattern = 0x55)

The difference between error_list.txt and station_list.txt is that station_list.txt include Action Items and Control Items (both Action Items and Control Items has no return error code, only Test Item Type has return error code).

5.4. All In One Test

To do…

6. BLTC TEST ITEMS

6.1. Items introduction

BLTC defines 3 type of items. They are Test Item, Action Item and Control Item. Test Item will return error code (PASS or FAIL) and no parameter to pass; Action Item won’t return error code and no parameter to pass. Control Item no return error code and has a U32 type of parameter to pass. Pick up examples from QIP7K as follow,

Test Item 88 : "Rear panel USB port Test"

Action Item 51 : "Ask SIE to do VCXO test"

Control Item 253 : "delay_100ms(x)" // delay_100ms(20) => delay 2 seconds

These 3 type of items map to dir Bltc\Items. Above examples, they are implemented in files, i088.cpp, i051.cpp and i253.cpp, and in function name TI088 (Test Item 088), AI051 (Action Item 051) and CI253 (Control Item 253). As follow,

i088.cpp

```
U32 Items::TI088()
{
        U32 lError = 1;
        U32 (*pf)() = &USB_RearFanelPortTest;
```

```
        int timeOut = 5;   // unit:1s, => 5 seconds
        lError = PROCESS_CREATE_IF_DEF(pf, timeOut);

        return lError;
}
```

i253.cpp

```
void Items::CI253(U32 arg)
{
        if (arg != 0) {
                lib.DelayMS(arg*100);
        }

        return;
}
```

The file, Bltc\build\Items.txt, include all the test items we have implemented. Follows are the format for it,

```
/* --- Test Item List ---
# 1st number is test item number
# 2nd string is description
*/

TESTITEM_DECLARATION
{
// Test Items : Items with return Pass/Fail declared here
…
}

ACTIONITEM_DECLARATION
{
// Action Items : Items without return Pass/Fail declared here
// For example,
201 "Display error code on LED"
202 "Display error code on LED and return Error Code to SIE"
}
```

```
CONTROLITEM_DECLARATION
{
// Control Items : Items without return Pass/Fail and has a U32 type of argument declared here
253 "delay_100ms(x)"            // delay_100ms(20) => delay 2 seconds
}
```

Items defined in TESTITEM_DECLARATION {…} meaning it's Test Item Type, for example, 88:"Rear panel USB port Test"; Defined in ACTIONITEM_ DECLARATION {…} meaning it's Action Item Type, for example, 201: " Display error code on LED "; Defined in CONTROLITEM_DECLARATION {…} meaning it's Control Item Type, for example, 253:"delay_100ms(x)".