HSDK Getting Started Guide

Software Development Kit
BCM56880 B0 (Trident4 X11)
BCM56990 A0/B0 (Tomahawk4)
BCM56996 A0 (Tomahawk4G)
BCM56780 A0 (Trident4 X9)

GS-SDK6520-R
Jul, 2020

1. Introduction

This document provides the basic information that HSDK users need to know about the build environment requirements and the different debug shells that HSDK supports. It also provides step-by-step guidelines about how to build the HSDK image and run it on different platforms including XGSSIM, BCMSIM and Broadcom SLK/GTS SVKs.

2. Build Considerations

2.1 YAML Installation

An open source library (LibYAML) is used in HSDK, which is available at http://pyyaml.org/download/libyaml/yaml-0.1.7.tar.gz.

The SHA256 checksum for yaml-0.1.7.tar.gz is 8088e457264a98ba451a90b8661fcb4f9d6f478f7265d48322a196cec2480729.

The YAML library may be built and installed for the local host as shown below:

- `wget http://pyyaml.org/download/libyaml/yaml-0.1.7.tar.gz`

Let's assume you install this package in the directory /home/$USER/build

- `cd /home/$USER/build`
- `tar xf yaml-0.1.7.tar.gz`
- `cd yaml-0.1.7`
- `./configure --prefix=/home/$USER/build/x86_64/yaml && make install`

When building the HSDK library, the HSDK expects to find the YAML header file as `$YAML/include/yaml.h`. For example, if the header file is installed as `/home/$USER/build/yaml-0.1.7/include/yaml.h`, then the `$YAML` variable must be set as follows:

- `export YAML=/home/$USER/build/yaml-0.1.7`

The linker will look for the YAML library in the linker's default library path `$YAML.`

2.2 Build tools

The **minimum gcc version** required for HSDK is 4.6.3.

3. Broadcom Applications

3.1. BCM Shell

The BCM shell is an interactive application. It provides a means of higher-level configuration, and also can be used for diagnosis.

- Primary BCM (Broadcom Command Monitor) shell

  After initialization, the system enters primary BCM shell automatically.

```
BCM.0>
```

  Use ? to show the current supported commands.

```
BCM.0> ?
help: "??" or "help" for summary
Commands common to all modes:
?              ??             ASSert         Attach         BackGround
break          CASE           CD             cint           ClearScreen
CONFig         CONSole        CoPy           CPUDB          CTEcho
CTInstall      CTSetup        DATE           DBDump         DBParse
DeBug          DELAY          DIR            DISPatch       Echo
EXIT           EXPR           FLASHINIT      FLASHSYNC      FOR
Help           HISTory        IF             IMPORT         JOBS
KILL           LED            LOCal          LOG            LOOP
LS             MKDIR          MODE           MORe           MoVe
NOEcho         Pause          PRINTENV       PROBE          PTP
PWD            QUIT           RCCache        RCLoad         REBOOT
REName         RESET          RM             RMDIR          RPC
SalProfile     SAVE           SET            SETENV         SHell
SLeep          TDPLL          TIME           Version
Commands for current mode:
ALPM           ASF            BcmltSHell     CLEAR          COS
CounTeR        CustomSTAT     DebugSHell     DETach         Dump
EventSTAT      FieldProcessor FlexCTR        FlexDigest     FLEXport
```

```
FlowDb         HA             INIT           IPMC           L2
L3             LINKscan       MemSCAN        MIRror         MPLS
MultiCast      PacketWatcher  PBMP           PORT           PortStat
PVlan          RATE           RATEBW         SER            SHOW
SramSCAN       StackPortGet   StackPortSet   STG            STKMode
SwitchControl  TRace          TRUNK          TX             VLAN
WARMBOOT
Dynamic commands for all modes:
xmem
```

3.2. SDKLT shell

SDKLT shell can be invoked from the primary BCM shell and provides a set of low level commands that are intended for specialized LT(Logical Table) use cases. Please note that SDKLT shell should be only used for advanced pipeline debugging, knowledge of SDKLT is required in order to use this shell. **Do not modify Logical or Physical tables via this shell unless directed by Broadcom support.**

Use bsh (abbreviation for **B**cmlt**SH**ell) in order to enter the SDKLT shell.

```
BCM.0> bsh
BCMLT.0>
```

Use exit in order to return to primary BCM shell.

```
BCMLT.0> exit
BCM.0>
```

Use ? to show the supported commands.

```
BCMLT.0> ?
Available commands: pmddecode, knet, PortStatus, TEChSUPport, ConFiG, CounTeR,
LtCAPture, pt, lt, DebugSHell, history, local, setenv, printenv, sort, grep, shell,
version, rmdir, mkdir, ls, pwd, cd, RCLoad, RCCache, time, sleep, delay, expr, case,
if, each, for, loop, echo, ?, Help, Exit, Quit
```

The command dsh (abbreviation for **D**ebug**SH**ell) can be used to enter the SDKLT debug shell for TR test.

```
BCMLT.0> dsh
sdklt.0>
```

HSDK configures the Logical Table Pipeline using SDKLT APIs. The Logical Table definitions are documented in the Logical Table Reference Guide.

3.3. BCM CINT

CINT is a Broadcom proprietary interactive and script language interpreter of a C-like language. The CINT interpreter provides a way to enable rapid prototyping and debugging of applications based on the BCM API. It allows user to write an entire C script and execute it or enter statements and function definitions interactively.

Use cint in order to enter the CINT.

```
BCM.0> cint
Entering C Interpreter. Type 'exit;' to quit.

cint>
```

Use exit in order to enter the primary BCM shell.

```
cint> exit;
BCM.0>
```

4. Building and Executing

4.1. XGSSIM

A built-in register / memory simulator is embedded in XGSSIM, which can be used for the most basic API testing. The benefit of the simulator is that it allows a user to develop their application without the need for a physical switch device. **The simulator does not provide any packet modeling and is mainly useful for control plane API testing**.

4.1.1 Building HSDK for XGSSIM

The HSDK image for XGSSIM can be built and executed on your local linux system. For example in linux X86, here are the detailed steps.

1. Compile HSDK for XGSSIM
    a. Export the `SDK` environment variable to the top of the HSDK source tree:
        - `export SDK=$PWD`
    b. Export the `MAKE_LOCAL` environment variable to make.local file:
        - For BCM56880 device
            - `export MAKE_LOCAL=$SDK/make/local/hsdk/Make.pkg.56880`
        - For BCM56990 device
            - `export MAKE_LOCAL=$SDK/make/local/hsdk/Make.pkg.56990`
        - For BCM56996 device
            - `export MAKE_LOCAL=$SDK/make/local/hsdk/Make.pkg.56996`
        - For BCM56780 device
            - `export MAKE_LOCAL=$SDK/make/local/hsdk/Make.pkg.56780`
    c. Change directory to the `SDK`:
        - `cd $SDK`
    d. Build the HSDK image for the XGSSIM target (Note if using the precompiled binary you can skip this step):
        - `make -C systems/xgssim`
2. Once the build is complete, the HSDK image will be available in `$SDK/systems/xgssim`
3. Copy `ltsw.soc` file
    - `cp $SDK/rc/ltsw.soc $SDK/systems/xgssim`
4. Copy the port configuration YAML file
    - For BCM56880 device

    - `cp $SDK/rc/yaml/bcm56880_a0/bcm56880_a0-generic-32x400.config.yml $SDK/systems/xgssim`
- For BCM56990 A0 device
    - `cp $SDK/rc/yaml/bcm56990_a0/bcm56990_a0-generic-64x400.config.yml  $SDK/systems/xgssim`
    - `cp $SDK/rc/yaml/bcm56990_a0/bcm56990_a0-generic-system_port.config.yml $SDK/systems/xgssim`
- For BCM56990 B0 device
    - `cp $SDK/rc/yaml/bcm56990_b0/bcm56990_b0-generic-64x400.config.yml  $SDK/systems/xgssim`
    - `$SDK/rc/yaml/bcm56990_b0/bcm56990_b0-generic-system_port.config.yml $SDK/systems/xgssim`
- For BCM56996 device
    - `cp $SDK/rc/yaml/bcm56996_a0/bcm56996_a0-generic-64x400.config.yml  $SDK/systems/xgssim`
    - `cp $SDK/rc/yaml/bcm56996_a0/bcm56996_a0-generic-system_port.config.yml $SDK/systems/xgssim`
- For BCM56780 device
    - `cp $SDK/rc/yaml/bcm56780_a0/bcm56780_a0-generic-20x400.config.yml $SDK/systems/xgssim`
    - `cp $SDK/rc/yaml/bcm56780_a0/bcm56780_a0-generic-bcm-device.config.yml $SDK/systems/xgssim`

5. Export `SOC` environment variables
    - `export SOC_TARGET_COUNT=0`
    - `export SOC_BOOT_FLAGS=0x800000`
6. Invoke the HSDK image from `$SDK/systems/xgssim` as follows:
    - `cd $SDK/systems/xgssim`
    - Run BCM56880
        - `./bcm.xgssim -o bcm56880_b0 -phymodsim -y bcm56880_a0-generic-32x400.config.yml`
    - Run BCM56990 A0
        - `./bcm.xgssim -o bcm56990_a0 -phymodsim -y bcm56990_a0-generic-64x400.config.yml -y bcm56990_a0-generic-system_port.config`
    - Run BCM56990 B0

  - `./bcm.xgssim -o bcm56990_b0 -phymodsim -y bcm56990_b0-generic-64x400.config.yml -y bcm56990_b0-generic-system_port.config`
- Run BCM56996
  - `./bcm.xgssim -o bcm56996_a0 -phymodsim -y bcm56996_a0-generic-64x400.config.yml -y bcm56996_a0-generic-system_port.config`
- Run BCM56780
  - `./bcm.xgssim -o bcm56780_a0 -phymodsim -y bcm56780_a0-generic-20x400.config.yml -y bcm56780_a0-generic-bcm-device.config.yml`

The message below is the output for the BCM56880 device, which may be slightly different based on the devices you are running.

```
Platform: unix-xgssim
OS: Unix (Posix)
Found 1 device.
Unit 0: BCM56880
NGBDE unit 0 (PCI), Dev 0xb880, Rev 0x11, Chip BCM56880_B0, Driver LTSW
Boot flags: Cold boot
rc: unit 0 device BCM56880_B0
rc: BCM driver initialized
rc: Port modes initialized
BCM.0>
```

4.2. BCMSIM

BCMSIM provides a device level control and data plane simulator which can be used for API testing. The benefit of the simulator is that it allows a user to develop their application without the need for a physical switch device.

4.2.1 Building HSDK for BCMSIM

The following steps can be used to build HSDK image for BCMSIM.

1. Export the `SDK` environment variable to the top of the HSDK source tree:
   - `export SDK=$PWD`
   - `ls`
     `Top of HSDK Source Tree`

```
Makefile  RELDOCS  RELEASE  doc  include  libs  make  rc  src  systems  tools
```

2. Export the `MAKE_LOCAL` environment variable to make.local file:
   - For BCM56880 device
     - `export MAKE_LOCAL=$SDK/make/local/hsdk/Make.pkg.56880`
   - For BCM56990 device
     - `export MAKE_LOCAL=$SDK/make/local/hsdk/Make.pkg.56990`
   - For BCM56996 device
     - `export MAKE_LOCAL=$SDK/make/local/hsdk/Make.pkg.56996`
   - For BCM56780 device
     - `export MAKE_LOCAL=$SDK/make/local/hsdk/Make.pkg.56780`
3. Change directory to the `SDK`:
   - `cd $SDK`
4. Build the HSDK image for the BCMSIM target (If using the precompile binary then skip this step):
   - `make -C systems/sim bcm.sim`

The following is a quick reference guide to start the BCMSIM. For more details on BCMSIM, please refer to BCMSIM documentation that comes with BCMSIM release package. Running BCMSIM requires **three** terminals. Setup the environment. In this example we will assume that the BCMSIM was installed at `/projects/bcmsim.` And the component description file can also be found in the bcmsim release folder.

- For example on BCM56880 device
  - `/projects/bcmsim/framework/configuration/bcm56880.cdf`

1. Start the BCMSIM framework in Linux **terminal 1**:

   - `cd $SDK`
   - `export BCMSIM_ROOT=/projects/bcmsim/release/framework`
   - `export BCMSIM_CXM_SERVER=localhost`
   - `export BCMSIM_CXM_PORT=7777`
   - For BCM56880 device
     - `$BCMSIM_ROOT/bin/bcmsim -c $BCMSIM_ROOT/configuration/bcm56880.cdf`
   - For BCM56990 device
     - `$BCMSIM_ROOT/bin/bcmsim -c $BCMSIM_ROOT/configuration/bcm56990.cdf`
   - For BCM56996 device
     - `$BCMSIM_ROOT/bin/bcmsim -c $BCMSIM_ROOT/configuration/bcm56996.cdf`
   - For BCM56780 device
     - `$BCMSIM_ROOT/bin/bcmsim -c $BCMSIM_ROOT/configuration/bcm56780.cdf`

```
CLI>
```

2. Start the BCMSIM model in Linux **terminal 2**:

    - `cd $SDK`
    - `export BCMSIM_ROOT=/projects/bcmsim/release/framework`
    - `export BCMSIM_CXM_SERVER=localhost`
    - `export BCMSIM_CXM_PORT=7777`
    - `export SOC_TARGET_PORT=8888`
    - Run BCM56880
        - `$BCMSIM_ROOT/bin/bcm56880_bcmsim.linux $BCMSIM_CXM_SERVER $BCMSIM_CXM_PORT 10 $SOC_TARGET_PORT`
    - Run BCM56990_A0
        - `$BCMSIM_ROOT/bin/bcm56990_bcmsim.linux $BCMSIM_CXM_SERVER $BCMSIM_CXM_PORT 10 $SOC_TARGET_PORT`
    - Run BCM56990_B0
        - `$BCMSIM_ROOT/bin/bcm56990_b0_bcmsim.linux $BCMSIM_CXM_SERVER $BCMSIM_CXM_PORT 10 $SOC_TARGET_PORT`
    - Run BCM56996
        - `$BCMSIM_ROOT/bin/bcm56996_bcmsim.linux $BCMSIM_CXM_SERVER $BCMSIM_CXM_PORT 10 $SOC_TARGET_PORT`
    - Run BCM56780
        - `$BCMSIM_ROOT/bin/bcm56780_bcmsim.linux $BCMSIM_CXM_SERVER $BCMSIM_CXM_PORT 10 $SOC_TARGET_PORT`

The message below may be slightly different depending on the version of BCMSIM you are running.

```
*******************************************************
* BROADCOM
* Copyright (c) 2002-2019
* ALL RIGHTS RESERVED
*******************************************************
*
*******************************************************
* BCMSIM DEVICE MODEL
* BCMSIM_2020_04_03_02_22_07
* Build Timestamp : Apr  3 2020 03:58:24
*******************************************************

INFO: Logger enabled.

Setting current debug level to 1.
cmic_module0:Starting simulation
```

3. Start the HSDK in Linux **terminal 3**:

- `cd $SDK`
- `export BCMSIM_CXM_SERVER=localhost`
- `export SOC_TARGET_PORT=8888`
- `cd $SDK/systems/sim`
- `cp $SDK/rc/ltsw.soc ./`
- For BCM56880 device
  - `cp $SDK/rc/yaml/bcm56880_a0/bcm56880_a0-generic-32x400.config.yml ./`
- For BCM56990 A0 device
  - `cp $SDK/rc/yaml/bcm56990_a0/bcm56990_a0-generic-64x400.config.yml ./`
  - `cp $SDK/rc/yaml/bcm56990_a0/bcm56990_a0-generic-system_port.config.yml ./`
- For BCM56990 B0 device
  - `cp $SDK/rc/yaml/bcm56990_b0/bcm56990_b0-generic-64x400.config.yml ./`
  - `cp $SDK/rc/yaml/bcm56990_b0/bcm56990_b0-generic-system_port.config.yml ./`
- For BCM56996 device
  - `cp $SDK/rc/yaml/bcm56996_a0/bcm56996_a0-generic-64x400.config.yml ./`
  - `cp $SDK/rc/yaml/bcm56996_a0/bcm56996_a0-generic-system_port.config ./`
- For BCM56780 device
  - `cp $SDK/rc/yaml/bcm56780_a0/bcm56780_a0-generic-20x400.config.yml ./`
  - `cp $SDK/rc/yaml/bcm56780_a0/bcm56780_a0-generic-bcm-device.config.yml ./`
- `export SOC_TARGET_COUNT=1`
- `export SOC_BOOT_FLAGS=0x420000`
- Run BCM56880
  - `./bcm.sim -phymodsim -y bcm56880_a0-generic-32x400.config.yml`
- Run BCM56990 A0
  - `./bcm.sim -phymodsim -y bcm56990_a0-generic-64x400.config.yml -y bcm56990_a0-generic-system_port.config`
- Run BCM56990 B0
  - `./bcm.sim -phymodsim -y bcm56990_b0-generic-64x400.config.yml -y bcm56990_b0-generic-system_port.config`
- Run BCM56996
  - `./bcm.sim -phymodsim -y bcm56996_a0-generic-64x400.config.yml -y bcm56996_a0-generic-system_port.config`
- Run BCM56780

  - `./bcm.sim -phymodsim -y bcm56780_a0-generic-20x400.config.yml -y bcm56780_a0-generic-bcm-device.config.yml`

The message below is the output for the BCM56880 device, which may be slightly different based on the devices you are running.

```
Platform: unix-linux-64
OS: Unix (Posix)
Starting DMA service...
[DMA-listener]DMA Controller listening on port[45903]
Starting Interrupt service...
[Interrupt-listener]ISR dispatcher listening on port[40081]
Found 1 device.
Unit 0: BCM56880
NGBDE unit 0 (PCI), Dev 0xb880, Rev 0x01, Chip BCM56880_A0, Driver LTSW
Boot flags: Cold boot
rc: unit 0 device BCM56880_A0
rc: BCM driver initialized
rc: Port modes initialized
BCM.0>
```

4.3. SVK support

SVK is a real device for system verification. Multiple CPU platforms can be supported. Below are the steps of building and running HSDK image on Broadcom SVKs based on SLK or GTS CPU platform.

4.3.1. Building HSDK for SVK

HSDK image can be built based on the type of CPU platform and executed on SVK. For example in SLK platform, here are the detailed steps.

1. Compile HSDK for SLK
   a. Export the `SDK` environment variable to the top of the HSDK source tree:
      - `export SDK=$PWD`
   b. Export the `MAKE_LOCAL` environment variable to make.local file:
      - For BCM56880 device
        - `export MAKE_LOCAL=$SDK/make/local/hsdk/Make.pkg.56880`
      - For BCM56990 device
        - `export MAKE_LOCAL=$SDK/make/local/hsdk/Make.pkg.56990`
      - For BCM56996 device
        - `export MAKE_LOCAL=$SDK/make/local/hsdk/Make.pkg.56996`
      - For BCM56780 device
        - `export MAKE_LOCAL=$SDK/make/local/hsdk/Make.pkg.56780`

Note - the make.local file can be created locally for specific build. For example, `Make.pkg.hybrid` file can be created locally in `$SDK/make/local/hsdk/` for hybrid build (supporting BCM56880, BCM56980, BCM56990, BCM56780 devices) .

```
FEATURE_LIST=PTP CINT L3 I2C MEM_SCAN EDITLINE BCM_SAL_PROFILE CUSTOMER TEST CHASSIS
MSTP KNET TCB PSTATS FLOWTRACKER IFA COLLECTOR TELEMETRY INT

BCM_PTL_SPT = 1
BCM_56880_A0 = 1
BCM_56980_A0 = 1
BCM_56990_A0 = 1
BCM_56990_B0 = 1
BCM_56996_A0 = 1
BCM_56780_A0 = 1

CFGFLAGS += -DBCM_WARM_BOOT_SUPPORT
CFGFLAGS += -DBCM_WARM_BOOT_SUPPORT_SW_DUMP
CFGFLAGS += -DALPM_ENABLE
CFGFLAGS += -DNO_CONTROL_C
```

Accordingly `MAKE_LOCAL` environment shall be defined as follows

- `export MAKE_LOCAL=$SDK/make/local/hsdk/Make.pkg.hybrid`

c. Change directory to the `SDK`:
   - `cd $SDK`

d. Build the HSDK image for the SLK LE target (Note if using the precompiled binary you can skip this step):
   - `make -C systems/linux/user/slk`

   Note - please change the target path for other CPU targets. For example, the image for GTS target can be built as follows.
   - `make -C systems/linux/user/gts`

2. Once the build is complete, the HSDK image will be available in `$SDK/systems/linux/user/slk`
3. Copy `ltsw.soc` file
   - `cp $SDK/rc/ltsw.soc $SDK/systems/linux/user/slk`
4. Copy the port configuration YAML file
   - For BCM56880 device
     - `cp $SDK/rc/yaml/bcm56880_a0/bcm56880_a0-generic-32x400.config.yml $SDK/systems/linux/user/slk`
   - For BCM56990 A0 device
     - `cp $SDK/rc/yaml/bcm56990_a0/bcm56990_a0-generic-64x400.config.yml $SDK/systems/linux/user/slk`

    - `cp $SDK/rc/yaml/bcm56990_a0/bcm56990_a0-generic-system_port.config.yml $SDK/systems/linux/user/slk`
  - For BCM56990 B0 device
    - `cp $SDK/rc/yaml/bcm56990_b0/bcm56990_b0-generic-64x400.config.yml $SDK/systems/linux/user/slk`
    - `cp $SDK/rc/yaml/bcm56990_b0/bcm56990_b0-generic-system_port.config $SDK/systems/linux/user/slk`
  - For BCM56996 device
    - `cp $SDK/rc/yaml/bcm56996_a0/bcm56996_a0-generic-64x400.config.yml $SDK/systems/linux/user/slk`
    - `cp $SDK/rc/yaml/bcm56996_a0/bcm56996_a0-generic-system_port.config $SDK/systems/linux/user/slk`
  - For BCM56780 device
    - `cp $SDK/rc/yaml/bcm56780_a0/bcm56780_a0-generic-20x400.config.yml $SDK/systems/linux/user/slk`
    - `cp $SDK/rc/yaml/bcm56780_a0/bcm56780_a0-generic-bcm-device.config.yml $SDK/systems/linux/user/slk`
5. Go to SVK and invoke the HSDK image from `$SDK/systems/linux/user/slk` as follows:
  - `cd $SDK/systems/linux/user/slk`
  - Run BCM56880
    - `./bcm.user -y bcm56880_a0-generic-32x400.config.yml`
  - Run BCM56990 A0
    - `./bcm.user -y bcm56990_a0-generic-64x400.config.yml -y bcm56990_a0-generic-system_port.config`
  - Run BCM56990 B0
    - `./bcm.user -y bcm56990_b0-generic-64x400.config.yml -y bcm56990_b0-generic-system_port.config`
  - Run BCM56996
    - `./bcm.user -y bcm56996_a0-generic-64x400.config.yml -y bcm56996_a0-generic-system_port.config`
  - Run BCM56780
    - `./bcm.user -y bcm56780_a0-generic-20x400.config.yml -y bcm56780_a0-generic-bcm-device.config.yml`

The message below is the output for the BCM56880 device, which may be slightly different based on the devices/platforms you are running.

```
Platform: SLK_BCM957812
OS: Unix (Posix)
Found 1 device.
Unit 0: BCM56880
NGBDE unit 0 (PCI), Dev 0xb880, Rev 0x11, Chip BCM56880_B0, Driver LTSW
linux-user-bde: no devices
DMA pool size: 33554432
Boot flags: Cold boot
rc: unit 0 device BCM56880_B0
rc: BCM driver initialized
rc: Port modes initialized
BCM.0>
```

5. Example CINT scripts

The following CINT scripts located in

- `$SDK/src/examples/ltsw/trident4`
- `$SDK/src/examples/ltsw/tomahawk4`
- `$SDK/src/examples/ltsw/tomahawk4g`

can be used to test some basic HSDK functionality.

The CINT scripts are not available yet for the device BCM56780 in preview.

5.1. Troubleshooting

If you see the following error when running any of the example scripts, be sure that the init script ltsw.soc and bcm.user are in the same directory.

```
Error: Could not get port config, info: Invalid unit
Invalid port bitmap "cd6"
```