VISION SDK
Use-Case Auto-Generation Tool

User Guide

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Table of Contents

1 Introduction

Vision SDK Use-Case Auto Generation tool allows users to generate C code for Vision SDK use-cases from configuration file.

This document explains procedure for writing configuration files and generating use-case files

This document assumes that the reader is familiar with basics of links and chains architecture used in Vision SDK.

2 Requirements

Install the following to use Auto Generation tool:

- Graphviz : Graphviz version 2.38.0 or higher
    - http://www.graphviz.org/Download.php

Install the below tools to compile and build the Auto Generation tool:

For windows,

- Install GCC compiler (v4.8.1 or higher) for Windows (ex, http://www.codeblocks.org/)
- Install GNU Make (v3.81 or higher) for Windows (ex, "gmake" is available as part XDC install at $(xdc_PATH)/gmake) or http://gnuwin32.sourceforge.net/packages/make.htm
- Install bash shell in Windows via tool like https://msysgit.github.io/ or Cygwin
- Flex : flex version 2.5.* or higher
    - http://gnuwin32.sourceforge.net/packages/flex.htm
- Bison : bison version 2.4.* or higher
    - http://gnuwin32.sourceforge.net/packages/bison.htm

3 Generating Use-Case files

Generating Use-Case files involves:

1. Writing Configuration file
2. Generating files

3.1 Configuration files

3.1.1 Usecase Name

- Use-case name can be mentioned in configuration file. It is used as prefix in files generated and struct and function names.
  Example:
  UseCase: chains_vipSingleCam_Display

3.1.2 Naming Of Link

- Every link has a particular basename, i.e. all instances of a particular type of link starts with basename. E.g.: All links of Capture type should start with basename Capture
- Basename information is available in help option (./vsdk.exe -help)
- A Link is named as Basename or Basename_suffix. For e.g Capture, Capture_1
- Different instances of a particular link have same basename but different suffix, i.e. Display_Video, Display_Grpx
- If it is algorithm link it has to be named Alg_<plugin name>_suffix. For e.g: Alg_FrameCopy_xyz
- In case link does not match any of the supported links an error will be shown.

3.1.3 Connections

- Grammar of Connections:
  Connection : ID | ID -> Connection | ID ([CPU]) | ID ([CPU]) -> Connection

- Example: Single camera display

```
1 UseCase: chains_vipSingleCam_Display
2
3 Capture -> Display_Video
4 GrpxSrc -> Display_Grpx
```

- Intermediate IPC are autogenerated. So, no need to mention in config file.
  Example:

```
UseCase: chains_vipSingleCameraEdgeDetection
Capture -> Alg_EdgeDetect (EVE1) -> Display
```

  Above Configuration file generates IPC links, which makes overall connections:

Capture -> IPCOut_IPU1_0_EVE1_0 -> IPCIn_EVE1_IPU_0_0 (EVE1) ->
Alg_EdgeDetect(EVE1) -> IPCOut_ EVE1_IPU1_0 _0 (EVE1)->
IPCIn _IPU_0_ EVE1_0 -> Display

3.2 Generating Files

- To generate usecase files, type:
  - ./vsdk.exe –file configFile
  - This generates file in the folder where command is executed
- To generate usecase files in an "output" folder, type:
  - ./vsdk –file configFile –path ./output
- To generate image along with file, type:
  - ./vsdk –file –img configFile
- Other options supported are:
  - -help Shows help regarding supported cmd line options, links and CPU
  - -v Verbose

3.3 Error Handling

Error is handled in following cases:

- Input file is not present
- Wrong number of input or output is provided to a link
- Link is assigned Invalid CPU or two different CPU
- Naming of Link does not follow the rules, i.e. Basename, Basename_suffix

4 Tool Development

This section describes how to extend the tool by modifying its source code.

If you are a user of the tool, then you can skip this section

4.1 Adding support for new link in the tool

To create a new Link class:

1. In link.h create new class in following format:

```
class LinkName: public Link {
        ~LinkName ();
public:
        LinkName(string nm);
        void genIncludes(ostream &fp);
        void genLinkID(ostream &fp);
        void genCreatePrms(ostream &fp);
        void genResetLinkPrms(ostream &fp, string obj);
        void genSetLinkPrms(ostream &fp, string obj);

        int setInLink(Link* obj);
        int setOutLink(Link* obj);
};
```

2. In processor.h, introduce extra enum in ClassType, cLinkName
3. Implement the functions in link.cpp file:
   a. Constructor:

```
LinkName(string nm){
        cType = cLinkName; //cType is classType which is set in Processor.h
        name = nm;
        linkIDName = name + string("LinkID");
        prmName = name + string("Prm");
        execPos = -1;
        procID = -1;
        pType = IPU1_0; //default processor type
        mulInQue = false; //set to true if the link can have multiple input
        mulOutQue = false; //set to true if the link can have multiple output
}
```

   b. genIncludes: Include the header file where the link is implemented

```
void LinkName::genIncludes(ostream &fp) {
        fp << "headerName.h"<< endl;
}
```

   c. genLinkID : Not required to change

```
void LinkName::genLinkID(ostream &fp) {
        fp << BLOCK_SPACE << setw(10) << left << "UInt32" << linkIDName << ";"
<< endl;
}
```

d. genCreatePrms : Modify LinkName _CreateParams with actual CreateParams struct name

```
void LinkName::genCreatePrms(ostream &fp) {
        fp << BLOCK_SPACE << setw(40) << left << " LinkName _CreateParams " <<
prmName<< ";" << endl;
}
```

e. genResetLinkPrms : Modify LinkName _CreateParams_Init function name

```
void LinkName::genResetLinkPrms(ostream &fp, string obj) {
        fp << BLOCK_SPACE << " LinkName _CreateParams_Init(&" << obj << "->"
                << prmName << ");" << endl;
}
```

f. setInLink : Leave the function as it is. Uncomment in case you want to introduce error if number of incoming links exceeds maxIncoming. Also, replace maxIncoming with actual number

```
int LinkName::setInLink(Link* obj) {
//CHECK_ERROR_ABORT(inLink.size() >= maxIncoming, "Error: "+name+" Link
//should not have more than "+maxIncoming+" ingoing links");
        inLink.push_back(make_pair(obj, -1));
        if(inLink.size() > 1)
                mulInQue = true;
        return (inLink.size() - 1);
}
```

g. setOutLink: Same rules as setInLink

```
int LinkName::setOutLink(Link* obj) {
        //CHECK_ERROR_ABORT(outLink.size() >= maxOutGoing, "Error:
//"+name+" Link should not have more than "+maxOutGoing +" outgoing links");
        outLink.push_back(make_pair(obj, -1));
        return (outLink.size() - 1);
}
```

h. genSetLinkPrms: Can set any parameters for the link

```
void LinkName::genSetLinkPrms(ostream &fp, string obj)
{
}
```

4. In processor.cpp getLinkID function, introduce an extra case in switch. Replace ID with LinkID which needs to be assigned. Other switch cases serve as an example.

```
case cDecode:
        linkIDName = ID;
        linkIDAsgn[cType]++;
        break;
```

5. In usecase.cpp in createNewObj function, include a new condition. Where "NewLinkBase" is the base name and NewLink is the class created.

```
else if (root == "NewLinkBase")
```

```
obj = new NewLink(name);
```

6. In options.cpp in process_Options function, add a new text string for the newly added link. Where "NewLinkBase" is the new link that is added

```
string usage =
…
" Supported Links:  \n"
…
"   NewLinkBase\n"
```

4.2 Adding support for new Algorithm Plugin in the tool

To develop a new Algorithm:

1. Follow all the steps in developing new link above except 5th. Preferably name class as Alg_LinkName
2. In processor.cpp getProcID introduce a new case to validate the CPU.
3. In usecase.cpp, in createNewObj function, include a new condition inside Alg condition.

e.g:

```
else if (root == "Alg") { //insert condition inside Alg Condition
      string sec = getSecRoot(name);
                  if (sec == "NewLinkBase") //NewLinkBase is the base name of Alg link
                              obj = new Alg_NewLink(name); // Alg_NewLink is class of new Alg
}
```

5 Revision History

Version	Date	Revision History
0.1	03rd March 2016	Draft
0.2	29th June 2017	Updated for Vision SDK rel 3.0