Building HSR RedBox on ZedBoard using FMC-GbE-RJ45

Version 0 Revision 3

March 29, 2019 (Jan. 7, 2019)

Future Design Systems
www.future-ds.com / contact@future-ds.com

Copyright © 2019 Future Design Systems, Inc.

Abstract

This manual describes details of building HSR RedBox on Avnet ZedBoard using Future Design Systems FMC-GbE-RJ45 board.

Table of Contents

1 HSR RedBox

As shown in Figure 1, HSR (High-availability Seamless Redundancy) RedBox (Redundancy Box) has three Ethernet ports.

- Upstream port: it is normal Ethernet port
- Link-A port: it is one of two HSR Ethernet ports
- Link-B port: it is one of two HSR Ethernet ports

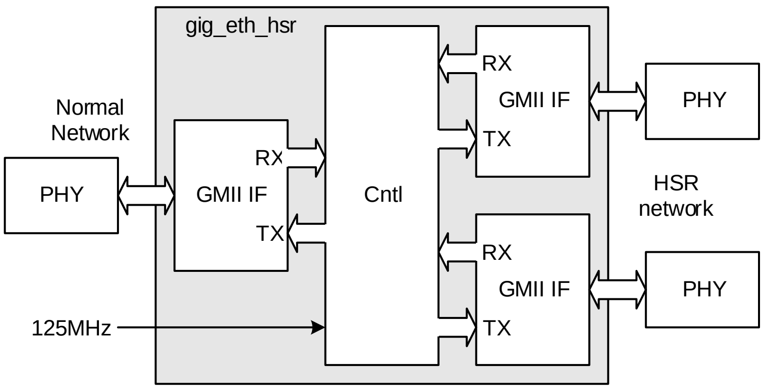

Figure 1: HSR RedBox structure

Egtherent packet received from the upstream port is duplicated and sent through two HSR port after adding HSR header as shown in Figure 2. HSR packet received from the HSR Link-A/B is forwared to the upstream and the other HSR Link depending on MAC address and other information. More detaails should be referred to HSR specification[1].

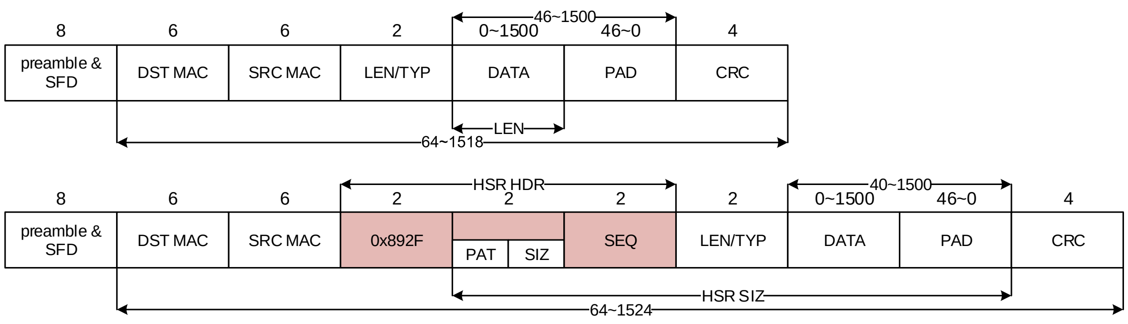

Figure 2: Normal packet and HSR packet

This document describes a details of building RedBox using Future Design Systems HSR RTL[2] on the following hardware borads.

- Future Design Systems, FMC-GbE-RJ45[3]
- Avnet, ZedBoard[4]

2 HSR RedBox on ZedBoard

Figure 3 shows how HSR RedBox is implemented on ZedBoard.

- Zedboard: FPGA boarad to implement HSR design
- FMC-GbE-RJ45: 3 port Ethernet board
- Raspberry Pi 3 Model B+[5]: Single board computer to access network throgh HSR

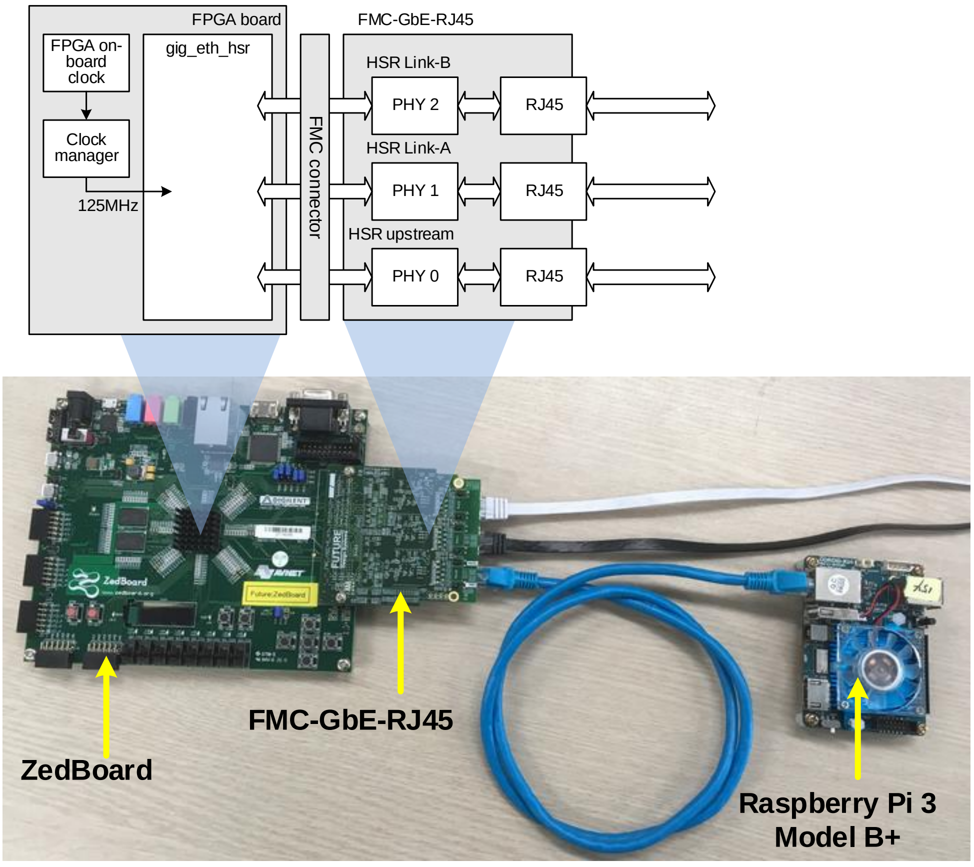

Figure 3: HSR RedBox implementation on ZedBoard

2.1 Directory structure

directory			remarks
design	RTL design directory		
	verilog	fpga_zed.v	
bench	Test-bench directory		
	verilog	top.v	
sim	RTL simulation directory		
	modelsim.ise		Simulation for ISE devices
	modelsim.vivado		Simulation for Vivado devices
pnr	implementation		
	vivado.zedboard.lpc		

2.2 Testing structure

Figure 4 shows hardware structure to simulate a HSR system, where several HSR nodes are connected to build ring and each HSR node consists of 'fpga' and 'tester'.

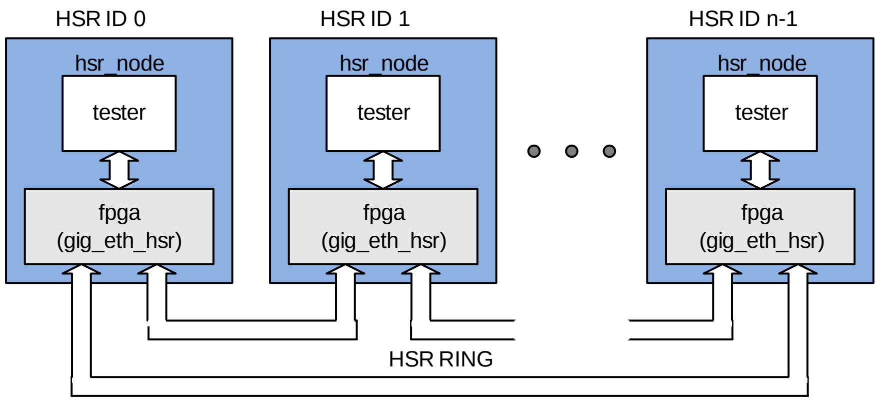

Figure 4: Testing structure

HSR nodes are connected through GMII instead of PHY and each 'tester' in the HSR node is assigned a unique MAC address related to node id such that '48'hF0_12_34_56_78_{node_id[7:0]'.

2.3 Simulation

Simulation requires Mentor Graphics ModelSim and Xilinx Vivado suites.

- ✧ HDL simulator: Mentor Graphics ModelSim
- ✧ FPGA related library: Vivado 2017.4 for Zynq

It requires following environment variables and refer to 'Makefile' and 'sim_define.v' in 'sim/modelsim.vivado'.

- 'XILINX_VIVADO' environment variable to point Vivado installation directory, such as '/opt/Xilinx/Vivado/2017.4' for Linux or 'C:/Xilinx/Vivado/2017.4' for Windows.
- 'FIP_HOME' environment variable in the "Makefile" to point 'gig_eth_hsr' directory, such as '../../../../FIP'.
- 'FPGA_TYPE' environment variable in the "Makefile" to specify FPGA type, such as 'z7' for Zynq 7000.
- 'BOARD_ZED' environment variable in "sim/modelsim.vivado/sim_define.v" to specify FPGA board.

Do as follows to simulate this HSR system.

1. Go to 'sim/modelsim.vivado' directory

2. Run ‘make’ for Linux
   - ✧ If any errors occurs, have a close look at the message.
   - ✧ Reasons of most of errors may be come from mismatches including path or environment variables.
   - ✧ Simulation version specific options may cause problems.
3. Invoke VCD waveform viewer to check simulation if simulation completes without any errors.
   - ✧ GTKwave would be a good choice

‘NUM_OF_HSR_NODE’ in ‘sim_define.v’ specifies the number of HSR nodes to simulate.

Figure 5 shows an example case that HSR NODE 0 is only enabled[1] to generate HSR packets. Set ‘boardcast’ 0 at ‘tester_gmii.v’ in ‘bench/verilog’ directory.

1. ‘tester’ in NODE 0 prepares a packet targeted to NODE 1.
2. ‘gig_eth_hsr’ in NODE 0 receives the packet and forward it to LINK-A and B after inserting HSR header.
3. NODE 1 receives the packet through LINK-B and forwards to the LINK-A. Although this packet is headed to NODE 1, it cannot make hit on MAC address since NODE 1 does not know its MAC address yet, i.e., ‘tester’ did not generate any packet.
4. NODE 2 receives the packet through LINK-A and forwards to the LINK-B.
5. NODE 1 receives the packet from NODE 2 through LINK-A, but discard it since it has been received already. (see step 3)
6. NODE 2 receives the packet from NODE 1 through LINK-B, but discard it since it has been received already. (see step 4)

[1] Refer to code to instantiate ‘hsr_node’ in the ‘bench/Verilog/top.v’.

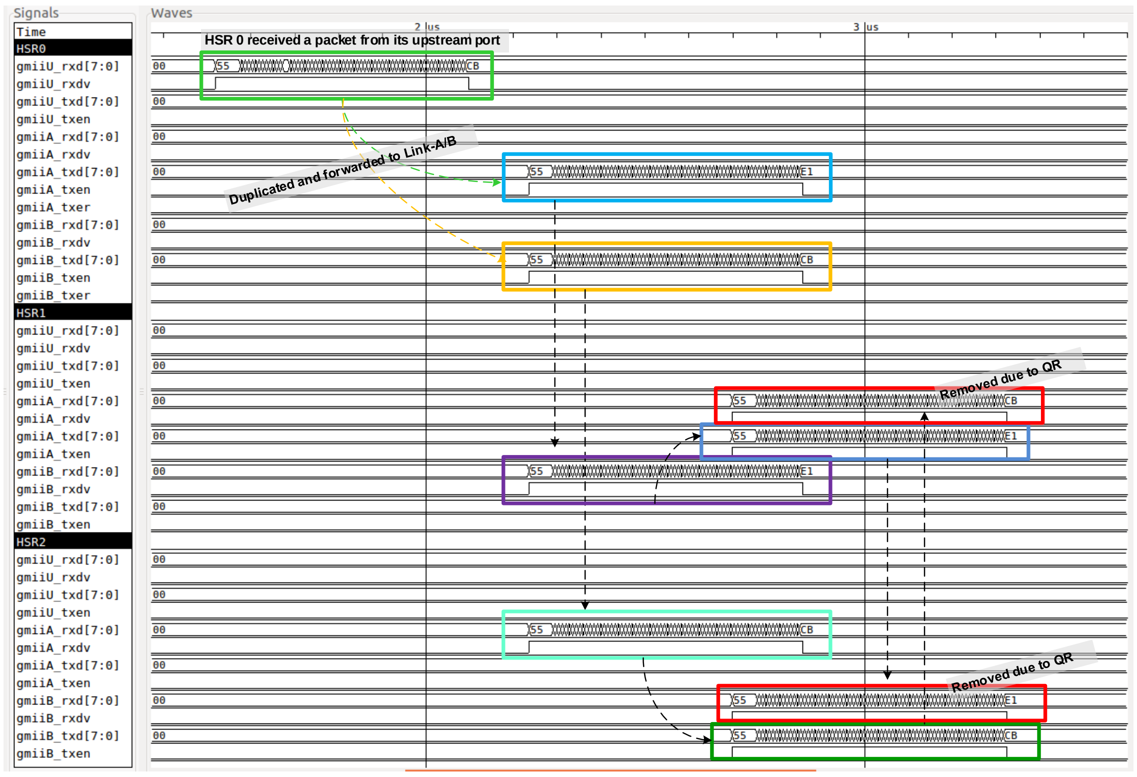

Figure 5: Simulation case NODE 0 generates a packet

Figure 6 shows an example with broadcasting case. Set ‘boardcast’ 1 at ‘tester_gmii.v’ in ‘bench/verilog’ directory.

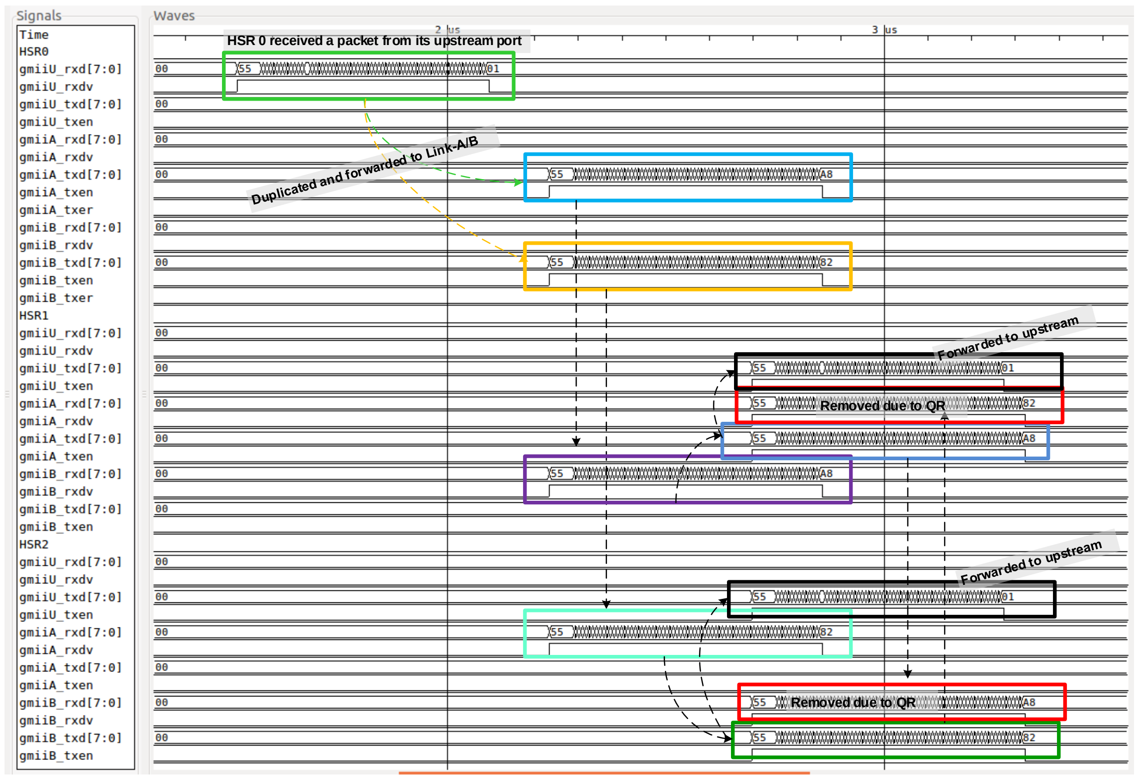

Figure 6: Simulation case NODE 0 generates a broadcasting packet

3 Implementation

Implementation requires Vivado and SDK suites.

- ✧ FPGA related library: Vivado 2017.4 for Zynq
- ✧ ARM related library: SDK

It requires following environment variables and refer to ‘Makefile’ and ‘syn_define.v’ in ‘pnr/vivado.zedboard.lpc’.

- ‘XILINX_VIVADO’ environment variable to point Vivado installation directory, such as ‘/opt/Xilinx/Vivado/2017.4’ for Linux or ‘C:/Xilinx/Vivado/2017.4’ for Windows.
- ‘XILINX_SDK’ environment variable to point SDK installation directory, such as ‘/opt/Xilinx/SDK/2017.4’ for Linux or ‘C:/Xilinx/SDK/2017.4’.
- ‘FIP_HOME’ environment variable to point ‘gig_eth_hsr’ directory, such as ‘../../../../FIP’.

Do as follows to implement this HSR system, where implementation means logic synthesis and placement & routing.

1. Make sure all necessary files are ready

    - Go to ‘FIP/gig_eth_hsr/fifo_async/z7/vivado.2017.4’ and run ‘make’. (It takes time, so that do not run if files exist.)
    - Go to ‘FIP/gig_eth_hsr/fifo_sync/z7/vivado.2017.4’ and run ‘make’. (It takes time, so that do not run if files exist.)
2. Go to ‘pnr/vivado.zedboard.lpc’ directory
3. Run ‘make’ for Linux
    - If any errors occurs, have a close look at the message.
    - Reasons of most of errors may be come from mismatches including path or environment variables.
4. There should be ‘fpga.bit’ ready.

Now you can config FPGA by using ‘fpga.bit’

1. Go to ‘pnr/vivado.zedboard.lpc/download’ directory
2. Connect you PC and ZedBoard through USB-JTAB
3. Run ‘make’
4. If any errors occur, please consult Vivado user manual.

3.1 Prepare SD Card

ZedBoard also can be configured using SD Card. The boot configuration DIP switches should be set properly; consult manual for details.

1. Prepare ZedBoard with FMC-GbE-RJ45
    - Port 0 for Upstream
    - Port 1 for HSR Link-A
    - Port 2 for HSR Link-B
2. Go to ‘pnr/vivado.zedboard.lpc/bootgen’ directory
3. Run ‘make clean’ and then ‘make’
4. ‘BOOT.bin’ should be ready
5. Copy ‘BOOT.bin’ to the SD Card
6. Insert SD Card to the ZedBoard
    - Make sure ZedBoard is turned off
    - Make sure Boothing mode selection switches are properly set as shwon in Figure 7.
7. Turn on ZedBoard
    - Configuration DONE LED should be on.
    - Two LEDs should be on as shown in Figure 7; one for HSR ready and the other for RedBox indicator.

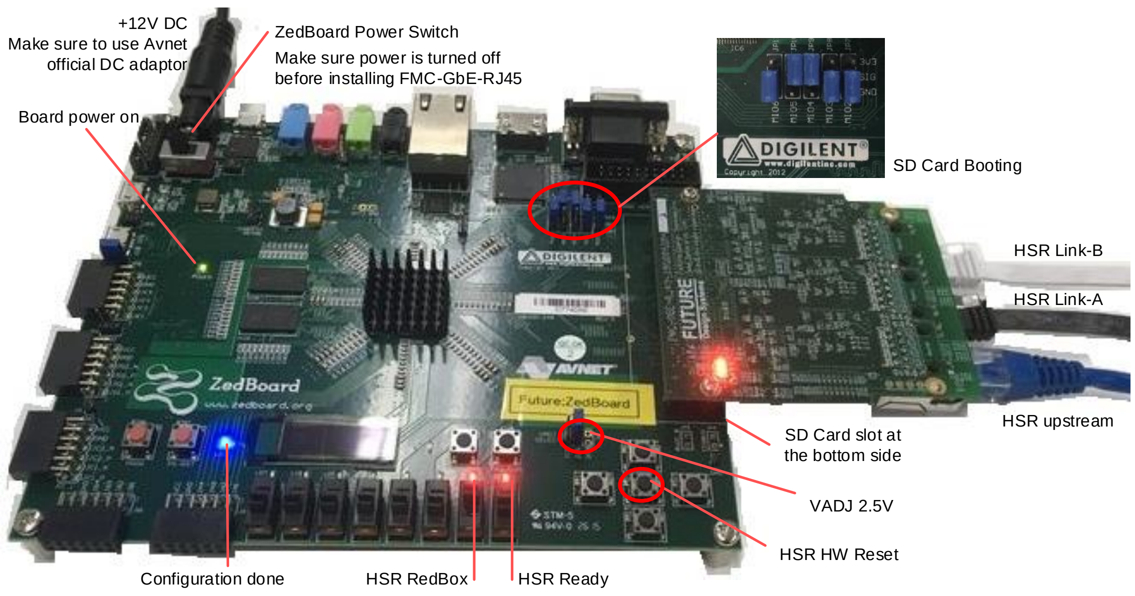

Figure 7: ZedBoard and FMC-GbE-RJ45

4 Testing with two RedBoxes and Raspberry Pi 3 Model B+

Raspberry Pi should support Gigabit Ethernet.

4.1 Tips for Raspberry Pi

4.1.1 Install Rasbian

Get Raspbian Stretch Image from ‘www.raspberrypi.org/downloads/raspbian’ and copy it to uSD card.

Login account will be ‘pi’.
Password will be ‘raspberry’

4.1.2 Setting locale and install Korean fonts

If text font is not visible, set locale to ‘en_US.UTF-8 UTF-8’ or ‘ko_KR.UTF-8 UTF-8’. This step requires network connection.

```
$ sudo apt-get install ibus ibus-hangul fonts-unfonts-core
$ sudo raspi-config
Then select ‘Localization options’ by pressing arrow key
Then select ‘en_US.UTF-8 UTF-8’ by pressing space key
Then select ‘select’ button using tap key
Then reboot
$ sudo reboot
```

4.2 Setup

When all has been done as shown in Figure 7, a HSR system shown in Figure 8 can be set up using two RedBoxes and two Raspberry Pi.

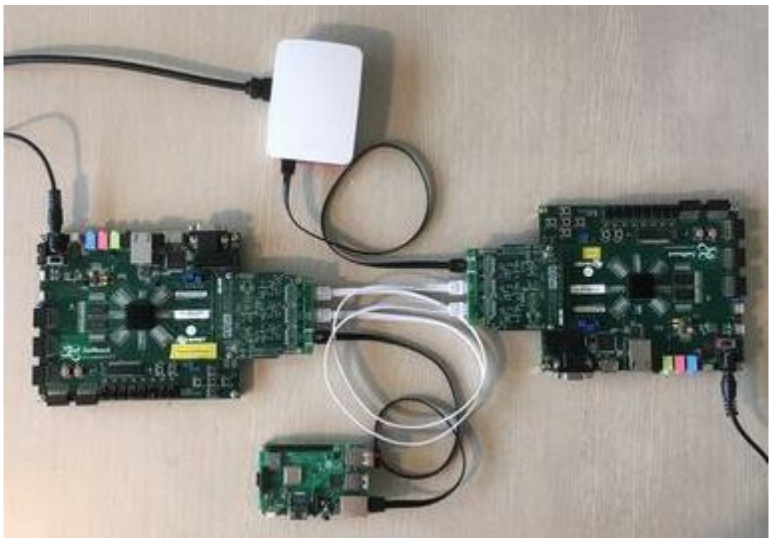

Figure 8: Two RedBox with Raspberry Pi

References

[1] IEC 62439-3, Industrial communication networks –High availability automation networks –Part 3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR), Edition 2.0, 2012-07.
[2] Future Design Systems, High-availability Seamless Redundancy Controller on Gigabit Ethernet, TD-2018-10-002, 2018. (company confidential)
[3] Future Design Systems, FMC-GbEGbE -RJ45 User Manual- Avnet ZedBoard, TD-2018-10-004, 2018.
[4] Avnet, ZedBoard (Zynq Evaluation and Development) Hardware User’s Guide, Jan. 2014.
[5] Raspberry Pi Foundation, Raspberry Pi Document (www.raspberrypi.org/documentation).

Revision history

- 2019.03.29: Updated.
- 2019.01.07: Document started by Ando Ki (adki@future-ds.com)

– End of document –