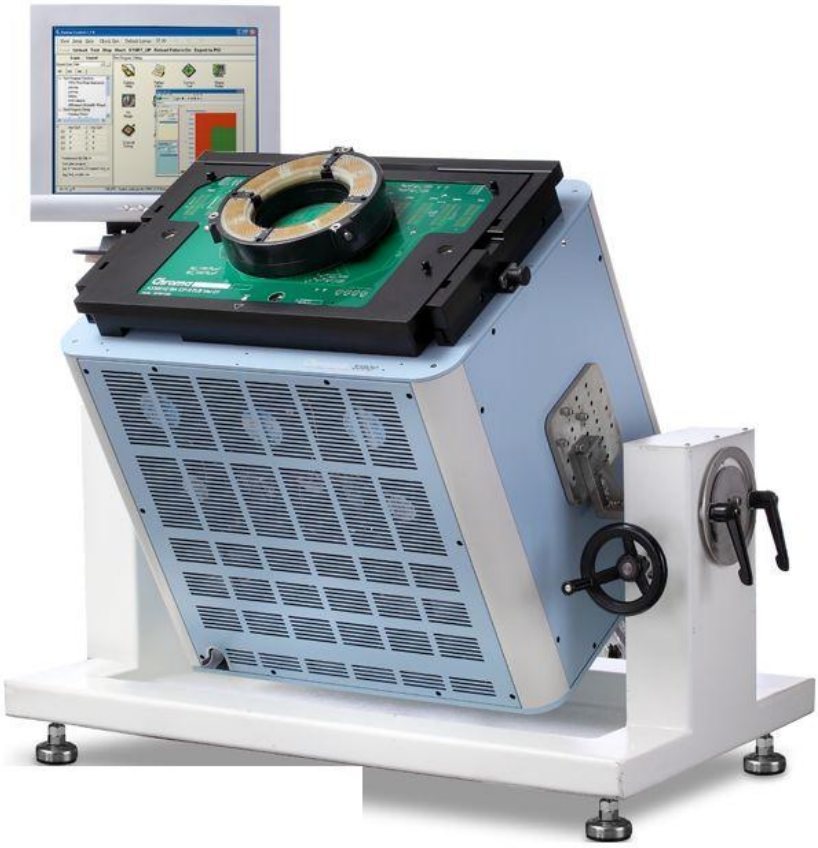
Chroma

VLSI Test System

3380 Series

Language Manual

VLSI Test System
3380 Series
Language Manual



Edition 3.14
Jan. 2019

Legal Notices

The information in this document is subject to change without notice.

Chroma ATE INC. makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Chroma ATE INC. shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

CHROMA ATE INC.
66 Huaya 1st Road, Guishan, Taoyuan 33383, Taiwan

Copyright Notices. Copyright 2012 Chroma ATE INC., all rights reserved. Reproduction, adaptation, or translation of this document without prior written permission is prohibited, except as allowed under the copyright laws.

Warranty

All Chroma instruments are warranted against defects in material and workmanship for a period of one year after date of shipment. Chroma agrees to repair or replace any assembly or component found to be defective, under normal use during this period. Chroma's obligation under this warranty is limited solely to repairing any such instrument, which in Chroma's sole opinion proves to be defective within the scope of the warranty when returned to the factory or to an authorized service center. Transportation to the factory or service center is to be prepaid by purchaser. Shipment should not be made without prior authorization by Chroma.

This warranty does not apply to any products repaired or altered by persons not authorized by Chroma, or not in accordance with instructions furnished by Chroma. If the instrument is defective as a result of misuse, improper repair, or abnormal conditions or operations, repairs will be billed at cost.

Chroma assumes no responsibility for its product being used in a hazardous or dangerous manner either alone or in conjunction with other equipment. High voltage used in some instruments may be dangerous if misused. Special disclaimers apply to these instruments. Chroma assumes no liability for secondary charges or consequential damages and in any event, Chroma's liability for breach of warranty under any contract or otherwise, shall not exceed the purchase price of the specific instrument shipped and against which a claim is made.

Any recommendations made by Chroma for use of its products are based upon tests believed to be reliable, but Chroma makes no warranty of the results to be obtained. This warranty is in lieu of all other warranties, expressed or implied, and no representative or person is authorized to represent or assume for Chroma any liability in connection with the sale of our products other than set forth herein.

CHROMA ATE INC.
6F No 5, Technology Road., Science-Based Ind. Park, Hsinchu City 300,Taiwan
Tel: 886 -3-563-5788
Fax: 886-3-563-5758

http://www.chromaate.com
Language system: CRAFT 3380 release version

Material Contents Declaration

The recycling label shown on the product indicates the Hazardous Substances contained in the product as the table listed below.

: See <Table 1>.

: See <Table 2>.

<Table 1>

Part Name	Hazardous Substances					
	Lead	Mercury	Cadmium	Hexavalent Chromium	Polybrominated Biphenyls	Polybromodiphenyl Ethers
	Pb	Hg	Cd	Cr^{6+}	PBB	PBDE
PCBA	O	O	O	O	O	O
CHASSIS	O	O	O	O	O	O
ACCESSORY	O	O	O	O	O	O
PACKAGE	O	O	O	O	O	O

"O" indicates that the level of the specified chemical substance is less than the threshold level specified in the standards of SJ/T-11363-2006 and EU 2005/618/EC.

"×" indicates that the level of the specified chemical substance exceeds the threshold level specified in the standards of SJ/T-11363-2006 and EU 2005/618/EC.

Disposal

Do not dispose of electrical appliances as unsorted municipal waste, use separate collection facilities. Contact your local government for information regarding the collection systems available. If electrical appliances are disposed of in landfills or dumps, hazardous substances can leak into the groundwater and get into the food chain, damaging your health and well-being. When replacing old appliances with new one, the retailer is legally obligated to take back your old appliances for disposal at least for free of charge.

<Table 2>

Part Name	Hazardous Substances					
	Lead	Mercury	Cadmium	Hexavalent Chromium	Polybrominated Biphenyls	Polybromodiphenyl Ethers
	Pb	Hg	Cd	Cr^{6+}	PBB	PBDE
PCBA	×	O	O	O	O	O
CHASSIS	×	O	O	O	O	O
ACCESSORY	×	O	O	O	O	O
PACKAGE	O	O	O	O	O	O

"O" indicates that the level of the specified chemical substance is less than the threshold level specified in the standards of SJ/T-11363-2006 and EU 2005/618/EC.

"×" indicates that the level of the specified chemical substance exceeds the threshold level specified in the standards of SJ/T-11363-2006 and EU 2005/618/EC.

1. Chroma is not fully transitioned to lead-free solder assembly at this moment; however, most of the components used are RoHS compliant.
2. The environment-friendly usage period of the product is assumed under the operating environment specified in each product's specification.

Disposal

Do not dispose of electrical appliances as unsorted municipal waste, use separate collection facilities. Contact your local government for information regarding the collection systems available. If electrical appliances are disposed of in landfills or dumps, hazardous substances can leak into the groundwater and get into the food chain, damaging your health and well-being. When replacing old appliances with new one, the retailer is legally obligated to take back your old appliances for disposal at least for free of charge.

Safety Summary

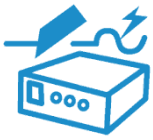
The following general safety precautions must be observed during all phases of operation, service, and repair of this instrument. Failure to comply with these precautions or specific WARNINGS given elsewhere in this manual will violate safety standards of design, manufacture, and intended use of the instrument. *Chroma* assumes no liability for the customer's failure to comply with these requirements.

BEFORE APPLYING POWER

Verify that the power is set to match the rated input of this power supply.

PROTECTIVE GROUNDING

Make sure to connect the protective grounding to prevent an electric shock before turning on the power.



NECESSITY OF PROTECTIVE GROUNDING

Never cut off the internal or external protective grounding wire, or disconnect the wiring of protective grounding terminal. Doing so will cause a potential shock hazard that may bring injury to a person.

FUSES

Only fuses with the required rated current, voltage, and specified type (normal blow, time delay, etc.) should be used. Do not use repaired fuses or short-circuited fuse holders. To do so could cause a shock or fire hazard.

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE

Do not operate the instrument in the presence of flammable gases or fumes. The instrument should be used in an environment of good ventilation.

DO NOT REMOVE THE COVER OF THE INSTRUMENT

Operating personnel must not remove the cover of the instrument. Component replacement and internal adjustment can be done only by qualified service personnel.

⚠WARNING

1. Lethal voltage. AC source may output 426 V peak voltage.
2. Touching the connected circuit or output terminal on the front or rear panel when power is on may result in death.

Safety Symbols

	DANGER – High voltage.
	Explanation: To avoid injury, death of personnel, or damage to the instrument, the operator must refer to an explanation in the instruction manual.
	Protective grounding terminal: To protect against electrical shock in case of a fault. This symbol indicates that the terminal must be connected to ground before operation of equipment.
WARNING	The WARNING sign denotes a hazard. It calls attention to a procedure, practice, or the like, which, if not correctly performed or adhered to, could result in personal injury. Do not proceed beyond a WARNING sign until the indicated conditions are fully understood and met.
CAUTION	The CAUTION sign denotes a hazard. It may result in personal injury or death if not noticed timely. It calls attention to procedures, practices and conditions.

Revision History

The following lists the additions, deletions and modifications in this manual at each revision.

Date	Version	Revised Sections
Oct 2019	3.16	Add statement FORCE_V_PPMU_X, FORCE_I_PPMU_X, SET_WG_OFFSET, Add SET_SHMOO new parameter "FREQ_SHMOO" Add MLDPS JUDGE/MEAS statement new parameter "RMS_X"
Mar 2019	3.15	Add PIN_INC option parameter, LOOP_BY_PIN, example.
Jan. 2019	3.14	Add system marco define for user usage
Sep. 2018	3.13	Add new ALPG Micro-Instruction to list
Aug. 2018	3.12	Add statement SET_ALPG_HOLDING_REG, READ_ALPG_REG_COUNTER, READ_ALPG_LOG_MEM and SET_ALPG_WORK_MODE_X.
Mar. 2018	3.11	Add ALPG Micro-Instruction to list The modify HSTMU function using limitation. "MXLPC board using the front 0 ~ 31 pins to do HSTMU function, try not to use 32 ~ 63 pins, otherwise the test speed will be limited to 25MHZ in DBL mode or 50MHZ in DBL_2X mode."
Feb. 2018	3.10	Add statement RESULT_MLDPS_EX, RESULT_PIN_EX for PINSHARE usage
Oct. 2017	3.9	JUDGE_SPI_WRITE / JUDGE_SPI_READ, RUN_SHMOO_EX
Feb. 2017	3.8	Modify the RELAY ON / RELAY OFF circuit diagram Modify plus C function C Language Library
Dec. 2016	3.7	Add RF statements RF Subsystem (Option), add READ_SYSTEM_VENDOR_ID & CRAFT_c_shuttle_relay_w
Nov.2016	3.6	Add SET_ECM_MODE, READ_ECM_BLOCK,PRINT_ECM_DATA
Aug. 2016	3.5	Modify PMU force current range(-250mA~250mA),add MXPMU V/I Curve
Mar. 2016	3.4	Modify statements for DBL MODE Pattern rate less ,RPTN X0~X7 mini value, JUDGE_PAT fail_count_limit. Add LOAD_FOCUS_CAL_FILE set value of TDR_ADV, add FORCE_CURRENT set value of positive/negative, add new statements of CARRAY_DIM.
Jan. 2016	3.3	Modify statements for IO_ALLPINS, MLDPS/FORCE_VOLT I_Clamp value Spec.
Oct. 2015	3.2	Add READ_LOAD_BOARD_ID_2; modify ALARM_ON (don't support MLDPS) & FORCE_I_PPMU (small current must be > 60nA), & MEAS_TMU_RESULT(new parameter)
May 2015	3.1	Add AWI2 of tool instruction. Appendix F, MAWI2 of tool instruction
Dec.2014	3.0	Add new statements of AWI2 setting.MAWI2 Generator / Digitizer
Jun. 2014	2.3	Modify statements
Mar. 2014	2.2	Remove CRAFT_c_set_dps_burst_mode, CRAFT_c_get_load_board_id, CRAFT_c_force_c_stdps_w ; Add statements of plus-c function
Dec. 2013	2.1	Add MLDPS special statements, add PMU new current range(@250mA)
Sep. 2013	2.0	Modify value about clamp Voltage of MXUVI, MXPMU & MXREF, Add GET_TRM_PATH, Modify LOAD_FOCUS_CAL_FILE
Aug. 2013	2.0	Modify TMU statement, SET_CBIT, PIN_INC, PAT_SECURITY
Aug. 2012	1.0	Complete this manual.

Table of Contents

1. Overview of Test Program 1-1
   1.1 Overview 1-1
   1.2 Architecture of the Program 1-1

2. Device Definition File 2-1
   2.1 Overview 2-1
      2.1.1 Format 2-1
      2.1.2 Explanation 2-1
      2.1.3 Example 2-2
   2.2 DEC_MODE 2-3
      2.2.1 Format 2-3
      2.2.2 Example 2-3
   2.3 PIN_LIST Declaration 2-4
      2.3.1 Format 2-4
      2.3.2 Explanation 2-5
      2.3.3 Example 2-6
   2.4 PIN_GROUP 2-7
      2.4.1 Format 2-7
      2.4.2 Explanation 2-7
      2.4.3 Example 2-8
   2.5 UR_PIN_GROUP 2-9
      2.5.1 Format 2-9
      2.5.2 Explanation 2-9
      2.5.3 Example 2-10
   2.6 POWER_PIN_GROUP 2-11
      2.6.1 Format 2-11
      2.6.2 Example 2-12
   2.7 TIME_NAME_DEF 2-13
      2.7.1 Format 2-13
      2.7.2 Example 2-13
   2.8 LEVEL_NAME_DEF 2-15
      2.8.1 Format 2-15
      2.8.2 Example 2-15

3. Pattern Files 3-1
   3.1 Overview 3-1
      3.1.1 Format 3-1
      3.1.2 Example 3-2
   3.2 SET_DEC_FILE 3-3
      3.2.1 Format 3-3
      3.2.2 Explanation 3-3
      3.2.3 Example 3-3
   3.3 HEADER 3-4
      3.3.1 Format 3-4
      3.3.2 Example 3-5
   3.4 PATTERN 3-6
      3.4.1 SPM_PATTERN 3-6
      3.4.2 APM_PATTERN 3-10
      3.4.3 Pattern Micro-instruction 3-17

4. Plan Program 4-1
   4.1 Overview 4-1
      4.1.1 Format 4-1
      4.1.2 Explanation 4-2
   4.2 Basic Control Statement 4-4
      4.2.1 SET_DEC_FILE 4-4

4.2.2 HW_BIN_DEF 4-5
4.2.3 SW_BIN_DEF 4-7
4.2.4 START_UP( ) { … } 4-9
4.2.5 TEST_PRO {….} 4-11
4.2.6 RELEASE(){ … } 4-18
4.2.7 CONDITION_SET 4-20
4.2.8 SPEC_DEF 4-22
4.2.9 TEST_NO 4-25
4.2.10 DATALOG_MSG 4-26
4.2.11 WAIT 4-27
4.2.12 GLOBAL 4-28
4.3 MLDPS Control Statement 4-30
4.3.1 FORCE_I_MLDPS 4-30
4.3.2 FORCE_V_MLDPS 4-32
4.3.3 JUDGE_I_MLDPS 4-35
4.3.4 JUDGE_V_MLDPS 4-37
4.3.5 MEAS_I_MLDPS 4-39
4.3.6 MEAS_V_MLDPS 4-42
4.3.7 RESULT_MLDPS 4-43
4.3.8 MIN_RESULT_MLDPS 4-44
4.3.9 MAX_RESULT_MLDPS 4-45
4.3.10 RESULT_MLDPS_MS 4-46
4.3.11 MIN_RESULT_MLDPS_MS 4-47
4.3.12 MAX_RESULT_MLDPS_MS 4-48
4.3.13 RESULT_MLDPS_EX 4-49
4.3.14 MIN_RESULT_MLDPS_EX 4-50
4.3.15 MAX_RESULT_MLDPS_EX 4-51
4.4 MLDPS High Speed Function 4-52
4.4.1 LOAD_VI_WAVEFORM (Statement) 4-52
4.4.2 CLEAR_VI_WAVEFORM (Statement) 4-54
4.4.3 SET_VI_WAVEFORM (Statement) 4-55
4.4.4 READ_VI_CAPTURE_MEM_BLOCK (Statement) 4-56
4.4.5 MLDPS_FORCE_INC (Micro-instruction) 4-58
4.4.6 MLDPS_MEAS (Micro-instruction) 4-59
4.4.7 MLDPS_SET_VOLT (Micro-instruction) 4-61
4.5 MXDPS Control Statement(Option) 4-62
4.5.1 FORCE_I_DPS 4-62
4.5.2 FORCE_V_DPS 4-64
4.5.3 JUDGE_I_DPS 4-68
4.5.4 JUDGE_V_DPS 4-71
4.5.5 MEAS_I_DPS 4-73
4.5.6 MEAS_V_DPS 4-75
4.5.7 RESULT_DPS 4-77
4.5.8 MIN_RESULT_DPS 4-78
4.5.9 MAX_RESULT_DPS 4-79
4.5.10 RESULT_DPS_MS 4-80
4.5.11 MIN_RESULT_DPS_MS 4-81
4.5.12 MAX_RESULT_DPS_MS 4-82
4.5.13 SET_CONST_CURRENT_LOAD 4-83
4.6 MXUVI Control Statement(Option) 4-85
4.6.1 FORCE_V_UVI 4-85
4.6.2 FORCE_I_UVI 4-88
4.6.3 MEAS_I_UVI 4-90
4.6.4 MEAS_V_UVI 4-91
4.6.5 JUDGE_I_UVI 4-92
4.6.6 JUDGE_V_UVI 4-94
4.6.7 RESULT_UVI 4-96
4.6.8 MIN_RESULT_UVI 4-97
4.6.9 MAX_RESULT_UVI 4-98

4.6.10 RESULT_UVI_MS 4-99
4.6.11 MIN_RESULT_UVI_MS 4-100
4.6.12 MAX_RESULT_UVI_MS 4-101
4.7 MXREF Control Statement(Option) 4-102
4.7.1 FORCE_I_PREF 4-102
4.7.2 FORCE_V_PREF 4-104
4.7.3 JUDGE_I_PREF 4-108
4.7.4 JUDGE_V_PREF 4-109
4.7.5 MEAS_I_PREF 4-111
4.7.6 MEAS_V_PREF 4-112
4.7.7 RESULT_PREF 4-113
4.7.8 MIN_RESULT_PREF 4-114
4.7.9 MAX_RESULT_PREF 4-115
4.7.10 RESULT_PREF_MS 4-116
4.7.11 MIN_RESULT_PREF_MS 4-117
4.7.12 MAX_RESULT_PREF_MS 4-118
4.8 Complex Control Statement 4-119
4.8.1 FORCE_VOLT 4-119
4.8.2 MEAS_CURRENT 4-127
4.8.3 JUDGE_CURRENT 4-129
4.8.4 FORCE_CURRENT 4-132
4.8.5 MEAS_VOLT 4-137
4.8.6 JUDGE_VOLT 4-139
4.8.7 SET_DVM_MODE 4-141
4.9 MXPMU Control Statement 4-143
4.9.1 FORCE_I_PMU 4-143
4.9.2 JUDGE_I_PMU 4-147
4.9.3 MEAS_I_PMU 4-149
4.9.4 FORCE_V_PMU 4-151
4.9.5 JUDGE_V_PMU 4-154
4.9.6 MEAS_V_PMU 4-156
4.9.7 PARAL_INC 4-158
4.9.8 PIN_INC 4-159
4.9.9 READ_PIN_INC 4-161
4.10 PPMU Control Statement 4-162
4.10.1 FORCE_V_PPMU 4-162
4.10.2 FORCE_V_PPMU_X 4-163
4.10.3 FORCE_I_PPMU 4-164
4.10.4 FORCE_I_PPMU_X 4-166
4.10.5 JUDGE_I_PPMU 4-167
4.10.6 JUDGE_V_PPMU 4-170
4.10.7 MEAS_I_PPMU 4-172
4.10.8 MEAS_V_PPMU 4-174
4.11 Input & Output Condition Setting 4-175
4.11.1 SET_LEVELN 4-175
4.11.2 LEVELS 4-178
4.11.3 INIT_DRIVE_STATE 4-180
4.11.4 JUDGE_STATE 4-182
4.11.5 RATEN 4-183
4.11.6 MARKERN 4-185
4.11.7 FRUN_CLK 4-187
4.11.8 FRUN_CLK_MODE 4-188
4.11.9 PIN_MODE 4-190
4.12 Relay Setting 4-193
4.12.1 SET_CBIT 4-193
4.12.2 RELAY_ON 4-196
4.12.3 RELAY_OFF 4-200
4.13 Oscillator Function Test 4-203

4.13.1 JUDGE_FREQ 4-203
4.13.2 MEAS_FREQ 4-207
4.13.3 JUDGE_TIME 4-210
4.13.4 MEAS_TIME 4-212
4.13.5 STATUS_PIN 4-214
4.13.6 SET_TFTM_MODE 4-215
4.13.7 OSC_struct 4-216
4.13.8 GET_OSC_CLK_DATA 4-217
4.13.9 SET_OSC_CLK 4-219
4.13.10 OSC_CLK_LOCK 4-220
4.13.11 OSC_CLK_MODE 4-221
4.14 TMU Control Statement 4-222
4.14.1 TMU Pin declare 4-222
4.14.2 SET_TMU_WORK_MODE 4-224
4.14.3 JUDGE_TMU_RESULT 4-227
4.14.4 MEAS_TMU_RESULT 4-234
4.14.5 READ_TMU_LOG_MEM_BLOCK 4-241
4.14.6 TMU_DATA_ARITH 4-244
4.14.7 CARRAY_3DIM 4-246
4.14.8 CARRAY_2DIM 4-247
4.14.9 CARRAY_DIM 4-248
4.15 Function Test 4-249
4.15.1 JUDGE_PAT 4-249
4.15.2 PAT_CONTINUE 4-251
4.15.3 LOAD_PAT 4-252
4.15.4 VECTOR_EXTRACT 4-254
4.15.5 SET_JUDGE_MODE 4-256
4.16 SOCKET Control function 4-260
4.16.1 GET_ACTIVE_SOCKET 4-260
4.16.2 DISABLE_SOCKET 4-261
4.16.3 ENABLE_SOCKET 4-262
4.16.4 SOCKET_SET 4-263
4.16.5 SOCKET_INC 4-265
4.16.6 POWER_DOWN_FAIL_SITE 4-268
4.16.7 PLAN_RESULT_MS 4-270
4.16.8 JUDGE_RESULT_MS 4-271
4.16.9 BACKUP_PLAN_RESULT 4-272
4.16.10 FAIL_REJUDGE 4-273
4.16.11 ENABLE_SOCKET_EX 4-275
4.16.12 DISABLE_SOCKET_EX 4-276
4.16.13 SOCKET_SET_EX 4-277
4.16.14 GET_ACTIVE_SOCKET_EX 4-279
4.16.15 POWER_DOWN_FAIL_SITE_X 4-280
4.17 Measure Result Getting 4-282
4.17.1 RESULT_PIN_MS 4-282
4.17.2 MIN_RESULT_PIN_MS 4-283
4.17.3 MAX_RESULT_PIN_MS 4-284
4.17.4 RESULT_PIN 4-285
4.17.5 READ_PIN_INC 4-286
4.17.6 MAX_RESULT_PIN 4-288
4.17.7 MIN_RESULT_PIN 4-289
4.17.8 JUDGE_VARIABLE 4-290
4.17.9 JUDGE_VARIABLE_MS 4-292
4.17.10 RESULT_PIN_EX 4-293
4.17.11 MIN_RESULT_PIN_EX 4-294
4.17.12 MAX_RESULT_PIN_EX 4-295
4.18 Scan Function (Option) 4-296
4.18.1 SCAN PIN Declaration 4-296

4.18.2 Scan Pattern 4-297
4.18.3 SET_SCAN_WORK_MODE 4-300
4.19 ALPG Function (Option) 4-301
4.19.1 ALPG_PIN_GROUP 4-301
4.19.2 ALPG Pattern 4-305
4.19.3 SET_ALPG_WORK_MODE 4-307
4.19.4 ALPG Micro Instruction 4-308
4.19.5 SET_ALPG_HOLDING_REG 4-311
4.19.6 READ_ALPG_REG_COUNTER 4-312
4.19.7 READ_ALPG_LOG_MEM 4-313
4.19.8 SET_ALPG_WORK_MODE_X 4-315
4.20 AWG/Digitizer (Option) 4-317
4.20.1 AWI Pins Declaration 4-317
4.20.2 AWI_HEADER – CRAFT declaration 4-320
4.20.3 LOAD_AWG_PAT 4-322
4.20.4 SET_WG_WORK_MODE 4-325
4.20.5 SET_WD_WAVEFORM_SET 4-328
4.20.6 SET_WD_WORK_MODE 4-330
4.20.7 READ_WD_DATA 4-332
4.20.8 AWI_DATA_AC_ARITH 4-333
4.20.9 AWI_DATA_DC_ARITH 4-336
4.20.10 AWI_SPECTRUM 4-338
4.20.11 WG_PAT_STOP 4-339
4.20.12 LOAD_FOCUS_CAL_FILE 4-340
4.20.13 AWI_SPECTRUM_EX 4-342
4.20.14 AWI_DATA_AC_ARITH_EX 4-343
4.21 Capture Memory 4-346
4.21.1 SET_CAPTURE_MEM_MODE 4-346
4.21.2 READ_CAPTURE_MEM 4-349
4.21.3 READ_CAPTURE_MEM_BLOCK 4-350
4.21.4 READ_CAPTURE_MEM_BLOCK_X 4-352
4.21.5 SET_ECM_MODE 4-354
4.21.6 READ_ECM_BLOCK 4-360
4.21.7 PRINT_ECM_DATA 4-364
4.22 Shmoo statement 4-366
4.22.1 SET_SHMOO_TG 4-366
4.22.2 SET_SHMOO_LEVEL 4-368
4.22.3 SET_SHMOO_POWER 4-370
4.22.4 RUN_SHMOO 4-371
4.22.5 RUN_SHMOO_EX 4-374
4.22.6 SET_SHMOO_COMPARE 4-377
4.22.7 SET_SHMOO_X(Y) 4-378
4.22.8 SET_SHMOO_COOR 4-379
4.23 MAWI2 Generator / Digitizer 4-382
4.23.1 Architecture 4-382
4.23.2 AWI2 Pins Declaration 4-383
4.23.3 Pattern Form 4-384
4.23.4 LOAD_ADDA_WAVEFORM 4-391
4.23.5 USE_WG_WAVEFORM 4-394
4.23.6 USE_WD_WAVEFORM 4-404
4.23.7 READ_WD_DATA_MS 4-415
4.23.8 WG_RELAY_ON 4-418
4.23.9 WG_RELAY_OFF 4-422
4.23.10 WD_RELAY_ON 4-425
4.23.11 WD_RELAY_OFF 4-428
4.23.12 WG_LS_TRIG 4-431
4.23.13 WD_LS_TRIG 4-438
4.23.14 INIT_WG_STATE 4-442

4.23.15 SET_WG_OFFSET 4-445
4.23.16 ADDA_PARAL_INC 4-447
4.23.17 ADDA_PIN_INC 4-449

5. Plan Program of Extension 5-1
5.1 Others 5-1
5.1.1 R_REG 5-1
5.1.2 W_REG 5-2
5.1.3 READ_LOAD_BOARD_ID 5-3
5.1.4 WRITE_LOAD_BOARD_ID 5-5
5.1.5 SHOW_TEST_TIME 5-7
5.1.6 TIMER_ON 5-8
5.1.7 TIMER_OFF 5-9
5.1.8 SPEC_DEF 5-10
5.1.9 DEBUG_BREAK 5-12
5.1.10 DO_WHEN_EOW 5-13
5.1.11 DATALOG_MSG 5-14
5.1.12 PRE-DEFINED PIN_GROUP 5-15
5.1.13 PRE-DEFINED TYPES 5-15
5.1.14 READ_WAFER_ID 5-16
5.1.15 READ_PROBER_XY 5-17
5.1.16 SET_DATALOG_MODE 5-18
5.1.17 ALARM_ON 5-20
5.1.18 ALARM_OFF 5-22
5.1.19 GET_ALARM_STATUS 5-23
5.1.20 WRITE_PAT_MEM 5-25
5.1.21 READ_PAT_MEM 5-26
5.1.22 READ_LOG_MEM 5-28
5.1.23 READ_LOG_MEM_BLOCK 5-29
5.1.24 SET_UPAT_REG 5-31
5.1.25 READ_DMM_VOLT 5-33
5.1.26 SET_STEP_LEVEL_MODE 5-34
5.1.27 SET_FAIL_BIN 5-37
5.1.28 TDO_PRINTF 5-38
5.1.29 CRAFT_STATEMENT 5-39
5.1.30 WRITE_FILE 5-40
5.1.31 READ_FILE 5-42
5.1.32 JUDGE_ARRAY 5-44
5.1.33 INPUT_BOX 5-46
5.1.34 ACCEPT_VER_EQ 5-47
5.1.35 ACCEPT_VER_GE 5-48
5.1.36 GET_LABEL_ADDR 5-49
5.1.37 READ_TESTER_MODEL 5-50
5.1.38 ABORT_TEST_PLN 5-51
5.1.39 PAT_SECURITY 5-52
5.1.40 TRM_PRINTF 5-54
5.1.41 READ_SYSTEM_LOAD_BOARD_ID 5-55
5.1.42 READ_LOAD_BOARD_ID_2 5-57
5.1.43 READ_SYSYEM_VENDOR_ID 5-59
5.1.44 JUDGE_SPI_WRITE 5-60
5.1.45 JUDGE_SPI_READ 5-63
5.2 CRAFT Macro Define 5-66
5.2.1 SELECTED_PIN_LIST 5-66
5.2.2 SELECTED_CONDITION 5-66
5.2.3 SELECTED_SPEC_DEF 5-66
5.2.4 TEST_LOT_ID 5-67
5.2.5 TEST_ROM_CODE 5-68
5.2.6 TEST_DEVICE_NAME 5-69
5.2.7 TEST_OPERATOR 5-70

5.2.8 TEST_SFT 5-71
5.2.9 TEST_PROBE_CARD_NO 5-72
5.2.10 TEST_PROBER_NAME 5-73
5.2.11 TEST_STATION 5-74
5.2.12 TEST_RUN_CARD_NO 5-75
5.2.13 TEST_TEMPERATURE 5-76
5.2.14 DISABLE_SYSTEM_EOT_RESET 5-77
5.2.15 GET_TEST_COUNT 5-78
5.2.16 GLOBAL PRODID 5-80
5.2.17 PLAN_VERSION 5-81
5.2.18 DISABLE_EOT_RESET 5-82
5.2.19 SET_PE_RELAY_XOR 5-83
5.2.20 CHECK_PE_RELAY_HOT_SWITCHING 5-84
5.2.21 DISABLE_RELAY_HOT_SWITCHING_MESSAGE 5-85
5.2.22 ALLOW_MULTI_JUDGE_IN_FAIL_REJUDGE 5-86
5.2.23 HWBIN 5-87
5.2.24 SWBIN 5-87
5.2.25 PHIF_X 5-87
5.2.26 PHIF_Y 5-87
5.2.27 TEST_WAFER_ID 5-88
5.2.28 TEST_WAFERNO 5-88
5.3 C Language Library 5-89
5.3.1 Method 5-89
5.3.2 GetPGChannel 5-90
5.3.3 GetPGSiteChannel 5-92
5.3.4 GetPinSiteAllIOChannel 5-93
5.3.5 GetPinSiteAllDPSChannel 5-95
5.3.6 GetPinSiteAllPREFChannel 5-97
5.3.7 GetPinSiteAllUVIChannel 5-99
5.3.8 GetPinSiteAllTRGChannel 5-100
5.3.9 DPS2SITE 5-101
5.3.10 PREF2SITE 5-102
5.3.11 UVI2SITE 5-103
5.3.12 GetPinNameByChannel 5-104
5.3.13 GetDutPin 5-106
5.3.14 GetAllPinName 5-108
5.3.15 GetPinChannel 5-110
5.3.16 CRAFT_c_get_brd_type 5-111
5.3.17 CRAFT_c_get_slot_config 5-112
5.3.18 CRAFT_c_pin_mode_w 5-114
5.3.19 CRAFT_c_relay_on_w 5-117
5.3.20 CRAFT_c_relay_off_w 5-119
5.3.21 CRAFT_c_frun_clk_w 5-121
5.3.22 CRAFT_c_set_leveln_w 5-123
5.3.23 CRAFT_c_meter_GPIB_Init 5-125
5.3.24 CRAFT_c_meter_GPIB_Close 5-127
5.3.25 CRAFT_c_meter_GPIB_Meas_volt 5-129
5.3.26 CRAFT_c_pma_address_r 5-132
5.3.27 GET_TRM_PATH 5-134
5.3.28 CRAFT_c_get_slot_temperature 5-135
5.3.29 CRAFT_c_shuttle_relay_w 5-137
5.3.30 CRAFT_c_get_load_board_id 5-139
5.3.31 CRAFT_c_set_frun_clk_mode 5-142
5.3.32 CRAFT_c_get_system_id 5-146
5.3.33 CRAFT_c_get_capture_sector_mode 5-148
5.3.34 CRAFT_c_set_capture_sector_mode 5-150
5.3.35 CRAFT_c_set_awi_wf_mem_point_mode 5-153
5.3.36 CRAFT_c_get_awi_wf_mem_point_mode 5-155
5.4 Gang-mode Function 5-157

5.4.1 POWER_GANG_PIN_GROUP 5-157
5.5 PE HV Function 5-158
5.5.1 PIN_MODE_HV 5-158
5.5.2 SET_LEVELN_HV 5-160
5.5.3 MARKERN_HV 5-163
5.6 RF Subsystem (Option) 5-166
5.6.1 RF Pins Declaration 5-166
5.6.2 RF_Initialize 5-171
5.6.3 RF_System_Alignment 5-172
5.6.4 RF_Load_Modulation_File 5-172
5.6.5 RF_System_Temperature 5-173
5.6.6 RF_Measure 5-174
5.6.7 RF_JUDGE 5-177
5.6.8 RF_Modulation_Burst 5-180
5.6.9 RF_Stop_Modulation 5-182
Appendix A How to Use 4820 to Add Relay Control Signal 1
Appendix B How to Use ALPG (Including *.DEC, *.PAT, *.PLN) 1
Appendix C The Instruction of Scan Function 1
Appendix D How to Debug Scan Function 1
Appendix E Micro Instruction "JNZ, JMP, JSR Limitation 1
Appendix F MAWI2 of Tool Instruction 1

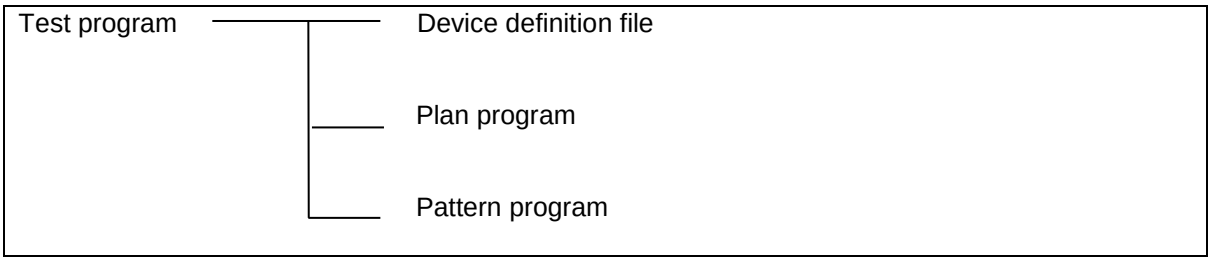
1. Overview of Test Program

1.1 Overview

The chapter contains a complete description of all of the language statements which apply to the test program code. 3380 language program is a C language and it provides a convenient way to specify the stimulus and expected response data for function testing. Device pins are grouped into meaningful sets. We can easily set the test condition, control the test sequence, execute the DC parametric test, and function test. The program specifies the plan of the test for DUT, please refers the below section.

1.2 Architecture of the Program

A 3380 test program consists of device definition, plan program and pattern program. Test program must be written for each device, and for each test method. It is necessary to prepare the following set of programs for each device.



Device definition file:

The file defines the correspondence between DUT pin numbers and ATE pin numbers, pin names and pin group names. The plan program and pattern program can use the names defined in this file.

Plan program:

The plan program defines a test flow that determines the execution order of test items including DC, function and AC testing, and control overall test sequence with test function.

Pattern program:

The pattern program uses the original pattern language to describe pattern data and expected value data for each pin. Pattern files must be separated from the plan program file. In a pattern file, the user defines wave format and pattern.

Device definition	:A file which extend name is "dec".
Plan program	: A file which extend name is "pln".
Pattern program	: A file which extend name is "pat".

2. Device Definition File

2.1 Overview

The file defines how to use each DUT pin in a test. It defines the correspondence between DUT pin numbers and ATE pin numbers, pin names and pin group names. The main program can use the names to define a device definition file.

Editor : General-purpose editor.
File name : A file which extend name is "dec".
Directory : Arbitrarily directory.

2.1.1 Format

Parameter list:

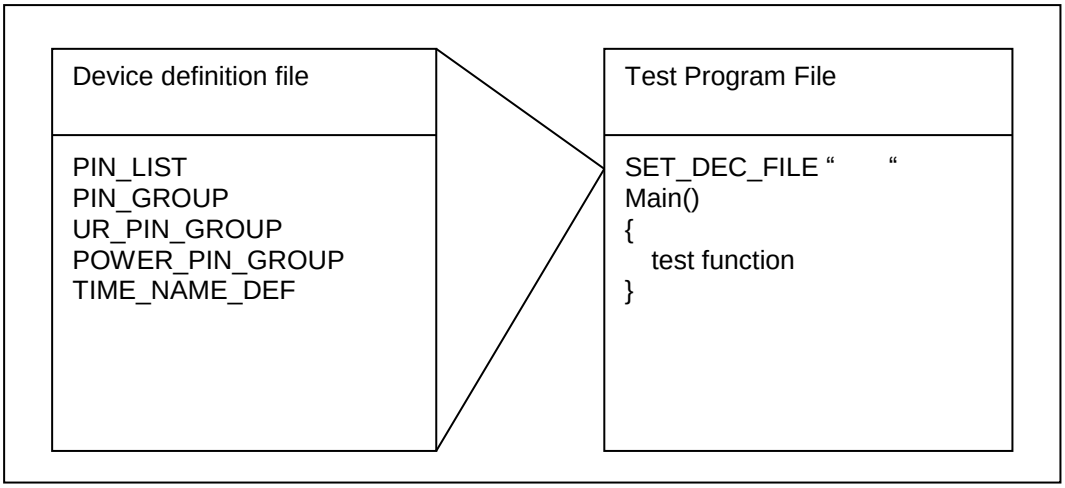
```
PIN_LIST
PIN_GROUP
UR_PIN_GROUP
POWER_PIN_GROUP
TIME_NAME_DEF
```

Parameter comment:

PIN_LIST : Define the relationship between test pin and DUT pin
PIN_GROUP : Be used to define groups of I/O pins.
UR_PIN_GROUP : Be used to define groups of user relay.
POWER_PIN_GROUP : Be used to define groups of power pin.
TIME_NAME_DEF : Specify a name to represent timing name.

Default value:
None.

2.1.2 Explanation



2.1.3 Example

```
PIN_LIST     (LPC_BOARD_00) {
  /*name = ATE channel = DUT channel = Type */
p0        =0          =1        =IO;
p1        =1          =2        =IO;
p2        =2          =3        =IO;
p3        =3          =4        =IO;
p4        =4          =5        =IO;
p5        =5          =6        =IO;
p6        =6          =7        =IO;
p7        =7          =8        =IO;
.       .       .       .
.       .       .       .
.       .       .       .
P15        =15          =16        =IO;
}
PIN_GROUP    {
       pg_pmu0_7 = p0+p1+p2+p3+p4+p5+p6+p7;
       Allpins = p0+p1+p2+p3+p4+p5+p6+p7+p8+p9+p10+p11+p12+p13+p14+p15;
       P10_15 = p10+p11+p12+p13+p14+p15;
}
TIME_NAME {
       TM1 = 1;
}
```

2.2 DEC_MODE

If the test program not declarae the mode, then default is NORM.

2.2.1 Format

DEC_MODE pin_mode ;

pin_mode: **APAS** (Any Pin to Any Site)
NORM (none APAS mode)
**no_entry: default NORM mode

2.2.2 Example

```
DEC_MODE   APAS;

PIN_LIST     (LPC_BOARD_00_2sites) {
  /*name = ATE channel = DUT channel = Type */
p0        =0   : 4       =1        =IO;
p1        =1   : 5       =2        =IO;
p2        =2   : 6       =3        =IO;
p3        =3   : 7       =4        =IO;
p4        =8   : 12      =5        =IO;
p5        =9   : 13      =6        =IO;
p6        =10 : 14      =7        =IO;
p7        =11 : 15       =8        =IO;
}
```

2.3 PIN_LIST Declaration

This section explains how to define device pins.

2.3.1 Format

Parameter list

```
PIN_LIST  ( loadboard )
 {
        pin_name =   ate_pin [:ate_pin] = dut_pin = pin_type ;
        .
        .
        pin_name =   ate_pin [:ate_pin] = dut_pin = pin_type ;
 }
```

Parameter comment:

Load board: The name of load board to be used by the test program.
Maximum: an array of 32 characters.
Type: string
No entry – illegal

pin_name: Specify the pin name for each defined DUT pin. This definition cannot be omitted.
Maximum: an array of 64 characters.
Type: string
No entry – illegal

ate_pin: Refer to the tester pin number. Declaring multiple pins within a range is possible. To describe the range, insert a colon sign " : " between pin boundaries.
Numbers – from 0 to 575 for each pin in 3380P
Numbers – from 0 to 1279 for each pin in 3380
Type: integer
This entry required for IN, OUT, IO, DPS, PREF, UR, TRG, and EXT.
Must be omitted for GND.

dut_pin: Refers to the DUT pin number. Declaring multiple pins within a range is possible. To describe the range, insert a colon sign " : " between pin boundaries. This value cannot be omitted if the DUT pin number equals to the tester pin number.
Maximum: an array of 8 characters for single pin

Type: string

No entry: Resource not connects to a device pin

Must be omitted for EXT, TRG, UR,

pin_Type: Select and specify the desired pin type for each defined DUT pin. This definition cannot be omitted. The lists of valid selections are:

IN	Pin used as input pin to the DUT.
OUT	Pin used as output from the DUT.
IO	Pin used as bi-directional (I/O) by the DUT.
MLDPS	Pin is a DUT Power Supply.
DPS	Pin is a DUT Power Supply.
UVI	Pin is a DUT Power pin.
PREF	Pin is a DUT Power pin.
TMU	Time measurement unit pin.
UR	Utility Relay pin
TRG	Trigger Pin.
GND	Pin is a DUT Ground pin.
WG	Waveform Generator
WD	Waveform Digitizer

Maximum: an array of 5 characters.

Type: string

No entry: illegal

Default value: None.

2.3.2 Explanation

- Uses define pin names, pin type and corresponding ATE pin number for each DUT pin. The DUT pin numbers and pin names defined here will be used in the test program.
- An error will occur if the same DUT or ATE pin numbers are defined more than once.
- An error will occur if the same pin name is given to more than one DUT pin. Pin names must be unique within the device definition.

Use a column sign " : "to declaring multi-sites

2.3.3 Example

< Device Definition File >

```
PIN_LIST   ( LS299_4_Sites )
{
        SEL0      =  0 : 288 : 320 : 352  =  1  =    IN  ;
        G1        =  2 : 290 : 322 : 354  =  2  =    IN  ;
        G2        =  4 : 292 : 324 : 356  =  3  =    IN  ;
        QG        =  6 : 294 : 326 : 358  =  4  =    IO  ;
        QE        =  8 : 296 : 328 : 360  =  5  =    IO  ;
        QC        = 10 : 298 : 330 : 362  =  6  =    IO  ;
        QA        = 12 : 300 : 332 : 364  =  7  =    IO  ;
        CLR       = 14 : 302 : 334 : 366  =  9  =    IN  ;
        SR        = 15 : 303 : 335 : 367  = 11  =    IN  ;
        CLK       = 13 : 301 : 333 : 365  = 12  =    IN  ;
        QB        = 11 : 299 : 331 : 363  = 13  =    IO  ;
        QD        =  9 : 297 : 329 : 361  = 14  =    IO  ;
        QF        =  7 : 295 : 327 : 359  = 15  =    IO  ;
        QH        =  5 : 293 : 325 : 357  = 16  =    IO  ;
        SL        =  3 : 291 : 323 : 355  = 18  =    IN  ;
        SEL1      =  1 : 289 : 321 : 353  = 19  =    IN  ;
        Vdps    =  576 : 577 : 578 : 579 = 21  =    DPS  ;
}

PIN_GROUP
{
  CTRL        = CLR+SEL0+SEL1+G1+G2+SL+SR;
  SEL01       = SEL0+SEL1;
  QQ          = QA+QB+QC+QD+QE+QF+QG+QH;
}
POWER_PIN_GROUP
{
    DPS_OS_PINS = Vdps;
}
TIME_NAME_DEF
{
    TM1=1;
}
```

2.4 PIN_GROUP

This section defines pin group name are given to the pins declared by PINLIST as a group.

[Notice]: Type IN and OUT can not be assigned in the same PIN_GROUP

2.4.1 Format

Parameter list:

```
PIN_GROUP {

        pin_group_name = pin_name operand pin_name operand …;

        pin_group_name = pin_group_name operand pin_name operand…;

                .

        .

        pin_group_name = pin_name operand pin_name operand…;

}
```

Parameter comment:

pin_group_name: Refer to a pin group name with a maximum 64 characters which begins with an alphabetical character.
Type: string
No entry: illegal

pin_name: Refer to a pin name with a maximum 64 characters which begins with an alphabetical character.

operand:

Type	Contents
+	Add a pin into this group
–	Remove a pin from this group.

Default Value:
IO_ALLPINS – A pin_group name, which include all pins of "IO", "IN", "OUT" pin-type.

[Notice]: TMU pin-type & pin_group can not be assigned in the same IO_ALLPINS

2.4.2 Explanation

Be used to define groups of pin. Grouping pins per each test condition facilitates programming. Pin group names be defined here will be used in the test program. An error will occur if the same pin group names are defined more than once. Pin group names must be unique within the device definition.
The same pin cannot be assigned to more than one group.

2.4.3 Example

< Device Definition File >

```
PIN_LIST   ( LS299_4_Sites )
{
        SEL0      =  0 : 288 : 320 : 352  =  1  =    IN  ;
        G1        =  2 : 290 : 322 : 354  =  2  =    IN  ;
        G2        =  4 : 292 : 324 : 356  =  3  =    IN  ;
        QG        =  6 : 294 : 326 : 358  =  4  =    IO  ;
        QE        =  8 : 296 : 328 : 360  =  5  =    IO  ;
        QC        = 10 : 298 : 330 : 362  =  6  =    IO  ;
        QA        = 12 : 300 : 332 : 364  =  7  =    IO  ;
        CLR       = 14 : 302 : 334 : 366  =  9  =    IN  ;
        SR        = 15 : 303 : 335 : 367  = 11  =    IN  ;
        CLK       = 13 : 301 : 333 : 365  = 12  =    IN  ;
        QB        = 11 : 299 : 331 : 363  = 13  =    IO  ;
        QD        =  9 : 297 : 329 : 361  = 14  =    IO  ;
        QF        =  7 : 295 : 327 : 359  = 15  =    IO  ;
        QH        =  5 : 293 : 325 : 357  = 16  =    IO  ;
        SL        =  3 : 291 : 323 : 355  = 18  =    IN  ;
        SEL1      =  1 : 289 : 321 : 353  = 19  =    IN  ;
        Vdps      =  576 : 577 : 578 : 579   = 21  =    DPS  ;
}

PIN_GROUP
{
  CTRL        = CLR+SEL0+SEL1+G1+G2+SL+SR;
  SEL01       = SEL0+SEL1;
  QQ          = QA+QB+QC+QD+QE+QF+QG+QH;
}

POWER_PIN_GROUP
{
    DPS_OS_PINS = Vdps;
}
```

[Notice]: Type IN and OUT can not be assigned in the same Pin group Name

2.5 UR_PIN_GROUP

Provides the ability to refer a set of user relay pin by a single name.

2.5.1 Format

Parameter list:

```
UR_PIN_GROUP {

        ur_pin_group_name = ur_pin_name operand ur_pin_name operand …;


        ur_pin_group_name = ur_pin_name operand ur_pin_name operand…;

}
```

Parameter comment:

ur_pin_group_name: Refer to a user relay pin group name with a maximum 64 characters which begins with an alphabetical character.

Type: string

No entry – illegal

ur_pin_name: Specify the pin name for each defined DUT pin. This definition cannot be omitted and the pin type must be UR (user relay).

operand:

Type	Contents
+	Add a pin into this group
–	Remove a pin from this group.

Default value:

UR_ALLPINS –A pin_group name, which include all pins of "UR" pin-type.

2.5.2 Explanation

Provides the ability to refer a set of user relay pin by a single name.

2.5.3 Example

< Device Definition File >

```
PIN_LIST   ( LS299_4_Sites )
{
        SEL0      =  0 : 288 : 320 : 352     =  1  =    IN  ;
        G1        =  2 : 290 : 322 : 354     =  2  =    IN  ;
        G2        =  4 : 292 : 324 : 356     =  3  =    IN  ;
        QG        =  6 : 294 : 326 : 358     =  4  =    IO  ;
        QE        =  8 : 296 : 328 : 360     =  5  =    IO  ;
        QC        = 10 : 298 : 330 : 362     =  6  =    IO  ;
        QA        = 12 : 300 : 332 : 364     =  7  =    IO  ;
        CLR       = 14 : 302 : 334 : 366     =  9  =    IN  ;
        SR        = 15 : 303 : 335 : 367     = 11  =    IN  ;
        CLK       = 13 : 301 : 333 : 365     = 12  =    IN  ;
        QB        = 11 : 299 : 331 : 363     = 13  =    IO  ;
        QD        =  9 : 297 : 329 : 361     = 14  =    IO  ;
        QF        =  7 : 295 : 327 : 359     = 15  =    IO  ;
        QH        =  5 : 293 : 325 : 357     = 16  =    IO  ;
        SL        =  3 : 291 : 323 : 355     = 18  =    IN  ;
        SEL1      =  1 : 289 : 321 : 353     = 19  =    IN  ;
        Vdps      =  576 : 577 : 578 : 579       = 21  =    DPS ;
        UR_C0     =  0 : 1  :  2  :  3       =     =    UR ;
        UR_C1     =  4 : 5  :  6  :  7       =     =    UR ;
}

UR_PIN_GROUP {

        UR_all      =UR_C0 + UR_C1 ;


}
```

2.6 POWER_PIN_GROUP

Provides the ability to refer a set of power pin by a single name.

2.6.1 Format

Parameter list:

```
POWER_PIN_GROUP {
        ps_pin_group_name = ps_pin_name operand ps_pin_name operand …;

        ps_pin_group_name = ps_pin_name operand ps_pin_name operand…;
}
```

Parameter comment:

ps_pin_group_name: Refer to a power pin group name with a maximum 64 characters which begins with an alphabetical character.

Type: string

No entry – illegal

ps_pin_name: Specify the pin name for each defined DUT pin. This definition cannot be omitted and the pin type must be DPS or PREF.

operand:

Type	Contents
+	Add a pin into this group
–	Remove a pin from this group.

Default value:

DPS_ALLPINS – A pin_group name, which include all pins of "DPS" pin-type.
PREF_ALLPINS – A pin_group name, which include all pins of "PREF" pin-type.

2.6.2 Example

< Device Definition File >

```
PIN_LIST   ( LS299_4_Sites )
{
        SEL0        =  0 : 288 : 320 : 352  =  1  =    IN  ;
        G1          =  2 : 290 : 322 : 354  =  2  =    IN  ;
        G2          =  4 : 292 : 324 : 356  =  3  =    IN  ;
        QG          =  6 : 294 : 326 : 358  =  4  =    IO  ;
        QE          =  8 : 296 : 328 : 360  =  5  =    IO  ;
        QC          = 10 : 298 : 330 : 362  =  6  =    IO  ;
        QA          = 12 : 300 : 332 : 364  =  7  =    IO  ;
        CLR         = 14 : 302 : 334 : 366  =  9  =    IN  ;
        SR          = 15 : 303 : 335 : 367  = 11  =    IN  ;
        CLK         = 13 : 301 : 333 : 365  = 12  =    IN  ;
        QB          = 11 : 299 : 331 : 363  = 13  =    IO  ;
        QD          =  9 : 297 : 329 : 361  = 14  =    IO  ;
        QF          =  7 : 295 : 327 : 359  = 15  =    IO  ;
        QH          =  5 : 293 : 325 : 357  = 16  =    IO  ;
        SL          =  3 : 291 : 323 : 355  = 18  =    IN  ;
        SEL1        =  1 : 289 : 321 : 353  = 19  =    IN  ;
        Vdps        =  576 : 577 : 578 : 579 = 21  =   DPS  ;
        UR0         =  0 :  1  :  2 :  3     =  =    UR  ;
        UR1         =  4 :  5  :  6 :  7     =  =    UR  ;
}

PIN_GROUP
{
   CTRL        = CLR+SEL0+SEL1+G1+G2+SL+SR;
   SEL01       = SEL0+SEL1;
   QQ          = QA+QB+QC+QD+QE+QF+QG+QH;
}
POWER_PIN_GROUP
{
     DPS_OS_PINS = Vdps;
}
UR_PIN_GROUP
{
     Relay_all = RL0+RL1;
}
TIME_NAME_DEF
{
     TM1
```

2.7 TIME_NAME_DEF

Assign a name to the timing number be used by the timing statement.

2.7.1 Format

Parameter list:

```
TIME_NAME_DEF {
      time_name   =   no   ;
      ……
      time_name   =   no   ;
                        }
```

Parameter comment:

time_name: Refer to a name assign to the timing number with a maximum 64 characters which begins with an alphabetical character.

no: Refer to the timing number, specified in an integer value from 1 to 15.

Default value: None.

2.7.2 Example

2.7.2.1 Example 1

Define in Device Definition File (*.dec file)

< Device Definition File >

```
PIN_LIST   ( LS299_4_Sites ){
          SEL0        =   0 : 288 : 320 : 352   =   1   =      IN   ;
          G1          =   2 : 290 : 322 : 354   =   2   =      IN   ;
          G2          =   4 : 292 : 324 : 356   =   3   =      IN   ;
          QG          =   6 : 294 : 326 : 358   =   4   =      IO   ;
          QE          =   8 : 296 : 328 : 360   =   5   =      IO   ;
          QC          = 10 : 298 : 330 : 362   =   6   =      IO   ;
          QA          = 12 : 300 : 332 : 364   =   7   =      IO   ;
          CLR         = 14 : 302 : 334 : 366   =   9   =      IN   ;
          SR          = 15 : 303 : 335 : 367   = 11   =      IN   ;
          CLK         = 13 : 301 : 333 : 365   = 12   =      IN   ;
          QB          = 11 : 299 : 331 : 363   = 13   =      IO   ;
          QD          =   9 : 297 : 329 : 361   = 14   =      IO   ;
          QF          =   7 : 295 : 327 : 359   = 15   =      IO   ;
          QH          =   5 : 293 : 325 : 357   = 16   =      IO   ;
          SL          =   3 : 291 : 323 : 355   = 18   =      IN   ;
          SEL1        =   1 : 289 : 321 : 353   = 19   =      IN   ;
          Vdps        =   576 : 577 : 578 : 579   = 21   =      DPS   ;
}
TIME_NAME_DEF
{
TM1= 1; // Define timing set 1 name is "TM1" for Test plan ( *.pln file )use;
}
```

<Test Plan used>

```
func_1MHZ_test()// Test Item
{
    DATALOG_MSG="func_1MHZ_test";
    FORCE_V_DPS(Vdps,5V,@8V, @100mA, 100mA, NORM, ON, 1mS) ;
    SET_LEVELN(func_lvl, IO_ALLPINS, 0V, 5V, 0.5V, 2.4V, 150uA, -150uA, 2.5V, , );
    LEVELS(func_lvl, 2mS);
    RELAY_ON(IO_ALLPINS, PDCL, 1mS);
    RATEN(TM1, 1uS); // TM1 define in DEC file
    MARKERN(TM1, CLK,250nS,750nS, ,0nS,0nS,850nS,850nS);
    MARKERN(TM1, CTRL,0nS,100nS, ,0nS,0nS,850nS,850nS);
    MARKERN(TM1, OUTPUTS,0nS,25nS, ,0nS,0nS,950nS,950nS);
    PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, NORM);
    PIN_MODE(CLK, RZ, EDGE, MASK, IO_NRZ, OFF);

    TEST_NO(20);
    JUDGE_PAT(clr_st, load_sp);
        :
        :
        :
        :
}
```

2.7.2.2 Example 2

Define in Test Plan and use it in test Plan (*.pln file)

Test pln file

```
    ..........
    ..........
TIME_NAME_DEF{
TM1= 1; // Define timing set 1 name is "TM1" for Test plan ( *.pln file )use;
}
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........

func_1MHZ_test() // Test Item
{
    DATALOG_MSG="func_1MHZ_test";
    FORCE_V_DPS(Vdps,5V,@8V, @100mA, 100mA, NORM, ON, 1mS) ;
    SET_LEVELN(func_lvl, IO_ALLPINS, 0V, 5V, 0.5V, 2.4V, 150uA, -150uA, 2.5V, , );
    LEVELS(func_lvl, 2mS);
    RELAY_ON(IO_ALLPINS, PDCL, 1mS);
    RATEN(TM1, 1uS); // TM1 define in pln file
    MARKERN(TM1, CLK,250nS,750nS, ,0nS,0nS,850nS,850nS);
    MARKERN(TM1, CTRL,0nS,100nS, ,0nS,0nS,850nS,850nS);
    MARKERN(TM1, OUTPUTS,0nS,25nS, ,0nS,0nS,950nS,950nS);
    PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, NORM);
    PIN_MODE(CLK, RZ, EDGE, MASK, IO_NRZ, OFF);

    TEST_NO(20);
    JUDGE_PAT(clr_st, load_sp);
        :
        :
        :
}
```

2.8 LEVEL_NAME_DEF

Assigns a name to the level number used by the level statements in "plan file".
(The level name will show on TCM IO-Pin level_set# to add the readability)

2.8.1 Format

Parameter list:

```
LEVEL_NAME_DEF {
    level_name  =  no  ;
    ……
    level_name  =  no  ;
}
```

Parameter comment:

level name: Refer to a name assign to the timing number with a maximum 64 characters which begins with an alphabetical character.
no: Refer to the timing number, specified in an integer value from 1 to 240.
Default value: None.

2.8.2 Example

2.8.2.1 Example 1

Define in Device Definition File (*.dec file)
< Device Definition File >

```
PIN_LIST   ( LS299_4_Sites ){
        SEL0      =  0 : 288 : 320 : 352  =  1  =    IN  ;
        G1        =  2 : 290 : 322 : 354  =  2  =    IN  ;
        G2        =  4 : 292 : 324 : 356  =  3  =    IN  ;
        QG        =  6 : 294 : 326 : 358  =  4  =    IO  ;
        QE        =  8 : 296 : 328 : 360  =  5  =    IO  ;
        QC        = 10 : 298 : 330 : 362  =  6  =    IO  ;
        QA        = 12 : 300 : 332 : 364  =  7  =    IO  ;
        CLR       = 14 : 302 : 334 : 366  =  9  =    IN  ;
        SR        = 15 : 303 : 335 : 367  = 11  =    IN  ;
        CLK       = 13 : 301 : 333 : 365  = 12  =    IN  ;
        QB        = 11 : 299 : 331 : 363  = 13  =    IO  ;
        QD        =  9 : 297 : 329 : 361  = 14  =    IO  ;
        QF        =  7 : 295 : 327 : 359  = 15  =    IO  ;
        QH        =  5 : 293 : 325 : 357  = 16  =    IO  ;
        SL        =  3 : 291 : 323 : 355  = 18  =    IN  ;
        SEL1      =  1 : 289 : 321 : 353  = 19  =    IN  ;
        Vdps      =  576 : 577 : 578 : 579  = 21  =  DPS  ;
}
TIME_NAME_DEF{
TM1= 1;
}
LEVEL_NAME_DEF {
        rst_lvl     = 1;
        os_lvl      = 2;
        func_lvl    = 3;
}
```

```
<Test Plan used>
func_1MHZ_test()// Test Item
{
      DATALOG_MSG="func_1MHZ_test";
      FORCE_V_DPS(Vdps,5V,@8V, @100mA, 100mA, NORM, ON, 1mS) ;
      SET_LEVELN(func_lvl, IO_ALLPINS, 0V, 5V, 0.5V, 2.4V, 150uA, -150uA, 2.5V, , );
      LEVELS(func_lvl, 2mS);
      RELAY_ON(IO_ALLPINS, PDCL, 1mS);
      RATEN(TM1, 1uS);
      MARKERN(TM1, CLK,250nS,750nS, ,0nS,0nS,850nS,850nS);
      MARKERN(TM1, CTRL,0nS,100nS, ,0nS,0nS,850nS,850nS);
      MARKERN(TM1, OUTPUTS,0nS,25nS, ,0nS,0nS,950nS,950nS);
      PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, NORM);
      PIN_MODE(CLK, RZ, EDGE, MASK, IO_NRZ, OFF);

      TEST_NO(20);
      JUDGE_PAT(clr_st, load_sp);
            :
            :
            :
            :
}
```

2.8.2.2 Example 2

Test pln file

Define in Test Plan and use it in test Plan (*.pln file)

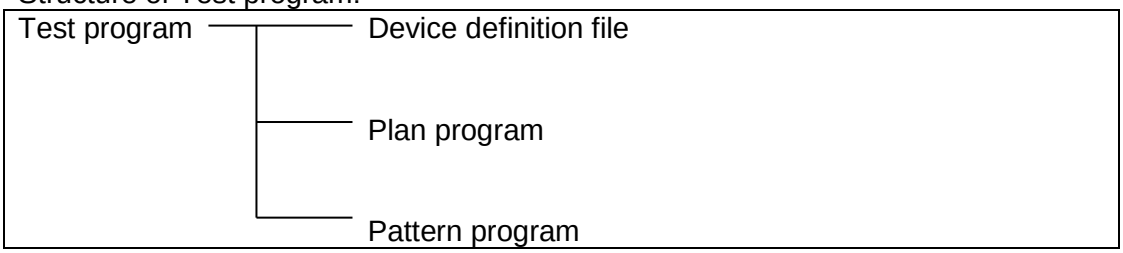
```
      ..........
      ..........
      ..........
TIME_NAME_DEF{
TM1= 1; // Define timing set 1 name is "TM1" for Test plan ( *.pln file )use;
}
LEVEL_NAME_DEF {
          rst_lvl          = 1;
          os_lvl           = 2;
          func_lvl         = 3;
}
      ..........
      ..........
      ..........
      ..........
func_1MHZ_test() // Test Item
{
      DATALOG_MSG="func_1MHZ_test";
      FORCE_V_DPS(Vdps,5V,@8V, @100mA, 100mA, NORM, ON, 1mS) ;
      SET_LEVELN(func_lvl, IO_ALLPINS, 0V, 5V, 0.5V, 2.4V, 150uA, -150uA, 2.5V, , );
      LEVELS(func_lvl, 2mS);
      RELAY_ON(IO_ALLPINS, PDCL, 1mS);
      RATEN(TM1, 1uS); // TM1 define in pln file
      MARKERN(TM1, CLK,250nS,750nS, ,0nS,0nS,850nS,850nS);
      PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, NORM);

      TEST_NO(20);
      JUDGE_PAT(clr_st, load_sp);
            :
            :
            :
}
```

3. Pattern Files

3.1 Overview

A pattern program comprises of pattern data that is used for function tests. Test pattern program is executing a call from the plan program.

Structure of Test program:



A pattern program source refers to a file, which is written in the pattern program language.
Editor: General-purpose editor
Extension of source file: ".pat"
Directory: Below "pattern" is in the data set directory.

3.1.1 Format

Parameter list:

SET_DEC_FILE "Device definition file"	Block
HEADER	Block
PATTERN	Block

Parameter comment:
SET_DEC_FILE: The block defines the declaration file name and its' directory.

HEADER: The block defines declaration that is used by the order of entry and display of the actual binary information, on a pin basis.

PATTERN: This block defines the pattern function and format.

3.1.2 Example

Include declaration file

```
SET_DEC_FILE "./ls299_16sites_pin.dec"

HEADER    CLR,%SEL0,SEL1,%G1,G2,%CLK,%SL,SR,%QA,QB,QC,
          QD,QE,QF,QG,QH;

SPM_PATTERN   (os_pat) {

os_st::                 *0 00 00 0 00 00000000 *TS15;
                        *0 00 00 0 00 00000000 *;
                        *0 00 00 0 00 00000000 *;
                        *0 00 00 0 00 00000000 * RPT 100;
                        *0 00 00 0 00 00000000 *;
                        *Z 00 00 0 00 00000000 *;
                        *0 Z0 00 0 00 00000000 *;
                        *0 0Z 00 0 00 00000000 *;
                        *0 00 Z0 0 00 00000000 *;
                        *0 00 0Z 0 00 00000000 *;
                        *0 00 00 Z 00 00000000 *;
                        *0 00 00 0 Z0 00000000 *;
                        *0 00 00 0 0Z 00000000 *;
                        *0 00 00 0 00 Z0000000 *;
                        *0 00 00 0 00 0Z000000 *;
                        *0 00 00 0 00 00Z00000 *;
                        *0 00 00 0 00 000Z0000 *;
                        *0 00 00 0 00 0000Z000 *;
                        *0 00 00 0 00 00000Z00 *;
                        *0 00 00 0 00 000000Z0 *;
                        *0 00 00 0 00 0000000Z *;
                        *0 00 00 0 00 00000000 * RPT 100;
                        *0 00 00 0 00 00000000 *;
        os_sp::         *0 00 00 0 00 00000000 *;

}
```

Specify the order the data pattern in terms of device pin names.

Use statement to declare the pattern memory section.

3.2 SET_DEC_FILE

The statement declares the declaration file name and its' directory.

3.2.1 Format

Parameter list:

SET_DEC_FILE "Device definition file"

Parameter comment:

Device definition file: A file that extension name is "dec".

3.2.2 Explanation

This section includes the device declaration file into this pattern file. (The device declaration file refers to "Device Declaration File"section.)

3.2.3 Example

```
SET_DEC_FILE "./ls299_16sites_pin.dec"

HEADER     CLR,%SEL0,SEL1,%G1,G2,%CLK,%SL,SR,%QA,QB,QC,
           QD,QE,QF,QG,QH;

SPM_PATTERN    (os_pat) {

os_st::                *0 00 00 0 00 00000000 *TS15;
                       *0 00 00 0 00 00000000 *;
                       *0 00 00 0 00 00000000 *;
                       *0 00 00 0 00 00000000 * RPT 100;
                       *0 00 00 0 00 00000000 *;
                       *Z 00 00 0 00 00000000 *;
                       *0 Z0 00 0 00 00000000 *;
                       *0 0Z 00 0 00 00000000 *;
                       *0 00 Z0 0 00 00000000 *;
                       *0 00 0Z 0 00 00000000 *;
                       *0 00 00 Z 00 00000000 *;
                       *0 00 00 0 Z0 00000000 *;
                       *0 00 00 0 0Z 00000000 *;
                       *0 00 00 0 00 Z0000000 *;
                       *0 00 00 0 00 0Z000000 *;
                       *0 00 00 0 00 00Z00000 *;
                       *0 00 00 0 00 000Z0000 *;
                       *0 00 00 0 00 0000Z000 *;
                       *0 00 00 0 00 00000Z00 *;
                       *0 00 00 0 00 000000Z0 *;
                       *0 00 00 0 00 0000000Z *;
                       *0 00 00 0 00 00000000 * RPT 100;
                       *0 00 00 0 00 00000000 *;
      os_sp::          *0 00 00 0 00 00000000 *;

 }
```

3.3 HEADER

The statement declaration is used by the order of entry and display of the actual binary information, on a pin basis.

3.3.1 Format

Parameter list:

HEADER	[%]pin_name, [%]pin_name, [%]**pin_group**, …;

Parameter comment:

Pin_name: A name of single pin defined in the PIN_LIST declaration. Only constant name from the PIN_LIST are allowed.
Type: STRING, 64 characters maximum.

%: A separator, indicating the compiler to use one space between the previous pin and next pin, in all of the output from the system, such as Pattern Editor Utility.

pin_group: This means the pin_group (for details refer to 3.4) is in Hexadecimal Format.

3.3.2 Example

The header declaration part states the input sequence of test pin names and data.

```
SET_DEC_FILE "./ls299_16sites_pin.dec"
HEADER     CLR,%SEL0,SEL1,%G1,G2,%CLK,%SL,SR,%QA,QB,QC,
           QD,QE,QF,QG,QH;




SPM_PATTERN    (os_pat) {

os_st::                *0 00 00 0 00 00000000 *TS15;
                       *0 00 00 0 00 00000000 *;
                       *0 00 00 0 00 00000000 *;
                       *0 00 00 0 00 00000000 * RPT 100;
                       *0 00 00 0 00 00000000 *;
                       *Z 00 00 0 00 00000000 *;
                       *0 Z0 00 0 00 00000000 *;
                       *0 0Z 00 0 00 00000000 *;
                       *0 00 Z0 0 00 00000000 *;
                       *0 00 0Z 0 00 00000000 *;
                       *0 00 00 Z 00 00000000 *;
                       *0 00 00 0 Z0 00000000 *;
                       *0 00 00 0 0Z 00000000 *;
                       *0 00 00 0 00 Z0000000 *;
                       *0 00 00 0 00 0Z000000 *;
                       *0 00 00 0 00 00Z00000 *;
                       *0 00 00 0 00 000Z0000 *;
                       *0 00 00 0 00 0000Z000 *;
                       *0 00 00 0 00 00000Z00 *;
                       *0 00 00 0 00 000000Z0 *;
                       *0 00 00 0 00 0000000Z *;
                       *0 00 00 0 00 00000000 * RPT 100;
                       *0 00 00 0 00 00000000 *;
       os_sp::         *0 00 00 0 00 00000000 *;
}
```

The sequence of PIN_PATTERN is equal to pattern sequence.

3.4 PATTERN

This section explains the function and format of SPM_Pattern.

3.4.1 SPM_PATTERN

The statement is used to define all of the test patterns, it is necessary to test a device functionally by using the Sequential Pattern Memory.

3.4.1.1 Format

Parameter list:

```
SPM_PATTERN   ( module_name [, NORM | DBL | DBL_2X, [ NORM_SET |K_SET |
Z_SET ] ])
 {
   Label ::  * pattern_data   * [ts_name, micro_ins ] ;
   Label :   * pattern_data   * [ts_name, micro_ins ] ;
             * pattern_data   # ape_field   * [ts_name, micro_ins ] ;
}
```

Parameter comment:

ape_field : AWI pattern field,
NOP, Exx or Pxx , xx:1~31
Pxx : Pattern trigger, waveform_set/memory_set :xx
Exx : External trigger, waveform_set/memory_set :xx

module_name: A name to be given to the set of patterns being defined in the body of this pattern statement. This name is not used in pattern references within the test program, but it is used by debug tools, such as Pattern Editor Tool.
Type: string, 32 characters maximum.
No entry: illegal.

NORM / DBL / DBL_2X : no-entry --> NORM

DBL_2X pattern setting has K_SET / Z_SET: no-entry --> NORM_SET
DBL limitation : only even cycle can use micro-instruction , and moudle pattern start from even address
DBL_2X: only 4*I th(i=0,1,2,3….) cycle can use micro-instruction

Label : A label can be local to the pattern module that is being defined (by filling the name with a single colon ":") or as a global label available to all pattern and test program files (by filling the name with a double colon "::")
"C" type : string, 16 characters maximum.

pattern_data : The information contained in the pattern vectors is supplied by a set of pre-defined characters. This set is designed for meaningful equivalence to the actual application it is used normally.

3.4.1.2 Pattern Symbol List

The set of characters as below:

Symbol	Meaning
0	Drive on and low
1	Drive on and high
L	Drive off and compare for a low
H	Drive off and compare for a high
X	Drive off and don't care comparator
2	Drive VHH in MXLPC board (HVIN pin type) Release version : **CRAFT_3380_2.00**
U	Drive low and compare low (underside)
S	Drive high and compare low (sink)
R	Drive low and compare high (rise)
T	Drive high and compare high (top)
V	Drive off and compare for valid state (V)
Z	Drive off and compare for tri-state (Z)
K	Drive trigger for statement "SET_CAPTURE_MEM_MODE"

Leading symbol for pattern in hexadecimal:

Symbol	Meaning
c	Compare data (L=0, H=1)
d	Drive data (0=0, 1=1)
z	Tri state (V=0, Z=1)
x	Don't care (X=1)
t	(U=0, T=1)
s	(R=0, S=1)

```
SET_DEC_FILE "./3360_ls299_pin.dec"

HEADER   CTRL1, %CLK, %QQ, %OAH;

SPM_PATTERN(func_pat) {
 sfr_st:   *1   01   00   1   X1   HLLLLLLL   HL*TS1; // shift H signal start
           *1   01   00   1   X1   HHLLLLLL   HL*;
           *1   01   00   1   X1   HHHLLLLL   HL*;
           *1   01   00   1   X1   HHHHLLLL   HL*;
           *1   01   00   1   X1   HHHHHLLL   HL*;
           *1   01   00   1   X1   HHHHHHLL   HL*;
           *1   01   00   1   X1   HHHHHHHL   HL*;
           *1   01   00   1   X1   HHHHHHHH   HL*;
}
```

HLLL➔c8 (**hexadecimal**)

```
SET_DEC_FILE "./3360_ls299_pin.dec"

HEADER   CTRL1, %CLK, %QQ, %OAH;
```

```
SPM_PATTERN(func_pat) {
 sfr_st:   *dA0   1   X1   c80   c8*TS1; // shift H signal start
           * dA0  1   X1   cC0   c8*;
           * dA0  1   X1   cE0   c8*;
           * dA0  1   X1   cF0   c8*;
           * dA0  1   X1   cF8   c8*;
           * dA0  1   X1   cFC   c8*;
           * dA0  1   X1   cFE   c8*;
           * dA0  1   X1   cFF   c8*;
}
```

ts_name: The time set pointer for the vector being specified. The range of valid time setting is from 1 to 15. This can be specified by the Keywords TS1 to TS15 or the time name defined by "TIME_NAME_DEF" .
This is an option entry. If not be specified, the time set name of the previous vector is also assign to this vector.

micro_instruction: A micro-instruction mnemonic, which is executed by the micro-controller in parallel with the use of defined vector data and time set pointer.

3.4.1.3 Micro-instruction

Following are very brief descriptions of all the micro-instruction.

Micro-instruction	Description
NOP	No operation
RPT	Repeat count: repeat the same address. RPT times 2 <= N <= 16777215 (24bit register)
STOP	Stop pattern execution.
SETM	Match count: set match count to index match register, match_count : 30 <= N <= 1677215 (24bit register)
IMATCH (only for normal mode pattern)	Immediately match and this instruction execute a single vector match. It will repeat the vector until a match occurs or the count specified in the data field of the **SETM** micro-instruction is decremented to zero .
DPS	High speed pattern trigger to measure DPS simultaneously.
DPSMST	Start marker for recording the maximum value.
PREF	High speed pattern trigger to measure PREF simultaneously
SETXi	count_value, i = 0~3: count_value : 2 <= N <= 16777215 (24bit register) .Need to be used with "JNZi"
JNZi	label , i = 0~3:

JMP label	Unconditional jump
JCONT label	Continuous loop jmp, execute the "PAT_CONTINUE()" statement in the test plan program.
SETS	count_value: Set Subroutine Counter (24 bits counter)
JSR label	Subroutine call
RET label	Returning from the subroutine to the label
TFTM	Trigger of Freq. And Time Measurement
UISYNC	Send trigger signal to test head from u-instruction.
LDXN	LDXN Xi+Xj : I, j = 0~7 : Load Xi index counter value from index holding register (X0~X7,setup by test plan)
RPTN	RPTN Xi : i = 0~7 : repeat count define by index holding register(X0 ~ X7 , setup by test plan)

3.4.1.4 Explanation

The Sequential Pattern Memory is provide to specify the content of the pattern memory, and is used as the main routine of the pattern program; each pattern memory section can be defined in each program.

SPM_PATTERN (func_pat , **NORM** , K_SET | Z_SET **)** → compiler error
SPM_PATTERN (func_pat , **DBL** , K_SET | Z_SET **)** → compiler error
SPM_PATTERN (func_pat , **DBL_2X** , K_SET | Z_SET **)** → compiler correct

3.4.1.5 Example

```
SET_DEC_FILE "./Is299_16sites_pin.dec"
HEADER     CLR,%SEL0,SEL1,%G1,G2,%CLK,%SL,SR,%QA,QB,QC,
           QD,QE,QF,QG,QH;

SPM_PATTERN    (os_pat) {
os_st::            *0 00 00 0 00 00000000 *TS15;
                   *0 00 00 0 00 00000000 *;
                   *0 00 00 0 00 00000000 *;
                   *0 00 00 0 00 00000000 * RPT 100;
                   *0 00 00 0 00 00000000 *;
                   *Z 00 00 0 00 00000000 *;
                   *0 Z0 00 0 00 00000000 *;
                   *0 0Z 00 0 00 00000000 *;
                   *0 00 Z0 0 00 00000000 *;
                   *0 00 0Z 0 00 00000000 *;
                   *0 00 00 Z 00 00000000 *;
                   *0 00 00 0 Z0 00000000 *;
                   *0 00 00 0 0Z 00000000 *;
                   *0 00 00 0 00 Z0000000 *;
                   *0 00 00 0 00 0Z000000 *;
                   *0 00 00 0 00 00Z00000 *;
```

Define use SPM_PATTERN

```
                    *0 00 00 0 00 000Z0000 *;
                    *0 00 00 0 00 0000Z000 *;
                    *0 00 00 0 00 00000Z00 *;
                    *0 00 00 0 00 000000Z0 *;
                    *0 00 00 0 00 0000000Z *;
                    *0 00 00 0 00 00000000 * RPT 100;
                    *0 00 00 0 00 00000000 *;
      os_sp::       *0 00 00 0 00 00000000 *;
}
```

3.4.2 APM_PATTERN

The statement is used to define all of the test patterns necessary to test a device functionally using the ALPG Pattern Memory .

3.4.2.1 Format

Parameter list:

```
APM_PATTERN   ( module_name [, NORM | DBL ] )   {
  Label ::  * pattern_data   * [ts_name, micro_ins] ; [ [ X-alpg-micro_ins , Y-alpg-micro_ins ,
                                                          Z-alpg-micro_ins ] ]
  Label :   * pattern_data   * [ts_name, micro_ins] ; [ [ X-alpg-micro_ins , Y-alpg-micro_ins ,
                                                          Z-alpg-micro_ins ] ]
}
```

Parameter comment:

```
APM_PATTERN (ZeroOne) {
  ZeroOne_st::

    *XXXXXXXX 111111 11* TS1;          NOP;

  ZeroOne_w0::

    *CCCCCCCC BBBAAA 01* TS1,SETX0 7; [X=XA:0,Y=YA:0,ZA:0,ZB:0,Z=ZA]
    *00000000 BBBAAA 01* TS1,RPT 7;       [ X=++XA ,     Y=YA ;     ]
    *00000000 BBBAAA 01* TS1,JNZ0 .-1; [ X=XA:0 ,       Y=++YA ; ]
    *00000000 BBBAAA 01* TS1,RPT 7;       [ X=++XA ,       Y=YA ;     ]
     ....
 }
```

ALPG Pattern Symbol :

address	symbol
X-Address driver pattern symbol	"A" (0/1)
Y-Address driver pattern symbol	"B" (0/1)
Z-Address Driver pattern symbol	"D" (0/1)
Z-Address Compare pattern symbol	"C" (L/H)

ALPG-Micro-Instruction-List :

symbol	define
XA@R	XACNT clear to 0s
XA:data	XACNT load from immediately data
XA=X	XACNT load from XALU

++XA	XACNT +1
--XA	XACNT -1
++XA<	XACNT +1 if carry
--XA<	XACNT -1 if borrow
XA=HAn	XACNT load from Hoding register(HA0~HA7)
XA=XB	XACNT load from XB
XA=XC	XACNT load from XC
XA=XD	XACNT load from XD
XB@R	----", 4, 0, "0001", 4,39,36),-- 07,AUI[39:36] XBCNT clear to 0s
XB:data	XBCNT load from immediately data
XB=X	XBCNT load from XALU
++XB	XBCNT +1
--XB	XBCNT -1
++XB<	XBCNT +1 if carry
--XB<	XBCNT -1 if borrow
XB=HBn	XBCNT load from Hoding register(HB0~HB7)
XB=XA	XBCNT load from XA
XB=XC	XBCNT load from XC
XB=XD	XBCNT load from XD
XC@R	XCCNT clear to 0s
XC:data	XCCNT load from immediately data
XC=X	XCCNT load from XALU
++XC	XCCNT +1
--XC	XCCNT -1
++XC<	XCCNT +1 if carry
--XC<	XCCNT -1 if borrow
XC=HAn	XDCNT load from Hoding register(HA0~HA7)
XC=XA	XDCNT load from XA
XC=XB	XDCNT load from XB
XC=XD	XDCNT load from XD
XD@R	XDCNT clear to 0s
XD:data	XDCNT load from immediately data
XD=X	XDCNT load from XALU
++XD	XDCNT +1
--XD	XDCNT -1
++XD<	XDCNT +1 if carry
--XD<	XDCNT -1 if borrow
XD=HBn	XDCNT load from Hoding register(HB0~HB7)
XD=XA	XDCNT load from XA
XD=XB	XDCNT load from XB
XD=XC	XDCNT load from XC
X=XA	XALU XA
X=~XA	XALU !XA
X=2XA	XALU 2XA=(XA+XA)
X=0	XALU 0=(XA-XA)
X=XB	XALU XB
X=~XB	XALU !XB
X=XA+XB	XALU XA+XB
X=XA-XB	XALU XA-XB
X=X<<N	XALU X barrel Shift N(N=1/2/4/8),load from immediate data
X=XC	XALU XC
X=~XC	XALU !XC
X=2XC	XALU 2XC=(XC+XC)
X=0	XALU 0=(XC-XC)
X=XD	XALU XD

X=~XD	XALU !XD
X=XC+XD	XALU XC+XD
X=XC-XD	XALU XC-XD
XMK:data	XADDR load XA/XB mask via immediately data
XF0	XADDR force XADDR to 0s
XF1	XADDR force XADDR to 1s
YA@R	YACNT clear to 0s
YA:data	YACNT load from immediately data
YA=Y	YACNT load from YALU
++YA	YACNT +1
--YA	YACNT -1
++YA<	YACNT +1 if carry
--YA<	YACNT -1 if borrow
YA=HAn	YACNT load from Hoding register(HA0~HA7)
YA=YB	YACNT load from YB
YA=YC	YACNT load from YC
YA=YD	YACNT load from YD
YB@R	YBCNT clear to 0s
YB:data	YBCNT load from immediately data
YB=Y	YBCNT load from YALU
++YB	YBCNT +1
--YB	YBCNT -1
++YB<	YBCNT +1 if carry
--YB<	YBCNT -1 if borrow
YB=HBn	YBACNT load from Hoding register(HB0~HB7)
YB=YA	YBCNT load from YA
YB=YC	YBCNT load from YC
YB=YD	YBCNT load from YD
YC@R	YCCNT clear to 0s
YC:data	YCCNT load from immediately data
YC=Y	YCCNT load from YALU
++YC	YCCNT +1
--YC	YCCNT -1
++YC<	YCCNT +1 if carry
--YC<	YCCNT -1 if borrow
YC=HAn	YCACNT load from Hoding register(HA0~HA7)
YC=YA	YCCNT load from YA
YC=YB	YCCNT load from YB
YC=YD	YCCNT load from YD
YD@R	YDCNT clear to 0s
YD:data	YDCNT load from immediately data
YD=Y	YDCNT load from YALU
++YD	YDCNT +1
--YD	YDCNT -1
++YD<	YDCNT +1 if carry
--YD<	YDCNT -1 if borrow
YD=HBn	YDACNT load from Hoding register(HB0~HB7)
YD=YA	YDCNT load from YA
YD=YB	YDCNT load from YB
YD=YC	YDCNT load from YC
Y=YA	YALU YA
Y=~YA	YALU !YA
Y=2YA	YALU 2YA=(YA+YA)
Y=0	YALU 0=(YA-YA)
Y=YB	YALU YB

Y=~YB	YALU !YB
Y=YA+YB	YALU YA+YB
Y=YA-YB	YALU YA-YB
Y=Y<<N	YALU Y barrel Shift N(N=1/2/4/8),load from immediate data
Y=YC	YALU YC
Y=~YC	YALU !YC
Y=2YC	YALU 2YC=(YC+YC)
Y=0	YALU 0=(YC-YC)
Y=YD	YALU YD
Y=~YD	YALU !YD
Y=YC+YD	YALU YC+YD
Y=YC-YD	YALU YC-YD
YMK:data	YADDR load YA/YB mask via immediately data
YF0	YADDR force YADDR to 0s
YF1	YADDR force YADDR to 1s
ZA@R	ZACNT clear to 0s
ZA:data	ZACNT load data from immediate data
ZA@S	ZACNT set to 1s
++ZA	ZACNT incrment
--ZA	ZACNT decrement
ZA<<	ZACNT rotate left
ZA>>	ZACNT rotate right
ZA<<0	ZACNT shift left & carry in with 0
ZA0>>	ZACNT shift right & carry in with 0
ZA<<1	ZACNT shift left & carry in with 1
ZA1>>	ZACNT shift right & carry in with 1
ZA=HAn	ZACNT load from Hoding register(HA0~HA7)
ZA=ZB	ZACNT load from ZB
ZA=ZC	ZACNT load from ZC
ZA=ZD	ZACNT load from ZD
ZB=0	ZBCNT clear to 0s
ZB:data	ZBCNT load data from immediate data
ZB=~0	ZBCNT set to 1s
++ZB	ZBCNT incrment
--ZB	ZBCNT decrement
ZB<<	ZBCNT rotate left
ZB>>	ZBCNT rotate right
ZB<<0	ZBCNT shift left & carry in with 0
ZB0>>	ZBCNT shift right & carry in with 0
ZB<<1	ZBCNT shift left & carry in with 1
ZB1>>	ZBCNT shift right & carry in with 1
ZB=HBn	ZBCNT load from Hoding register(HB0~HB7)
ZB=ZA	ZBCNT load from ZA
ZB=ZC	ZBCNT load from ZC
ZB=ZD	ZBCNT load from ZD
ZC@R	ZCCNT clear to 0s
ZC:data	ZCCNT load data from immediate data
ZC@S	ZCCNT set to 1s
++ZC	ZCCNT incrment
--ZC	ZCCNT decrement
ZC<<	ZCCNT rotate left
ZC>>	ZCCNT rotate right
ZC<<0	ZCCNT shift left & carry in with 0
ZC0>>	ZCCNT shift right & carry in with 0
ZC<<1	ZCCNT shift left & carry in with 1

ZC1>>	ZCCNT shift right & carry in with 1
ZC=HAn	ZCCNT load from Hoding register(HA0~HA7)
ZC=ZA	ZCCNT load from ZA
ZC=ZB	ZCCNT load from ZB
ZC=ZD	ZCCNT load from ZD
ZD@R	ZDCNT clear to 0s
ZD:data	ZDCNT load data from immediate data
ZD@S	ZDCNT set to 1s
++ZD	ZDCNT incrment
--ZD	ZDCNT decrement
ZD<<	ZDCNT rotate left
ZD>>	ZDCNT rotate right
ZD<<0	ZDCNT shift left & carry in with 0
ZD0>>	ZDCNT shift right & carry in with 0
ZD<<1	ZDCNT shift left & carry in with 1
ZD1>>	ZDCNT shift right & carry in with 1
ZD=HBn	ZDCNT load from Hoding register(HB0~HB7)
ZD=ZA	ZDCNT load from ZA
ZD=ZB	ZDCNT load from ZB
ZD=ZC	ZDCNT load from ZC
Z=0	ZBLU all 0s
Z=ZA^ZB	ZBLU ZA xor ZB
Z=ZA	ZBLU ZA
Z=ZB	ZBLU ZB
Z=~0	ZBLU all 1s
Z=~(ZA^ZB)	ZBLU ZA xnor ZB
Z=~ZA	ZBLU !ZA
Z=~ZB	ZBLU !ZB
Z=ZA==ZB	ZDATA ZA=ZB
Z=ZA!=ZB	ZDATA ZF!=ZB
Z=ZA==ZB ? ZC:ZD	if ZA==ZB then Z=ZC else Z=ZD
Z=ZA==ZB ? ZC:!ZC	if ZA==ZB then Z=ZC else Z=!ZC
Z=ZA==ZB ? ZD:!ZD	if ZA==ZB then Z=ZD else Z=!ZD
Z=ZC==ZD ? ZA:ZB	if ZC==ZD then Z=ZA else Z=ZB
Z=ZC==ZD ? ZA:!ZA	if ZC==ZD then Z=ZA else Z=!ZA
Z=ZC==ZD ? ZB:!ZB	if ZC==ZD then Z=ZB else Z=!ZB
ZMK:data	ZDATA load ZA/ZB mask via immediately data
ZF0	ZDATA force ZDATA to 0s
ZF1	ZDATA force ZDATA to 1s
NOP	nop,unused
Z=Z<<N	ZALU Z barrel Shift N(N=1/2/4/8),load from immediate data
Z=ZC^ZD	ZBLU ZC xor ZD
Z=ZC	ZBLU ZC
Z=ZD	ZBLU ZD
Z=~(ZC^ZD)	ZBLU ZC xnor ZD
Z=~ZC	ZBLU !ZC
Z=~ZD	ZBLU !ZD
Z=ZC==ZD	ZDATA ZC=ZD
Z=ZC!=ZD	ZDATA ZC!=ZD

counter [XA] operator :

symbol	define
XA@R	clear to 0s
XA:data	load from immediately data
XA=X	load from XALU

++XA	+1
--XA	-1
++XA<	+1 if carry YA
--XA<	-1 if borrow YA

counter [XB] operator :

Symbol	define
XB@R	clear to 0s
XB:data	load from immediately data
XB=X	load from XALU
++XB	+1
--XB	-1
++XB<	+1 if carry YA
--XB<	-1 if borrow YA

X-ALU operator :

Symbol	define
X=XA	XA
X=~XA	!XA
X=2XA	2XA=(XA+XA)
X=0	0=(XA-XA)
X=XB	XB
X=~XB	!XB
X=XA+XB	XA+XB
X=XA-XB	XA-XB
XF0	XADDR force XADDR to 0s
XF1	XADDR force XADDR to 1s

counter [YA] operator :

Symbol	define
YA@R	clear to 0s
YA:data	load from immediately data
YA=Y	load from YALU
++YA	+1
--YA	-1
++YA<	+1 if carry XA
--YA<	-1 if borrow XA

counter [YB] operator :

Symbol	define
YB@R	clear to 0s
YB:data	load from immediately data
YB=Y	load from YALU
++YB	+1
--YB	-1
++YB<	+1 if carry XA
--YB<	-1 if borrow XA

Y-ALU operator :

Symbol	define
Y=YA	YA
Y=~YA	!YA
Y=2YA	2YA=(YA+YA)
Y=0	0=(YA-YA)
Y=YB	YB
Y=~YB	!YB

Y=YA+XB	YA+YB
Y=YA-XB	YA-YB
YF0	YADDR force YADDR to 0s
YF1	YADDR force YADDR to 1s

shifter-counter [ZA] operation :

Symbol	define
ZA@R	clear to 0s
ZA:data	load data from immediate data
ZA@S	set to 1s
++ZA	+1
--ZA	-1
ZA<<	rotate left
ZA>>	rotate right
ZA<<0	shift left & carry in with 0
ZA0>>	shift right & carry in with 0
ZA<<1	shift left & carry in with 1
ZA1>>	shift right & carry in with 1

shifter-counter [ZB] operation :

Symbol	define
ZB@R	clear to 0s
ZB:data	load data from immediate data
ZB@S	set to 1s
++ZB	+1
--ZB	-1
ZB<<	rotate left
ZB>>	rotate right
ZB<<0	shift left & carry in with 0
ZB0>>	shift right & carry in with 0
ZB<<1	shift left & carry in with 1
ZB1>>	shift right & carry in with 1

Z-BLU (bit logic unit) operation :

Symbol	define
Z=0	Z@R, all 0s
Z=ZA^ZB	ZA xor ZB
Z=ZA	ZF
Z=ZB	ZB
Z=1	Z@S, all 1s
Z=~(ZA^ZB)	ZA xnor ZB
Z=~ZA	!ZA
Z=~ZB	!ZB
Z=ZA==ZB	ZA=ZB
Z=ZA!=ZB	ZF!=ZB
Z=ZA==ZB ? ZC:ZD	if ZA==ZB then Z=ZC else Z=ZD
Z=ZA==ZB ? ZC:!ZC	if ZA==ZB then Z=ZC else Z=!ZC
Z=ZA==ZB ? ZD:!ZD	if ZA==ZB then Z=ZD else Z=!ZD
Z=ZC==ZD ? ZA:ZB	if ZC==ZD then Z=ZA else Z=ZB
Z=ZC==ZD ? ZA:!ZA	if ZC==ZD then Z=ZA else Z=!ZA
Z=ZC==ZD ? ZB:!ZB	if ZC==ZD then Z=ZB else Z=!ZB

3.4.3 Pattern Micro-instruction

RPT repeat time: repeat the same address
Repeat time: specify the desired repeat times in a decimal or hexadecimal Value
Value: 2 <= N <= 16777215 (24bit register)

```
SET_DEC_FILE "./3360_ls299_pin.dec"

HEADER     CLR,%SEL1,SEL0,%OE1,OE2,%CLK,%SL,SR,%QA,QB,QC,
           QD,QE,QF,QG,QH,%OA,OH;

SPM_PATTERN (rpt_pat) {//RPT from 2 to 16777215

                    /*C SS OO C SS ABCDEFGH OO*/
                    /*L EE EE L LR QQQQQQQQ AH*/
                    /*R LL 12 K      ABCDEFGH     */
                    /*   10                      */
  rpt_st::          *0 X0 00 X XX LLLLLLLL LL*TS1;//-------clear data using clear1
                     *1 01 00 1 X1 XXXXXXXX HL*RPT 7;//Shift H signal start
                     *1 01 00 1 X1 HHHHHHHH HH*;
                     *1 01 00 1 X0 LHHHHHHH LH*;//Shift L one signal

                     *1 01 00 1 X1 XXXXXXXX HH*RPT 6;//Shift H continue

                     *1 01 00 1 X1 HHHHHHHL HL*;
                     *1 01 00 1 X1 HHHHHHHH HH*;
                     *1 01 00 1 X0 XXXXXXXX LH*RPT 7;//Shift L signal start

                     *1 01 00 1 X0 LLLLLLLL LL*;
                     *1 01 00 1 X1 HLLLLLLL HL*;//Shift H one signal
                     *1 01 00 1 X0 XXXXXXXX LL*RPT 6;//Shift L continue
                     *1 01 00 1 X0 LLLLLLLH LH*;
                     *1 01 00 1 X0 LLLLLLLL LL*;

                     *0 0X 00 X XX LLLLLLLL LL*;//-------clear data using clear2
                     *1 10 00 1 1X XXXXXXXX LH*RPT 7;//Shift H signal start
                     *1 10 00 1 1X HHHHHHHH HH*;
                     *1 10 00 1 0X HHHHHHHL HL*;//Shift L one signal

                     *1 10 00 1 1X XXXXXXXX HH*RPT 6;//Shift H continue

                     *1 10 00 1 1X LHHHHHHH LH*;
                     *1 10 00 1 1X HHHHHHHH HH*;
                     *1 10 00 1 0X XXXXXXXX HL*RPT 7;//Shift L signal start

                     *1 10 00 1 0X LLLLLLLL LL*;
                     *1 10 00 1 1X LLLLLLLH LH*;//Shift H one signal
                     *1 10 00 1 0X XXXXXXXX LL*RPT 6;//Shift L continue
                     *1 10 00 1 0X HLLLLLLL HL*;
  rpt_sp::          *1 10 00 1 0X LLLLLLLL LL*;

}
```

SETS count_value: count value (24 bits counter)
Use with "JSR" and RET
Refer to figure A

JSR label: This micro-instruction executes an unconditional jump to the address specified in the data field (label) of the micro-instruction, which is the first address of a subroutine.
label: This represent an address. It can be specified as an absolute integer address, as a label (a name is associated with another pattern address), as a "." (meaning this vector's address), and as label + integer, label – integer, "." + integer, "." – integer.

Note:
System provide another way to replace "label"
by ".- number " and ".+ number".
".- number" :back number
".+number" :Go forward number
Ex: JSR .+3(from here go forward three lines)

Use with "SETS" and "RET"
Refer to figure A

RET label : Returning from the subroutine to the label

Note:
System provide a another way to replace "label"
by ".- number " and ".+ number".
".- number" :back number
".+number" :Go forward number
Ex: RET .-3(from here back three lines)

Use with "SETS" and "RET"
Refer to figure A

<Figure A>

```
SPM_PATTERN (jsr_pat) {//JSR
                        //SETS from 2 to 16777215

                    /*C SS OO C SS ABCDEFGH OO*/
                    /*L EE EE L LR QQQQQQQQ AH*/
                    /*R LL 12 K      ABCDEFGH     */
                    /*   10                      */

jsr_st:            *0 X0 00 X XX LLLLLLLL LL*TS1,SETS 2;
                     //-------clear data using clear1
                     *1 01 00 1 X1 HLLLLLLL HL*;//Shift H signal start
                     *1 01 00 1 X1 HHLLLLLL HL*;
                     *1 01 00 1 X1 HHHLLLLL HL*;
                     *1 01 00 1 X1 HHHHLLLL HL*;
                     *1 01 00 1 X1 HHHHHLLL HL*;
                     *1 01 00 1 X1 HHHHHHLL HL*;
                     *1 01 00 1 X1 HHHHHHHL HL*;
                     *1 01 00 1 X1 HHHHHHHH HH*;

                     *1 01 00 1 X0 LHHHHHHH LH*;//Shift L one signal

                     *1 01 00 1 X1 HLHHHHHH HH*;//Shift H continue

                     *1 01 00 1 X1 HHLHHHHH HH*;
                     *1 01 00 1 X1 HHHLHHHH HH*;
                     *1 01 00 1 X1 HHHHLHHH HH*;
                     *1 01 00 1 X1 HHHHHLHH HH*;
                     *1 01 00 1 X1 HHHHHHLH HH*;
                     *1 01 00 1 X1 HHHHHHHL HL*;
                     *1 01 00 1 X1 HHHHHHHH HH*JSR sr_pat;


                     *1 01 00 1 X0 LHHHHHHH LH*;//Shift L signal start

                     *1 01 00 1 X0 LLHHHHHH LH*;
                     *1 01 00 1 X0 LLLHHHHH LH*;
                     *1 01 00 1 X0 LLLLHHHH LH*;
                     *1 01 00 1 X0 LLLLLHHH LH*;
                     *1 01 00 1 X0 LLLLLLHH LH*;
                     *1 01 00 1 X0 LLLLLLLH LH*;
                     *1 01 00 1 X0 LLLLLLLL LL*;

                     *1 01 00 1 X1 HLLLLLLL HL*;//Shift H one signal
                     *1 01 00 1 X0 LHLLLLLL LL*;//Shift L continue
                     *1 01 00 1 X0 LLHLLLLL LL*;
                     *1 01 00 1 X0 LLLHLLLL LL*;
                     *1 01 00 1 X0 LLLLHLLL LL*;
                     *1 01 00 1 X0 LLLLLHLL LL*;
                     *1 01 00 1 X0 LLLLLLHL LL*;
                     *1 01 00 1 X0 LLLLLLLH LH*;
jsr_sp:            *1 01 00 1 X0 LLLLLLLL LL*;

sr_pat:            *0 X0 00 X XX LLLLLLLL LL*;//Clear
```

Set JSR count=2

JUMP to Label “sr_pat”

```
                    *0 X0 00 X XX LLLLLLLL LL*;
                    *1 11 XX 1 XX 11111111 11*;//-----load

                    *1 XX 00 0 XX HHHHHHHH HH*RET sr_pat;//-----hold

}
SPM_PATTERN (jsr_pat) {//JSR
                        //SETS from 2 to 16777215

                    /*C SS OO C SS ABCDEFGH OO*/
                    /*L EE EE L LR QQQQQQQQ AH*/
                    /*R LL 12 K      ABCDEFGH     */
                    /*   10                       */

  jsr_st:           *0 X0 00 X XX LLLLLLLL LL*TS1,SETS 2;
                     //-------clear data using clear1
                     *1 01 00 1 X1 HLLLLLLL HL*;//Shift H signal start
                     *1 01 00 1 X1 HHLLLLLL HL*;
                     *1 01 00 1 X1 HHHLLLLL HL*;
                     *1 01 00 1 X1 HHHHLLLL HL*;
                     *1 01 00 1 X1 HHHHHLLL HL*;
                     *1 01 00 1 X1 HHHHHHLL HL*;
                     *1 01 00 1 X1 HHHHHHHL HL*;
                     *1 01 00 1 X1 HHHHHHHH HH*;

                     *1 01 00 1 X0 LHHHHHHH LH*;//Shift L one signal

                     *1 01 00 1 X1 HLHHHHHH HH*;//Shift H continue

                     *1 01 00 1 X1 HHLHHHHH HH*;
                     *1 01 00 1 X1 HHHLHHHH HH*;
                     *1 01 00 1 X1 HHHHLHHH HH*;
                     *1 01 00 1 X1 HHHHHLHH HH*;
                     *1 01 00 1 X1 HHHHHHLH HH*;
                     *1 01 00 1 X1 HHHHHHHL HL*;
                     *1 01 00 1 X1 HHHHHHHH HH*JSR .+18;

                     *1 01 00 1 X0 LHHHHHHH LH*;//Shift L signal start

                     *1 01 00 1 X0 LLHHHHHH LH*;
                     *1 01 00 1 X0 LLLHHHHH LH*;
                     *1 01 00 1 X0 LLLLHHHH LH*;
                     *1 01 00 1 X0 LLLLLHHH LH*;
                     *1 01 00 1 X0 LLLLLLHH LH*;
                     *1 01 00 1 X0 LLLLLLLH LH*;
                     *1 01 00 1 X0 LLLLLLLL LL*;

                     *1 01 00 1 X1 HLLLLLLL HL*;//Shift H one signal
                     *1 01 00 1 X0 LHLLLLLL LL*;//Shift L continue
                     *1 01 00 1 X0 LLHLLLLL LL*;
                     *1 01 00 1 X0 LLLHLLLL LL*;
                     *1 01 00 1 X0 LLLLHLLL LL*;
                     *1 01 00 1 X0 LLLLLHLL LL*;
                     *1 01 00 1 X0 LLLLLLHL LL*;
```

Back to label “sr_pat”
After loop 2 times, go back to “AA”

```
                    *1 01 00 1 X0 LLLLLLLH LH*;
jsr_sp:            *1 01 00 1 X0 LLLLLLLL LL*;

sr_pat:            *0 X0 00 X XX LLLLLLLL LL*;//Clear
                    *0 X0 00 X XX LLLLLLLL LL*;
                    *1 11 XX 1 XX 11111111 11*;//-----load

                    *1 XX 00 0 XX HHHHHHHH HH*RET .-3; //-----hold

}
```

SETXi count_value, i = 0~3: this instruction set a "count_value" in the corresponding index constant register (0 to 3)
count_value: 2 <= N <= 16777215 (24bit register)

JNZi label , i = 0~3: The JNZi instruction is a conditional jump instruction.
If corresponding index count register is not equal to zero, then jump to "label".
Note:
System provide another way to replace "label"
by ".- number " and ".+ number".
".- number" :back number
".+number" :Go forward number
Ex: JNZ .-3(from here back three lines)

[Notice]: JNZ instruction limitation note

```
SPM_PATTERN (jnz_pat) {
                    /*C SS OO C SS ABCDEFGH OO*/
                    /*L EE EE L LR QQQQQQQQ AH*/
                    /*R LL 12 K      ABCDEFGH     */
                    /*   10                      */
                     *0 X0 00 X XX LLLLLLLL LL*TS1;
                     *0 X0 00 X XX LLLLLLLL LL*SETX1 5;

  //CLEAR Mode Check
  jnz_1:            *1 11 XX 1 XX 10101010 10*;//-----load
                     *1 00 00 X XX HLHLHLHL HL*;//-----hold
                     *1 11 XX 1 XX 01010101 01*;//-----load

                     *1 00 00 X XX LHLHLHLH LH*JNZ1 jnz_1;//-----hold
                     *0 X0 00 X XX LLLLLLLL LL*;
                     *0 X0 00 X XX LLLLLLLL LL*SETX2 2;
  //Hold   register mode check
  jnz_2:            *1 11 XX 1 XX 10101010 10*;//-----load
                     *1 00 00 X XX HLHLHLHL HL*SETX3 4;//-----hold

  jnz_3:            *1 11 XX 1 XX 11111111 11*;//-----load
                     *1 XX 00 0 XX HHHHHHHH HH*;//-----hold
                     *1 11 XX 1 XX 00000000 00*;//-----load

                     *1 00 00 X XX LLLLLLLL LL*JNZ3 jnz_3;//-----hold
                     *1 11 XX 1 XX 01010101 01*;//-----load
```

SET JNZ count=5

if ((count-1) !=0) back to “jnz_1”

```
                        *1 XX 00 0 XX LHLHLHLH LH*JNZ2 jnz_2;//-----hold
                        *0 X0 00 X XX LLLLLLLL LL*;

}
```

```
SPM_PATTERN (jnz_pat) {
                        /*C SS OO C SS ABCDEFGH OO*/
                        /*L EE EE L LR QQQQQQQQ AH*/
                        /*R LL 12 K       ABCDEFGH     */
                        /*    10                          */
                         *0 X0 00 X XX LLLLLLLL LL*TS1;
                         *0 X0 00 X XX LLLLLLLL LL*SETX1 5;

  //CLEAR Mode Check
  jnz_1:                *1 11 XX 1 XX 10101010 10*;//-----load
                         *1 00 00 X XX HLHLHLHL HL*;//-----hold
                         *1 11 XX 1 XX 01010101 01*;//-----load

                         *1 00 00 X XX LHLHLHLH LH*JNZ1 .-3;//-----hold
                         *0 X0 00 X XX LLLLLLLL LL*;
                         *0 X0 00 X XX LLLLLLLL LL*SETX2 2;
  //Hold    register mode check
  jnz_2:                *1 11 XX 1 XX 10101010 10*;//-----load
                         *1 00 00 X XX HLHLHLHL HL*SETX3 4;//-----hold

  jnz_3:                *1 11 XX 1 XX 11111111 11*;//-----load
                         *1 XX 00 0 XX HHHHHHHH HH*;//-----hold
                         *1 11 XX 1 XX 00000000 00*;//-----load

                         *1 00 00 X XX LLLLLLLL LL*JNZ3 jnz_3;//-----hold
                         *1 11 XX 1 XX 01010101 01*;//-----load

                         *1 XX 00 0 XX LHLHLHLH LH*JNZ2 jnz_2;//-----hold
                         *0 X0 00 X XX LLLLLLLL LL*;

}
```

JMP label: unconditional jump

Note:

System provide a extra way to replace "label" by ".- number " and ".+ number".

".- number" :back number

".+number" :Go forward number

Ex: JMP .+5(from here go forward five lines)

```
SPM_PATTERN (jmp_pat) {
                    /*C SS OO C SS ABCDEFGH OO*/
                    /*L EE EE L LR QQQQQQQQ AH*/
                    /*R LL 12 K      ABCDEFGH     */
                    /*   10                      */
  clr_1:            *0 X0 00 X XX LLLLLLLL LL*TS1;//-----Clear1
                     *1 11 XX 1 XX 10101010 10*;//-----load

                     *1 00 00 X XX HLHLHLHL HL*;//-----hold

                     *0 0X 00 X XX LLLLLLLL LL*JMP jmp1;//-----Clear2

  clr_2:            *0 11 XX X XX XXXXXXXX LL*;//-----Clear3

                     *1 11 XX 1 XX 10101010 10*;//-----load

                     *1 00 00 X XX HLHLHLHL HL*;//-----hold

                     *0 11 XX X XX XXXXXXXX LL*JMP jmp2;//-----Clear3

  //Hold and Load and Shift mode check
  jmp1:             *0 11 XX X XX XXXXXXXX LL*;//-----Clear register using Clear3

                     *1 11 XX 1 XX 10000000 XX*;//Load signal to register

                     *1 00 00 X XX HLLLLLLL HL*;//hold register
                     *1 01 00 1 X0 LHLLLLLL LL*;//Shift L to right
                     *1 01 00 1 X0 LLHLLLLL LL*;
                     *1 01 00 1 X0 LLLHLLLL LL*;
                     *1 01 00 1 X0 LLLLHLLL LL*;
                     *1 01 00 1 X0 LLLLLHLL LL*;
                     *1 01 00 1 X0 LLLLLLHL LL*;
                     *1 01 00 1 X0 LLLLLLLH LH*JMP clr_2;

  jmp2:              *1 11 XX 1 XX 01010101 XX*;//Load signal to register

                     *1 00 00 X XX LHLHLHLH LH*;//hold register
                     *1 10 00 1 1X HLHLHLHH HH*;//Shift H to left

                     *1 10 00 1 1X LHLHLHHH LH*;
                     *1 10 00 1 1X HLHLHHHH HH*;
                     *1 10 00 1 1X LHLHHHHH LH*;
                     *1 10 00 1 1X HLHHHHHH HH*;
                     *1 10 00 1 1X LHHHHHHH LH*;
                     *1 10 00 1 1X HHHHHHHH HH*;
                     *0 X0 00 X XX LLLLLLLL LL*;//-----Clear using Clear1

}
```

```
SPM_PATTERN (jmp_pat) {
                        /*C SS OO C SS ABCDEFGH OO*/
                        /*L EE EE L LR QQQQQQQQ AH*/
                        /*R LL 12 K       ABCDEFGH     */
                        /*    10                          */
   clr_1:               *0 X0 00 X XX LLLLLLLL LL*TS1;//-----Clear1
                         *1 11 XX 1 XX 10101010 10*;//-----load

                         *1 00 00 X XX HLHLHLHL HL*;//-----hold
                         *0 0X 00 X XX LLLLLLLL LL*JMP .+5;//-----Clear2
   clr_2:               *0 11 XX X XX XXXXXXXX LL*;//-----Clear3
                         *1 11 XX 1 XX 10101010 10*;//-----load

                         *1 00 00 X XX HLHLHLHL HL*;//-----hold
                         *0 11 XX X XX XXXXXXXX LL*JMP .+11;//-----Clear3
   //Hold and Load and Shift mode check
   jmp1:                 *0 11 XX X XX XXXXXXXX LL*;//-----Clear register using Clear3

                         *1 11 XX 1 XX 10000000 XX*;//Load signal to register

                         *1 00 00 X XX HLLLLLLL HL*;//hold register
                         *1 01 00 1 X0 LHLLLLLL LL*;//Shift L to right
                         *1 01 00 1 X0 LLHLLLLL LL*;
                         *1 01 00 1 X0 LLLHLLLL LL*;
                         *1 01 00 1 X0 LLLLHLLL LL*;
                         *1 01 00 1 X0 LLLLLHLL LL*;
                         *1 01 00 1 X0 LLLLLLHL LL*;
                         *1 01 00 1 X0 LLLLLLLH LH*JMP .-13;

   jmp2:                 *1 11 XX 1 XX 01010101 XX*;//Load signal to register

                         *1 00 00 X XX LHLHLHLH LH*;//hold register
                         *1 10 00 1 1X HLHLHLHH HH*;//Shift H to left
                         *1 10 00 1 1X LHLHLHHH LH*;
                         *1 10 00 1 1X HLHLHHHH HH*;
                         *1 10 00 1 1X LHLHHHHH LH*;
                         *1 10 00 1 1X HLHHHHHH HH*;
                         *1 10 00 1 1X LHHHHHHH LH*;
                         *1 10 00 1 1X HHHHHHHH HH*;
                         *0 X0 00 X XX LLLLLLLL LL*;//-----Clear using Clear1
}
```

```
SPM_PATTERN (jmp_pat) {
                    /*C SS OO C SS ABCDEFGH OO*/
                    /*L EE EE L LR QQQQQQQQ AH*/
                    /*R LL 12 K      ABCDEFGH     */
                    /*   10                      */
  clr_1:            *0 X0 00 X XX LLLLLLLL LL*TS1;//-----Clear1
                     *1 11 XX 1 XX 10101010 10*;//-----load

                     *1 00 00 X XX HLHLHLHL HL*;//-----hold

                     *0 0X 00 X XX LLLLLLLL LL*JMP .+5;//-----Clear2

  clr_2:            *0 11 XX X XX XXXXXXXX LL*;//-----Clear3
                     *1 11 XX 1 XX 10101010 10*;//-----load
                     *1 00 00 X XX HLHLHLHL HL*;//-----hold
                     *0 11 XX X XX XXXXXXXX LL*JMP .+11;//-----Clear3
                  //Hold and Load and Shift mode check
  jmp1:              *0 11 XX X XX XXXXXXXX LL*;//-----Clear register using Clear3

                     *1 11 XX 1 XX 10000000 XX*;//Load signal to register

                     *1 00 00 X XX HLLLLLLL HL*;//hold register
                     *1 01 00 1 X0 LHLLLLLL LL*;//Shift L to right
                     *1 01 00 1 X0 LLHLLLLL LL*;
                     *1 01 00 1 X0 LLLHLLLL LL*;
                     *1 01 00 1 X0 LLLLHLLL LL*;
                     *1 01 00 1 X0 LLLLLHLL LL*;
                     *1 01 00 1 X0 LLLLLLHL LL*;
                     *1 01 00 1 X0 LLLLLLLH LH*JMP .-13;

  jmp2:              *1 11 XX 1 XX 01010101 XX*;//Load signal to register

                     *1 00 00 X XX LHLHLHLH LH*;//hold register
                     *1 10 00 1 1X HLHLHLHH HH*;//Shift H to left

                     *1 10 00 1 1X LHLHLHHH LH*;
                     *1 10 00 1 1X HLHLHHHH HH*;
                     *1 10 00 1 1X LHLHHHHH LH*;
                     *1 10 00 1 1X HLHHHHHH HH*;
                     *1 10 00 1 1X LHHHHHHH LH*;
                     *1 10 00 1 1X HHHHHHHH HH*;
                     *0 X0 00 X XX LLLLLLLL LL*;//-----Clear using Clear1

}
```

TFTM: This instruction sends a trigger pulse to Frequency/Timing counter to initiate frequency/timing measurement.
Use with statement "JUDGE_FREQ/JUDGE_TIME"

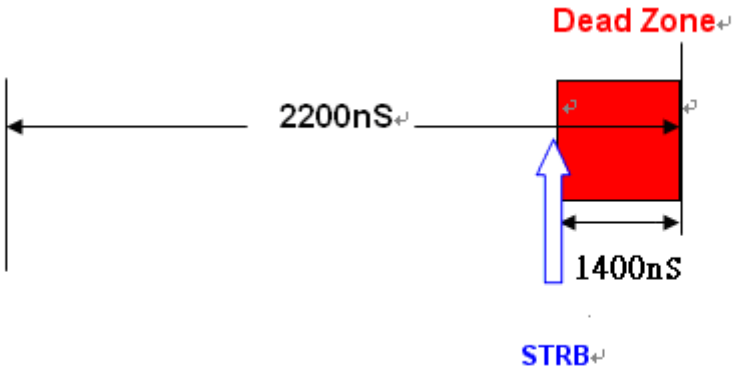
```
m_ft_pat_st:    *1 11 XX 1 XX 10101010 10*SETX0 1024;//-----load

                 *1 00 00 X XX HLHLHLHL HL*;//-----hold

                 *1 00 00 X XX HLHLHLHL HL*;//-----hold

                 *1 01 00 1 X0 LHLHLHLH LX*;//Shift L one signal

                 *1 01 00 1 X1 HLHLHLHL HX*;//Shift H signal start
                 *1 01 00 1 X0 LHLHLHLH LX*;//Shift L one signal

                 *1 01 00 1 X1 HLHLHLHL HX*TFTM;//Shift H signal start
                 *1 01 00 1 X0 LHLHLHLH LX*;//Shift L one signal

                 *1 01 00 1 X1 HLHLHLHL HX*;//Shift H signal start
                 *1 01 00 1 X0 LHLHLHLH LX*;//Shift L one signal

                 *1 01 00 1 X1 HLHLHLHL HX*;//Shift H signal start
jnz_label:      *1 01 00 1 X0 LHLHLHLH LX*;//Shift L one signal

                 *1 01 00 1 X1 HLHLHLHL HX*;//Shift H signal start
                 *1 01 00 1 X0 LHLHLHLH LX*;//Shift L one signal

                 *1 01 00 1 X1 HLHLHLHL HX*JNZ0 jnz_label;//Shift H signal
start
                 *1 01 00 1 X0 LHLHLHLH LX*;//Shift L one signal

m_ft_pat_sp:    *1 01 00 1 X1 HLHLHLHL HX*;//Shift H signal start
```

Pattern trig to measure frequency

SETM: match_count: set match count to index match register
match_count: 2 <= N <= 1677215 (24bit register)

IMATCH: This instruction executes a single vector to match. It will repeat the vector until a match occurs or the count specified in the data field of the

1. Set the match expire count with u-instruction "SETM".
2. Insert 30 dummy cycles (disable all clock pins) to clear every stage of the pipe line.
3. Make sure there are at least 1400nS between STRB and END of cycle.
   The dead zone : CRAFT 3380 → 1400nS (Maximum Site No. : 32)
4. Only support normal mode pattern.
5. IMATCH must follow internal clock of ATE only that means can't use external clock of chip, R/C, etc.




```
SPM_PATTERN (match_pat) {
                    /*C SS OO C SS ABCDEFGH OO*/
                    /*L EE EE L LR QQQQQQQQ AH*/
                    /*R LL 12 K      ABCDEFGH     */
                    /*   10                      */

                     *0 X0 00 X XX LLLLLLLL LL*TS1;//-------clear data using clear1
                     *0 X0 00 X XX LLLLLLLL LL*SETM 20;//-------clear data using
clear1
                     *0 X0 00 X XX XXXXXXXX XX*RPT 30;//-------clear data using
clear1
                     *1 01 00 1 X1 XXXXXXXX XH*TS2,IMATCH;

                     *1 01 00 1 X0 LHHHHHHH LH*TS1;//Shift L one signal
                     *1 01 00 0 X0 XXXXXXXX XX*RPT 30;//Shift L one signal
                     *1 01 00 1 X1 XXXXXXXX XL*TS2,IMATCH;//Shift H continue
                     *1 01 00 1 X1 HHHHHHHH HH*TS1;

                     *1 01 00 0 X1 XXXXXXXX XX*RPT 30;

                     *1 01 00 1 X0 XXXXXXXX XL*TS2,IMATCH;//Shift L signal
start

                     *1 01 00 1 X1 HLLLLLLL HL*TS1;//Shift H one signal
                     *1 01 00 0 X1 XXXXXXXX XX*RPT 30;
                     *1 01 00 1 X0 XXXXXXXX XH*TS2,IMATCH;

                     *1 01 00 1 X0 LLLLLLLL LL*TS1;
                     *0 X0 00 X XX LLLLLLLL LL*TS1;//-------clear data using clear1
                     *0 X0 00 X XX LLLLLLLL LL*;//-------clear data using clear1
}
```

Set match count

Before executing match instruction, need 30 dummy cycles

JCONT label: this instruction passes control to jump destination address at the end of execution of the vector, and transfers control to test program. To release this continuous loop, execute the "PAT_CONTINUE()" statement in the test plan program. To provide user use the instruction to measure "dynamic current".

Note:

System provide a extra way to replace "label" by ".- number " and ".+ number".

".- number" :back number

".+number" :Go forward number

Ex: JCONT .-3(from here back three lines)

```
SPM_PATTERN (jcont1_pat) {
                    /*C SS OO C SS ABCDEFGH OO*/
                    /*L EE EE L LR QQQQQQQQ AH*/
                    /*R LL 12 K      ABCDEFGH     */
                    /*   10                       */

  //Shift Right mode check
  jcont1_st:        *0 X0 00 X XX LLLLLLLL LL*TS1;//-------clear data using clear1
                     *1 01 00 1 X1 HLLLLLLL HL*;//Shift H signal start
                     *1 01 00 1 X1 HHLLLLLL HL*;
                     *1 01 00 1 X1 HHHLLLLL HL*;
                     *1 01 00 1 X1 HHHHLLLL HL*;
                     *1 01 00 1 X1 HHHHHLLL HL*;
                     *1 01 00 1 X1 HHHHHHLL HL*;
                     *1 01 00 1 X1 HHHHHHHL HL*;
                     *1 01 00 1 X1 HHHHHHHH HH*;
                     *1 01 00 1 X0 LHHHHHHH LH*;//Shift L one signal
                     *1 01 00 1 X1 HLHHHHHH HH*;//Shift H continue
                     *1 01 00 1 X1 HHLHHHHH HH*;
                     *1 01 00 1 X1 HHHLHHHH HH*;
                     *1 01 00 1 X1 HHHHLHHH HH*;
                     *1 01 00 1 X1 HHHHHLHH HH*;
                     *1 01 00 1 X1 HHHHHHLH HH*;
                     *1 01 00 1 X1 HHHHHHHL HL*;
                     *1 01 00 1 X1 HHHHHHHH HH*;
                     *1 01 00 1 X0 LHHHHHHH LH*;//Shift L signal start
                     *1 01 00 1 X0 LLHHHHHH LH*;
                     *1 01 00 1 X0 LLLHHHHH LH*;
                     *1 01 00 1 X0 LLLLHHHH LH*;
                     *1 01 00 1 X0 LLLLLHHH LH*;
                     *1 01 00 1 X0 LLLLLLHH LH*;
                     *1 01 00 1 X0 LLLLLLLH LH*;
                     *1 01 00 1 X0 LLLLLLLL LL*;
                     *1 01 00 1 X1 HLLLLLLL HL*;//Shift H one signal
                     *1 01 00 1 X0 LHLLLLLL LL*;//Shift L continue
                     *1 01 00 1 X0 LLHLLLLL LL*;
                     *1 01 00 1 X0 LLLHLLLL LL*;
                     *1 01 00 1 X0 LLLLHLLL LL*;
                     *1 01 00 1 X0 LLLLLHLL LL*;
                     *1 01 00 1 X0 LLLLLLHL LL*;
                     *1 01 00 1 X0 LLLLLLLH LH*;
                     *1 01 00 1 X0 LLLLLLLL LL*JCONT jcont1_st;
                     *0 X0 00 X XX LLLLLLLL LL*;
}
```

Return to label "jcont1_st" and do loop, until execute PAT_CONTINUE()" statement in the test plan program.pattern_con

< TEST PLAN >

```
//u_in : JCONT
DATALOG_MSG="IDD Dynamic Test and u_in : JCONT";
TEST_NO(50);
JUDGE_PAT(jcont1_pat, __jcont1_pat);//JCONT

DATALOG_MSG="Judge I mode";
TEST_NO(51);
JUDGE_I_DPS ( Vddd, idd.min, idd.max, 2mS);
JUDGE_I_PREF(__Vdd, idd.min, idd.max, 2mS);
PAT_CONTINUE();
```

UISYNC: From u-instruction send trigger signal to test head.

```
SPM_PATTERN (rpt_pat) {//RPT from 2 to 16777215

                    /*C SS OO C SS ABCDEFGH OO*/
                    /*L EE EE L LR QQQQQQQQ AH*/
                    /*R LL 12 K      ABCDEFGH     */
                    /*   10                      */

  //Shift Right mode check
  rpt_st::          *0 X0 00 X XX LLLLLLLL LL*TS1;//-------clear data using clear1

                     *1 01 00 1 X1 XXXXXXXX HL*RPT 7;//Shift H signal start
                     *1 01 00 1 X1 HHHHHHHH HH*;

                     *1 01 00 1 X0 LHHHHHHH LH*;//Shift L one signal

                     *1 01 00 1 X1 XXXXXXXX HH*RPT 6;//Shift H continue

                     *1 01 00 1 X1 HHHHHHHL HL*;
                     *1 01 00 1 X1 HHHHHHHH HH* UISYNC;


                     *1 01 00 1 X0 XXXXXXXX LH*RPT 7;//Shift L signal start

                     *1 01 00 1 X0 LLLLLLLL LL*;

                     *1 01 00 1 X1 HLLLLLLL HL*;//Shift H one signal
                     *1 01 00 1 X0 XXXXXXXX LL*RPT 6;//Shift L continue
                     *1 01 00 1 X0 LLLLLLLH LH*;
  rpt_sp::          *1 01 00 1 X0 LLLLLLLL LL*;
  }
```

Tester will send a trigger signal during the cycle to trig “Oscilloscope”

LDXN: Load X0 ~ X7 values to register (0~7) for RPTN used.
X0 ~ X7 values are stetted in statement "SET_UPAT_REG".

```
sw_test()
{
      RATEN(TS2, raten);
      MARKERN(TS2, CLK,dt0,dt1, ,0nS,0nS,ct0,ct1);
      MARKERN(TS2, CTRL,0nS,0nS, ,0nS,0nS,ct0,ct1);
      MARKERN(TS2, OUTPUTS,0nS,0nS, ,0nS,0nS,ct0,ct1);
      PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, NORM);
      PIN_MODE(CLK, RZ, EDGE, MASK, IO_NRZ, OFF);
      SET_LEVELN(func_lvl, IO_ALLPINS, 0V, 5V, 0.5V, 2.4V, 150uA, -150uA,
2.5V, , );
      LEVELS(func_lvl, 2mS);
      RELAY_ON(IO_ALLPINS, PDCL, 1mS);

       SET_UPAT_REG( X0, 2 );
       SET_UPAT_REG( X1, 3 );
       SET_UPAT_REG( X2, 4 );
       SET_UPAT_REG( X3, 5 );
       SET_UPAT_REG( X4, 6 );
       SET_UPAT_REG( X5, 7 );
       SET_UPAT_REG( X6, 8 );
       SET_UPAT_REG( X7, 9 );

       JUDGE_PAT(SET_UPAT,__ SET_UPAT)
}
```

```
SPM_PATTERN (SET_UPAT)
{
Start :     * 001000*TS1;
            * 001000*;
            * 001000*;
            * 001000*;
            * 001000*LDXN    X0;
            * 001000*;
            * 001000*;
            * 001000*;
            * 001000*;
            * 000000*RPTN    X0;
            * 001000*;
            * 001000*;
            * 001000*;
            * 001000*LDXN    X1;
            * 001000*;
            * 001000*;
            * 000000*RPTN    X1;
            * 001000*;
}
```

RPTN: Vector repeat

Use with Micro instruction "LDXN" and statement "SET_UPAT_REG"

< TEST PLAN >

```
sw_test()
{
      RATEN(TS2, raten);
      MARKERN(TS2, CLK,dt0,dt1, ,0nS,0nS,ct0,ct1);
      MARKERN(TS2, CTRL,0nS,0nS, ,0nS,0nS,ct0,ct1);
      MARKERN(TS2, OUTPUTS,0nS,0nS, ,0nS,0nS,ct0,ct1);
      PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, NORM);
      PIN_MODE(CLK, RZ, EDGE, MASK, IO_NRZ, OFF);
      SET_LEVELN(func_lvl, IO_ALLPINS, 0V, 5V, 0.5V, 2.4V, 150uA, -150uA,
2.5V, , );
      LEVELS(func_lvl, 2mS);
      RELAY_ON(IO_ALLPINS, PDCL, 1mS);

      SET_UPAT_REG( X0, 2 );
      SET_UPAT_REG( X1, 3 );
      SET_UPAT_REG( X2, 4 );
      SET_UPAT_REG( X3, 5 );
      SET_UPAT_REG( X4, 6 );
      SET_UPAT_REG( X5, 7 );
      SET_UPAT_REG( X6, 8 );
      SET_UPAT_REG( X7, 9 );

      JUDGE_PAT(SET_UPAT,__ SET_UPAT)
}
```

Set X0-X7 count

< Pattern >

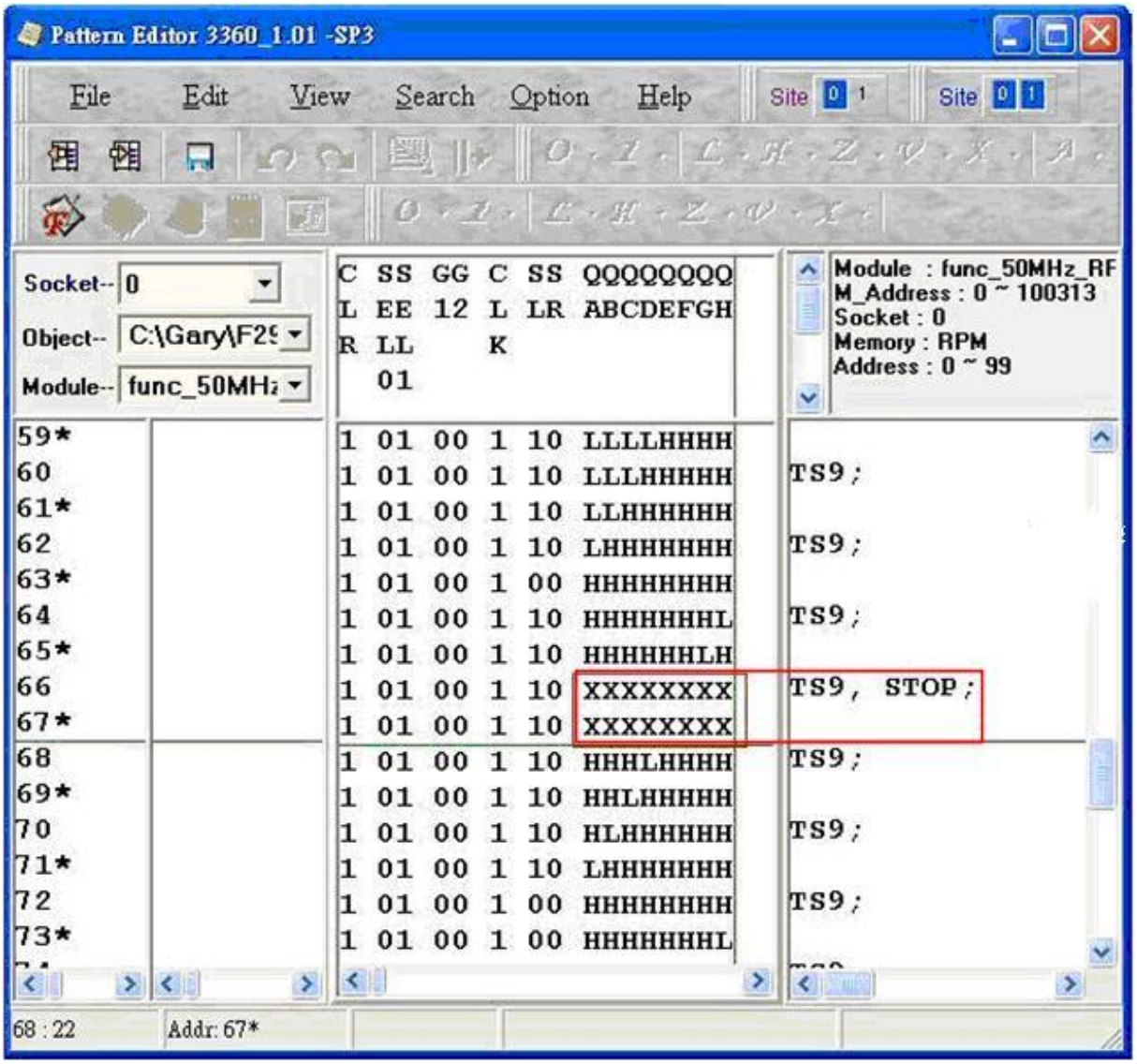
```
SPM_PATTERN (SET_UPAT)
{
Start :     * 001000*TS1;
            * 001000*;
            * 001000*;
            * 001000*;
            * 001000*LDXN    X0;
            * 001000*;
            * 001000*;
            * 001000*;
            * 001000*;
            * 000000*RPTN    X0;
            * 001000*;
            * 001000*;
            * 001000*;
            * 001000*LDXN    X1;
            * 001000*;
            * 001000*;
            * 000000*RPTN    X1;
            * 001000*;
}
```

Load X0 value

NOP: no operation

STOP: Stop pattern execution.

[Notice]: when using "DBL Mode" and pattern rate less than "40nS",
Stop vector and next vector must be "Don't care (X) ".
Except Stop vector is last vector.



4. Plan Program

4.1 Overview

In a plan program, the users write the program with test functions. The plan program sets the test condition, executes DC parametric tests and function tests, and controls the overall test sequence.

4.1.1 Format

Parameter list:

SET_DEC_FILE	block
HW_BIN_DEF	block
SW_BIN_DEF	block
START_UP(){ … }	block
TEST_PRO {….}	block
TEST_ITEM1(){….}	block
RELEASE	block

Parameter comment:

SET_DEC_FILE : The block indicates the declaration file.

HW_BIN_DEF: The block indicates the associated user given names with the actual hardware bins.

SW_BIN_DEF: The block indicates the associated user given names with the actual software bins.

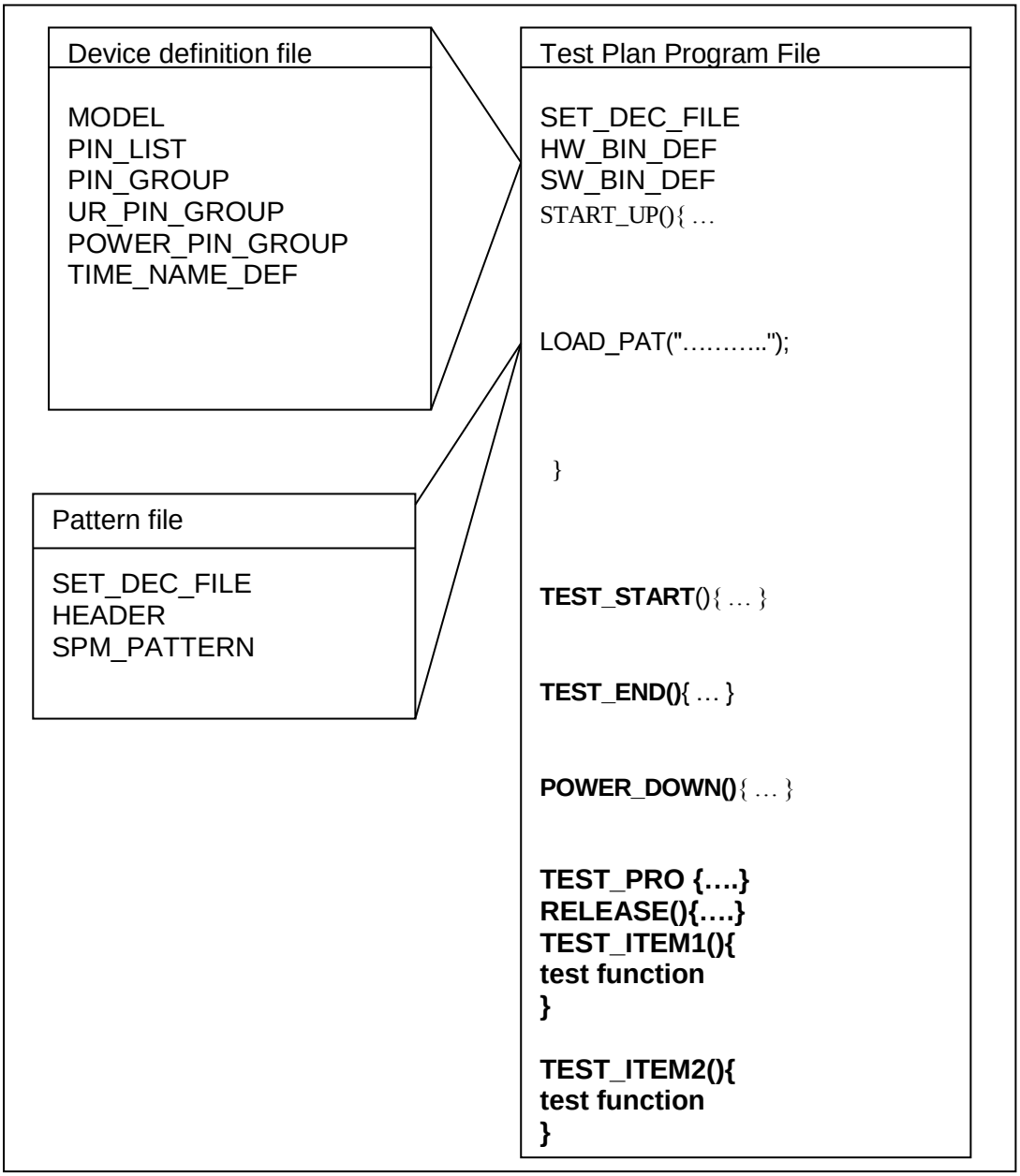
START_UP(){ … }: The block is a special function. It being called automatically when the first time the test is done, or upon a menu request. Any valid statements can be used in this function.

RELEASE(){ … } : The block is a special function. It being called automatically at the program release time. Any valid statements can be used in this function. This function follows by a default power down function.

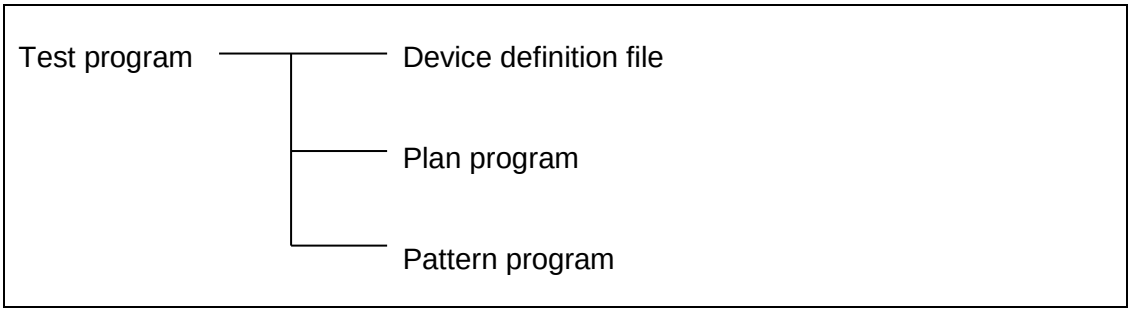
TEST_PRO {….} : The block is a declaration and the actual body of the test program (It replaces main in the common C program). It defines the test sequence as well as the bin strategy (both hardware and software bins).

TEST_ITEM1(){….}: The block is actual test program.
[Notice]: **TEST_ITEM name must be less than 64 characters.**

4.1.2 Explanation



Test program must be written for each device and for each test method. It is necessary to prepare the following sets of program for each device.



In the general case, the test program includes three parts. One is device definition file, the second is pattern file and the last is plan program. The device definition file can refer the Chapter 2. The pattern file can refer the Chapter 3. And the plan program can refer this chapter.

Test Program for single or Multi-DUT Tests
A "Single DUT Test' is for testing one DUT at a time, and a "Multi-DUT Test" is for testing multiple DUTs at a time. Whether a test program is "Single" or "Multi" defined in the device definition file by the "Multi_DUT Definition". The "Multi-DUT Definition"defines number of DUTs; channel and pin numbers allocated to each DUT. For more detailed information, please refer to chapter 2.

4.2 Basic Control Statement

4.2.1 SET_DEC_FILE

The statement declares the file name and its' directory.

4.2.1.1 Format

Parameter list:

SET_DEC_FILE **"**declaration file name**"**

Parameter comment:
declaration file name: The file which extension name is "dec". Alphanumeric and under scope (_) can be used.

4.2.1.2 Explanation

The statement declares the declaration file name and its' directory.

4.2.1.3 Example

```
SET_DEC_FILE    ".\PAT\FW_ls299_4sites_pin.dec"

GLOBAL    int    os_test_by_OST              = 1,
                 os_test_by_Pattern        = 0,
                 os_test_by_PMU              = 1,
                 os_test_by_PPMU             = 0;

   /**declaration for IO level set use**/
    int rst_lvl        = 1,
          os_lvl          = 2;
HW_BIN_DEF {
    all_pass                          =    1 ;
    fail_os_test                    = !2 ;
    }


SW_BIN_DEF {
    all_pass                          =    1 ;
    fail_os_test                    = !2 ;
    }
```

Include declaration file for Pin define

4.2.2 HW_BIN_DEF

This declaration associates the user given names with the actual hardware bins.

4.2.2.1 Format

Parameter list:

```
HW_BIN_DEF   {
      hardware_bin_name1 = [!]hardware_bin_num1;

       hardware_bin_name2 = [!]hardware_bin_num2;

       ..

       hardware_bin_namen = [!]hardware_bin_numn;

  }
```

Parameter comment:

hardware_bin_name: The name to be associated with the specified hardware_bin_num.
Maximum: an array of 32 characters.
C Type: string
No entry – illegal

hardware_bin_num: The actual bin value, which corresponds to the given hardware_bin_name.
The valid range of value is 1 to 255.
C Type: string
No entry – illegal

When user program"ALARM_ON", then must program a Alarm-Bin" **SYS_ALARM_HW_BIN**"
Hardware_bin_name Keyword : **SYS_ALARM_HW_BIN**

4.2.2.2 Explanation

- This declaration allows the test programmers and users to refer to these hardware bins by names. It is simple to attach an integer value to a name.
- The major difference is that these names are not only used by the test program, but also by the user, since they are displayed on the menu system as well as on output reports, such as the summary sheet.

4.2.2.3 Example

```
SET_DEC_FILE    ".\PAT\FW_ls299_4sites_pin.dec"

GLOBAL    int    os_test_by_OST              = 1,
                 os_test_by_Pattern        = 0,
                 os_test_by_PMU              = 1,
                 os_test_by_PPMU             = 0;

   /**declaration for IO level set use**/
    int rst_lvl        = 1,
          os_lvl          = 2;
HW_BIN_DEF {
    all_pass                          =    1 ;
    fail_os_test                      = !2 ;
    SYS_ALARM_HW_deBIN     =!255;
    }


SW_BIN_DEF {
    all_pass                          =    1 ;
    fail_os_test                      = !2 ;
    }
```

Hardware bin setting “1~255”

4.2.3 SW_BIN_DEF

This declaration associates the user given names with the actual software bins.

4.2.3.1 Format

Parameter list:

```
SW_BIN_DEF   {

        sw_bin_name1 = [!]sw_bin_number1;

        sw_bin_name2 = [!]sw_bin_number2;

        .       .

        sw_bin_namen = [!]sw_bin_numbern;

  }
```

Parameter comment:

software_bin_name: The name to be associated with the specified software_bin_num.
Maximum: an array of 32 characters.
C Type: string
No entry – illegal

software_bin_num: The actual bin value, which corresponds to the given hardware_bin_name.
The valid range of value is from 1 to 1023.
C Type: string
No entry – illegal

When user program"ALARM_ON", then must program a Alarm-Bin" **SYS_ALARM_SW_BIN**"
Software_bin_name Keyword : **SYS_ALARM_SW_BIN**

4.2.3.2 Explanation

- These hardware bins are limited in number by the physical constraints of the handing equipment. The addition of software bins, which are nothing more than user named counters, holding information about a much more detailed failures than it is possible with the hardware bins.
- This declaration allows the test programmers and users to refer to these hardware bins by names. It is simple to attach an integer value to a name.
- The major difference is that these names are not only used by the test program, but also by the user, since they are displayed on the menu system as well as on output reports such as the summary sheet.

4.2.3.3 Example

```
SET_DEC_FILE    ".\PAT\FW_ls299_4sites_pin.dec"

GLOBAL    int    os_test_by_OST              = 1,
                 os_test_by_Pattern        = 0,
                 os_test_by_PMU              = 1,
                 os_test_by_PPMU             = 0;

   /**declaration for IO level set use**/
    int rst_lvl        = 1,
          os_lvl          = 2;
HW_BIN_DEF {
    all_pass                          =    1 ;
    fail_os_test                    = !2 ;
    }

SW_BIN_DEF {
    all_pass                          =    1 ;
    fail_os_test                    = !2 ;
    SYS_ALARM_SW_BIN    = !1023;
    }
```

Software bin setting “1~1023”

4.2.4 START_UP( ) { … }

This is a special function. When the test is done at the first time, this function being called automatically. Any valid statements can be used in this function.

4.2.4.1 Format

Parameter list:

```
START_UP( ) {

    Statements

            }
```

Parameter comment:

Statements: Any valid TPG (Test Program Generator) or "C" language list of statements.

4.2.4.2 Explanation

- The START_UP is a special pre-declared test program function. The system software (CRAFT) invokes it automatically, the first time the Test button is pressed.
- You only need to declare the function and it will be called by the system. Of course, START_UP id also like any other function, and thus can be called by the test program as well.
- This function is an ideal place to set-up any condition, which should remain constant throughout the remaining time the programs are in use. This is an ideal place to load the test pattern.
- Any of the tester statements are allowed in this function, but recursion is not possible, do not call START_UP from within the function.

4.2.4.3 Example

```
SET_DEC_FILE  ".\PAT\3360_ls299_pin.dec"//include declaration file (PIN_define)

HW_BIN_DEF {//HW bin defined have 0~255
   all_pass                  =  0 ;
   fail_os_test              = !3 ;
   fail_leakage_test         = !4 ;
   }

SW_BIN_DEF {//SW bin defined have 0~1023
   all_pass                  =  0  ;
   fail_os_test              = !30 ;
   fail_leakage_test         = !40 ;
   }

START_UP() {
    LOAD_PAT("./PAT/ls299_pat.ppo");//Load pattern to ATE
}
test_start(){
    RELAY_OFF(IO_ALLPINS,ALL,1mS);
    FORCE_V_DPS ( Vddd, 0V, @8V, @100uA, 100uA, NORM, OFF, 1mS);
}

test_end(){
    SHOW_TEST_TIME(list_Test_Time);//show test time
}
TEST_PRO {
  BEFORE_TEST: test_start;
             //==*  test item     ? pass_do                  : fail_do ; *==//
             open_short_test     ?                          : #C(fail_os_test, fail_os_test);
             leakage_test        ? #C(all_pass_2,all_pass_2): #C(fail_leakage_test, fail_leakage_test);
  AFTER_TEST: test_end ;
 }
open_short_test()
{

}
```

```
SET_DEC_FILE    ".\PAT\FW_ls299_4sites_pin.dec"

   /**declaration for IO level set use**/
     int rst_lvl        = 1,
           os_lvl          = 2;

HW_BIN_DEF {
     all_pass                        =   1 ;
     fail_os_test                  = !2 ;
     }

SW_BIN_DEF {
     all_pass                        =   1 ;
     fail_os_test                  = !2 ;
     }

START_UP() {
       LOAD_PAT("./PAT/ls299_pat.ppo");//Load pattern to ATE
}
```

"START_UP" will be executed one time at **first test**.

4.2.5 TEST_PRO {….}

This is a declaration and the actual body of test program. It replaces main in the common C program. It defines the test sequence as well as the bin strategy (both hardware and software bins).

4.2.5.1 Format

Parameter list:

```
TEST_PRO {
      BEFORE_TEST : before_test_function ;
      test_item#   ?   passdo     :      faildo ,
      ...
      test_item#   ?   passdo     :      faildo ;
      AFTER_TEST : after_test_function ;
}
```

Parameter comment:

BEFORE_TEST: This is a keyword that tells the compiler that the following function name to be executed prior to the test routines.

before_test_function: Any smaller than 64 characters test function name. This is the name of any defined function to follow the BEFORE_TEST keywords. This is optional entry. If the BEFORE_TEST keyword is used, then this becomes a required entry.

test_item# Any smaller than 64 characters test function name.

passdo [[[flag =>] passtodo],[[flag =>] passtodo],....]
When the first true flag meet, Relative passtodo will be performed.It could be left blank, and assign next test_item# automatically.

faildo [[[flag =>] failtodo],[[flag =>] failtodo],....]
When the first true flag meet, Relative failtodo will be performed.It could be left blank, and assign next test_item# automatically.

flag condition# '&' test_item# result
condition# : strings defines at CONDITION_SET statement
test_item# result
'!' represent test_item fail

AFTER_TEST: This is a keyword that tells the compiler that the following function name to be executed prior to the tests routines.

after_test_function Any smaller than 64 characters test function name. This is the name of any defined function to follow the AFTER_TEST keywords. This is optional entry. If the AFTER_TEST keyword is used then this becomes a required entry.

passtodo/failtodo	*#C (hw_category, sw_category)* or *#F (test_item#)* category #C and test_item #F can't exist at the same flag assign. hw_category is assigned at HW_CATEGORY_NO, sw_category is assigned at SW_CATEGORY_NO.
Keywords:	TEST_PRO, BEFORE_TEST and AFTER_TEST are keywords.

4.2.5.2 Explanation

This is a declaration and the actual body of the test program. It replaces the main in the common C program. It defines the test sequence as well as the bin strategy (both hardware and software bins).

4.2.5.3 Example

<1>

```
SET_DEC_FILE    ".\PAT\FW_ls299_4sites_pin.dec"

   /**declaration for IO level set use**/
     int rst_lvl        = 1,
           os_lvl          = 2;

HW_BIN_DEF {
     all_pass                          =    1 ;
     fail_os_test                    = !2 ;
     }

SW_BIN_DEF {
     all_pass                          =    1 ;
     fail_os_test                    = !2 ;
     }

START_UP() {
       LOAD_PAT("./PAT/ls299_pat.ppo");//Load pattern to ATE
}

test_start()
{
}
test_end()
{
}

TEST_PRO {
   BEFORE_TEST: test_start;
       open_short_test        ?                          : #C(fail_os_test, fail_os_test);
   AFTER_TEST: test_end ;
  }
```

<2>

```
TEST_PRO {
  BEFORE_TEST: test_start;
       //==*  test item  ? pass_do                                  : fail_do ; *==//
       open_short_test   ?                                          : #C(fail_os_test, fail_os_test);
       leakage_test      ?                                          : #C(fail_leakage_test, fail_leakage_test);
       func_1MHZ_test    ?                                          : #C(fail_func_test, fail_func_test);
       func_2MHZ_test    ?                                          : #F(drive_sink_test);
       drive_sink_test   ?                                          : #C(fail_drive_sink_test, fail_drive_sink_test);
       u_in_test         ?                                          : #C(fail_u_in_test, fail_u_in_test);
       idd_dynamic1_test ? #F(match_test)                           : #F(idd_dynamic2_test);
       idd_dynamic2_test ?                                          : #C(fail_idd_dynamic_test, fail_idd_dynamic_test);
       match_test        ?                                          : #C(fail_match_test,fail_match_test);
       time_freq         ?                                          : #C(fail_time_freq,fail_time_freq);
       func_Rom1_test    ?                                          : #C(fail_func_Rom_test,fail_func_Rom_test);
       func_Rom2_test    ?                                          : #C(fail_func_Rom_test,fail_func_Rom_test);
       func_Rom3_test    ? !func_2MHZ_test=>#C(all_pass, all_pass),
                           func_2MHZ_test=>#C(all_pass_2,all_pass_2): #C(fail_func_Rom_test,fail_func_Rom_test);
  AFTER_TEST: test_end ;
}
```

#F mean: jump to do test item.

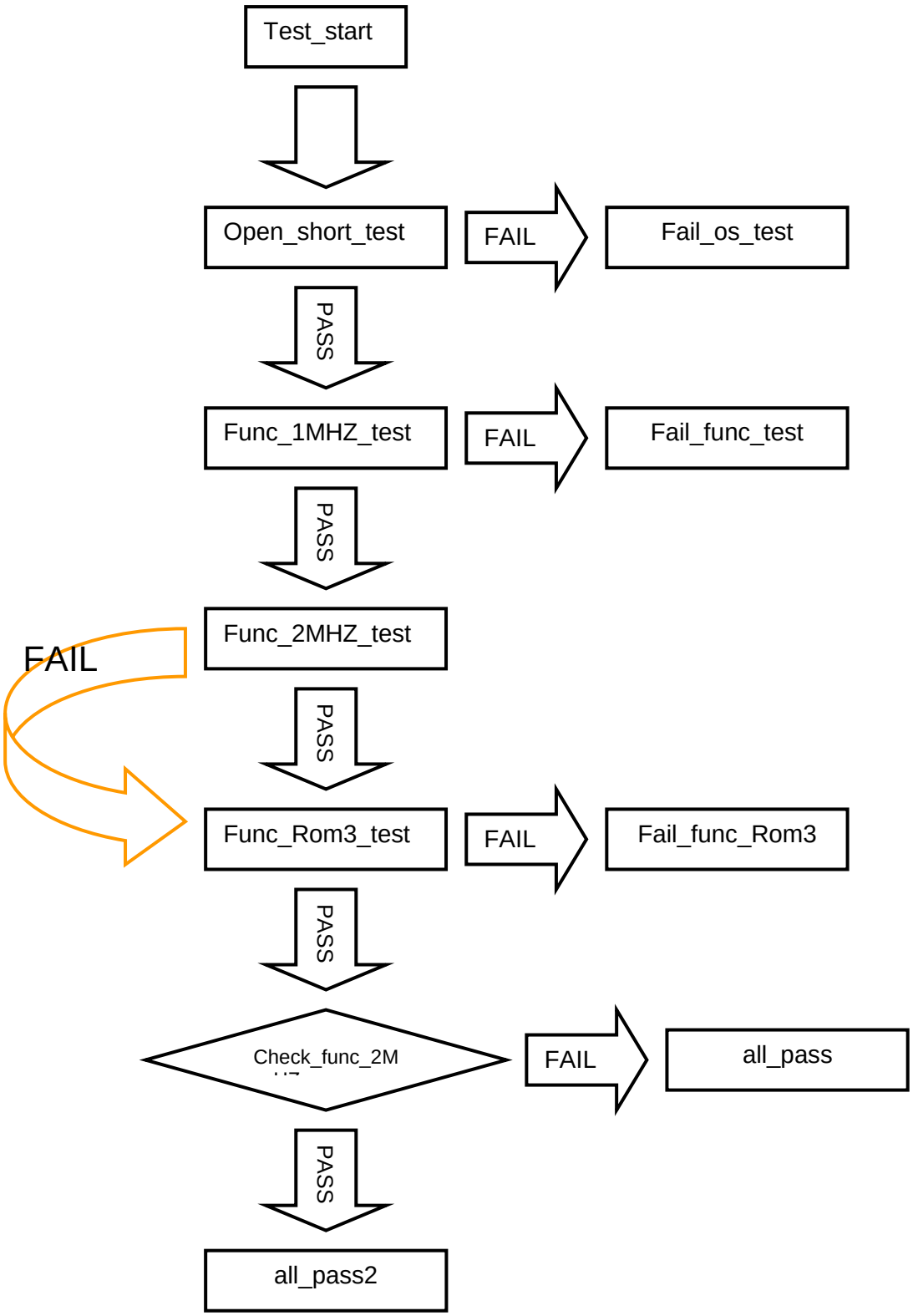
#C mean: after test to do categorize

```
TEST_PRO {
   BEFORE_TEST: test_start;
                  //==*   test item       ? pass_do        : fail_do ; *==//
     open_short_test     ?                          : #C(fail_os_test, fail_os_test);
     leakage_test        ?                          : #C(fail_leakage_test,
fail_leakage_test);
     func_1MHZ_test      ?                          : #C(fail_func_test, fail_func_test);
     func_2MHZ_test      ?                          : #F(drive_sink_test);
     drive_sink_test     ?                          : #C(fail_drive_sink_test,
fail_drive_sink_test);
     u_in_test           ?                          : #C(fail_u_in_test, fail_u_in_test);
     idd_dynamic1_test   ?   #F(match_test) : #F(idd_dynamic2_test);
     idd_dynamic2_test   ?                  : #C(fail_idd_dynamic_test,
fail_idd_dynamic_test);
     match_test          ?                          : #C(fail_match_test,fail_match_test);
     time_freq           ?                          : #C(fail_time_freq,fail_time_freq);
     func_Rom1_test      ?                  :
#C(fail_func_Rom_test,fail_func_Rom_test);
     func_Rom2_test      ?                  :
#C(fail_func_Rom_test,fail_func_Rom_test);
     func_Rom3_test      ? !func_2MHZ_test=>#C(all_pass, all_pass),
                             func_2MHZ_test=>#C(all_pass_2,all_pass_2)   :
                                                #C(fail_func_Rom_test,fail_func_Rom_te
                                                st);
   AFTER_TEST: test_end ;
 }
```

<3>

```
TEST_PRO {
  BEFORE_TEST: test_start;
            //==*  test item     ? pass_do                                  : fail_do ; *==//
            open_short_test      ?                                          : #C(fail_os_test, fail_os_test);
            func_1MHZ_test       ?                                          : #C(fail_func_test, fail_func1_test);
            func_2MHZ_test       ?                                          : #F(func_Rom3_test);
            func_Rom3_test       ? !func_2MHZ_test=>#C(all_pass, all_pass),
                                   func_2MHZ_test=>#C(all_pass_2,all_pass_2)  : #C(fail_func_Rom_test,fail_func_Rom_test);
  AFTER_TEST: test_end ;
 }
```

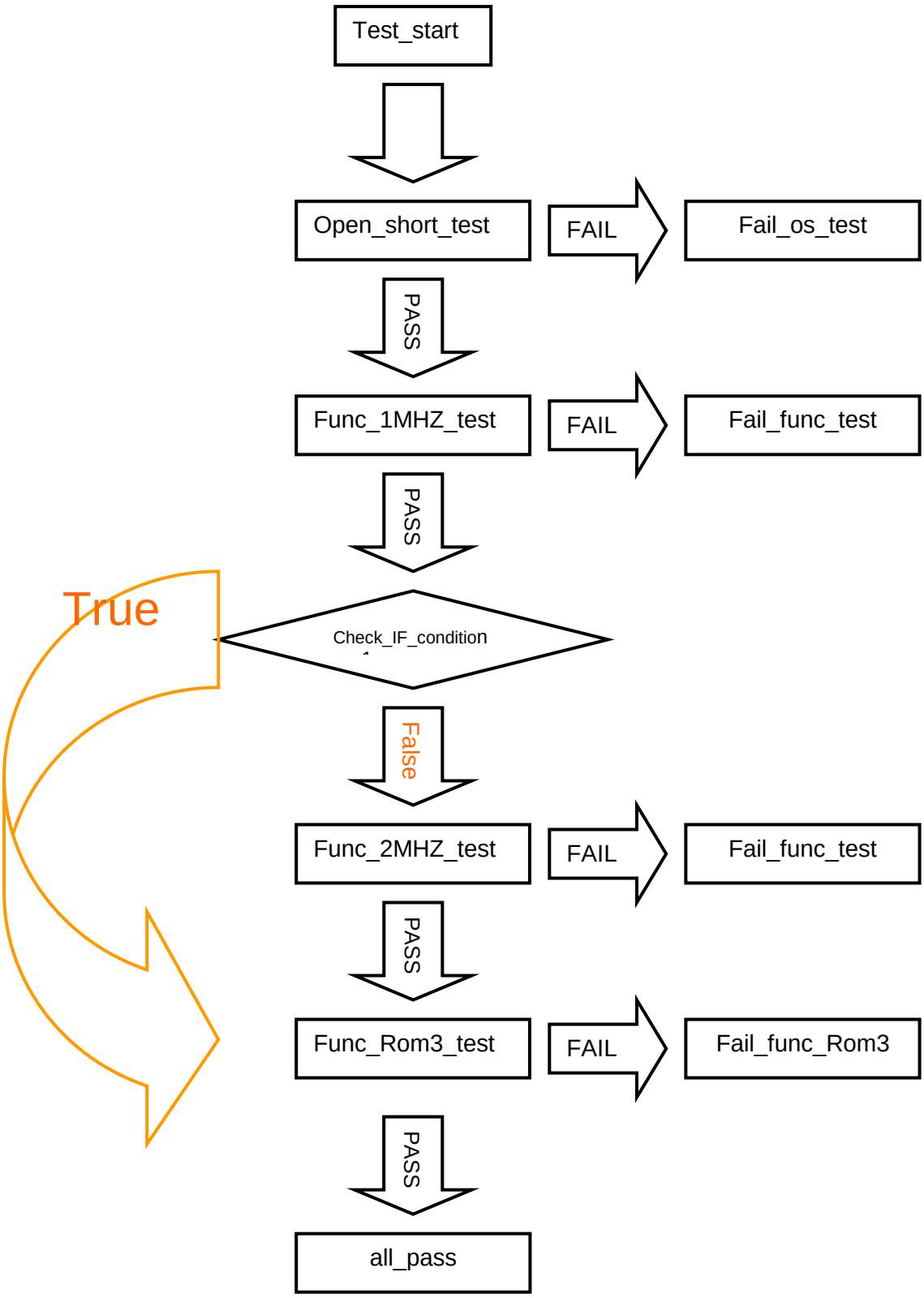
```
TEST_PRO {
   BEFORE_TEST: test_start;
                    //==*   test item       ? pass_do        : fail_do ; *==//
     open_short_test     ?                    : #C(fail_os_test, fail_os_test);
     func_1MHZ_test      ?                         : #C(fail_func_test, fail_func_test);
     func_2MHZ_test      ?                         : #F(drive_sink_test);
     func_Rom3_test        ? !func_2MHZ_test=>#C(all_pass, all_pass),
                             func_2MHZ_test=>#C(all_pass_2,all_pass_2)    :
                                                  #C(fail_func_Rom_test,fail_func_Rom_te
                                                  st);
   AFTER_TEST: test_end ;
  }
```



<4> use condition set to change test flow

```
CONDITION_SET  {
    IF_NO_CONDITION : (PART = NOCOND);
    IF_condition_1 : (PART = condition_1);
    IF_condition_2 : (PART = condition_2);
    IF_condition_3 : (PART = condition_3);
}

TEST_PRO {
  BEFORE_TEST: test_start;
            //==*  test item     ? pass_do                                  : fail_do ; *==//
            open_short_test     ?                                          : #C(fail_os_test, fail_os_test);
            func_1MHZ_test      ?  IF_condition_1=> #F(func_Rom3_test)     : #C(fail_func_test, fail_func1_test);
            func_2MHZ_test      ?                                          : #C(fail_func_test, fail_func2_test);
            func_Rom3_test      ? #C(all_pass,all_pass)                    : #C(fail_func_Rom_test,fail_func_Rom_test);

  AFTER_TEST: test_end ;
 }
```

```
CONDITION_SET{
      IF_NO_CONDITION : (PART = NOCOND) ;
      IF_condition_1 : (PART = condition_1) ;
      IF_condition_2 : (PART = condition_2) ;
      IF_condition_3 : (PART = condition_3) ;

}

TEST_PRO {
   BEFORE_TEST: test_start;
                  //==*   test item       ? pass_do        : fail_do ; *==//
    open_short_test     ?                                  : #C(fail_os_test, fail_os_test);
    func_1MHZ_test     ? IF_condition_1 =>#F(func_Rom3_test) : #C(fail_func_test,
fail_func1_test);
    func_2MHZ_test     ?                                  : #C(fail_func_test, fail_func2_test);
    func_Rom3_test        ? #C(all_pass, all_pass ):
#C(fail_func_Rom_test,fail_func_Rom_test);
   AFTER_TEST: test_end ;
 }
```


Test_start
Open_short_test
FAIL
Fail_os_test
PASS
Func_1MHZ_test
FAIL
Fail_func_test
PASS
True
Check_IF_condition
False
Func_2MHZ_test
FAIL
Fail_func_test
PASS
Func_Rom3_test
FAIL
Fail_func_Rom3
PASS
all_pass

4.2.6 RELEASE(){ … }

This is a special function. At program release time, it is called automatically. Any valid statements can be used in the body of this function. This function is then followed by a default power down function.

4.2.6.1 Format

Parameter list:

```
RELEASR ()
{
        statements
}
```

Parameter comment:

Statements: Any valid TPG (Test Program Generator) or "C" statements list.

Default value:
None.

4.2.6.2 Explanation

This is a special function. At program release time, it is called automatically. Any valid statements can be used in the body of this function. This function is then followed by a default power down function.

4.2.6.3 Example

```
SET_DEC_FILE    ".\PAT\FW_ls299_4sites_pin.dec"

   /**declaration for IO level set use**/
     int rst_lvl        = 1,
           os_lvl          = 2;

HW_BIN_DEF {
     all_pass                          =    1 ;
     fail_os_test                    = !2 ;
     }

SW_BIN_DEF {
     all_pass                          =    1 ;
     fail_os_test                    = !2 ;
     }

START_UP() {
        LOAD_PAT("./PAT/ls299_pat.ppo");//Load pattern to ATE
}

test_start()
{
}
test_end()
{
}
RELEASE()
{
        FORCE_V_DPS ( Vddd, 0V, @8V, @100uA, 100uA, NORM, OFF, 1mS);
}
TEST_PRO {
   BEFORE_TEST: test_start;
        open_short_test        ?                          : #C(fail_os_test, fail_os_test);
   AFTER_TEST: test_end ;
  }
```

It will be executed when unload program

4.2.7 CONDITION_SET

This section explains how to select one condition to assign the three-system environment variably.

4.2.7.1 Format

Parameter list:

```
CONDITION_SET {
    condition#   :   ( environment assign );
    ...
    condition#   :   ( environment assign );
}
```

Parameter comment:

CONDITION_SET: This is a keyword.

condition#: Any smaller than 64 characters string

environment assign: Any combination of system environments assign, and use blank to separate them.

System environment variables: PART, JOB, ENV

Assign value: Any smaller than 64 characters.

Default value:
None.

4.2.7.2 Explanation

We select one condition to assign the three-system environment variably (PART/JOB/ENV) when load test program. User can use "GetPART", "GetJOB", and "GetENV" read back string. Please refer to Example.

4.2.7.3 Example

User can select these conditions when loading program. And selected condition will be **true**

Use GetPART, GetJOB , GetENV read back string

```
CONDITION_SET  {
    IF_NO_CONDITION : (PART = NOCOND,JOB=nomal,ENV=nomal);
    IF_ROM1 : (PART = ROM1,JOB=ROM1_JOB,ENV=ROM1_ENV);
    IF_ROM2 : (PART = ROM2,JOB=ROM2_JOB,ENV=ROM2_ENV);
    IF_ROM3 : (PART = ROM3,JOB=ROM2_JOB,ENV=ROM2_ENV);
}
test_start()
{
    if(IF_ROM1)printf("device code=%s,JOB=%s,ENV=%s\n",IF_ROM1.GetPART(),IF_ROM1.GetJOB(),IF_ROM1.GetENV());
    if(IF_ROM2)printf("device code=%s,JOB=%s,ENV=%s\n",IF_ROM2.GetPART(),IF_ROM2.GetJOB(),IF_ROM2.GetENV());
    if(IF_ROM3)printf("device code=%s,JOB=%s,ENV=%s\n",IF_ROM3.GetPART(),IF_ROM3.GetJOB(),IF_ROM3.GetENV());

    LEVELS(rst_lvl,5mS) ;
    RELAY_OFF(IO_ALLPINS,ALL,1mS);

    FORCE_V_DPS ( Vddd, 0V, @8V, @100uA, 100uA, NORM, OFF, 1mS);
    FORCE_V_PREF(  Vdd, 0V, @6V, @100uA, 100uA, NORM, OFF, 1mS);
}
```

```
CONDITION_SET   {
      IF_NO_CONDITION : (PART = NOCOND,JOB=nomal,ENV= nomal);
      IF_ROM1 : (PART = ROM1,JOB=ROM1_JOB,ENV=ROM1_ENV);
      IF_ROM2 : (PART = ROM2,JOB=ROM2_JOB,ENV=ROM2_ENV);
      IF_ROM3 : (PART = ROM3,JOB=ROM3_JOB,ENV=ROM3_ENV);
}
test_start()
{
       if(IF_ROM1)printf("device
code=%s,JOB=%s,ENV=%s\n",IF_ROM1.GetPART(),IF_ROM1.GetJOB(),IF_ROM1
.GetENV());
       if(IF_ROM2)printf("device
code=%s,JOB=%s,ENV=%s\n",IF_ROM2.GetPART(),IF_ROM2.GetJOB(),IF_ROM2
.GetENV());
       if(IF_ROM3)printf("device
code=%s,JOB=%s,ENV=%s\n",IF_ROM3.GetPART(),IF_ROM3.GetJOB(),IF_ROM3
.GetENV());
      LEVELS(rst_lvl,5mS) ;
      RELAY_OFF(IO_ALLPINS,ALL,1mS);

      FORCE_V_DPS ( Vddd, 0V, @8V, @100uA, 100uA, NORM, OFF, 1mS);
      FORCE_V_PREF(   Vdd, 0V, @6V, @100uA, 100uA, NORM, OFF, 1mS);
}
```

4.2.8 SPEC_DEF

Pre-define spec variable for using in test program.

```
SPEC_DEF   ( FT_Test) {
    VOLTAGE   vdd   = 4.5V    : 5.0V    : 5.5V   ;//SPEC min : typical : max ;
    VOLTAGE   vddd = 4.5V    : 5.0V    : 5.5V   ;
    VOLTAGE   vih   = 4.5V    : 5.0V    : 5.5V   ;
    VOLTAGE   vil   = 0.0V    : 0.0V    : 0.8V   ;
    VOLTAGE   voh   = 2.4V    : 2.4V    : 3.2V   ;
    VOLTAGE   vol   = 0.35V   : 0.8V    : 0.8V   ;
    ...

    ...
    CURRENT   iil   = -0.2mA : -0.2mA : -0.2mA;
    CURRENT   iih   = 20.0uA : 20.0uA : 20.0uA;
    CURRENT   iol   = 12mA    : 12mA    : 12mA   ;
    CURRENT   ioh   = -2.6mA : -2.6mA : -2.6mA;
    VOLTAGE   vref = 1.8V    : 1.8V    : 1.8V   ;
    CURRENT   idd   = 22mA    : 35mA    : 50mA   ;
}
SPEC_DEF   (QC_Test) {
    VOLTAGE   vdd   = 4.5V    : 5.0V    : 5.5V   ;
    VOLTAGE   vddd = 4.5V    : 5.0V    : 5.5V   ;
    VOLTAGE   vih   = 4.5V    : 5.0V    : 5.5V   ;
    VOLTAGE   vil   = 0.0V    : 0.0V    : 0.8V   ;
    VOLTAGE   voh   = 2.4V    : 2.4V    : 3.2V   ;
    VOLTAGE   vol   = 0.35V   : 0.8V    : 0.8V   ;
    CURRENT   iil   = -0.2mA : -0.2mA : -0.2mA;
    CURRENT   iih   = 20.0uA : 20.0uA : 20.0uA;
    CURRENT   iol   = 12mA    : 12mA    : 12mA   ;
    CURRENT   ioh   = -2.6mA : -2.6mA : -2.6mA;
    VOLTAGE   vref = 1.8V    : 1.8V    : 1.8V   ;
    CURRENT   idd   = 30mA    : 35mA    : 42mA   ;

}
```

The Var_type of maximum value is 50 settings.

4.2.8.1 Format

Parameter list:

```
SPEC_DEF   (spec_table_name) {
    Var_type   var_name = min_value : typ_value : max_value ;
    . . . . .
    Var_type   var_name = min_value : typ_value : max_value ;
}
```

Parameter comment:

usage: var_name.min (= min_value)
"var_name" (equal "var_name.typ") (= typ_value)
var_name.max (= max_value)

spec_table_name: The name of load board to be used by the test program.
Max. 32 chars.
Type: string

No entry – illegal

Var_type: TIME, CURRENT, VOLTAGE, FREQUENCY or double.
No entry: illegal

var_name: Variable name with the value (min_value/typ_value/max_value)
No entry: illegal

min_value: Only constant is allowed.
No entry: default 0

typ_value: Only constant is allowed.
No entry: default 0

max_value: Only constant is allowed.
No entry: default 0

4.2.8.2 Explanation

Pre-define spec variable for using in test program.

vdd.min : vdd (vdd.typ):vdd.max
vddd.min: vddd (vddd.typ):vddd.max

Example
<1>

```
//SPEC define    //Test item please refer "idd_dynamic_test"
SPEC_DEF    ( FT_Test) {
      VOLTAGE   vdd    = 4.5V     : 5.0V     : 5.5V    ;//SPEC min : typical : max ;
      VOLTAGE   vddd = 4.5V     : 5.0V     : 5.5V    ;
      VOLTAGE   vih    = 4.5V     : 5.0V     : 5.5V    ;
      VOLTAGE   vil    = 0.0V     : 0.0V     : 0.8V    ;
      VOLTAGE   voh    = 2.4V     : 2.4V     : 3.2V    ;
      VOLTAGE   vol    = 0.35V    : 0.8V     : 0.8V    ;
      CURRENT   iil    = -0.2mA : -0.2mA : -0.2mA;
      CURRENT   iih    = 20.0uA : 20.0uA : 20.0uA;
      CURRENT   iol    = 12mA     : 12mA     : 12mA    ;
      CURRENT   ioh    = -2.6mA : -2.6mA : -2.6mA;
      VOLTAGE   vref = 1.8V     : 1.8V     : 1.8V    ;
      CURRENT   idd    = 22mA     : 35mA     : 50mA    ;
}

SPEC_DEF    (QC_Test) {
      VOLTAGE   vdd    = 4.5V     : 5.0V     : 5.5V    ;
      VOLTAGE   vddd = 4.5V     : 5.0V     : 5.5V    ;
      VOLTAGE   vih    = 4.5V     : 5.0V     : 5.5V    ;
      VOLTAGE   vil    = 0.0V     : 0.0V     : 0.8V    ;
      VOLTAGE   voh    = 2.4V     : 2.4V     : 3.2V    ;
      VOLTAGE   vol    = 0.35V    : 0.8V     : 0.8V    ;
      CURRENT   iil    = -0.2mA : -0.2mA : -0.2mA;
      CURRENT   iih    = 20.0uA : 20.0uA : 20.0uA;
      CURRENT   iol    = 12mA     : 12mA     : 12mA    ;
      CURRENT   ioh    = -2.6mA : -2.6mA : -2.6mA;
      VOLTAGE   vref = 1.8V     : 1.8V     : 1.8V    ;
      CURRENT   idd    = 30mA     : 35mA     : 42mA    ;
}
```

idd.min: idd (idd.typ): idd.max

<2>

```
FORCE_V_DPS ( Vddd, vddd, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
FORCE_V_PREF(    Vdd,    vdd, @6V, @100mA, 100mA, NORM, ON, 1mS);

LEVELS(func_lvl, 3mS);

PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE(            CLK,    RZ, EDGE,      MASK, IO_NRZ, OFF);

RELAY_ON( IO_ALLPINS, PDCL, 3mS);

//u_in : JCONT
DATALOG_MSG="IDD Dynamic Test and u_in : JCONT";
TEST_NO(50);
JUDGE_PAT(jcont1_pat, __jcont1_pat);//JCONT

DATALOG_MSG="Judge I mode";
TEST_NO(51);
JUDGE_I_DPS ( Vddd, idd.min, idd.max, 2mS);
JUDGE_I_PREF(    Vdd, idd.min, idd.max, 2mS);

DATALOG_MSG="Judge I Average mode ";
TEST_NO(52);
JUDGE_I_DPS ( Vddd, idd.min, idd.max, 2mS, 32, AVE, 50uS, , );
JUDGE_I_PREF(    Vdd, idd.min, idd.max, 2mS, 32, AVE, 50uS, , );
```

4.2.9 TEST_NO

This statement sets the test index in the test plan program.

4.2.9.1 Format

Parameter list:

```
TEST_NO (test_number) ;
```

Parameter comment:
test_number: A interval from 1 to 33000000.
C Type: integer
No entry: illegal
Total TEST_NO number limit to 8000

4.2.9.2 Explanation

The test_no statement sets the test No. in the test plan program. The test No. itself does not affect the test contents. However, it provides great convenience when uses to identify the status of the test execution.

4.2.9.3 Example

```
FORCE_V_DPS ( Vddd, vddd, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
FORCE_V_PREF(   Vdd,   vdd, @6V, @100mA, 100mA, NORM, ON, 1mS);

LEVELS(func_lvl, 3mS);

PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE(          CLK,   RZ, EDGE,     MASK, IO_NRZ, OFF);

RELAY_ON( IO_ALLPINS, PDCL, 3mS);

//u_in : JCONT
DATALOG_MSG="IDD Dynamic Test and u_in : JCONT";
TEST_NO(50);
JUDGE_PAT(jcont1_pat, __jcont1_pat);//JCONT

DATALOG_MSG="Judge I mode";
TEST_NO(51);
JUDGE_I_DPS ( Vddd, idd.min, idd.max, 2mS);
JUDGE_I_PREF(   Vdd, idd.min, idd.max, 2mS);

DATALOG_MSG="Judge I Average mode ";
TEST_NO(52);
JUDGE_I_DPS ( Vddd, idd.min, idd.max, 2mS, 32, AVE, 50uS, , );
JUDGE_I_PREF(   Vdd, idd.min, idd.max, 2mS, 32, AVE, 50uS, , );
```

Identify test number 50

4.2.10 DATALOG_MSG

A datalog message to be printed on next judge_statement.

4.2.10.1 Format

Parameter list:

```
DATALOG_MSG ( "string " ) ;
```

Parameter comment:
string : any string to describe next judgement

4.2.10.2 Explanation

A datalog message to be printed on next judge_statement;

4.2.10.3 Example

```
FORCE_V_DPS ( Vddd, vddd, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
FORCE_V_PREF(   Vdd,   vdd, @6V, @100mA, 100mA, NORM, ON, 1mS);

LEVELS(func_lvl, 3mS);

PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE(          CLK,   RZ, EDGE,     MASK, IO_NRZ, OFF);

RELAY_ON( IO_ALLPINS, PDCL, 3mS);

//u_in : JCONT
DATALOG_MSG="IDD Dynamic Test and u_in : JCONT";
TEST_NO(50);
JUDGE_PAT(jcont1_pat, __jcont1_pat);//JCONT

DATALOG_MSG="Judge I mode";
TEST_NO(51);
JUDGE_I_DPS ( Vddd, idd.min, idd.max, 2mS);
JUDGE_I_PREF(   Vdd, idd.min, idd.max, 2mS);

DATALOG_MSG="Judge I Average mode ";
TEST_NO(52);
JUDGE_I_DPS ( Vddd, idd.min, idd.max, 2mS, 32, AVE, 50uS, , );
JUDGE_I_PREF(   Vdd, idd.min, idd.max, 2mS, 32, AVE, 50uS, , );
```

Use “DATALOG_MSG” to print message on datalog

4.2.11 WAIT

This statement inserts a waiting time between processes.

4.2.11.1 Format

Parameter list:

WAIT (wait_time);

Parameter comment:

wait_time: Specify the waiting time after conditions setup. Variable and constant are both allowed. Constant must be terminated with the unit "S", which can be
preceded by the prefixes "S, pS, nS, uS or mS". The value of this entry must be a positive floating value.

Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: illegal

4.2.11.2 Explanation

Executing this instruction means that no other function or program will be executed until the waiting time is end.

4.2.11.3 Example

Use Variable or constant value
Variable TYPE : "**TIME**" or "**double**"

```
DATALOG_MSG="Judge I Average mode ";
TEST_NO(52);
WAIT(20mS);
JUDGE_I_DPS ( Vddd, idd.min, idd.max, 2mS, 32, AVE, 50uS, , );
JUDGE_I_PREF(   Vdd, idd.min, idd.max, 2mS, 32, AVE, 50uS, , );

DATALOG_MSG="Judge I Average mode and Average Min Max clamp";
TEST_NO(53);
//declare===\\
GLOBAL TIME delay_time=20mS;   (need to declare)
// GLOBAL double delay_time=20.0E-3;
WAIT(delay_time);
JUDGE_I_DPS ( Vddd, idd.min, idd.max, 2mS, 32, AVE, 50uS, 0mA, 50mA);
JUDGE_I_PREF(   Vdd, idd.min, idd.max, 2mS, 32, AVE, 50uS, 0mA, 50mA);
```

4.2.12 GLOBAL

This section explains how to define variables are used in the plan program.

4.2.12.1 Format

Parameter list:

```
GOBAL variable_type    variable_name;
```

Parameter name:

Variable_type: One of the pre-defiined "C" language variable type (such as int, and float), one of the pre-defined CRAFT type (such as TIME, and VOLTAGE), or any programmer defined type.

Variable_name: A valid name of the following all of "C" language naming rules. Since this entry specifies the name of variable, it is a required entry.

Default Value:

CURRENT:	Variable with A/a (current) unit declaration
FREQUENCY:	Variable with HZ/hz (hertz) unit declaration
TIME:	Variable with S/s (time) unit declaration
VOLTAGE:	Variable with V/v (voltage) unit declaration
RESISTANCE:	Variable with OHM/ohm (ohm) unit declaration
STRING:	Variable with char array, if no instant then assign to size 15
INT:	Variable with integer.

4.2.12.2 Explanation

Keyword is added before any variable declaration, to make it global. A list of all Global variables appears on the menu system, and the user to control or modify the test program execution can change their value. If the same name is defined as a global variable and as a local variable, the local variable is valid within the test program.

4.2.12.3 Example

```
GLOBAL double delay_time=20.0E-3;
GLOBAL    int     list_Test_Time            = 0,//show test time flag
                  os_test_by_PMU            = 0,
                  os_test_by_pat            = 0,
                  os_test_by_ost            = 1,//LPC32
                  os_test_by_ppmu           = 0,
                  Is_Test_On_Fail_Return    = 0;
```

These variable can be changed in Craft System(Plan_Debug) .it will be useful and commodiously

```
DATALOG_MSG="Judge I Average mode ";
TEST_NO(52);
WAIT(20mS);
JUDGE_I_DPS ( Vddd, idd.min, idd.max, 2mS, 32, AVE, 50uS, , );
JUDGE_I_PREF(    Vdd, idd.min, idd.max, 2mS, 32, AVE, 50uS, , );

DATALOG_MSG="Judge I Average mode and Average Min Max clamp";
TEST_NO(53);
//declare===\\
// GLOBAL TIME delay_time=20mS;    (need to declare)
// GLOBAL double delay_time=20.0E-3;
WAIT(delay_time);
JUDGE_I_DPS ( Vddd, idd.min, idd.max, 2mS, 32, AVE, 50uS, 0mA, 50mA);
JUDGE_I_PREF(    Vdd, idd.min, idd.max, 2mS, 32, AVE, 50uS, 0mA, 50mA);
```

“delay_time” is a global variable .

4.3 MLDPS Control Statement

4.3.1 FORCE_I_MLDPS

This statement sets the selected MLDPS to the I-force mode.
Release version : **CRAFT_3380_2.00_SP1**

4.3.1.1 Format

```
FORCE_I_MLDPS(MLDPS_name, f_curent, i_range, v_range, v_clamp, m_mode,
                    ON/OFF, wait_time) ;
```

Parameter list:

MLDPS_name: Define in *.dec file
MLDPS channels are defined by slot number.
EX:
Slot8 channel 512~ 543
Slot9 (default MLDPS SLOT) 576~607
And Type declared "MLDPS" in declaration file.

Both variables and constants are allowed.
Type: integer
No entry: illegal

f_curent: The amount of force current from -1A to +1A by the selected MLDPS.
Both variables and constants are allowed.
Type: CURRENT or double
Unit: A, pA, nA, uA, mA
No entry: illegal

I_range	Minimum	Maximum
@1A	-1A	+1A
@500mA	-500mA	+500mA
@25mA	-25mA	+25mA

I_range: One of eight possible hardware ranges. For the best accuracy, you should program the current by using the closest range possible.

I_range	Minimum	Maximum
@1A	-1A	+1A
@500mA	-500mA	+500mA
@25mA	-25mA	+25mA

//only @6V support @1A i-range

V_range: No entry: illegal

I_range	Minimum	Maximum
@6V	-6V	+6V
@12V	-12.5V	+12.5V

V_clamp: This value must be in the specified V_range, as following:

I_range	Minimum	Maximum
@6V	-6V	+6V
@12V	-12.5V	+12.5V

Variables and constants are allowed.
Type: VOLTAGE or double
Unit: V, pV, nV, uV, mV
No entry: illegal

ON/OFF: ON – turns on MLDPS by marking MLDPS utility relay after conditions setup.
OFF – turns off MLDPS by breaking MLDPS utility relay after conditions setup.
Type: integer
No entry: keep voltage state

[Notice]: Force ON and Force OFF procedure
Force ON : Relay closes first and force setting value
Force OFF : Force setting value first and Relay open

m_mode : Specify the MLDPS measure mode ;
NORM (CPU trigger)
No entry: Default = NORM

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 400uS

Vi_mode: IFVM (default)

IFIM

4.3.1.2 Example

```
FORCE_I_MLDPS (Vcc, 1mA, @25mA, @12V, 6V, NORM, ON, 2mS);
JUDGE_V_MLDPS(Vcc, 2.9V, 3.1V, 2mS, 32, AVE, 50uS, ,);
```

4.3.1.3 Relative

FORCE_CURRENT

4.3.2 FORCE_V_MLDPS

This statement sets the selected MLDPS to the programmed voltage condition, as well as ranges and clamp currents. Release version : **CRAFT_3380_2.00_SP1**

4.3.2.1 Format

Parameter list:

FORCE_V_MLDPS(MLDPS_name, f_volt, v_range, i_range, I_clamp, m_mode, ON/OFF, wait_time) ;

Parameter comment:

MLDPS_name: Define in *.dec file
MLDPS channels are defined by slot number.
EX:
Slot8 chanenl 512~ 543
Slot9 (default MLDPS SLOT) 576~607
And Type declared "MLDPS" in declaration file.
Type: integer
No entry: illegal

f_volt:
Variable is allowed.
Type: VOLTAGE or double
Constant values must be followed by the unit 'V', pV, nV, uV, mV
No entry: illegal

I_range	Minimum	Maximum
@6V	-6V	+6V
@12V	-6V	+12V

V_range: No entry: illegal

I_range	Minimum	Maximum
@6V	-6V	+6V
@12V	-6V	+12V

I_range: One of eight possible hardware ranges. For the best accuracy, you should program the voltage by using the closest range possible.

- Possible hardware range:

I_range	Minimum	Maximum
@5uA	-5uA	+5uA
@25uA	-25uA	+25uA
@250uA	-250uA	+250uA
@2.5mA	-2.5mA	+2.5mA
@25mA	-25mA	+25mA
@500mA	-500mA	+500mA
@1A	-1A	+1A

//only @6V support @1A i-range

Type: integer
No entry: illegal

I_clamp: The maximum current to be supplied by the selected MLDPS. One of eight possible Hardware ranges. This value must be in the specified I_range, as following:

- Possible hardware range:

I_calmp	Range
5uA	Disable clamp function
25uA	Disable clamp function
250uA	±250uA ∗ (10% ~ 102.5%)
2.5mA	±2.5mA ∗ (10% ~ 102.5%)
25mA	±25mA ∗ (10% ~ 102.5%)
500mA	±500mA ∗ (10% ~ 102.5%)
1A	±1A ∗ (10% ~ 102.5%)

Variables are allowed. Constant values must be followed by the unit 'A', which can be preceded by the prefixes A, pA, nA, uA, mA.
Type: CURRENT or double
No entry: set as the maximum value of the present I_range

m_mode: Select the mode.
NORM (CPU trigger)
No entry: NORM

ON/OFF: The state of the specified MLDPS output buffer. There are two possible states:
ON – turns on MLDPS by marking MLDPS utility relay after conditions setup.
OFF – turns off MLDPS by breaking MLDPS S utility relay after conditions setup.
Type: integer
No entry: keep current state
[Notice]: Force ON and Force OFF procedure
Force ON : Relay closes first and force setting value
Force OFF : Force setting value first and Relay open

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
**no_entry: Default relay ON Delay = 500uS
Default relay OFF Delay : 2mS (to ensure the discharge ok)

4.3.2.2 Example

```
idd_static_test()
{
	DATALOG_MSG="idd_static_test";
	SET_LEVELN(com_lvl, IO_ALLPINS, 0V, 4.5V,1V,3V,0uA,0uA,3.0V, , );
	LEVELS(com_lvl, 2mS);
	FORCE_V_MLDPS(Vcc, 6V, @6V, @250uA, 250uA, NORM, ON, 2mS);
	RELAY_ON(IO_ALLPINS,PDCL, 1mS);
	TEST_NO(32);
	JUDGE_PAT(static_idd_st, static_idd_sp );
	TEST_NO(33);
	JUDGE_I_MLDPS(Vcc,0uA , 80uA, 2mS);
	POWER_DOWN_FAIL_SITE( );
}
```

4.3.2.3 Relative

FORCE_VOLT

4.3.3 JUDGE_I_MLDPS

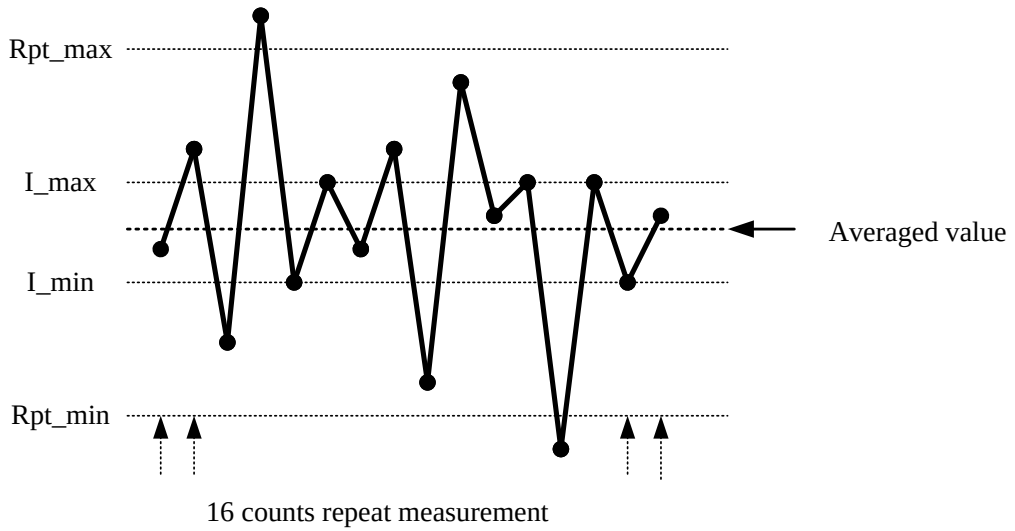
Perform a MLDPS current measurement and compare the value to the specified limits.

4.3.3.1 Format

Parameter list:

```
JUDGE_I_MLDPS(MLDPS_name, I_min, I_max, wait_time[, ave_cnt, ave_mode,
sampling_interval , rpt_min, rpt_max ]) ;
```

// driver description : 1. delay "wait_time"
2. iteration "ave_cnt" to measure current and do average
3. each measured value must between rpt_min and rpt_max,
otherwise the judgement will be set to fail .



Parameter comment:

MLDPS_name: Define in *.dec file
MLDPS channels are defined by slot number.
EX:
Slot8 chanenl 512~ 543
Slot9 (default MLDPS SLOT) 576~607
And Type declared "MLDPS" in declaration file.
Type: integer
No entry: illegal

I_min: Refer to the minimum current value to generate the pass or fail.
Omitting the parameter is possible, but the comparison will be made only regarding I_max.
When this happens, omitting I_max description is not allowed. This is lower limit value.
Variables and constants are allowed.
Type: CURRENT or double
Unit: A, pA, nA, uA, mA

No entry: Only high limit.

I_max: Refer to the maximum current value to generate the pass or fail. Omitting the parameter is possible, but the comparison will be made only regarding I_min. When this happens, omitting I_min description is not allowed. This is upper limit value.
Variables and constants are allowed.
Type: CURRENT or double
Unit: A, pA, nA, uA, mA
No entry: Only low limit.

wait_time; Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 500uS

ave_cnt : the average count
no_entry: Default = 1

ave_mode : the average mode
AVE : average
RMS : root mean square
RMS_X : root mean square; //calculation formula is different from "RMS"; Calibration table import to hw-driver-function (not in fw-driver-function)
(RMS_X Release version : CRAFT_3380_2.50-SP3)
type: int
no_entry: Default = AVE

sampling_interval : the sampling interval between each measurement
no_entry: Default = 50uS ; minimum value >5uS.

rpt_min: the low limit of each measurement ; if any measurement smaller than this value, then the judge result will be set to fail

rpt_max: the high limit of each measurement ; if any measurement larger than this value, then the judge result will be set to fail

if rpt_min & rpt_max no_entry : skip the rpt_min/rpt_max judgement

4.3.3.2 Example

```
idd_static_test()
{
    DATALOG_MSG="idd_static_test";
    SET_LEVELN(com_lvl, IO_ALLPINS, 0V, 4.5V,1V,3V,0uA,0uA,3.0V, , );
    LEVELS(com_lvl, 2mS);
    FORCE_V_MLDPS(Vcc, 6V, @6V, @250uA, 250uA, NORM, ON, 2mS);
    RELAY_ON(IO_ALLPINS,PDCL, 1mS);
    TEST_NO(32);
    JUDGE_PAT(static_idd_st, static_idd_sp );
    TEST_NO(33);
    JUDGE_I_MLDPS(Vcc,0uA , 80uA, 2mS);
    POWER_DOWN_FAIL_SITE( );
}
```

4.3.3.3 Relative

JUDGE_CURRENT

4.3.4 JUDGE_V_MLDPS

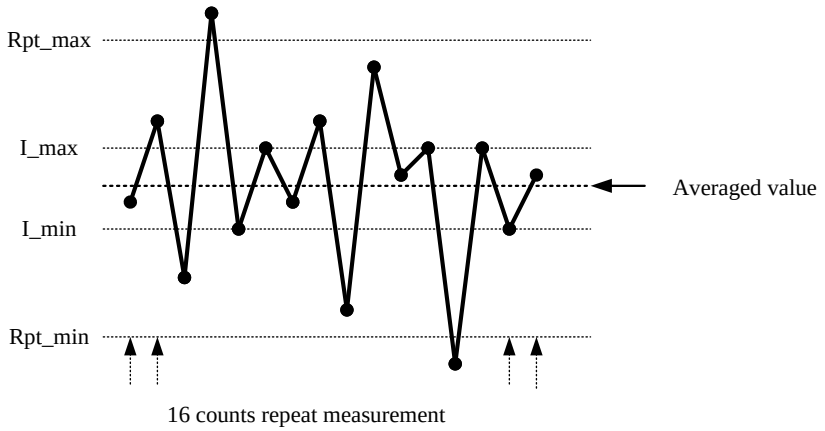
Perform a MLDPS voltage measurement and compare the value to the specified limits.
Release version : **CRAFT_3380_2.00_SP1**

4.3.4.1 Format

Parameter list:

```
JUDGE_V_MLDPS(MLDPS_name, V_min, V_max, wait_time[, ave_cnt, ave_mode,
                    sampling_interval , rpt_min, rpt_max ]) ;
```

// driver description : 1. delay "wait_time"
2. iteration "ave_cnt" to measure voltage and do average
3. each measured value must between rpt_min and rpt_max,
otherwise the judgement will be set to fail .



Parameter comment:

MLDPS_name: Define in *.dec file
MLDPS channels are defined by slot number.
EX:
Slot8 chanenl 512~ 543
Slot9 (default MLDPS SLOT) 576~607
And Type declared "MLDPS" in declaration file.
Type: integer
No entry: illegal

v_min: Refer to the minimum voltage to generate the pass or fail. Omitting the parameter is possible, but the comparison will be made only regarding I_max.
When this happens, omitting I_max description is not allowed. This is lower limit value. Variables and constants are allowed.
Type: VOLTAGE or double
Unit: V, mV

No entry: only high limit

v_max: Refer to the maximum voltage value to generate the pass or fail. Omitting the parameter is possible, but the comparison will be made only regarding I_min.
When this happens, omitting I_min description is not allowed. This is the upper limit value. Variables and constants are allowed.
Type: VOLTAGE or double

Unit: V, mV
No entry: Only low limit.

wait_time; Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 500uS

ave_cnt: The average count.
No entry: Default = 1

ave_mode: The average mode
AVE: average
RMS: root mean square
RMS_X : root mean square; //calculation formula is different from "RMS"; Calibration table import to hw-driver-function (not in fw-driver-function)
(RMS_X Release version : CRAFT_3380_2.50-SP3)
Type: int
No entry: Default = AVE

sampling_interval: the sampling interval is between each measurement.
**no_entry: Default = 50uS ; minimum value >5uS.

rpt_min: the low limit of each measurement ; if any measurement smaller than this value, then the judge result will be set to fail

rpt_max: the high limit of each measurement ; if any measurement larger than this value, then the judge result will be set to fail
if rpt_min & rpt_max no_entry : skip the rpt_min/rpt_max judgement

4.3.4.2 Example

```
FORCE_I_MLDPS (Vcc, 1mA, @25mA, @12V, 6V, NORM, ON, 2mS);
JUDGE_V_MLDPS(Vcc, 2.9V, 3.1V, 2mS, 32, AVE, 50uS, ,);
```

4.3.4.3 Relative

JUDGE_VOLT

4.3.5 MEAS_I_MLDPS

Perform a power supply current measurement and save the measured value to the system buffer. Read the result by the "RESULT_MLDPS" statement. Release version : **CRAFT_3380_2.00_SP1**

4.3.5.1 Format

Parameter list:

```
MEAS_I_MLDPS(MLDPS_name, wait_time[, ave_cnt, ave_mode, sampling_interval] ) ;
```

// driver description : 1. delay "wait_time"
2. iteration "ave_time" to measure current and do average

Parameter comment:

MLDPS_name: Define in *.dec file
MLDPS channels are defined by slot number.
EX:
Slot8 chanenl 512~ 543
Slot9 (default MLDPS SLOT) 576~607
And Type declared " MLDPS" in declaration file.
Type: integer
No entry: illegal

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 400uS
// Notice : if use average mode, then meas_type should use NORM mode,
otherwise compiler error

ave_cnt: The average count.
No entry: Default = 1

ave_mode: The average mode.

AVE: average
RMS: root mean square
RMS_X : root mean square; //calculation formula is different from "RMS"; Calibration table import to hw-driver-function (not in fw-driver-function)
(RMS_X Release version : CRAFT_3380_2.50-SP3)
Type: int
No entry: Default = AVE

sampling_interval: The sampling interval is between each measurement.
// H/W ADC convert-time = 5uS ;
**no_entry: Default = 50uS ; minimum value >5uS.

4.3.5.2 Example

```
idd_static_test()
{
        double imeas;
        FORCE_V_MLDPS(Vcc, 6V, @12V, @25uA, 25uA, NORM, ON, 2mS);
        SET_LEVELN(com_lvl, IO_ALLPINS, 0V, 4.5V,1V,3V,0uA,0uA,3.0V, , );
        LEVELS(com_lvl, 2mS);
        RELAY_ON(IO_ALLPINS,PDCL, 1mS);
        TEST_NO(32);
        JUDGE_PAT(static_idd_st, static_idd_sp );
        TEST_NO(33);
        MEAS_I_MLDPS (Vcc, 1mS, 16, AVE, 50uS);
       imeas = RESULT_MLDPS(S0, Vcc);
        JUDGE_VARIABLE (S0, imeas, 1uA, 10uA, "idd_static_test");
         ............
}
```

4.3.5.3 Relative

RESULT_MLDPS , JUDGE_VARIABLE

4.3.6 MEAS_V_MLDPS

Perform a power supply voltage measurement. Save the measured value to the system buffer. Read the result by the "RESULT_MLDPS" statement. Release version : **CRAFT_3380_2.00_SP1**

4.3.6.1 Format

Parameter list:

```
MEAS_V_MLDPS(MLDPS_name, wait_time[, ave_cnt, ave_mode, sampling_interval]) ;
```

// driver description : 1. delay "wait_time"
2. iteration "ave_time" to measure voltage and do average

Parameter comment:

MLDPS_name: Define in *.dec file
MLDPS channels are defined by slot number.
EX:
Slot8 chanenl 512~ 543
Slot9 (default MLDPS SLOT) 576~607
And Type declared " MLDPS" in declaration file.
Type: integer
No entry: illegal

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 400uS

// Notice : if use average mode, then meas_type should use NORM mode,
otherwise compiler error

ave_cnt: The average count.
No entry: Default = 1

ave_mode: The average mode.
AVE: average
RMS: root mean square
RMS_X : root mean square; //calculation formula is different from "RMS"; Calibration table import to hw-driver-function (not in fw-driver-function)
(RMS_X Release version : CRAFT_3380_2.50-SP3)
Type: int
No entry: Default = AVE

sampling_interval: The sampling interval is between each measurement.
// H/W ADC convert-time = 5uS ;
**no_entry: Default = 50uS ; minimum value >5uS.

4.3.6.2 Example

```
//MEAS_V_MLDPS(Vdd, 1mS, 32, AVE, 50uS);
//MEAS_V_MLDPS(Vee, 1mS);
MEAS_V_MLDPS(Vdd+Vee, 1mS);
vmeas = RESULT_MLDPS(S0, Vdd);
JUDGE_VARIABLE (S0, imeas, 1uA, 10uA, "rst");
```

4.3.6.3 Relative

RESULT_MLDPS , JUDGE_VARIABLE

4.3.7 RESULT_MLDPS

The statement extracts the result (measured value) after the MEAS_I_MLDPS/MEAS_V_MLDPS statement executed. **Release version :** CRAFT_3380_2.00_SP1

4.3.7.1 Format

Parameter list:

```
Value =RESULT_MLDPS(socket_no, MLDPS_name);
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

MLDPS_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

4.3.7.2 Example

```
double   v_meas_value;
v_meas_value=RESULT_MLDPS(S0,Vdd);
```

4.3.8 MIN_RESULT_MLDPS

The statement extracts the result (measured value) after the MEAS_I_MLDPS/MEAS_V_MLDPS statement executed. **Release version :** CRAFT_3380_2.00_SP1

4.3.8.1 Format

Parameter list:

```
Value =MIN_RESULT_MLDPS(socket_no,MLDPS_name);
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

MLDPS_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

4.3.8.2 Example

```
double   v_meas_min_value;
v_meas_min_value=MIN_RESULT_MLDPS(S0,Vdd);
```

4.3.9 MAX_RESULT_MLDPS

The statement extracts the result (measured value) after the MEAS_I_MLDPS/MEAS_V_MLDPS statement executed. **Release version :** CRAFT_3380_2.00_SP1

4.3.9.1 Format

Parameter list:

```
Value =MAX_RESULT_MLDPS(socket_no, MLDPS_name);
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

MLDPS_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

4.3.9.2 Example

```
double   v_meas_max_value;
v_meas_max_value=MAX_RESULT_MLDPS(S0,Vdd);
```

4.3.10 RESULT_MLDPS_MS

The statement extracts the result (measured value) after the MEAS_I_MLDPS/MEAS_V_MLDPS statement executed. **Release version :** CRAFT_3380_2.00_SP1

4.3.10.1 Format

Parameter list:

```
RESULT_MLDPS_MS(MLDPS_name, value_ms);
```

Parameter comment:

MLDPS_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

value_ms : Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value
Type: array

4.3.10.2 Example

```
double   value_ms[4];   //suppose 4-site
double   high_limit, low_limit;

RESULT_MLDPS_MS( Vdd, value_ms );
JUDGE_VARIABLE_MS(value_ms, high_limit, low_limit, "vdd test");
```

4.3.11 MIN_RESULT_MLDPS_MS

The statement extracts the minimum result (measured value) after the MEAS_I_MLDPS/MEAS_V_MLDPS statement executed. Release version : **CRAFT_3380_2.00_SP1**

4.3.11.1 Format

Parameter list:

```
MIN_RESULT_MLDPS_MS(MLDPS_name, value_ms ) ;
```

Parameter comment:

MLDPS_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

value_ms: Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value
Type: array

4.3.11.2 Example

```
double   min_value [4];   //suppose 4-site
double   high_limit, low_limit;

MIN_RESULT_MLDPS_MS( Vdd, min_value );
JUDGE_VARIABLE_MS(min_value, high_limit, low_limit, "vdd min test");
```

4.3.12 MAX_RESULT_MLDPS_MS

The statement extracts the maximun result (measured value) after the MEAS_I_MLDPS/MEAS_V_MLDPS statement executed. Release version : **CRAFT_3380_2.00_SP1**

4.3.12.1 Format

Parameter list:

```
MAX_RESULT_MLDPS_MS(MLDPS_name, value_ms ) ;
```

Parameter comment:

MLDPS_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

value_ms: Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value

Type: array

4.3.12.2 Example

```
double   value_ms[4];   //suppose 4-site
double   high_limit, low_limit;

MAX_RESULT_MLDPS_MS( Vdd, value_ms );
JUDGE_VARIABLE_MS(value_ms, high_limit, low_limit, "vdd max test");
```

4.3.13 RESULT_MLDPS_EX

The statement extracts the result (measured value) after the MEAS_I_MLDPS/MEAS_V_MLDPS statement executed. **Release version : CRAFT_3380_2.40_SP7**

4.3.13.1 Format

Parameter list:

```
Value = RESULT_MLDPS_EX(socket_no, DPS_name ) ;
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
Type : integer
**no_entry: illegal

DPS_name : pin-def
variables & constants allowed

type: interger

**no_entry: illegal

4.3.13.2 Example

```
double vdd_value;
   FORCE_I_MLDPS(Vdd, -200uA, @25mA, @6V, 2V, NORM, ON, 2mS);
   MEAS_V_MLDPS(Vdd, 1mS, 32, AVE, 50uS)

   SOCKET_INC(FRZ_OFF)
   {
        vdd_value = RESULT_MLDPS_EX(SOCKET, Vdd);
        JUDGE_VARIABLE (SOCKET,   vdd_value, -1.2V, -0.2V, "RESULT_MLDPS_EX ");

        vdd_value = MIN_RESULT_MLDPS_EX(SOCKET, Vdd);
        JUDGE_VARIABLE (SOCKET,   vdd_value, -1.2V, -0.2V, "MIN_RESULT_MLDPS_EX ");

        vdd_value = MAX_RESULT_MLDPS_EX(SOCKET, Vdd);
```

```
        JUDGE_VARIABLE (SOCKET,    vdd_value, -1.2V, -0.2V, "MAX_RESULT_MLDPS_EX ");
    }
```

4.3.14 MIN_RESULT_MLDPS_EX

The statement extracts the result (measured value) after the MEAS_I_MLDPS/MEAS_V_MLDPS statement executed. **Release version : CRAFT_3380_2.40_SP7**

4.3.14.1 Format

Parameter list:

```
Value = MIN_RESULT_MLDPS_EX(socket_no, DPS_name ) ;
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
Type : integer
**no_entry: illegal

DPS_name : pin-def
variables & constants allowed

type: interger

**no_entry: illegal

4.3.14.2 Example

```
double vdd_value;
   FORCE_I_MLDPS(Vdd, -200uA, @25mA, @6V, 2V, NORM, ON, 2mS);
   MEAS_V_MLDPS(Vdd, 1mS, 32, AVE, 50uS)

   SOCKET_INC(FRZ_OFF)
   {
        vdd_value = RESULT_MLDPS_EX(SOCKET, Vdd);
        JUDGE_VARIABLE (SOCKET,    vdd_value, -1.2V, -0.2V, "RESULT_MLDPS_EX ");

        vdd_value = MIN_RESULT_MLDPS_EX(SOCKET, Vdd);
        JUDGE_VARIABLE (SOCKET,    vdd_value, -1.2V, -0.2V, "MIN_RESULT_MLDPS_EX ");

        vdd_value = MAX_RESULT_MLDPS_EX(SOCKET, Vdd);
        JUDGE_VARIABLE (SOCKET,    vdd_value, -1.2V, -0.2V, "MAX_RESULT_MLDPS_EX ");
```

```
}
```

4.3.15 MAX_RESULT_MLDPS_EX

The statement extracts the result (measured value) after the MEAS_I_MLDPS/MEAS_V_MLDPS statement executed. **Release version : CRAFT_3380_2.40_SP7**

4.3.15.1 Format

Parameter list:

```
Value = MAX_RESULT_MLDPS_EX(socket_no, DPS_name ) ;
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
Type : integer
**no_entry: illegal

DPS_name : pin-def
variables & constants allowed

type: interger

**no_entry: illegal

4.3.15.2 Example

```
double vdd_value;
  FORCE_I_MLDPS(Vdd, -200uA, @25mA, @6V, 2V, NORM, ON, 2mS);
  MEAS_V_MLDPS(Vdd, 1mS, 32, AVE, 50uS)

  SOCKET_INC(FRZ_OFF)
  {
      vdd_value = RESULT_MLDPS_EX(SOCKET, Vdd);
      JUDGE_VARIABLE (SOCKET,   vdd_value, -1.2V, -0.2V, "RESULT_MLDPS_EX ");

      vdd_value = MIN_RESULT_MLDPS_EX(SOCKET, Vdd);
      JUDGE_VARIABLE (SOCKET,   vdd_value, -1.2V, -0.2V, "MIN_RESULT_MLDPS_EX ");

      vdd_value = MAX_RESULT_MLDPS_EX(SOCKET, Vdd);
      JUDGE_VARIABLE (SOCKET,   vdd_value, -1.2V, -0.2V, "MAX_RESULT_MLDPS_EX ");
  }
```

4.4 MLDPS High Speed Function

Usage limitation :

- Hi-speed function only supports FV、MV、MI，no FI
- Please set force mode、range、clamp value before executing the hi-speed function
- Within a single test plan , there are only 32 pattern modules could carry on HS_MLPDS Micro-instruction. And each pattern module only has 32 PMAs(pattern memory address) could use HS_MLPDS Micro-instruction.

4.4.1 LOAD_VI_WAVEFORM (Statement)

Load the VI board waveform data into the hardware waveform memory.

4.4.1.1 Format

```
LOAD_VI_WAVEFORM( pin_name, MLDPS
[ _RAMP, force_type, @range,   start_value,   stop_value,   steps , vi_waveform_label    ]
[ _ARRAY, force_type, @range,   data_array[],   data_cnt, vi_waveform_label   ]
```

Parameter list:

pin_name: pin_def, pin_group
**no_entry; illegal

force_type: **_FORCE_V** → force voltage
**no_entry: illegal

vi_waveform_label: to represent the loaded-waveform
type : string
**no_entry: illegal

@range: FV range : @6V, @12V
**no_entry: illegal

start_value: RAMP waveform start value
type: double
**no_entry: illegal

stop_value: RAMP waveform stop value
type: double
**no_entry: illegal

steps: RAMP waveform total steps (include begin/end point)

type: int
**no_entry: illegal

4.4.1.2 Explanation

Each channel waveform memory size : total 1024
(include micro-instruction “MLDPS_SET_VOLT” setting value)

usage notice:

- Maximum number of waveform label is 16.
- If reload the same label , system will confirm if the numbers of waveform point is over the occupied one. If over , system will pop-up error message.

4.4.1.3 Example

```
double fv_data[6] = { 6.0, 5.0, 4.0, 3.0, 2.0, 1.0 };
int    read_length= 6;
CUSTOM_3DIM Cap_Current;      // Cap_Current[site_no][pin_count][cap_length]

LOAD_VI_WAVEFORM(Vmlp00, MLDPS, _ARRAY, _FORCE_V, @6V, fv_data, 6,
"waveform_0");
LOAD_VI_WAVEFORM(Vmlp00, MLDPS, _RAMP,    _FORCE_V, @6V, 0.0, 5.0, 16, "
waveform_1");
…
SET_VI_WAVEFORM(Vmlp00, MLDPS, "waveform_0");
…
JUDGE_PAT(mldps_HS_MVMI_pat_st, mldps_HS_MVMI_pat_sp);

//----- Read back capture memory -----//
READ_VI_CAPTURE_MEM_BLOCK(Vmlp00, MLDPS, 0, read_length, Cap_Current);

for(int j=0; j< read_length; j++)
{
      JUDGE_VARIABLE (0, Cap_Current[0][0][j], 2.0e-3, 10.0e-3, “Idd”);
}

SPM_PATTERN (mldps_HS_MVMI_pat) {
     mldps_HS_MVMI_pat_st::
                              *00000000* TS1;
                              *11111111* ;
                              *00000000* SETX0 6;
                              *00000000* ;
                              *00000000* ;
                    AA:       *11111111* TS2, MLDPS_FORCE_INC(Vmlp00);
                              *00000000* RPT 15;
                              *11111111* MLDPS_MEAS(Vmlp00,1);
                              *00000000* RPT 15;
                              *11111111* MLDPS_MEAS(Vmlp04,1);
                              *00000000* RPT 2;
                              *00000000* ;
                              *00000000* JNZ0 AA;

     mldps_HS_MVMI_pat_sp::
                              *XXXXXXXX* ;
}
```

4.4.1.4 Relative

SET_VI_WAVEFORM , CLEAR_VI_WAVEFORM

4.4.2 CLEAR_VI_WAVEFORM (Statement)

Clear the VI board waveform memory by specified waveform-label.

4.4.2.1 Format

```
CLEAR_VI_WAVEFORM( pin_name, MLDPS  [ , vi_waveform_label ] ) ;
```

vi_waveform_label: type : string
**no_entry: to clear the pin all waveform label

4.4.2.2 Explanation

The statement will clear vi-memory to 0.0V, and release the memory size for the other application. (not include micro-instruction “MLDPS_SET_VOLT” setting value)

4.4.2.3 Example

```
LOAD_VI_WAVEFORM(VMLP_1, MLDPS, _RAMP,   _FORCE_V, @6V, 0.0,   5.0, 10,
"Test01");
…
…
CLEAR_VI_WAVEFORM( VMLP_1, MLDPS , "Test01") ;
LOAD_VI_WAVEFORM(VMLP_1, MLDPS, _RAMP,   _FORCE_V, @6V, 0.0,   3.3, 10,
"Test01");
```

4.4.2.4 Relative

4.4.3 SET_VI_WAVEFORM (Statement)

Set the wavefrom data address point to specified waveform-label.

4.4.3.1 Format

```
SET_VI_WAVEFORM( pin_name, MLDPS , vi_waveform_label [ , offset ]);
```

offset : offset value ;
type : int
**no_entry: default 0
// final wavefrom data address = “vi_waveform_label” + offset;

4.4.3.2 Explanation

4.4.3.3 Example

```
LOAD_VI_WAVEFORM(Vmlp00, MLDPS, _ARRAY, _FORCE_V, @6V, fv_data, 6,
"waveform_0");
…
SET_VI_WAVEFORM(Vmlp00, MLDPS, "waveform_0");
…
JUDGE_PAT(mldps_HS_MVMI_pat_st, mldps_HS_MVMI_pat_sp);
```

4.4.3.4 Relative

LOAD_VI_WAVEFORM

4.4.4 READ_VI_CAPTURE_MEM_BLOCK (Statement)

Read VI capture memory block.

4.4.4.1 Format

READ_VI_CAPTURE_MEM_BLOCK(pin_name, **MLDPS** , start_address, read_length, return_vi_cap_data) ;

pin_name; pin_def, pin_group,
**no_entry: illegal

start_address :
MLDPS: 0 ~ 32K
type: integer
**no_entry: illegal

read_length : //MLDPS capture memory: per-channel 32K
MLDPS: 0 ~ 32K
type: integer
**no_entry: illegal

return_vi_cap_data :
type: **CUSTOM_3DIM** ;
//CRAFT auot-allocate array size [site_no][pin_count][read_length];
**no_entry: illegal

4.4.4.2 Explanation

While running every JUDGE_PAT, capture memory will write data from 0.
Each channel capture memory size : 32K

4.4.4.3 Example

```
double fv_data[6] = { 6.0, 5.0, 4.0, 3.0, 2.0, 1.0 };
int    cap_length= 6;
CUSTOM_3DIM Cap_Current;      // Cap_Current[site_no][pin_count][read_length]


LOAD_VI_WAVEFORM(Vmlp00, MLDPS, _ARRAY, _FORCE_V, @6V, fv_data, 6,
"waveform_0");
LOAD_VI_WAVEFORM(Vmlp00, MLDPS, _RAMP,    _FORCE_V, @6V, 0.0, 5.0, 16, "
waveform_1");
…
SET_VI_WAVEFORM(Vmlp00, MLDPS, "waveform_0");
…
JUDGE_PAT(mldps_HS_MVMI_pat_st, mldps_HS_MVMI_pat_sp);

//----- Read back capture memory -----//
READ_VI_CAPTURE_MEM_BLOCK(Vmlp00, MLDPS, 0, read_length, Cap_Current);

for(int j=0; j< read_length; j++)
{
      JUDGE_VARIABLE (0, Cap_Current[0][0][j], 2.0e-3, 10.0e-3, “Idd”);
}
```

4.4.4.4 Relative

4.4.5 MLDPS_FORCE_INC (Micro-instruction)

The micro-instruction force voltage one point by vector rate running.

4.4.5.1 Format

```
MLDPS_FORCE_INC( pin_name );
```

4.4.5.2 Explanation

MLDPS need to have sufficient execution time between two HS micro-instruction; otherwise run-time will pop-up error message.

HS CMD	Execute Time
MLDPS_FORCE_INC	5uS
MLDPS_MEAS	5uS
MLDPS_SET_VOLT	5uS

//the Execute-Time of MLDPS_FORCE_INC not equal settling time;

4.4.5.3 Example

```
SPM_PATTERN (mldps_HS_FV_pat) {
    mldps_HS_FV_pat_st::

                        *00000000* TS2;
                        *00000000* ;
                        *11111111* TS2,MLDPS_FORCE_INC(Vmlp00);
                        *00000000* TS1;
                        *11111111*      MLDPS_FORCE_INC(Vmlp00);
                        *00000000* ;
                        *11111111*      MLDPS_FORCE_INC(Vmlp00);
                        *00000000* ;
                        *11111111*      MLDPS_FORCE_INC(Vmlp00);
                        *00000000* ;
                        *11111111*      MLDPS_FORCE_INC(Vmlp00);
                        *00000000* ;
                        *11111111*      MLDPS_FORCE_INC(Vmlp00);
                        *00000000* ;
                        *11111111* ;
      mldps_HS_FV_pat_sp::
                        *00000000* ;
}
```

4.4.5.4 Relative

4.4.6 MLDPS_MEAS (Micro-instruction)

The micro-instruction will trigger measurement-event once by vector rate running. Measure-V or Measure-I is programmed by FORCE-statement.

4.4.6.1 Format

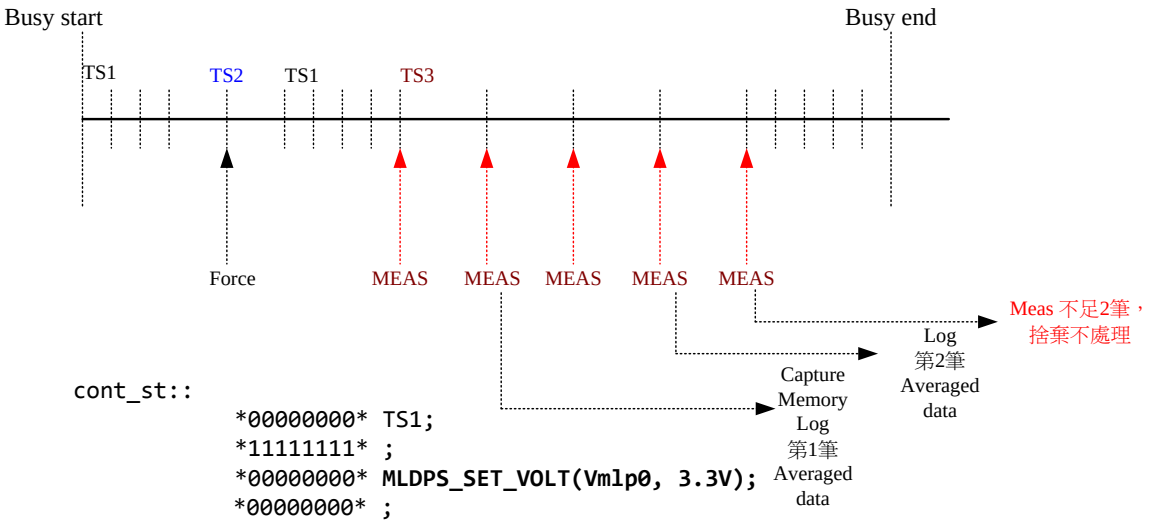
```
MLDPS_MEAS( pin_name, average_cnt);
```

average_cnt: average_count; must 2^n (1,2,4,8,…1024)
type: integer
maximum value : 1024
**no_entry: illegal

4.4.6.2 Explanation

MLDPS need to have sufficient execution time between two HS micro-instruction; otherwise run-time will pop-up error message.

HS CMD	Excute Time
MLDPS_FORCE_INC	5uS
MLDPS_MEAS	5uS
MLDPS_SET_VOLT	5uS




```
cont_st::
            *00000000* TS1;
            *11111111* ;
            *00000000* MLDPS_SET_VOLT(Vmlp0, 3.3V);
            *00000000* ;
            *00000000* TS2, MLDPS_FORCE_INC(Vmlp0);
        Aa: *11111111* TS1;
            *00000000* ;
            *11111111* ;
            *0K000000* ;
            *11111111* TS3, MLDPS_MEAS(Vmlp0,2), RPT 5; //Measure
            *00000000* ;
            *11111111* ;
            *00000000* ;
            *11111111* ;
cont_sp::
            *00000000* ;
```

4.4.6.3 Example

```
SPM_PATTERN (mldps_HS_MIpp_pat) {
    mldps_HS_MIpp_pat_st::
                        *00000000* TS2; //10uS
                        *00000000* ;

                        *11111111* SETX0 1024;
                        *00000000* ;
            AA:         *11111111* MLDPS_MEAS(VMLP_1, 1);
                        *00000000* RPT 18;
                        *11111111* JNZ0 AA;
     mldps_HS_MIpp_pat_sp::
                        *00000000* ;
}
```

4.4.6.4 Relative

4.4.7 MLDPS_SET_VOLT (Micro-instruction)

The micro-instruction set voltage directly in pattern file.
(setup force-voltage to waveform memory)

4.4.7.1 Format

```
MLDPS_SET_VOLT( pin_name, voltage_value);
```

Voltage_value: type: double
variable not allowed.
**no_entry: illegal

4.4.7.2 Explanation

MLDPS need to have sufficient execution time between two HS micro-instruction; otherwise run-time will pop-up error message.

HS CMD	Execute Time
MLDPS_FORCE_INC	5uS
MLDPS_MEAS	5uS
MLDPS_SET_VOLT	5uS

Each MLDPS_SET_VOLT needs one waveform memory address even though the voltage value is the same . User should use either MLDPS_SET_VOLT or SET_VI_WAVEFORM.

4.4.7.3 Example

```
SPM_PATTERN (mldps_HS_SETV_pat) {
    mldps_HS_SETV_pat_st::
                        *00000000* TS1;
                        *00000000* ;
                        *11111111* ;
                        *00000000* ;
                        *11111111* MLDPS_SET_VOLT(Vmlp00,5.0);
                        *00000000* ;
                        *11111111* MLDPS_FORCE_INC(Vmlp00);
                        *00000000* ;
                        *11111111* ;
      mldps_HS_SETV_pat_sp::
                        *00000000* ;
}
```

4.4.7.4 Relative

MLDPS_FORCE_INC , MLDPS_MEAS

4.5 MXDPS Control Statement(Option)

4.5.1 FORCE_I_DPS

This statement sets the values of current running at the power source during the execution of the function test.

4.5.1.1 Format

```
FORCE_I_DPS (DPS_name, f_curent, i_range, v_range, v_clamp, m_mode,
             ON/OFF, wait_time) ;
```

Parameter list:

DPS_name: Define in *.dec file
DPS channels are defined by slot number.
EX:
Slot8 channel 512~ 519
Slot9 (default DPS SLOT) 576~583
And Type declared "DPS" in declaration file.

Both variables and constants are allowed.
Type: integer
No entry: illegal

f_curent: The amount of force current from -2A to +2A by the selected DPS.
Both variables and constants are allowed.
Type: CURRENT or double
Unit: A, pA, nA, uA, mA
No entry: keep current value

I_range	Minimum	Maximum
@100uA	-100uA	+100uA
@1mA	-1mA	+1mA
@10mA	-10mA	+10mA
@100mA	-100mA	+100mA
@1A	-1A	+1A
@2A:	-2A	+2A

I_range: One of eight possible hardware ranges. For the best accuracy, you should program the current by using the closest range possible.

- Possible hardware range:

I_range	Minimum	Maximum
@100uA	-100uA	+100uA
@1mA	-1mA	+1mA

@10mA	-10mA	+10mA
@100mA	-100mA	+100mA
@1A	-1A	+1A
@2A:	-2A	+2A

V_range: No entry: keep current range

I_range	Minimum	Maximum
@4V	-4V	+4V
@8V	-8V	+8V
@12V	-12V	+12V
@16V	-16V	+16V

V_clamp:This value must be in the specified V_range, as following:

I_range	Minimum	Maximum
@4V	-4V	+4V
@8V	-8V	+8V
@12V	-8V	+12V
@16V	-8V	+16V

Variables and constants are allowed.
Type: VOLTAGE or double
Unit: V, pV, nV, uV, mV
No entry: keep current value

ON/OFF: ON – turns on DPS by marking DPS utility relay after conditions setup.
OFF – turns off DPS by breaking DPS utility relay after conditions setup.
Type: integer
No entry: keep voltage state

[Notice]: Force ON and Force OFF procedure
Force ON : Relay closes first and force setting value
Force OFF : Force setting value first and Relay open

m_mode : Specify the DPS measure mode ;
NORM (CPU trigger)
No entry: Default = NORM

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 400uS

4.5.1.2 Explanation

The FORCE_I_DPS statement sets the selected Devices Power Supply (DPS) to the current force mode, and programs the amount of current to be forced.

4.5.1.3 Example

```
FORCE_I_DPS (Vcc, 1mA, @10mA, @8V, 8V, NORM, ON, 2mS);
```

4.5.1.4 Relative

JUDGE_V_DPS

4.5.2 FORCE_V_DPS

This statement sets the selected DPS to the programmed voltage condition as well as ranges and clamp currents.

4.5.2.1 Format

Parameter list:

```
FORCE_V_DPS(DPS_name, f_volt, v_range, i_range, I_clamp, m_mode,
                    ON/OFF, wait_time [, slew_rate ]) ;
```

Parameter comment:

DPS_name: Define in *.dec file
DPS channels are defined by slot number.
EX:
Slot8 chanenl 512~ 519
Slot9 (default DPS SLOT) 576~583
And Type declared "DPS" in declaration file.
Type: integer
No entry: illegal

f_volt: The voltage to be applied by the selected DPS. This is a value that can be Programmed from -8V to +16V.
Variable is allowed.
Type: VOLTAGE or double
Constant values must be followed by the unit 'V', pV, nV, uV, mV
No entry: keep current value.

I_range	Minimum	Maximum
@4V	-4V	+4V
@8V	-8V	+8V
@12V	-8V	+12V
@16V	-8V	+16V

V_range: No entry: keep current range

I_range	Minimum	Maximum
@4V	-4V	+4V
@8V	-8V	+8V
@12V	-8V	+12V
@16V	-8V	+16V

I_range: One of eight possible hardware ranges. For the best accuracy, you should program the voltage by using the closest range possible.

- Possible hardware range:

I_range	Minimum	Maximum
@1uA	-1uA	+1uA
@10uA	-10uA	+10uA
@100uA	-100uA	+100uA
@1mA	-1mA	+1mA
@10mA	-10mA	+10mA
@100mA	-100mA	+100mA
@1A	-1A	+1A
@2A:	-2A	+2A

Type: integer
No entry: keep current range

I_clamp: The maximum current to be supplied by the selected DPS. One of eight possible Hardware ranges. This value must be in the specified I_range, as following:

- Possible hardware range:

I_calmp	Minimum	Maximum
@1uA	50uA	100uA
@10uA	50uA	100uA
@100uA	0A	100uA
@1mA	0A	1mA
@10mA	0A	10mA
@100mA	0A	100mA
@1A	0A	1A
@2A:	0A	2A

Variables are allowed. Constant values must be followed by the unit 'A', which can be preceded by the prefixes A, pA, nA, uA, mA.
Type: CURRENT or double
No entry: set as the maximum value of the present I_range

m_mode: Select the mode.
NORM (CPU trigger)
No entry: NORM

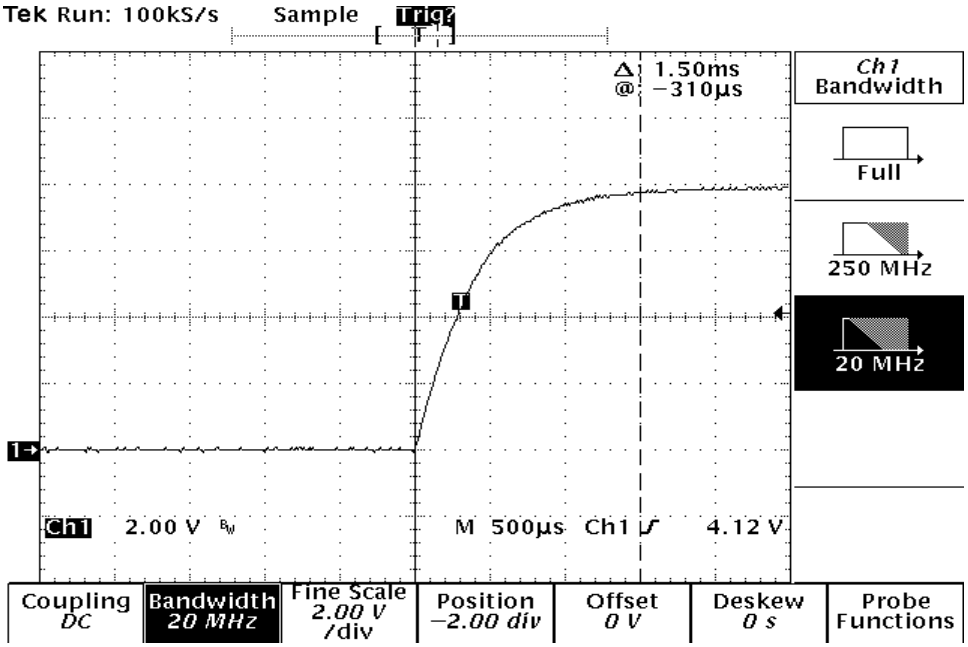
ON/OFF: The state of the specified DPS output buffer. There are two possible states:
ON – turns on DPS by marking DPS utility relay after conditions setup.
OFF – turns off DPS by breaking DPS utility relay after conditions setup.
Type: integer
No entry: keep current state
[Notice]: Force ON and Force OFF procedure
Force ON : Relay closes first and force setting value
Force OFF : Force setting value first and Relay open

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
**no_entry: Default relay ON Delay = 500uS

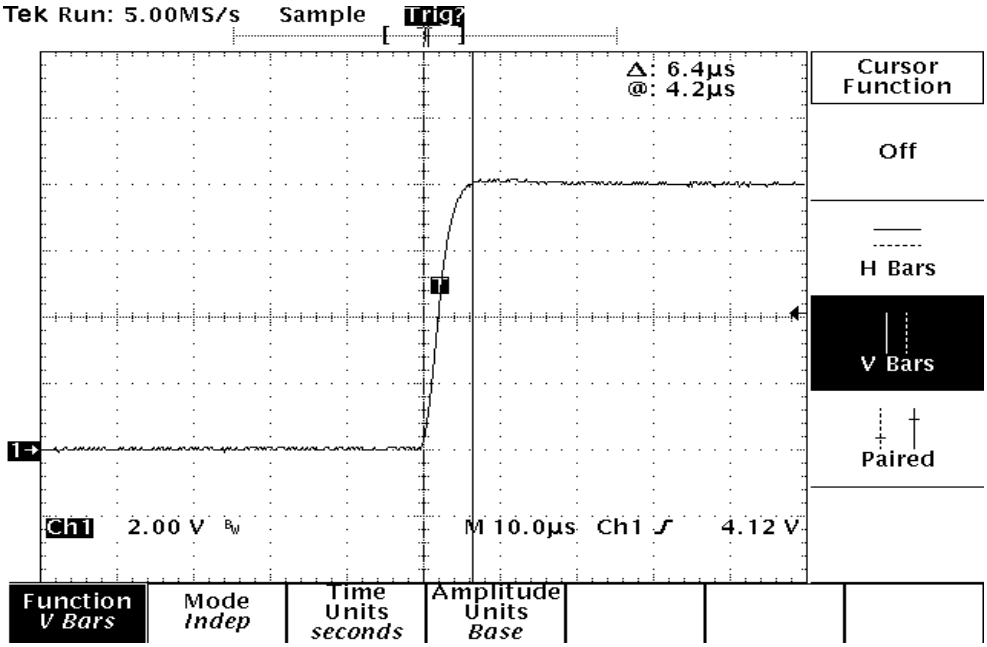
Default relay OFF Delay : 2mS (to ensure the discharge ok)

slew_rate: FAST
NORM
**no_entry: NORM (system default)

** slew_rate = NORM waveform



** slew_rate = FAST waveform



4.5.2.2 Explanation

This statement sets the selected DPS to the programmed voltage condition, as well as ranges and clamp currents.

4.5.2.3 Example

```
idd_static_test()
{
	DATALOG_MSG="idd_static_test";
	SET_LEVELN(com_lvl, IO_ALLPINS, 0V, 4.5V,1V,3V,0uA,0uA,3.0V, , );
	LEVELS(com_lvl, 2mS);
	FORCE_V_DPS(Vcc, 6V, @8V, @100mA, 100mA, NORM, ON, 2mS);
	RELAY_ON(IO_ALLPINS,PDCL, 1mS);
	TEST_NO(32);
	JUDGE_PAT(static_idd_st, static_idd_sp );
	TEST_NO(33);
	JUDGE_I_DPS(Vcc,0uA , 80uA, 2mS);
	POWER_DOWN_FAIL_SITE( );
}
```

4.5.2.4 Relative

JUDGE_I_DPS

4.5.3 JUDGE_I_DPS

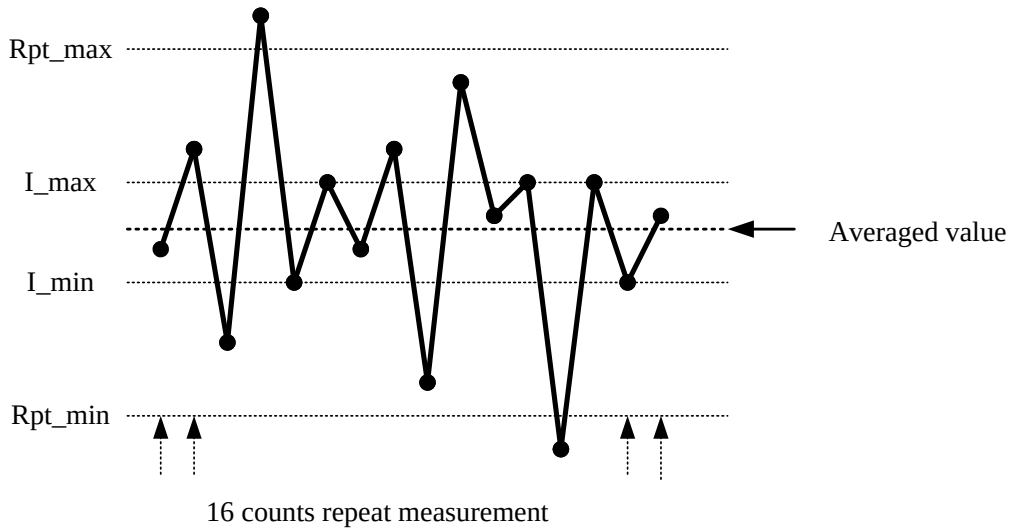
Perform a DPS current measurement and compare the value to the specified limits.

4.5.3.1 Format

Parameter list:

```
JUDGE_I_DPS (DPS_name, I_min, I_max, wait_time[, ave_cnt, ave_mode,
                    sampling_interval, rpt_min, rpt_max ]) ;
```

// driver description : 1. delay "wait_time"
2. iteration "ave_cnt" to measure current and do average
3. each measured value must between rpt_min and rpt_max,
otherwise the judgement will be set to fail .



Parameter comment:

DPS_name: Define in *.dec file
DPS channels are defined by slot number.
EX:
Slot8 chanenl 512~ 519
Slot9 (default DPS SLOT) 576~583
And Type declared "DPS" in declaration file.
Type: integer
No entry: illegal

I_min: Refer to the minimum current value to generate the pass or fail.
Omitting the parameter is possible, but the comparison will be made only regarding I_max.
When this happens, omitting I_max description is not allowed. This is lower limit value.
Variables and constants are allowed.
Type: CURRENT or double
Unit: A, pA, nA, uA, mA
No entry: Only high limit.

I_max: Refer to the maximum current value to generate the pass or fail. Omitting the parameter is possible, but the comparison will be made only regarding I_min. When this happens, omitting I_min description is not allowed. This is upper limit value.
Variables and constants are allowed.
Type: CURRENT or double
Unit: A, pA, nA, uA, mA
No entry: Only low limit.

wait_time; Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 500uS

ave_cnt : the average count
no_entry: Default = 1

ave_mode : the average mode
AVE : average
RMS : root mean square
type: int
no_entry: Default = AVE

sampling_interval : the sampling interval between each measurement
no_entry: Default = 50uS (minimum value)

rpt_min: the low limit of each measurement ; if any measurement smaller than this value, then the judge result will be set to fail
rpt_max: the high limit of each measurement ; if any measurement larger than this value, then the judge result will be set to fail
if rpt_min & rpt_max no_entry : skip the rpt_min/rpt_max judgement

4.5.3.2 Explanation

This statement generates the pass or fails result by checking whether the current value produced as the test result comes within the scope of the current value setting. When the test result comes within the set scope, a pass result is generated, but when the test result comes outside the set scope, the fail result is generated.

4.5.3.3 Example

```
JUDGE_I_DPS (PVdd, 2mA, 40mA, 2mS);
JUDGE_I_DPS (PVdd, 2mA, 40mA, 2mS);
JUDGE_I_DPS (PVdd, 0uA, 10uA, 2mS, 32, AVE, 50uS, , );
```

```
idd_static_test()
{
	DATALOG_MSG="idd_static_test";
	SET_LEVELN(com_lvl, IO_ALLPINS, 0V, 4.5V,1V,3V,0uA,0uA,3.0V, , );
	LEVELS(com_lvl, 2mS);
	FORCE_V_DPS(Vcc, 6V, @8V, @100mA, 100mA, NORM, ON, 2mS);
	RELAY_ON(IO_ALLPINS,PDCL, 1mS);
	TEST_NO(32);
	JUDGE_PAT(static_idd_st, static_idd_sp );
	TEST_NO(33);
	JUDGE_I_DPS(Vcc,0uA , 80uA, 2mS);
	POWER_DOWN_FAIL_SITE( );
}
```

4.5.3.4 Relative

FORCE_V_DPS

4.5.4 JUDGE_V_DPS

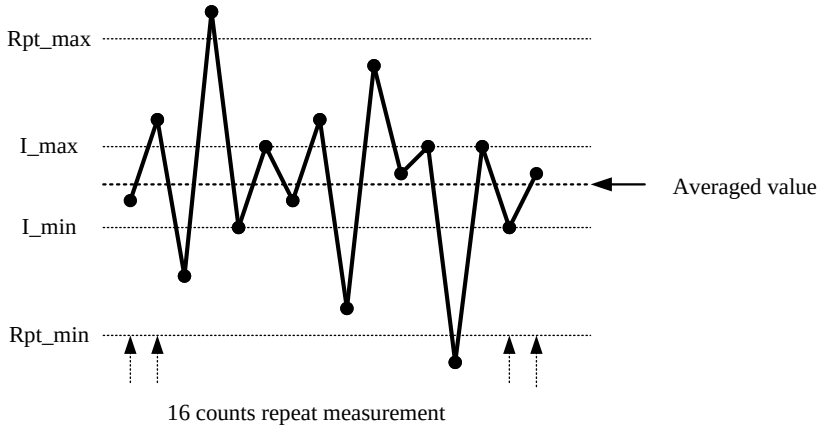
Perform a DPS voltage measurement and compare the value to the specified limits.

4.5.4.1 Format

Parameter list:

```
JUDGE_V_DPS (DPS_name, V_min, V_max, wait_time[, ave_cnt, ave_mode,
                    sampling_interval, rpt_min, rpt_max ]) ;
```

// driver description : 1. delay "wait_time"
2. iteration "ave_cnt" to measure voltage and do average
3. each measured value must between rpt_min and rpt_max,
otherwise the judgement will be set to fail .



Parameter comment:

DPS_name: Define in *.dec file
DPS channels are defined by slot number.
EX:
Slot8 chanenl 512~ 519
Slot9 (default DPS SLOT) 576~583
And Type declared "DPS" in declaration file.
Type: integer
No entry: illegal

v_min: Refer to the minimum voltage to generate the pass or fail. Omitting the parameter is possible, but the comparison will be made only regarding I_max.
When this happens, omitting I_max description is not allowed. This is lower limit value. Variables and constants are allowed.
Type: VOLTAGE or double
Unit: V, mV
No entry: only high limit

v_max: Refer to the maximum voltage value to generate the pass or fail. Omitting the parameter is possible, but the comparison will be made only regarding I_min.

When this happens, omitting I_min description is not allowed. This is the upper limit value. Variables and constants are allowed.
Type: VOLTAGE or double

Unit: V, mV
No entry: Only low limit.

wait_time; Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 500uS

ave_cnt: The average count.
No entry: Default = 1

ave_mode: The average mode
AVE: average
RMS: root mean square
Type: int
No entry: Default = AVE

sampling_interval: the sampling interval is between each measurement.
**no_entry: Default = 50uS (minimum value).

rpt_min: the low limit of each measurement ; if any measurement smaller than this value, then the judge result will be set to fail

rpt_max: the high limit of each measurement ; if any measurement larger than this value, then the judge result will be set to fail
if rpt_min & rpt_max no_entry : skip the rpt_min/rpt_max judgement

4.5.4.2 Explanation

Perform a DPS voltage measurement and compare the value to the specified limits. Judges PASS or FAIL by comparing the data with the upper and lower limits.

4.5.4.3 Example

```
JUDGE_V_DPS (Vdd , 2V, 3V, 2mS, 32, AVE, 50uS, ,);
JUDGE_V_DPS (Vee, 2V, 3V, 2mS);
JUDGE_V_DPS (Vdd+Vee , 2V, 3V, 2mS);
```

4.5.4.4 Relative

FORCE_I_DPS

4.5.5 MEAS_I_DPS

Perform a power supply current measurement and save the measured value to the system buffer. Read the result by the "RESULT_DPS" statement.

4.5.5.1 Format

Parameter list:

MEAS_I_DPS (DPS_name, wait_time**[**, ave_cnt, ave_mode, sampling_interval**]**) ;

// driver description : 1. delay "wait_time"
2. iteration "ave_time" to measure current and do average

Parameter comment:

DPS_name: Define in *.dec file
DPS channels are defined by slot number.
EX:
Slot8 chanenl 512~ 519
Slot9 (default DPS SLOT) 576~583
And Type declared "DPS" in declaration file.
Type: integer
No entry: illegal

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 400uS
// Notice : if use average mode, then meas_type should use NORM mode, otherwise compiler error

ave_cnt: The average count.
No entry: Default = 1

ave_mode: The average mode.
AVE: average
RMS: root mean square
Type: int
No entry: Default = AVE

sampling_interval: The sampling interval is between each measurement.
// H/W ADC convert-time = 50uS ;
**no_entry: Default = 50uS (minimum value).

4.5.5.2 Explanation

- If use average mode, then meas_type should use NORM mode, otherwise compiler will be error.

- Perform a power supply current measurement and save the measured value to system buffer. Read the result by the "RESULT_DPS" statement.

4.5.5.3 Example

```
idd_static_test()
{
        double imeas;
        FORCE_V_DPS(Vcc, 6V, @8V, @100mA, 100mA, NORM, ON, 2mS);
        SET_LEVELN(com_lvl, IO_ALLPINS, 0V, 4.5V,1V,3V,0uA,0uA,3.0V, , );
        LEVELS(com_lvl, 2mS);
        RELAY_ON(IO_ALLPINS,PDCL, 1mS);
        TEST_NO(32);
        JUDGE_PAT(static_idd_st, static_idd_sp );
        TEST_NO(33);
        MEAS_I_DPS (Vcc, 1mS, 16, AVE, 50uS);
       imeas = RESULT_DPS(S0, Vcc);
        JUDGE_VARIABLE (S0, imeas, 1uA, 10uA, "idd_static_test");
         ............
}
```

4.5.5.4 Relative

RESULT_DPS , JUDGE_VARIABLE

4.5.6 MEAS_V_DPS

Perform a power supply voltage measurement. Save the measured value to the system buffer. Read the result by the "RESULT_DPS" statement.

4.5.6.1 Format

Parameter list:

MEAS_V_DPS (DPS_name, wait_time**[**, ave_cnt, ave_mode, sampling_interval**]**) ;

// driver description : 1. delay "wait_time"
2. iteration "ave_time" to measure voltage and do average

Parameter comment:

DPS_name: Define in *.dec file
DPS channels are defined by slot number.
EX:
Slot8 chanenl 512~ 519
Slot9 (default DPS SLOT) 576~583
And Type declared "DPS" in declaration file.
Type: integer
No entry: illegal

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 400uS
// Notice : if use average mode, then meas_type should use NORM mode,
otherwise compiler error

ave_cnt: The average count.
No entry: Default = 1

ave_mode: The average mode.
AVE: average
RMS: root mean square
Type: int
No entry: Default = AVE

sampling_interval: The sampling interval is between each measurement.
// H/W ADC convert-time = 50uS ;
**no_entry: Default = 50uS (minimum value)

4.5.6.2 Explanation

- If use average mode, then meas_type should use NORM mode, otherwise compiler will be error

- Perform a power supply voltage measurement. Save the measured value to system buffer. Read the result by the "RESULT_DPS" statement.

4.5.6.3 Example

```
//MEAS_V_DPS(Vdd, 1mS, 32, AVE, 50uS);
//MEAS_V_DPS(Vee, 1mS);
MEAS_V_DPS(Vdd+Vee, 1mS);
vmeas = RESULT_DPS(S0, Vdd);
JUDGE_VARIABLE (S0, imeas, 1uA, 10uA, "rst");
```

4.5.6.4 Relative

RESULT_DPS , JUDGE_VARIABLE

4.5.7 RESULT_DPS

The statement extracts the result (measured value) after the MEAS_I_DPS/MEAS_V_DPS statement executed.

4.5.7.1 Format

Parameter list:

```
Value =RESULT_DPS(socket_no, DPS_name);
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

DPS_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

4.5.7.2 Example

```
double   v_meas_value;
v_meas_value=RESULT_DPS(S0,Vdd);
```

4.5.8 MIN_RESULT_DPS

The statement extracts the result (measured value) after the MEAS_I_DPS/MEAS_V_DPS statement executed.

4.5.8.1 Format

Parameter list:

```
Value =MIN_RESULT_DPS(socket_no, DPS_name);
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

DPS_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

4.5.8.2 Example

```
double    v_meas_min_value;
v_meas_min_value=MIN_RESULT_DPS(S0,Vdd);
```

4.5.9 MAX_RESULT_DPS

The statement extracts the result (measured value) after the MEAS_I_DPS/MEAS_V_DPS statement executed.

4.5.9.1 Format

Parameter list:

```
Value =MAX_RESULT_DPS(socket_no, DPS_name);
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

DPS_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

4.5.9.2 Example

```
double   v_meas_max_value;
v_meas_max_value=MAX_RESULT_DPS(S0,Vdd);
```

4.5.10 RESULT_DPS_MS

The statement extracts the result (measured value) after the MEAS_I_DPS/MEAS_V_DPS statement executed.

4.5.10.1 Format

Parameter list:

```
RESULT_DPS_MS(DPS_name, value_ms);
```

Parameter comment:

DPS_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

value_ms : Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value
Type: array

4.5.10.2 Example

```
double   value_ms[4];   //suppose 4-site
double   high_limit, low_limit;

RESULT_DPS_MS( Vdd, value_ms );
JUDGE_VARIABLE_MS(value_ms, high_limit, low_limit, "vdd test");
```

4.5.11 MIN_RESULT_DPS_MS

The statement extracts the minimum result (measured value) after the MEAS_I_DPS/MEAS_V_DPS statement executed.

4.5.11.1 Format

Parameter list:

```
MIN_RESULT_DPS_MS(DPS_name, value_ms ) ;
```

Parameter comment:

DPS_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

value_ms: Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value
Type: array

4.5.11.2 Example

```
double   min_value [4];   //suppose 4-site
double   high_limit, low_limit;

MIN_RESULT_DPS_MS( Vdd, min_value );
JUDGE_VARIABLE_MS(min_value, high_limit, low_limit, "vdd min test");
```

4.5.12 MAX_RESULT_DPS_MS

The statement extracts the maximun result (measured value) after the MEAS_I_DPS/MEAS_V_DPS statement executed.

4.5.12.1 Format

Parameter list:

```
MAX_RESULT_DPS_MS(DPS_name, value_ms ) ;
```

Parameter comment:

DPS_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

value_ms: Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value

Type: array

4.5.12.2 Example

```
double   value_ms[4];   //suppose 4-site
double   high_limit, low_limit;

MAX_RESULT_DPS_MS( Vdd, value_ms );
JUDGE_VARIABLE_MS(value_ms, high_limit, low_limit, "vdd max test");
```

4.5.13 SET_CONST_CURRENT_LOAD

This statement sets the selected MXDPS to the constant current load (non-active).

4.5.13.1 Format

Parameter list:

```
SET_CONST_CURRENT_LOAD(
        [ pin_name,
         MXDPS
        [, _CONST_I, current_value
        , _I_RNG , @100uA | @1mA | @10mA | @100mA | @1A | @2A ]
        [ , _MEAS_TYPE, MV, _V_RNG, @4V |@8V | @12V | @16V]
        [ , _MEAS_TYPE, MI , _I_RNG, @100uA | @1mA | @10mA | @100mA | @1A | @2A ]
        [ , _RELAY, ON | OFF]
        [ , _DELAY, delay_time]
        [,_GDELay, delay_time]
);
```

Parameter comment:

pin_name : pin-def or pin_group name
variables & constants allowed
type: interger
**no_entry: illegal

pin_type_name : **MXDPS**

current_value : **MXDPS** : -2A to +2A
variables & constants allowed
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry: keep current value

I_range: @100uA, @1mA, @10mA,@100mA,@1A , @2A ; (**from range @100uA**)
type: integer
**no_entry: keep current range

V_range: **@4V** |@8V | @12V | @16V
**no_entry: keep current range

ON/OFF: ON – turns on Power by marking Power utility relay after conditions setup
OFF – turns off Power by breaking Power utility relay after conditions setup
type: integer
**no_entry: keep current state

m_type : MV (measure output voltage)
MI (measure output current)
VMM (voltage meter measurement)
**no_entry: Default = MI

delay_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 400uS

4.5.13.2 Explanation

If DUT output positive voltage, (eLOAD sink 1mA), so Measure Current will be -1mA.
If DUT output nagative voltage, (eLOAD sink 1mA), so Measure Current will be 1mA.
When DUT outout floating → Measure Voltage will be 0V.

4.5.13.3 Example

```
SET_CONST_CURRENT_LOAD(
                        Vee, MXDPS
                            , _CONST_I, 1A
                            , _I_RNG , @1A
                             , _MEAS_TYPE, MV
                            , _V_RNG, @8V
                            , _RELAY, ON
                            , _DELAY, 2mS
);
JUDGE_VOLT( Vee, MXDPS, -1.5V, -0.2V,
                _AVG_CNT, 1
);
```

4.6 MXUVI Control Statement(Option)

4.6.1 FORCE_V_UVI

This statement sets the selected UVI to the programmed voltage condition, as well as Ranges and clamp currents.

4.6.1.1 Format

Parameter list:

```
FORCE_V_UVI(UVI_name, f_volt, v_range, i_range, i_clamp, m_mode, relay, wait_time [, slew_rate ]);
```

Parameter comment:

UVI_name : The UVI may be specified as a single pin from the PIN_DEF declaration.
UVI channels are defined by slot
EX: Slot8➔512~527(64*8~64*8+15)
And Type declared "UVI" in declaration file.

Both variables & constants are allowed
Type: integer
No entry: illegal

f_volt: The voltage to be applied by the selected UVI.This is a value that can be programmed from -12V to +12V.
Variable & constants allowed
Type: VOLTAGE or double
Unit: V, pV, nV, uV, mV
No_entry: keep current value

V_Range	Minimum	Maximum
@2V	-2V	+2V
@4V	-4V	+4V
@6V	-6V	+6V
@12V	-12V	+12V

V_range:

V_Range	Minimum	Maximum
@2V	-2V	+2V
@4V	-4V	+4V
@6V	-6V	+6V
@12V	-12V	+12V

Type: Integer

**10V~12V,maximum current is 250mA.
No_entry: illegal

I_range:

I_Range	Minimum	Maximum
@1uA	-1uA	+1uA
@10uA	-10uA	+10uA
@100uA	-100uA	+100uA
@1mA	-1mA	+1mA
@10mA	-10mA	+10mA
@100mA	-100mA	+100mA
@1A	-1A	+1A

Type: integer

**10V~12V,maximum current is 250mA.
No_entry: keep current range

I_calmp: This value must be in the specified I_range, as follows:

I_Range	Minimum	Maximum	Remark
@1uA	-1uA	+1uA	(I_clamp value must > 1uA * 5%)
@10uA	-10uA	+10uA	(I_clamp value must > 10uA * 5%)
@100uA	-100uA	+100uA	(I_clamp value must > 100uA * 5%)
@1mA	-1mA	+1mA	(I_clamp value must > 1mA * 5%)
@10mA	-10mA	+10mA	(I_clamp value must > 10mA * 5%)
@100mA	-100mA	+100mA	(I_clamp value must > 100mA * 5%)
@1A	-1A	+1A	(I_clamp value must > 1A * 5%)

Type: CURRENT or double and Variables & constnats allowed
Unit: A, pA, nA, uA, mA
No_entry: keep current value

m_mode : Select the mode:
NORM(CPU trigger)
No_entry: Default = NORM

Relay: ON – turns on UVI by marking UVI utility relay after conditions setup.
OFF – turns off UVI by breaking UVI utility relay after conditions setup.
Type: integer
No_entry: keep current state

wait_time: specify the waiting time after conditions setup
Variables & constants allowed
Type: TIME or double
Unit: S, pS, nS, uS, mS
No_entry: Default = 500uS

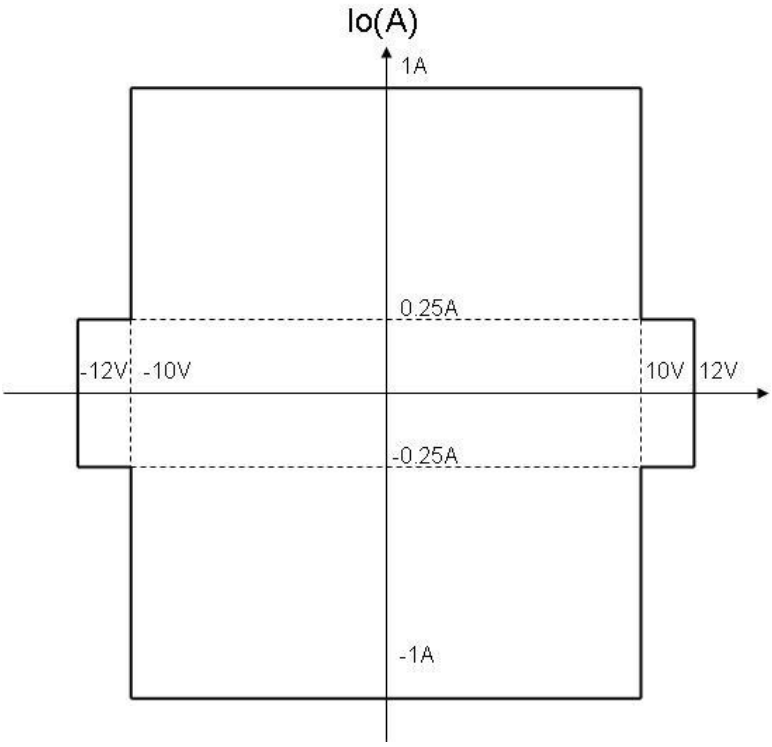
slew_rate: FAST

NORM
**no_entry: NORM (system default)

Current Gang specification

Gang mode group	channel 0~3, channel 4~7, channel 8~11, channel 12~15
Gang mode channel selection	continued 2 channels or 3 channels or 4channels in each Gang mode group , maximum 4 channels

** MXUVI operating V/I curve **



4.6.1.2 Example

```
idd_static_test()
{
      FORCE_V_UVI(Vcc, 5V, @6V, @100mA, 100mA, NORM, ON, 2mS);
      SET_LEVELN(com_lvl, IO_ALLPINS, 0V, 4.5V,1V,3V,0uA,0uA,3.0V, , );
      LEVELS(com_lvl, 2mS);
      RELAY_ON(IO_ALLPINS,PDCL, 1mS);
      TEST_NO(32);
      JUDGE_PAT(static_idd_st, static_idd_sp );
      TEST_NO(33);
      JUDGE_I_UVI(Vcc,0uA , 80uA, 2mS);
       .............
}
```

4.6.1.3 Relative

JUDGE_I_UVI

4.6.2 FORCE_I_UVI

FORCE_I_UVI -- CRAFT statement
This statement sets the selected UVI to the I-force mode

4.6.2.1 Format

Parameter list:

FORCE_I_UVI(UVI_name, f_curent, i_range, v_range, v_clamp, m_mode, ON/OFF, ait_time) ;

Parameter comment:

UVI_name : The UVI may be specified as a single pin from the PIN_DEF declaration.
UVI channels are defined by slot
EX: Slot8➔512~527(64*8~64*8+15)
And Type declared "UVI" in declaration file.
Both variables & constants are allowed
Type: integer
No entry: illegal

f_curent: The voltage to be applied by the selected UVI.This is a value that can be programmed from –1A to +1A
Variable & constants allowed
Type: CURRENT or double
Unit: A, pA, nA, uA, mA
No_entry: keep current value

I_Range	Minimum	Maximum
@1uA	-1uA	+1uA
@10uA	-10uA	+10uA
@100uA	-100uA	+100uA
@1mA	-1mA	+1mA
@10mA	-10mA	+10mA
@100mA	-100mA	+100mA
@1A	-1A	+1A

I_range: Possible hardware range:

I_Range	Minimum	Maximum
@1uA	-1uA	+1uA
@10uA	-10uA	+10uA
@100uA	-100uA	+100uA
@1mA	-1mA	+1mA
@10mA	-10mA	+10mA
@100mA	-100mA	+100mA

@1A	-1A	+1A

No entry: depend on f_current (auto-range)

V_range: One of two possible hardware ranges. For the best accuracy, you should program the current by using the closest range as possible.

V_Range	Minimum	Maximum
@2V	-2V	+2V
@4V	-4V	+4V
@6V	-6V	+6V
@12V	-12V	+12V

No entry: keep current range

V_clamp: This values must be in the specified V_range, as follows:

Possible hardware range:

V_range	V_clamp value	Remark
@2V	+2V*0.05 to +2V*1.1	(V_clamp value must > 2 V* 5%)
@4V	+4V*0.05 to +4V*1.1	(V_clamp value must > 4 V* 5%)
@6V	+6V*0.05 to +6V*1.1	(V_clamp value must > 6 V* 5%)
@12V	+12V*0.05 to +12V*1.1	(V_clamp value must > 12V * 5%)

Variables and constants are allowed.
Type: VOLTAGE or double
Unit: V, pV, nV, uV, mV
No entry: keep current value

ON/OFF: ON – turns on UVIs by marking UVI utility relay after conditions setup
OFF – turns off UVI by breaking UVI utility relay after conditions setup
Type: integer
no_entry: keep current state
[Notice]: Force ON and Force OFF procedure
Force ON : Relay closes first and force setting value
Force OFF : Force setting value first and Relay open

m_mode : specify the UVI measure mode ;
NORM(CPU trigger)
no_entry: Default = NORM

wait_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 400uS

4.6.2.2 Example

```
FORCE_I_UVI (Vee, 0uA, @10uA, @6V, 6V, NORM, ON, 1mS);
```

4.6.3 MEAS_I_UVI

MEAS_I_UVI -- CRAFT statement

Perform a UVI current measurement and save the measured value to system buffer. Read the result by the "RESULT_UVI" statement .

4.6.3.1 Format

Parameter list:

```
MEAS_I_UVI(UVI_name, wait_time[, ave_cnt, ave_mode, sampling_interval]);
```

Parameter comment:

UVI_name : The UVI may be specified as a single pin from the PIN_DEF declaration.
Using Slot will decide UVI channel
EX: (Slot3)96-111, (Slot8)256-271, (Slot18)576-591
And Type declared "UVI" in declaration file.

Both variables & constants are allowed
type: interger
no_entry: illegal

wait_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
no_entry: Default = 400uS

ave_cnt: the average count
no_entry: Default = 1

ave_mode: the average mode
AVE : average
RMS : root mean square
type: int
no_entry: Default = AVE

sampling_interval: the sampling interval between each measurement
no_entry: Default = 50uS (minimum value)

Example:

```
MEAS_I_UVI( VEE, 2mS, 16, AVE, 50uS);
imeas = RESULT_UVI(S0, VEE);
JUDGE_VARIABLE (S0, imeas, 1uA, 10uA, "rst");
```

4.6.4 MEAS_V_UVI

MEAS_V_UVI -- CRAFT statement

Perform a reference power supply Voltage measurement and save the measured value to system buffer. Read the result by the "RESULT_UVI" statement .

4.6.4.1 Format

Parameter comment:

```
MEAS_V_UVI(UVI_name, judge_mode, wait_time[, ave_cnt, ave_mode, sampling_interval]) ;
```

// driver description : 1. delay "wait_time"
2. iteration "ave_cnt" to measure voltage and do average

UVI_name : The UVI may be specified as a single pin from the PIN_DEF declaration.
Using Slot will decide UVI channel
EX: (Slot3)96-111, (Slot8)256-271, (Slot18)576-591
And Type declared "UVI" in declaration file.

Both variables & constants are allowed
type: interger
no_entry: illegal

wait_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
no_entry: Default = 400uS

judge_mode : IFVM (The current loading Voltage measurement is executed)
no_entry: default = IFVM ;

ave_cnt: the average count
no_entry: Default = 1

ave_mode: the average mode
AVE : average
RMS : root mean square
type: int
no_entry: Default = AVE

sampling_interval: the sampling interval between each measurement
no_entry: Default = 50uS (minimum value)

4.6.4.2 Example

```
MEAS_V_UVI(VEE, IFVM, 1mS);
```

```
MEAS_V_UVI(VEE, IFVM, 1mS, 16, AVE, 50uS);
vmeas = RESULT_UVI(S0, Vdd);
JUDGE_VARIABLE (S0, imeas, 3.5V, 5V, "rst");
```

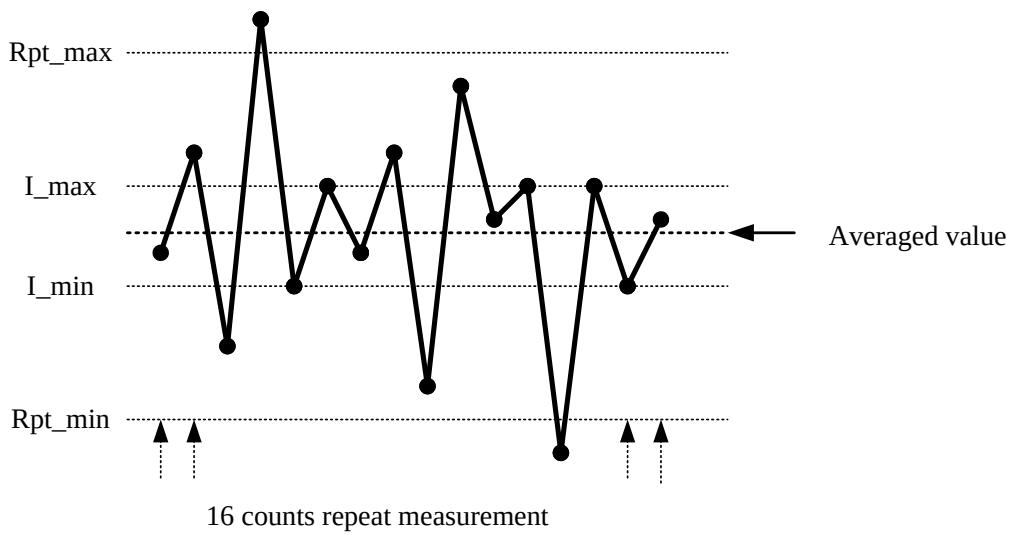
4.6.5 JUDGE_I_UVI

JUDGE_I_UVI -- CRAFT statement

Perform a UVI current measurement and compare the value against specified limits.

4.6.5.1 Format

```
JUDGE_I_UVI(UVI_name, I_min, I_max, wait_time[, ave_cnt, ave_mode, sampling_interval ,
rpt_min, rpt_max ]) ;
```



Parameter comment:

UVI_name : The UVI may be specified as a single pin from the PIN_DEF declaration.
Using Slot will decide UVI channel
EX: (Slot3)96-111, (Slot8)256-271, (Slot18)576-591
And Type declared "UVI" in declaration file.

Both variables & constants are allowed
type: interger
no_entry: illegal

I_min: lower limit value, variables & constants allowed
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry: only high limit

I_max: upper limit value, variables & constants allowed
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry: only low limit

wait_time; specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 500uS

ave_cnt: the average count
**no_entry: Default = 1

ave_mode: the average mode
AVE : average
RMS : root mean square
type: int
**no_entry: Default = AVE

sampling_interval: the sampling interval between each measurement
**no_entry: Default = 20uS (minimum value)

rpt_min: the low limit of each measurement ; if any measurement smaller than this value, then the judge result will be set to fail

rpt_max: the high limit of each measurement ; if any measurement larger than this value, then the judge result will be set to fail

if ave_min & ave_max no_entry : skip the ave_min/ave_max judgement

4.6.5.2 Example

```
JUDGE_I_UVI (VEE, 60uA, 150uA, 2mS, 32, AVE, 50uS, , );
```

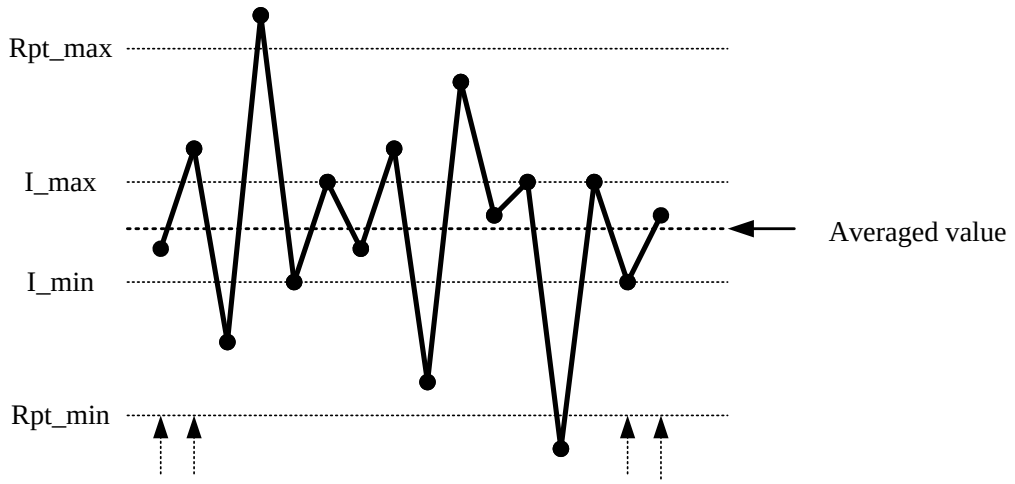
4.6.6 JUDGE_V_UVI

JUDGE_V_UVI -- CRAFT statement

Perform a UVI voltage measurement and compare the value against specified limits.

4.6.6.1 Format

JUDGE_V_UVI(UVI_name, judge_mode, V_min, V_max, wait_time[, ave_cnt, ave_mode, sampling_interval , rpt_min, rpt_max]) ;



Parameter comment:

UVI_name : The UVI may be specified as a single pin from the PIN_DEF declaration.
Using Slot will decide UVI channel
EX: (Slot3)96-111, (Slot8)256-271, (Slot18)576-591
And Type declared "UVI" in declaration file.

Both variables & constants are allowed
type: interger
no_entry: illegal

v_min: lower limit value, variables & constants allowed
type: VOLTAGE or double
unit: V, mV
no_entry: only high limit

v_max: upper limit value, variables & constants allowed
type: VOLTAGE or double
unit: V, mV
no_entry: only low limit

wait_time; specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
no_entry: Default = 500uS

judge_mode : IFVM (The current loading Voltage measurement is executed)
no_entry: default = IFVM ;

ave_cnt: the average count
no_entry: Default = 1

ave_mode: the average mode
AVE : average
RMS : root mean square
type: int
no_entry: Default = AVE

sampling_interval: the sampling interval between each measurement
no_entry: Default = 20uS (minimum value)

rpt_min: the low limit of each measurement ; if any measurement smaller than this value, then the judge result will be set to fail

rpt_max: the high limit of each measurement ; if any measurement larger than this value, then the judge result will be set to fail

if ave_min & ave_max no_entry : skip the ave_min/ave_max judgement

4.6.6.2 Example

```
JUDGE_V_UVI( VEE, IFVM, 1V, 2V, 2mS, 16, AVE, 50uS, , );
```

4.6.7 RESULT_UVI

RESULT_UVI -- CRAFT statement
The statement extracts the result (measured value) of the MEAS_I_UVI/MEAS_V_UVI statement execution as the variable.

4.6.7.1 Format

```
Value = RESULT_UVI(socket_no, UVI_name) ;
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
….
S31 : socket No. 31
Type : integer
no_entry: illegal

UVI_name : The UVI may be specified as a single pin from the PIN_DEF declaration.
Both variables & constants are allowed
type: interger
no_entry: illegal

4.6.7.2 Example

```
v_meas_value=RESULT_UVI( S0, Vdd_pin );
```

4.6.8 MIN_RESULT_UVI

MIN_RESULT_UVI -- CRAFT statement
The statement extracts the minimum result (measured value) of the MEAS_I_UVI/MEAS_V_UVI statement execution as the variable.

4.6.8.1 Format

```
Value = MIN_RESULT_UVI(socket_no, UVI_name) ;
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

UVI_name : The UVI may be specified as a single pin from the PIN_DEF declaration.
Both variables & constants are allowed
type: interger
no_entry: illegal

4.6.8.2 Example

```
v_meas_value= MIN_RESULT_UVI( S0, Vdd_pin );
```

4.6.9 MAX_RESULT_UVI

MAX_RESULT_UVI -- CRAFT statement

The statement extracts the maximum result (measured value) of the MEAS_I_UVI/MEAS_V_UVI statement execution as the variable.

4.6.9.1 Format

```
Value = MAX_RESULT_UVI(socket_no, UVI_name) ;
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

UVI_name : pin-def
variables & constants allowed
type: interger
no_entry: illegal

4.6.9.2 Example

```
v_meas_value= MAX_RESULT_UVI( S0, Vdd_pin );
```

4.6.10 RESULT_UVI_MS

The statement extracts the minimum result (measured value) after the MEAS_I_UVI/MEAS_V_UVI statement executed.

4.6.10.1 Format

Parameter list:

```
RESULT_UVI_MS(DPS_name, value_ms ) ;
```

Parameter comment:

UVI_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

value_ms: Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value
Type: array

4.6.10.2 Example

```
double   value_ms [4];   //suppose 4-site
double   high_limit, low_limit;

RESULT_UVI_MS( Vdd, value_ms );
JUDGE_VARIABLE_MS(value_ms, high_limit, low_limit, "vdd test");
```

4.6.11 MIN_RESULT_UVI_MS

The statement extracts the minimum result (measured value) after the MEAS_I_UVI/MEAS_V_UVI statement executed.

4.6.11.1 Format

Parameter list:

```
MIN_RESULT_UVI_MS(DPS_name, value_ms ) ;
```

Parameter comment:

UVI_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

value_ms: Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value
Type: array

4.6.11.2 Example

```
double   value_ms [4];   //suppose 4-site
double   high_limit, low_limit;

MIN_RESULT_UVI_MS( Vdd, value_ms );
JUDGE_VARIABLE_MS(value_ms, high_limit, low_limit, "vdd min test");
```

4.6.12 MAX_RESULT_UVI_MS

The statement extracts the minimum result (measured value) after the MEAS_I_UVI/MEAS_V_UVI statement executed.

4.6.12.1 Format

Parameter list:

```
MAX_RESULT_UVI_MS(DPS_name, value_ms ) ;
```

Parameter comment:

UVI_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

value_ms: Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value
Type: array

4.6.12.2 Example

```
double   value_ms [4];   //suppose 4-site
double   high_limit, low_limit;

MAX_RESULT_UVI_MS( Vdd, value_ms );
JUDGE_VARIABLE_MS(value_ms, high_limit, low_limit, "vdd max test");
```

4.7 MXREF Control Statement(Option)

4.7.1 FORCE_I_PREF

this statement sets the selected PREF to the I-force mode

4.7.1.1 Format

Parameter list:

FORCE_I_PREF(PREF_name, f_curent, i_range, v_range, v_clamp, m_mode, ON/OFF, wait_time) ;

Parameter comment:

PREF_name : pin-def.
variables & constants allowed
type: interger
**no_entry: illegal

f_curent:

I_range	Minimum	Maximum
@1uA	-1uA	+1uA
@10uA	-10uA	+10uA
@100uA	-100uA	+100uA
@1mA	-1mA	+1mA
@10mA	-10mA	+10mA
@100mA	-100mA	+100mA
@250mA	-250mA	+250mA

variables & constants allowed
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry: keep current value

I_range:

I_range	Minimum	Maximum
@1uA	-1uA	+1uA
@10uA	-10uA	+10uA
@100uA	-100uA	+100uA
@1mA	-1mA	+1mA
@10mA	-10mA	+10mA
@100mA	-100mA	+100mA
@250mA	-250mA	+250mA

**no_entry: depend on f_current(auto-range)

V_range:

V_range	Minimum	Maximum
@6V	-6V	+6V
@12V	-12V	+12V
@24V	-24V	+24V
@48V	-48V	+48V

**no_entry: keep current range

V_clamp: This values must be in the specified V_range, as follows:

V_range	V_clamp value	Remark
@6V	+6V*0.05 to +6V*1.1	(V_clamp value must > 6 V* 5%)
@12V	+12V*0.05 to +12V*1.1	(V_clamp value must > 12 V* 5%)
@24V	+24V*0.05 to +24V*1.1	(V_clamp value must > 24 V* 5%)
@48V	+48V*0.05 to +48V*1.1	(V_clamp value must > 48V * 5%)

variables & constants allowed
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: keep current value

ON/OFF:ON – turns on PREF by marking PREF utility relay after conditions setup
OFF – turns off PREF by breaking PREF utility relay after conditions setup
type: integer
**no_entry: keep current state

m_mode : specify the PREF measure mode ;
NORM(CPU trigger)
**no_entry: Default = NORM

wait_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 400uS

4.7.1.2 Example

```
FORCE_I_PREF (Vee, 0uA, @10uA, @6V, 6V, NORM, ON, 1mS);
```

4.7.2 FORCE_V_PREF

this statement sets the selected PREF to the programmed voltage condition, as well as ranges and clamp currents.

4.7.2.1 Format

Parameter list:

FORCE_V_PREF(PREF_name, f_volt, v_range, i_range, i_clamp, m_mode, relay, wait_time [, slew_rate]) :

Parameter comment:

PREF_name : pin-def.
variables & constants allowed
type: interger
**no_entry: illegal

f_volt: -48V to +48V
variable & constants allowed
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: keep current value

V_range:

V_range	Minimum	Maximum
@6V	-6V	+6V
@12V	-12V	+12V
@24V	-24V	+24V
@48V	-48V	+48V

**45V~48V,maximum current is 100mA.
**no_entry: depend on f_volt (auto-range)

I_range:

I_range	Minimum	Maximum
@1uA	-1uA	+1uA
@10uA	-10uA	+10uA
@100uA	-100uA	+100uA
@1mA	-1mA	+1mA
@10mA	-10mA	+10mA
@100mA	-100mA	+100mA
@250mA	-250mA	+250mA

**45V~48V,maximum current is 100mA.
**no_entry: keep current range

I_calmp: This value must be in the specified I_range, as follows:

- Possible hardware range:

I_range	I_clamp value Minimum	I_clamp value Maximum	Remark
@1uA	-1.00uA	1.00uA	(I_clamp value

			must 1.00uA * 5%)
@10uA	-10.0uA	10.0uA	(I_clamp value must 10.0uA * 5%)
@100uA	-100uA	100uA	(I_clamp value must 100uA * 5%)
@1mA	-1.00mA	1.00mA	(I_clamp value must 1.00mA * 5%)
@10mA	-10.0mA	10.0mA	(I_clamp value must 10.0mA * 5%)
@100mA	-100mA	100mA	(I_clamp value must 100mA * 5%)
@250mA	-250mA	250mA	(I_clamp value must 250mA * 5%)

variables & constnats allowed
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry: keep current value

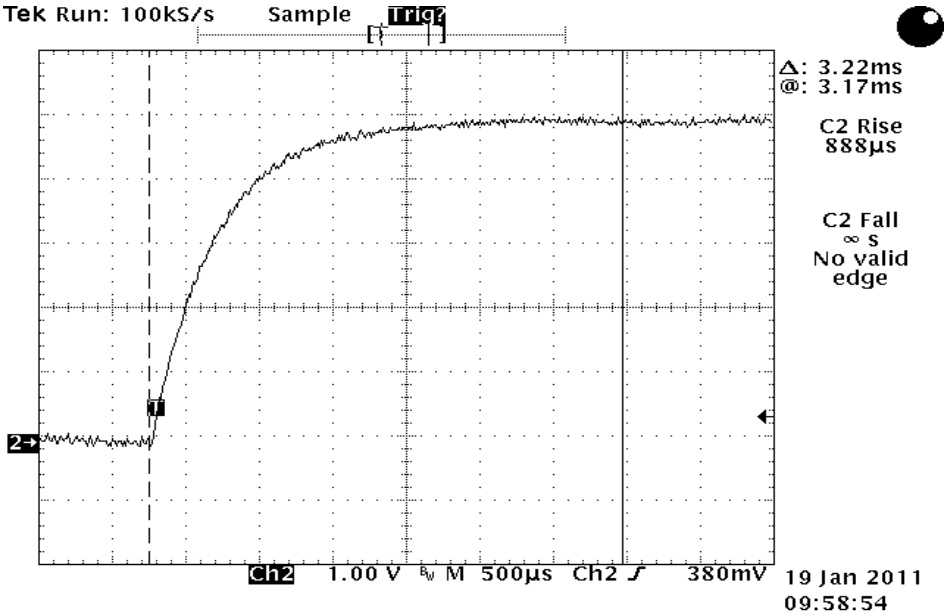
m_mode : NORM(CPU trigger)
**no_entry: Default = NORM

relay: ON – turns on PREF by marking PREF utility relay after conditions setup.
OFF – turns off PREF by breaking PREF utility relay after conditions setup.
type: integer
**no_entry: keep current state

wait_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 500uS

slew_rate: FAST
NORM
**no_entry: NORM (system default)

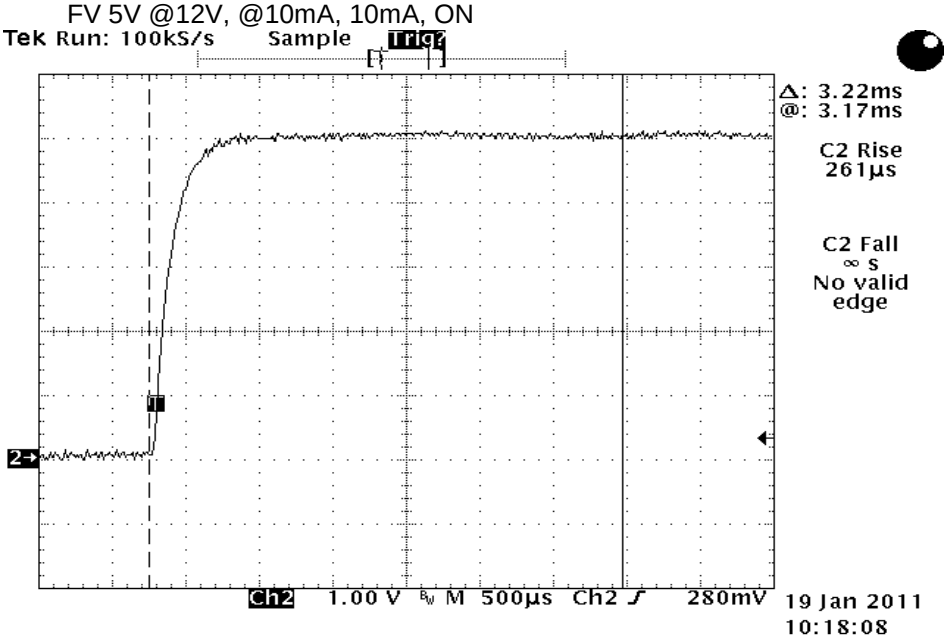
** slew_rate = NORM waveform
FV 0V @12V, @10mA, 10mA, ON //no loading
FV 5V @12V, @10mA, 10mA, ON



** slew_rate = FAST waveform

FV 0V @12V, @10mA, 10mA, ON //no loading

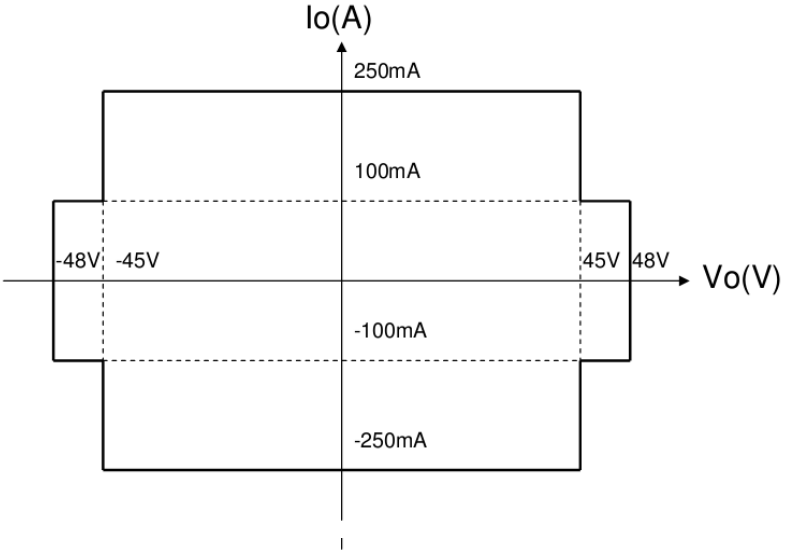
FV 5V @12V, @10mA, 10mA, ON



Current Gang specification

Gang mode group	channel 0~3, channel 4~7, channel 8~11, channel 12~15
Gang mode channel selection	continued 2 channels or 3 channels or 4channels in each Gang mode group , maximum 4 channels

** MXREF operating V/I curve **



4.7.2.2 Example

```
FORCE_V_PREF(Vee, 5V, @6V, @100mA, 100mA, NORM, ON, 2mS);
```

4.7.3 JUDGE_I_PREF

Perform a PREF current measurement and compare the value against specified limits.

4.7.3.1 Format

Parameter list:

```
JUDGE_I_PREF(PREF_name, I_min, I_max, wait_time[, ave_cnt, ave_mode,
                    sampling_interval , rpt_min, rpt_max ]) ;
```

Parameter comment:

PREF_name : pin-def or pin_group
variables & constants allowed
type: interger
**no_entry: illegal

I_min: lower limit value, variables & constants allowed
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry: only high limit

I_max: upper limit value, variables & constants allowed
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry: only low limit

wait_time; specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 500uS

ave_cnt: the average count
**no_entry: Default = 1

ave_mode: the average mode
AVE : average
RMS : root mean square
type: int
**no_entry: Default = AVE

sampling_interval: the sampling interval between each measurement
**no_entry: Default = 20uS (minimum value)

rpt_min: the low limit of each measurement ; if any measurement smaller than this value, then the judge result will be set to fail

rpt_max: the high limit of each measurement ; if any measurement larger than this value, then the judge result will be set to fail

if rpt_min & rpt_max no_entry : skip the rpt_min/rpt_max judgement

4.7.3.2 Example

```
JUDGE_I_PREF (Vee, 60uA, 150uA, 2mS, 32, AVE, 50uS, , );
```

4.7.4 JUDGE_V_PREF

Perform a PREF voltage measurement and compare the value against specified limits.

4.7.4.1 Format

Parameter list:

```
JUDGE_V_PREF(PREF_name, judge_mode, V_min, V_max, wait_time
                    [, ave_cnt, ave_mode, sampling_interval , rpt_min, rpt_max ]) ;
```

Parameter comment:

PREF_name : pin-def or pin-group.
variables & constants allowed
type: interger
**no_entry: illegal

v_min: lower limit value, variables & constants allowed
type: VOLTAGE or double
unit: V, mV
**no_entry: only high limit

v_max: upper limit value, variables & constants allowed
type: VOLTAGE or double
unit: V, mV
**no_entry: only low limit

wait_time; specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 500uS

judge_mode : IFVM (The current loading Voltage measurement is executed)

ave_cnt: the average count
**no_entry: Default = 1

ave_mode: the average mode
AVE : average
RMS : root mean square
type: int
**no_entry: Default = AVE

sampling_interval: the sampling interval between each measurement
**no_entry: Default = 20uS (minimum value)

rpt_min: the low limit of each measurement ; if any measurement smaller than this value, then the judge result will be set to fail

rpt_max: the high limit of each measurement ; if any measurement larger than this value, then the judge result will be set to fail

if rpt_min & rpt_max no_entry : skip the rpt_min/rpt_max judgement

4.7.4.2 Example

```
JUDGE_V_PREF( VEE , IFVM, -1.2V,   -0.8V, 2mS);
JUDGE_V_PREF( VEE, IFVM, 1V, 2V, 2mS, 16, AVE, 50uS, , );
```

4.7.5 MEAS_I_PREF

Perform a PREF current measurement and save the measured value to system buffer. Read the result by the "RESULT_PREF" statement .

4.7.5.1 Format

Parameter list:

```
MEAS_I_PREF(PREF_name, wait_time[, ave_cnt, ave_mode, sampling_interval]);
```

Parameter comment:

PREF_name : pin-def or pin_group
variables & constants allowed
type: interger
**no_entry: illegal

wait_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 400uS

// Notice : if use average mode, then meas_type should use NORM mode,
otherwise compiler error

ave_cnt: the average count
**no_entry: Default = 1

ave_mode: the average mode
AVE : average
RMS : root mean square
type: int
**no_entry: Default = AVE

sampling_interval: the sampling interval between each measurement
**no_entry: Default = 50uS (minimum value)

4.7.5.2 Example

```
MEAS_I_PREF( Vee, 2mS, 16, AVE, 50uS);
imeas = RESULT_PREF(S0, Vee);
```

4.7.6 MEAS_V_PREF

Perform a reference power supply Voltage measurement and save the measured value to system buffer. Read the result by the "RESULT_PREF" statement .

4.7.6.1 Format

Parameter list:

```
MEAS_V_PREF(PREF_name, judge_mode, wait_time[, ave_cnt, ave_mode,
                    sampling_interval]) ;
                    rpt_max ]) ;
```

Parameter comment:

```
PREF_name : pin-def.
            variables & constants allowed
            type: interger
            **no_entry: illegal

  wait_time:    specify the waiting time after conditions setup
                variables & constants allowed
                type: TIME or double
                unit: S, pS, nS, uS, mS
                **no_entry: Default = 400uS

  judge_mode : IFVM (The current loading Voltage measurement is executed )

  // Notice : if use average mode, then meas_type should use NORM mode,
                otherwise compiler error

  ave_cnt:      the average count
                **no_entry: Default = 1

  ave_mode:     the average mode
                AVE : average
                RMS : root mean square
                type: int
                **no_entry: Default = AVE

  sampling_interval:   the sampling interval between each measurement
                **no_entry: Default = 20uS (minimum value)
```

4.7.6.2 Example

```
MEAS_V_PREF(VEE,   , 1mS);

imeas = RESULT_PREF(S0, VEE);
```

4.7.7 RESULT_PREF

The statement extracts the result (measured value) of the MEAS_I_PREF/MEAS_V_PREF statement execution as the variable.

4.7.7.1 Format

Parameter list:

```
Value = RESULT_PREF(socket_no, PREF_name) ;
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

PREF_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

4.7.7.2 Example

```
v_meas_value=RESULT_PREF( S0, Vdd_pin );
```

4.7.8 MIN_RESULT_PREF

The statement extracts the minimum result (measured value) of the MEAS_I_PREF/MEAS_V_PREF statement execution as the variable.

4.7.8.1 Format

Parameter list:

```
Value = MIN_RESULT_PREF(socket_no, PREF_name) ;
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

PREF_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

4.7.8.2 Example

```
v_meas_value= MIN_RESULT_PREF( S0, Vdd_pin );
```

4.7.9 MAX_RESULT_PREF

The statement extracts the maximum result (measured value) of the MEAS_I_PREF/MEAS_V_PREF statement execution as the variable.

4.7.9.1 Format

Parameter list:

```
Value = MAX_RESULT_PREF(socket_no, PREF_name) ;
```

Parameter comment:

Value : Specifies the variable to store the measuring results. The description is given in the form of The current type or The voltage type variable.

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

PREF_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

4.7.9.2 Example

```
v_meas_value= MAX_RESULT_PREF( S0, Vdd_pin );
```

4.7.10 RESULT_PREF_MS

The statement extracts the result (measured value) of the MEAS_I_PREF/MEAS_V_PREF statement execution as the variable.

4.7.10.1 Format

Parameter list:

```
RESULT_PREF_MS(PREF_name, value_ms);
```

Parameter comment:

PREF_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

value_ms: Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value
Type: array

4.7.10.2 Example

```
double   value_ms[4];   //suppose 4-site
double   high_limit, low_limit;

RESULT_PREF_MS( Vdd, value_ms );
JUDGE_VARIABLE_MS(value_ms, high_limit, low_limit, "vdd test");
```

4.7.11 MIN_RESULT_PREF_MS

The statement extracts the result (measured value) of the MEAS_I_PREF/MEAS_V_PREF statement execution as the variable.

4.7.11.1 Format

Parameter list:

```
MIN_RESULT_PREF_MS(PREF_name, value_ms);
```

Parameter comment:

PREF_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

value_ms: Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value
Type: array

4.7.11.2 Example

```
double   value_ms[4];   //suppose 4-site
double   high_limit, low_limit;

MIN_RESULT_PREF_MS( Vdd, value_ms );
JUDGE_VARIABLE_MS(value_ms, high_limit, low_limit, "vdd min test");
```

4.7.12 MAX_RESULT_PREF_MS

The statement extracts the result (measured value) of the MEAS_I_PREF/MEAS_V_PREF statement execution as the variable.

4.7.12.1 Format

Parameter list:

```
MAX_RESULT_PREF_MS(PREF_name, value_ms);
```

Parameter comment:

PREF_name : pin-def
variables & constants allowed
type: interger
**no_entry: illegal

value_ms: Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value
Type: array

4.7.12.2 Example

```
double   value_ms[4];   //suppose 4-site
double   high_limit, low_limit;

MAX_RESULT_PREF_MS( Vdd, value_ms );
JUDGE_VARIABLE_MS(value_ms, high_limit, low_limit, "vdd max test");
```

4.8 Complex Control Statement

4.8.1 FORCE_VOLT

This statement sets the selected MLDPS or other board to the programmed voltage condition, as well as ranges and clamp currents.

4.8.1.1 Format

Parameter list:
When we use this statement have to use one and more ([ ]+ , [ ]+) condition. Another is no and more ([ ]*) condition.Option parameter input is no sequencing, but must be pairs ([ ]) to use.

```
FORCE_VOLT(
[
[ pin_name,
  MLDPS
[
[ , _FORCE_V, f_volt , _V_RNG, @6V | @12V ]
[ , _I_RNG, @5uA | @25uA | @250uA | @2.5mA | @25mA | @500mA | @1A
  , _MEAS_TYPE, MV | MI , _CLAMP_I, clamp_value ]
[ , _RELAY, ON | OFF]
[ , _DELAY, delay_time]
[ , _GANG_MODE, GANG | NORM ] // gang_mode_flag, gang-mode function OFF
]+
]*,
[ pin_name,
  MXDPS
[
[ , _FORCE_V, f_volt , _V_RNG, @4V |@8V | @12V | @16V ]
[ , _I_RNG, @1uA | @10uA | @100uA | @1mA | @10mA | @100mA | @1A | @2A
  , _MEAS_TYPE, MV | MI , _CLAMP_I, clamp_value ]
[ , _SETTLING_TIME, @S0 | @S1 ]    //slew_rate
[ , _RELAY, ON | OFF]
[ , _DELAY, delay_time]
]+
]*,
[ pin_name,
  MXREF
[
[ , _FORCE_V, f_volt , _V_RNG, @6V | @12V | @24V | @48V ]
[ , _I_RNG, @1uA | @10uA | @100uA | @1mA | @10mA | @100mA | @250mA
  , _MEAS_TYPE, MI , _CLAMP_I, clamp_value ]
[ , _SETTLING_TIME, @S0 | @S1 ]    // slew_rate
[ , _RELAY, ON | OFF]
[ , _DELAY, delay_time]
[ , _GANG_MODE, GANG | NORM ] // gang_mode_flag, gang-mode function OFF
]+
]*,
[ pin_name,
  MXUVI
[
[ , _FORCE_V, f_volt    , _V_RNG, @2V | @4V | @6V | @12V ]
```

```
[ , _I_RNG, @1uA | @10uA | @100uA | @1mA | @10mA | @100mA | @1A
  , _MEAS_TYPE, MI , _CLAMP_I, clamp_value ]
[ , _SETTLING_TIME, @S0 | @S1 ]    // slew_rate
[ , _RELAY, ON | OFF]
[ , _DELAY, delay_time]     //
[ , _GANG_MODE, GANG | NORM ] // gang_mode_flag, gang-mode function OFF
]+
]*
[ _GDELAY, delay_time]
]+
);
```

Parameter comment:

pin_name : pin-def or pin_group name
variables & constants allowed
type: interger
**no_entry: illegal
//when _GANG_MODE exist→ pin_name must belong to "POWER_GANG_PIN_GROUP" definition

pin_type_name : **MLDPS/MXDPS/MXREF/MXUVI** (from .dec pin_type define)

f_volt: **MLDPS** : -6V to +12V
MXREF : -48V to +48V
MXDPS : -8V to +16V
MXUVI : -12V to +12V

variable & constants allowed
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: keep current value

v_range: **MLDPS** : @6V : **-6V** ~ +6V
@12V : **-6V** ~ +12V

MXREF : @6V: -6V to +6V
@12V: -12V to +12V
@24V: -24V to +24V
@48V: -48V to +48V

MXDPS : @4V : -4V to +4V
@8V : -8V to +8V
@12V : **-8V** to +12V
@16V : **-8V** to +16V

MXUVI : @2V: -2V to +2V
@4V: -4V to +4V
@6V: -6V to +6V
@12V: -12V to +12V
type: integer
**no_entry: depend on f_volt (auto-range)

i_range: **MLDPS** : @5uA: -5uA to +5uA
@25uA: -25uA to +25uA
@250uA: -250uA to +250uA
@2.5mA: -2.5mA to +2.5mA
@25mA: -25mA to +25mA
@500mA: -500mA to +500mA
@1A: -1A to +1A (only @6V range support)

MXREF : @1uA, @10uA, @100uA, @1mA, @10mA,@100mA, @250mA

MXDPS : @1uA, @10uA, @100uA, @1mA, @10mA,@100mA , @1A, @2A

MXUVI : @1uA, @10uA, @100uA, @1mA, @10mA,@100mA , @1A,

// type: integer
**no_entry: keep current range

clamp_value: // FV → must be Clamp-I

MLDPS :

Range	Range
5uA	Disable clamp function
25uA	Disable clamp function
250uA	±250uA ∗(10% ~ 102.5%)
2.5mA	±2.5mA ∗(10% ~ 102.5%)
25mA	±25mA ∗(10% ~ 102.5%)
500mA	±500mA ∗(10% ~ 102.5%)
1A	±1A ∗(10% ~ 102.5%)

MXREF :

Range	Max current	Notice
1uA	-1.0uA ~ +1.0uA	I_clamp value must > 1uA * 5%
10uA	-10.0uA ~ +10.0uA	I_clamp value must > 10uA * 5%
100uA	-100uA ~ +100uA	I_clamp value must > 100uA * 5%
1mA	-1.0mA ~ +1.0mA	I_clamp value must > 1 mA * 5%
10mA	-10.0mA ~ +10.0mA	I_clamp value must > 10mA * 5%
100mA	-100mA ~ +100mA	I_clamp value must > 100mA * 5%
250mA	-250mA ~ +250mA	I_clamp value must > 250mA * 5%

MXUVI :

Range	Max current	Notice
1uA	-1.0uA ~ +1.0uA	I_clamp value must > 1uA * 5%
10uA	-10.0uA ~ +10.0uA	I_clamp value must > 10uA * 5%
100uA	-100uA ~ +100uA	I_clamp value must > 100uA * 5%
1mA	-1.0mA ~ +1.0mA	I_clamp value must > 1 mA * 5%
10mA	-10.0mA ~ +10.0mA	I_clamp value must > 10mA * 5%
100mA	-100mA ~ +100mA	I_clamp value must > 100mA * 5%
1A	-1A ~ +1A	I_clamp value must > 1A * 5%

MXDPS :

Range	Max current	Notice
1uA	minimum 50uA ~ +100uA	None
10uA	minimum 50uA ~ +100uA	None
100uA	0 ~ +100uA	None
1mA	0 ~ +1.0mA	None
10mA	0 ~ +10.0mA	None
100mA	0 ~ +100mA	None
1A	0 ~ +1A	None
2A	0 ~ +2A	None

This value must be in the specified I_range, as follows:
variables & constnats allowed
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry : if i_range has setup i_range, then i_clamp will follow the i_range to set the maximum i_clamp in the i_range else keep current value .

meas_type : MV (measure output voltage, only MLDPS and MXDPS)
MI (measure output current)
**no_entry: Default = MI

relay: ON – turns on Power by marking Power utility relay after conditions setup.
OFF – turns off Power by breaking Power utility relay after conditions setup.
type: integer
**no_entry: keep current state

gang_mode_flag: **GANG** – use gang-mode function.
NORM – use normal function.
type: integer
**no_entry: NORM
[Notice]:To use "gang_mode_flag" in **GANG,** statement also be setting **"relay"** parameter in the same time**.**

delay_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 500uS

SETTLING_TIME :

MXREF
@S0 : (NORM)
@S1 : (FAST)

MXUVI
@S0 : (NORM)
@S1 : (FAST)

MXDPS
@S0 : (NORM)
@S1 : (FAST)

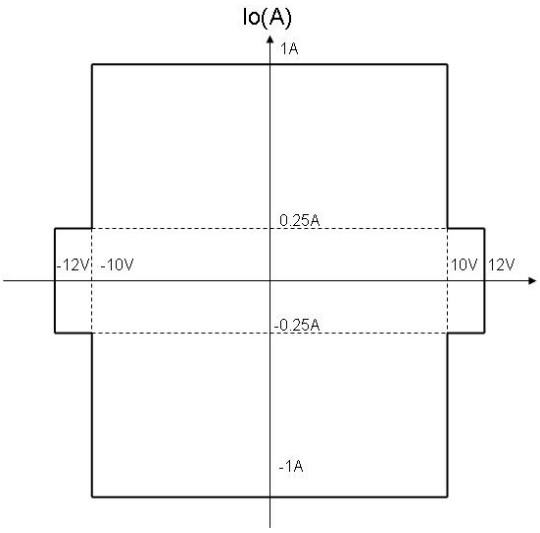
variables & constants allowed
**no_entry: Default = @S0

MXUVI settling time :

FV :

V_range	Settling time/Normal mode(@S0)	Settling time/Faster mode(@S1)
2V	1.6mS	450uS
4V	1.6mS	450uS
6V	1.6mS	450uS
12V	1.6mS	450uS

Operating V/I curve:

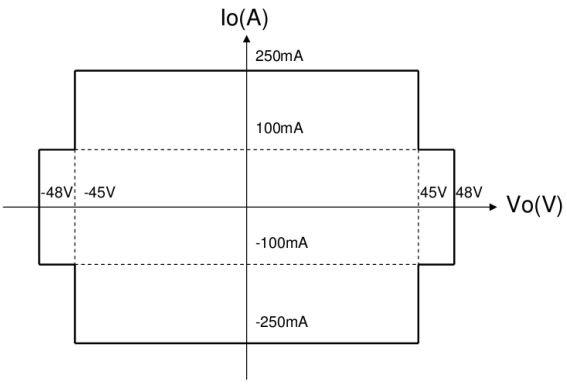


MXREF settling time :

FV :

V_range	Settling time/Normal mode(@S0)	Settling time/Faster mode(@S1)
6V	4.5mS	3mS
12V	4.5mS	3mS
24V	6mS	3mS
48V	6mS	3mS

Operating V/I curve:



MXDPS settling time :

FV :

V_range	Settling time/Normal mode(@S0)	Settling time/Faster mode(@S1)
4V	500uS	6uS
8V	550uS	6.5uS
12V	600uS	7uS
16V	650uS	7.5uS

4.8.1.2 Explanation

1. When use Gang-mode, don't use GANG mode ON → NORM mode OFF (error using, this will lead -48V spike).

   e.g, correct using : FV GANG relay-on 3V -> FV GANG relay-off 0V -> FV NORM relay-off 0V

2. FORCE_VOLT of pin_name can't combine one to one, plan compiler not support & will occur error. Error Using : FORCE_VOLT(Vdd_Gang+ Vee_Gang,….);

   Tne Gang-mode, FORCE_VOLT of pin_name, only use POWER_GANG_PIN_GROUP to define pin_group name.

   e.g, correct using : POWER_GANG_PIN_GROUP{Vdd_gang=Vdd1+Vdd2}, FORCE_VOLT(Vdd_gang, …);

MXREF/MXUVI gang mode using tips & limits :

1 MXREF/MXUVI has 16 channels per board. Using gang mode has to follow Channel number / 4=0, that we will have 4 groups. User could use 2 channels, 3 channels or 4 channels to execute gang mode in every group. But the channels of gang mode must be continuous channel number that the gang mode could be worked. And other of the gang mode channel is still using in normal mode.

Example for gang mode channel:

2.1 A board has 4 groups that channel number 0~3, channel number 4~7, channel number 8~11 and channel number 12~15.

2.2 For channel number 0~3 of group, 2 channels gang mode could use :

4.3.1 Channel 0 and channel 1 in group.

4.3.2 Channel 1 and channel 2 in group.

4.3.3 Channel 2 and channel 3 in group.

4.3.4 User couldn't use channel 0 and channel 2 in group or channel 1 and channel 3 in group. We must use continuous channel number in gang mode.

2.3 For channel number 0~3 of group, 3 channels gang mode could use :

4.3.1 Channel 0 , channel 1 and channel 2 in group.

4.3.2 Channel 1 , channel 2 and channel 3 in group.

4.3.3 User couldn't use channel 0, channel 1 and channel 3 in group. We must use continuous channel number in gang mode.

2 Total 16 channels in board have front channels of channel 0~7 and back channels of channel 8~15. The front channel 0~7 of gang mode channels, have use the maximum number to be master channel. And back channel 8~15 of gang mode channels, have use the minimum number to be master channel.

Example for master channel definition:

2.1 The channel 1, channel 2 and channel 3 are belonging to the front 8 channels. For gang mode, the channel 3 is the master channel in program.

2.2 Others, we select channel 13, channel 14 and channel 15 are belonging the back 8 channels. For gang mode, the channel 13 is the master channel in program.

3 MXUVI support gang-mode of I-range : @1A, @100mA (others range not verified)

4 MXREF support gang-mode of I-range : @250mA, @100mA (others range not verified)

5 MXUVI/MXREF Current gang specification：

Gang mode group	channel 0~3, channel 4~7, channel 8~11, channel 12~15
Gang mode channel selection	continued 2 channels or 3 channels or 4 channels in each gang mode group, maximum 4 channels

MLDPS gang mode using tips & limits:

1. MLDPS has 32 channels, that gang mode channel must follow continuous channel

number; we provide 8 channels of maximum gang-mode group in system. But 32 channels of maximum gang-mode group in release version: CRAFT_3380_2.0_SP5.

2. The minimum channel number of gang mode group is the master channel number in program.
3. MLDPS support gang-mode of I-range : @1A, @500mA (others range not verified)

4.8.1.3 Example

```
// NORM mode with two type power board
FORCE_VOLT(
        Vdd , MLDPS ,
                _FORCE_V, 5.25V,
                _V_RNG, @6V,
                _I_RNG, @1A,
                _MEAS_TYPE, MI,
                _CLAMP_I, 1A,
                _DELAY, 2mS,
                _RELAY , ON,

        Vee , MXREF ,
                _FORCE_V, 12.5V,
                _V_RNG, @24V,
                _I_RNG, @100mA,
                _MEAS_TYPE, MI,
                _CLAMP_I, 100mA
                _DELAY, 2mS,
                _RELAY , ON
        );
// Gang mode
    FORCE_VOLT(Vdd_Gang , MLDPS ,
                _FORCE_V, 5.25V,
                _V_RNG, @6V,
                _I_RNG, @1A,
                _MEAS_TYPE, MI,
                _GANG_MODE, GANG,
                _CLAMP_I, 1A,
                _DELAY, 2mS,
                _RELAY , ON
                );
```

4.8.2 MEAS_CURRENT

Perform a MLDPS or other board current measurement and return the measured value

4.8.2.1 Format

Parameter list:

```
MEAS_CURRENT(   pin_name, pin_type_name, measured_value
                [, _DELAY, delay_time,   ]
                [, _AVG_CNT, average_count,   ]
                [, _CPU_TRIG_INTERVAL, cpu_trig_interval,   ]
                [, _RET_MIN_MEAS, return_min_measured_value, ]
                [, _RET_MAX_MEAS, return_max_measured_value, ]
                );
```

Parameter comment:

pin_name : pin-def or pin_group
variables & constants allowed
type: interger
**no_entry: illegal

pin_type_name : **MLDPS, MXDPS, MXREF, MXUVI** (this from .dec pin_type define)

measured_value : 2-dimension array, array size: [site][pin]
return measured value,
measured_value[0][ ] : Site-0
measured_value[1][ ] : Site-1

delay_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 500uS

average_count: the average count
**no_entry: Default = 1

cpu_trig_interval: the sampling interval between each measurement
MLDPS : minimum 5uS
MXDPS : minimum 5uS
MXUVI : minimum 10uS
MXREF : minimum 10uS
**no_entry: Default = minimum value

return_min_measured_value : 2-dimension array, array size: [site][pin]
return minimum measured value,
measured_value[0][ ] : Site-0
measured_value[1][ ] : Site-1

return_max_measured_value : 2-dimension array, array size: [site][pin]
return maximum measured value,
measured_value[0][ ] : Site-0
measured_value[1][ ] : Site-1

4.8.2.2 Example

```
CARRAY_2DIM<double> imeas( 2 , 1 );    // same with double imeas[2][1];2-site, 1-pin

    FORCE_VOLT(
            Vdd , MLDPS ,
                    _FORCE_V, 5.25V,
                    _V_RNG, @6V,
                    _I_RNG, @1A,
                    _MEAS_TYPE, MI,
                    _CLAMP_I, 1A,
                    _DELAY, 2mS,
                    _RELAY , ON
            );
….
….
     FORCE_VOLT(
            Vdd , MLDPS ,
                    _I_RNG, @1mA,
                    _MEAS_TYPE, MI,
                    _CLAMP_I, 1mA,
            );

MEAS_CURRENT(   Vdd, MLDPS, imeas ,
                _DELAY, 2mS ,
                _AVG_CNT, 1 ,
                _CPU_TRIG_INTERVAL, 10uS
            );
```

4.8.3 JUDGE_CURRENT

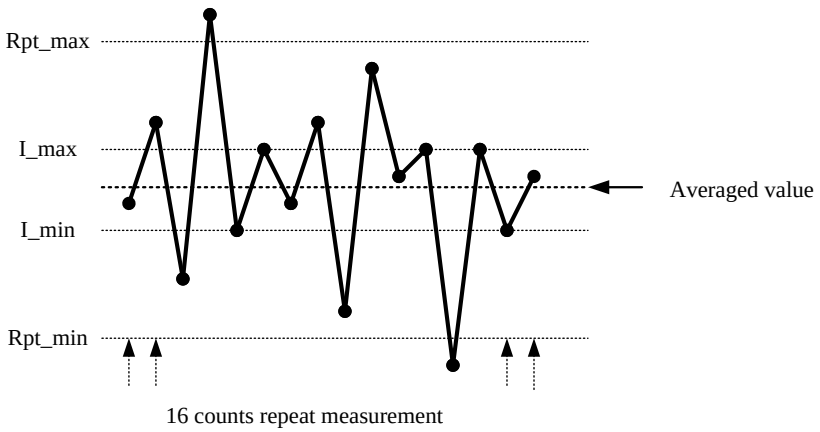
Perform a MLDPS or other board current measurement and compare the value against specified limits. Release version : **CRAFT_3380_1.30**

4.8.3.1 Format

Parameter list:

```
JUDGE_CURRENT(   pin_name, pin_type_name, L_limit, H_limit
                    [, _RET_MEAS, return_measured_value ,]
                  [, _RET_MIN_MEAS, return_min_measured_value ,]
                  [, _RET_MAX_MEAS, return_max_measured_value, ]
                  [, _DELAY,delay_time ,]
                  [, _AVG_CNT, average_count, ]
                  [, _CPU_TRIG_INTERVAL, cpu_trig_interval, ]
                  [, _SAMP_LOW_LIMIT, sample_low_limit, ]
                  [, _SAMP_HIGH_LIMIT, sample_high_limit, ]
      ) ;
```

// driver description : 1. delay "wait_time"
2. iteration "ave_cnt" to measure current and do average
3. each measured value must between rpt_min and rpt_max,
otherwise the judgement will be set to fail .



Parameter comment:

pin_name : pin-def or pin_group
variables & constants allowed
type: interger
**no_entry: illegal

pin_type_name : **MLDPS, MXDPS, MXUVI, MXREF, DPS, PREF, UVI** (this from .dec pin_type define)

L_limit: the low test limit
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry: only high limit

H_limit: the high test limit
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry: only low limit

return_measured_value : 2-dimension array, array size: [site][pin]
return measured value, // array size: [site][pin]
measured_value[0][ ] : Site-0
measured_value[1][ ] : Site-1

return_min_measured_value : 2-dimension array, array size: [site][pin]
return minimum measured value, // array size: [site][pin]
measured_value[0][ ] : Site-0
measured_value[1][ ] : Site-1

return_max_measured_value : 2-dimension array, array size: [site][pin]
return maximum measured value, // array size: [site][pin]
measured_value[0][ ] : Site-0
measured_value[1][ ] : Site-1

delay_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 500uS

average_count: the average count
**no_entry: Default = 1

cpu_trig_interval: the sampling interval between each measurement, for CPU trigger (NORM-mode)
MLDPS : minimum 5uS
MXDPS : minimum 5uS
MXUVI : minimum 10uS
MXREF : minimum 10uS

**no_entry: Default = (minimum value)

sample_low_limit: the low limit of each measurement ; if any measurement smaller than this value, then the judge result will be set to fail

sample_high_limit: the high limit of each measurement ; if any measurement larger than this value, then the judge result will be set to fail
if rpt_min & rpt_max no_entry : skip sample_low_limit / sample_high_limit judgement

4.8.3.2 Example

```
//NORM mode
JUDGE_CURRENT( Vdd, MLDPS, 0mA, 50mA,
                    _AVG_CNT, 1,
                    _CPU_TRIG_INTERVAL, 10uS,
                    _DELAY, 2mS
                  );

// GANG mode
JUDGE_CURRENT( Vdd_Gang, MLDPS, 1.2A, 2A,
                    _AVG_CNT, 1,
                    _CPU_TRIG_INTERVAL, 10uS,
                    _DELAY, 2mS
                  );
```

4.8.4 FORCE_CURRENT

This statement sets the selected MLDPS or other board to the I-force mode.
Release version : **CRAFT_3380_1.30**

4.8.4.1 Format

Parameter list:
When we use this statement have to use one and more ([ ]+ , [ ]+) condition. Another is no and more ([ ]*) condition.Option parameter input is no sequencing, but must be pairs([ ]) to use.

```
FORCE_CURRENT(
[
[ pin_name,
MLDPS
[
[ , _FORCE_I, f_current    , _I_RNG , @25mA | @500mA | @1A ]
[ , _V_RNG, @6V | @12V, _MEAS_TYPE, MV|MI
  , _CLAMP_V, clamp_value |
  , _CLAMP_PV, pclamp_value, _CLAMP_NV,
   nclamp_value]
[ , _RELAY, ON | OFF]
[ , _DELAY, delay_time]
]+
]*
[ pin_name,
  MXDPS
[
[ , _FORCE_I, f_current , _I_RNG , @100uA | @1mA | @10mA | @100mA | @1A |
@2A]
[ , _V_RNG, @4V | @8V | @12V | @16V , _MEAS_TYPE, MV|MI
  , _CLAMP_V, clamp_value]
[ , _RELAY, ON | OFF ]
[ , _DELAY, delay_time]
]+
]*
[ pin_name,
  MXREF
[
[ , _FORCE_I, f_current , _I_RNG , @1uA | @10uA | @100uA | @1mA | @10mA |
@100mA | @250mA ]
[ , _V_RNG, @6V | @12V | @24V | @48V, _MEAS_TYPE, MV
  , _CLAMP_V, clamp_value ]
[ , _SETTLING_TIME, @S0 | @S1 ]
[ , _RELAY, ON | OFF]
[ , _DELAY, delay_time]
]+
]*
[pin_name,
  MXUVI
[
[ , _FORCE_I, f_current , _I_RNG , @1uA | @10uA | @100uA | @1mA | @10mA |
@100mA | @1A ]
[ , _V_RNG, @2V | @4V | @6V | @12V , _MEAS_TYPE, MV
  , _CLAMP_V, clamp_value ]
```

```
[ , _SETTLING_TIME, @S0 | @S1 ]
[ , _RELAY, ON | OFF]
[ , _DELAY, delay_time]
]+
]*
[, _GDELay, delay_time]
]+
);
```

Parameter comment:
Pin_name : pin-def or pin_group name
variables & constants allowed
type: interger
**no_entry: illegal

pin_type_name : **MLDPS, MXDPS, MXREF, MXUVI** (this from .dec pin_type define)

f_curent: **MLDPS** : -1A to +1A
MXDPS : -2A to +2A
MXREF : -250mA to +250mA
MXUVI : -1A to +1A
variables & constants allowed
type: TIME or double
unit: A, pA, nA, uA, mA
**no_entry: keep current value

i_range: **MLDPS** :
@25mA: -25mA to +25mA
@500mA: -500mA to +500mA
@1A: -1A to +1A (only @6V range support)

MXDPS :
@100uA: -100uA to +100uA (start from @100uA)
@1mA: -1mA to +1mA
@10mA: -10mA to +10mA
@100mA: -100mA to +100mA
@1A: -1A to +1A
@2A: -2A to +2A

MXUVI :
@1uA: -1uA to +1uA
@10uA: -10uA to +10uA
@100uA: -100uA to +100uA
@1mA: -1mA to +1mA
@10mA: -10mA to +10mA
@100mA: -100mA to +100mA
@1A: -1A to +1A

MXREF :
@1uA: -1uA to +1uA
@10uA: -10uA to +10uA
@100uA: -100uA to +100uA
@1mA: -1mA to +1mA

@10mA: -10mA to +10mA
@100mA: -100mA to +100mA
@250mA: -250mA to +250mA
**no_entry: depend on f_current(auto-range)

v_range: **MLDPS** : @6V, @12V
MXDPS : @4V ,@8V ,@12V ,@16V
MXREF : @6V, @12V, @24V, @48V
MXUVI : @2V, @4V, @6V, @12V
**no_entry: keep current range

V_clamp: This values must be in the specified V_range, as follows:

MLDPS :

V_range	V_clamp value	Remark
@6V	-6V*1.1 to +6V*1.1	(V_clamp value must > ±6 V* 10%)
@12V	-12V*1.1 to +12V*1.1	(V_clamp value must > ±12 V* 10%)
@6V (pclamp _value)	+6V*0.1 to +6V*1.1	(V_clamp value must > +6 V* 10%)
@12V (pclamp _value)	+12V*0.1 to +12V*1.1	(V_clamp value must > +12V * 10%)
@6V (nclamp _value)	-6V*0.1 to -6V*1.1	(V_clamp value must > -6 V* 10%)
@12V (nclamp _value)	-12V*0.1 to -12V*1.1	(V_clamp value must > -12V * 10%)

MXDPS :

@4V : -4V to +4V
@8V : -8V to +8V
@12V : -12V to +12V
@16V : -16V to +16V

MXUVI :

V_range	V_clamp value	Remark
@2V	+2V*0.05 to +2V*1.1	(V_clamp value must > 2 V* 5%)
@4V	+4V*0.05 to +4V*1.1	(V_clamp value must > 4 V* 5%)
@6V	+6V*0.05 to +6V*1.1	(V_clamp value must > 6 V* 5%)
@12V	+12V*0.05 to +12V*1.1	(V_clamp value must > 12V * 5%)

MXREF :

V_range	V_clamp value	Remark
@6V	+6V*0.05 to +6V*1.1	(V_clamp value must > 6 V* 5%)
@12V	+12V*0.05 to +12V*1.1	(V_clamp value must > 12 V* 5%)
@24V	+24V*0.05 to +24V*1.1	(V_clamp value must > 24 V* 5%)
@48V	+48V*0.05 to +48V*1.1	(V_clamp value must > 48V * 5%)

variables & constants allowed
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: keep current value

ON/OFF: ON – turns on Power by marking Power utility relay after conditions setup

OFF – turns off Power by breaking Power utility relay after conditions setup
type: integer
**no_entry: keep current state

meas_type : MV (measure output voltage)
MI (measure output current)

**no_entry: Default = MV
// when m_type== VMM mode, then f_current will be 0uA (by FW setting)

SETTLING_TIME : This is the time of slew rate. @S0 has faster slew rate. If device has clamp current in @S0 (faster mode), user may get bigger ripple of current than @S1.

MXUVI Settling time :

FI :

I_range	@S0/ Settling time/(Faster mode)	@S1/Settling time/(Normal mode)
1uA	16ms	50ms
10uA	2.5ms	8ms
100uA	0.6ms	2ms
1mA	100us	250us
10mA	100us	250us
100mA	100us	250us
1A	100us	250us

**no_entry: Default = @S1

MXREF Settling time :

FI :

I_range	@S0/Settling time/(Faster mode)	@S1/Settling time/(Normal mode)
1uA	16ms	50ms
10uA	2.5ms	8ms
100uA	0.6ms	2ms
1mA	100us	250us
10mA	100us	250us
100mA	100us	250us
1A	100us	250us

**no_entry: Default = @S1

meas_type : MV (measure output voltage)
MI (measure output current)
**no_entry: Default = MV

delay_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 400uS

4.8.4.2 Example

```
FORCE_CURRENT(Vdd, MLDPS,
                    _FORCE_I, -100uA,
                    _I_RNG, @25mA,
                    _V_RNG, @12V,
                    _MEAS_TYPE, MV,
                    _CLAMP_V, 2V,
                    _DELAY, 1mS,
                    _RELAY , ON
                    );
JUDGE_VOLT( Vdd, MLDPS, -1.5V, -0.2V,
                _AVG_CNT, 1
            );
```

```
FORCE_CURRENT(Vdd, MLDPS,
                    _FORCE_I, -100uA,
                    _I_RNG, @25mA,
                    _V_RNG, @6V,
                    _MEAS_TYPE, MV,
                    _CLAMP_PV, 5V, _CLAMP_NV, -4V,
                    _DELAY, 1mS,
                    _RELAY , ON
                    );
JUDGE_VOLT( Vdd, MLDPS, -1.5V, -0.2V,
                _AVG_CNT, 1
            );
```

4.8.5 MEAS_VOLT

Perform a MLDPS or other board voltage measurement and return the measured value.
Release version : **CRAFT_3380_1.30**

4.8.5.1 Format

Parameter list:

```
MEAS_VOLT(   pin_name, pin_type_name, measured_values
                    [, _DELAY, delay_time ]
                    [, _AVG_CNT, average_count ]
                    [, _CPU_TRIG_INTERVAL, sampling_interval ]
                    [, _RET_MIN_MEAS, return_min_measured_value ]
                    [, _RET_MAX_MEAS, return_max_measured_value ]
);
```

Parameter comment:

DPS_name : pin-def or pin_group
variables & constants allowed
type: interger
**no_entry: illegal

Pin_type_name : **MLDPS, MXDPS, MXREF, MXUVI**

measured_value : 2-dimension array, array size: [site][pin]
return measured value,
measured_value[0][ ] : Site-0
measured_value[1][ ] : Site-1

delay_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 400uS

// Notice : if use average mode, then meas_type should use NORM mode,
otherwise compiler error

average_cnt: the average count
**no_entry: Default = 1

sampling_interval: the sampling interval between each measurement
**no_entry: Default= (minimum value)
MLDPS : minimum 5uS
MXDPS : minimum 5uS
MXUVI : minimum 10uS
MXREF : minimum 10uS

return_min_measured_value : 2-dimension array, array size: [site][pin]
return minimum measured value,
measured_value[0][ ] : Site-0

measured_value[1][ ] : Site-1

return_max_measured_value : 2-dimension array, array size: [site][pin]
return maximum measured value,
measured_value[0][ ] : Site-0
measured_value[1][ ] : Site-1

4.8.5.2 Example

```
CARRAY_2DIM<double> imeas( 2 , 1 );   // same with double imeas[2][1];2-site, 1-pin

FORCE_CURRENT(Vdd, MLDPS,
                    _FORCE_I, -100uA,
                    _I_RNG, @25mA,
                    _V_RNG, @12V,
                    _MEAS_TYPE, MV,
                    _CLAMP_V, 2V,
                    _DELAY, 1mS,
                    _RELAY , ON
                    );

MEAS_VOLT(   Vdd, MLDPS, imeas,
                    _DELAY, 2mS,
                    _AVG_CNT, 1,
                    _CPU_TRIG_INTERVAL,10uS
              );
```

4.8.6 JUDGE_VOLT

Perform a MLDPS or other board voltage measurement and compare the value against specified limits. Release version : **CRAFT_3380_1.30**

4.8.6.1 Format

Parameter list:

```
JUDGE_VOLT( pin_name, Brd_name, L_limit, H_limit
                    [,_DELAY, delay_time ]
                    [, _AVG_CNT, average_count ]
                    [, _CPU_TRIG_INTERVAL, sampling_interval ]
                    [, _SAMP_LOW_LIMIT, sample_low_limit ]
                    [, _SAMP_HIGH_LIMIT, sample_high_limit ]
                    [, _RET_MEAS,return_measured_value ]
                    [, _RET_MIN_MEAS, return_min_measured_value ]
                    [, _RET_MAX_MEAS, return_max_measured_value ]
);
```

Parameter comment:

pin_name : pin-def or pin_group
variables & constants allowed
type: interger
**no_entry: illegal

Brd_name : **MLDPS, MXDPS, MXREF, MXUVI**

low_limit: the low test limit
type: VOLTAGE or double
unit: V, mV
**no_entry: only high limit

high_limit: the high test limit
type: VOLTAGE or double
unit: V, mV
**no_entry: only low limit

delay_time; specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 500uS

average_cnt: the average count
**no_entry: Default = 1

Cpu_trig_interval: the sampling interval between each measurement
**no_entry: Default = (minimum value)
MLDPS : minimum 5uS
MXDPS : minimum 5uS
MXUVI : minimum 10uS
MXREF : minimum 10uS

sample_low_limit: the low limit of each measurement ; if any measurement smaller than this value, then the judge result will be set to fail

sample_high_limit: the high limit of each measurement ; if any measurement larger than this value, then the judge result will be set to fail

if rpt_min & rpt_max no_entry : skip sample_low_limit / sample_high_limit judgement

return_measured_value : 2-dimension array, array size: [site][pin]
return measured value,
measured_value[0][ ] : Site-0
measured_value[1][ ] : Site-1

return_min_measured_value : 2-dimension array, array size: [site][pin]
return minimum measured value,
measured_value[0][ ] : Site-0
measured_value[1][ ] : Site-1

return_max_measured_value : 2-dimension array, array size: [site][pin]
return maximum measured value,
measured_value[0][ ] : Site-0
measured_value[1][ ] : Site-1

4.8.6.2 Example

```
FORCE_CURRENT(Vdd, MLDPS,
                    _FORCE_I, -100uA,
                    _I_RNG, @25mA,
                    _V_RNG, @12V,
                    _MEAS_TYPE, MV,
                    _CLAMP_V, 2V,
                    _DELAY, 1mS,
                    _RELAY , ON
                    );

JUDGE_VOLT( Vdd, MLDPS, -1.5V, -0.2V,
              _AVG_CNT, 1
);
```

4.8.7 SET_DVM_MODE

System set power board as a voltage measurement. When program is in DVM mode, clamp voltage is not useful.

4.8.7.1 Format

Parameter list:
//When we use this statement have to use one and more ([ ]+) condition. Another is no and more ([ ]*) condition.

```
SET_DVM_MODE(
    [       [ pin_name,
              MLDPS
              , V_RNG, @6V | @12V
              , _RELAY, ON | OFF
              , _DELAY, delay_time
            ]*,
            [ pin_name,
              MXDPS
            , _V_RNG, @4V | @8V | @12V | @16V
            , _RELAY, ON | OFF
            , _DELAY, delay_time
            ]*,
            [ pin_name,
              MXREF
              , _V_RNG, @6V | @12V | @24V | @48V
              , _RELAY, ON | OFF
              , _DELAY, delay_time
            ]*,
            [ pin_name,
              MXUVI
            , _V_RNG, @2V | @4V | @6V | @12V
            , _RELAY, ON | OFF
            , _DELAY, delay_time
            ]*
  ]+
);
```

Parameter comment:

pin_name : pin-def or pin_group name
variables & constants allowed
type: interger
**no_entry: illegal

pin_type_name : **MLDPS/MXDPS/MXREF/MXUVI** //pin_type_name from .dec

v_range: **MLDPS** : @6V : -6V ~ +6V
@12V : -6V ~ +12V
// per-board ch0~ch15 must be the same range.
// per-board ch16~ch31 must be the same range.

MXREF : @6V: -6V to +6V
@12V: -12V to +12V
@24V: -24V to +24V

@48V: -48V to +48V

MXDPS :

@4V : -4V to +4V
@8V : -8V to +8V
@12V : -12V to +12V
@16V : -16V to +16V

MXUVI :

@2V: -2V to +2V
@4V: -4V to +4V
@6V: -6V to +6V
@12V: -12V to +12V

type: integer
**no_entry: depend on f_volt (auto-range)

relay: ON – turns on Power by marking Power utility relay after conditions setup.
OFF – turns off Power by breaking Power utility relay after conditions setup.
type: integer
**no_entry: keep current state

4.8.7.2 Example

```
SET_DVM_MODE( vdd,    MXDPS
                ,V_RNG, @12V
                , _RELAY, ON
                , _DELAY, 2mS
              );
…
…
JUDGE_VOLT( Vdd, MXDPS, -1.5V, -0.2V,
                _AVG_CNT, 32
              );
```

4.9 MXPMU Control Statement

4.9.1 FORCE_I_PMU

This statement sets the PMU in current force mode, and programs the respective current/voltage conditions. @250mA release version : **CRAFT_3380_2.00_SP1**

4.9.1.1 Format

Parameter list:

FORCE_I_PMU (PMU_number[|PMU_number], f_current, I_range, V_range, V_clamp, ON/OFF, wait_time);

Parameter comment:

PMU_number: "PMU0~63"
PMU : if outside of "PIN_INC" , or "PARAL_INC", then all PMU relay connected pins will be programmed.
if inside of "PIN_INC", or "PARAL_INC", then ONLY the PMU relay connected pins and must belong to the pin group of "PIN_INC" , "PARALLEL_INC" or "PARAL_INC" will be programmed .
Type: integer
No entry: Default = PMU

f_current: The amount of force current from –250mA to +250mA by the selected PMU.
Variables and constants are allowed.
Type: CURRENT or double
Unit: A, pA, nA, uA, mA
No entry: keep current value

I_range: One of six possible hardware ranges. For the best accuracy, you should program the current by using the closest range as possible.

- Possible hardware range:

I_range	Minimum	Maximum
@1uA	-1uA	+1uA
@10uA	-10uA	+10uA
@100uA	-100uA	+100uA
@1mA	-1mA	+1mA
@10mA	-10mA	+10mA
@100mA	-100mA	+100mA
@250mA	-250mA	+250mA

No entry: depend on f_current (auto-range)

V_range: One of four possible hardware ranges. For the best accuracy, you should program the current by using the closest range as possible.

V_range	V_clamp value	Remark
@6V	+6V*0.05 to +6V*1.1	(V_clamp value must > 6 V* 5%)
@12V	+12V*0.05 to +12V*1.1	(V_clamp value must > 12 V* 5%)
@24V	+24V*0.05 to +24V*1.1	(V_clamp value must > 24 V* 5%)
@48V	+48V*0.05 to +48V*1.1	(V_clamp value must > 48V * 5%)

Type: integer
No entry: keep current range

V_calmp: The maximum current to be supplied by the selected PMU. One of four possible hardware ranges. This value must be in the specified V_range, as following:

- Possible hardware range:

V_range	V_clamp value Minimum	V_clamp value Maximum	Remark
@6V	-6.0V	+6.0V	(V_clamp value must > 6 V* 5%)
@12V	-12V	+12V	(V_clamp value must > 12 V* 5%)
@24V	-24V	+24V	(V_clamp value must > 24 V* 5%)
@48V	-48V	+48V	(V_clamp value must > 48V * 5%)

Variables and constants are allowed.
Type: VOLTAGE or double
Unit: V, pV, nV, uV, mV
No entry: keep current value

ON/OFF: The state of specified PMU output buffer. There are two possible states:
ON: turns on PMU by marking PMU utility relay after conditions setup.
OFF: turns off PMU by breaking PMU utility relay after conditions setup.
Type: integer
No entry: keep current state

[Notice]: Force ON and Force OFF procedure
Force ON : Relay closes first and force setting value
Force OFF : Force setting value first and Relay open

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 3mS@1uA, 400uS for all other ranges.

4.9.1.2 Explanation

This statement allows the user to just change the current value previously programmed, to change the ranges and clamps, to turn it on, or to turn it off. Each of these fields are indecently programmed.

4.9.1.3 Example

```
RELAY_ON(P1_4_6_7,PDCL,2mS);
PIN_INC(P1_4_6_7)
{
 RELAY_OFF(PIN_DEF,PDCL,1mS);
 RELAY_ON(PIN_DEF,PMU,1mS);
 FORCE_I_PMU(PMU,-100uA,@1mA,@6V,2V,ON,3mS);
 JUDGE_V_PMU(PMU,IFVM,-1.2V,-0.2V,2mS);
 RELAY_OFF(PIN_DEF,PMU,1mS);
 RELAY_ON(PIN_DEF,PDCL,1mS);
}
```

4.9.1.4 Relative

JUDGE_V_PMU , PARAL_INC , PIN_INC

4.9.2 JUDGE_I_PMU

This statement prompts the execution of the voltage loading current measurement (VFIM) by PMU to generate the pass or fail judgment.

4.9.2.1 Format

Parameter list:

```
JUDGE_I_PMU (PMU_number, I_min, I_max, wait_time [, ave_cnt, ave_mode,
                    sampling_interval, ave_min, ave_max ]) :
```

Parameter comment:

PMU_number: "PMU0~31"
PMU : if outside of "PIN_INC" , or "PARAL_INC", then all PMU relay connected pins will be tested .
if inside of "PIN_INC", or "PARAL_INC", then ONLY the PMU relay connected pins and must belong to the pin group of "PIN_INC" , "PARALLEL_INC" or "PARAL_INC" will be tested .
Type: integer
No entry: Default = PMU

I_min: Refer to the minimum current value to generate the pass or fail. Omitting the parameter is possible, but the comparison will be made only regarding I_max. When this happens, omitting I_max description is not allowed.
This is lower limit value.
Variables and constants are allowed.
Type: CURRENT or double
Unit: A, pA, nA, uA, mA
No entry: only high limit

I_max: Refer to the maximum current value to generate the pass or fail. Omitting the parameter is possible, but the comparison will be made only regarding I_min. When this happens, omitting I_min description is not allowed.
This is upper limit value.
Variables and constants are allowed.
Type: CURRENT or double
Unit: A, pA, nA, uA, mA
No entry: only low limit

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 500uS

ave_cnt: The average count.

No entry: Default = 1

ave_mode: The average mode
AVE: average
RMS: root mean square
Type: int
No entry: Default = AVE

sampling_interval: The sampling interval is between each measurement.
**no_entry: Default = 50uS (minimum value)

ave_min: The low limit of each measurement ; if any measurement is smaller than this value, then the judge result will be set to fail.

ave_max: The high limit of each measurement ; if any measurement is larger than this value, then the judge result will be set to fail.

4.9.2.2 Explanation

This statement generates the pass or fail result by checking whether the current value produced as the test result comes within the scope of the current value setting. When the test result comes within the set scope, a pass result is generated, but when the test result comes outside the set scope, the fail result is generated. The statement "FORCE_I_PMU " must be executed in advance.

4.9.2.3 Example

```
RELAY_ON(P1_4_6_7,PDCL,2mS);
PIN_INC(P1_4_6_7)
{
  RELAY_OFF(PIN_DEF,PDCL,1mS);
  RELAY_ON(PIN_DEF,PMU,1mS);
  FORCE_V_PMU (PMU, 5V, @6V, @1mA, 1mA, ON, 2mS);
  JUDGE_I_PMU (PMU, 300uA, 700uA, 2mS, 16, AVE, 50uS, , );
  RELAY_OFF(PIN_DEF,PMU,1mS);
  RELAY_ON(PIN_DEF,PDCL,1mS);
}
```

4.9.2.4 Relative

FORCE_V_PMU , PARAL_INC , PIN_INC

4.9.3 MEAS_I_PMU

Perform a PMU current measurement, and save the measured current value to system buffer memory. Read the measured result from the "RESULT_PIN" statement.

4.9.3.1 Format

Parameter list:

MEAS_I_PMU (PMU_number, wait_time**[**, ave_cnt, ave_mode, sampling_interval**]**) :

Parameter comment:

PMU_number: "PMU0~31".
PMU : if outside of "PIN_INC" , or "PARAL_INC", then all PMU relay connected pins will be tested .
if inside of "PIN_INC", or "PARAL_INC", then ONLY the PMU relay connected pins and must belong to the pin group of "PIN_INC" , "PARALLEL_INC" or "PARAL_INC" will be tested .
Type: integer
No entry: Default = PMU

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 500uS

ave_cnt: The average count.
No entry: Default = 1

ave_mode: The average mode.

AVE: average

RMS: root mean square
Type: int
No entry: Default = AVE

sampling_interval: The sampling interval between each measurement

4.9.3.2 Explanation

This statement prompts the execution of the current measurement by PMU and stores the measured value as the system variable. The statement "FORCE_I_PMU " must be executed in advance.

4.9.3.3 Example

```
MEAS_I_PMU( PMU, 2mS, 16, AVE, 50uS);
imeas = RESULT_PIN(S0, VEE);
JUDGE_VARIABLE (S0, imeas, 1uA, 10uA, "rst");
```

4.9.3.4 Relative

RESULT_PIN , JUDGE_VARIABLE , PARAL_INC , PIN_INC

4.9.4 FORCE_V_PMU

This statement sets the PMU in voltage force mode, and programs the respective voltage/current conditions. @250mA release version : **CRAFT_3380_2.00_SP1**

4.9.4.1 Format

Parameter list:

FORCE_V_PMU (PMU_number[|PMU_number], voltage, V_range, I_range, I_clamp, ON/OFF, wait_time);

Parameter comment:

PMU_number: "PMU0~63".
PMU : if outside of "PIN_INC" , or "PARAL_INC", then all PMU relay connected pins will be programmed.
if inside of "PIN_INC", or "PARAL_INC", then ONLY the PMU relay connected pins and must belong to the pin group of "PIN_INC" , "PARALLEL_INC" or "PARAL_INC" will be programmed .
Type: integer
No entry: Default = PMU

voltage: The voltage to be applied by the selected PMU. This is a value that can be programmed from -48V to +48V.
Variables and constants are allowed
Type: VOLTAGE or double
Unit: V, pV, nV, uV, mV
No entry: keep current value

V_range: One of four possible hardware ranges. For the best accuracy, you should program the current by using the closest range as possible.

V_range	Minimum	Maximum
@6V	-6V	+6V
@12V	-12V	+12V
@24V	-24V	+24V
@48V	-48V	+48V

Type: integer48
**45V~48V,maximum current is 100mA
No entry: depend on voltage (auto-range)

I_range: One of six possible hardware ranges. For the best accuracy, you should program the current by using the closest range as possible.

- Possible hardware range:

I_range	Minimum	Maximum
@1uA	-1uA	+1uA
@10uA	-10uA	+10uA
@100uA	-100uA	+100uA
@1mA	-1mA	+1mA

@10mA	-10mA	+10mA
@100mA	-100mA	+100mA
@250mA	-250mA	+250mA

No entry: keep current range

I_calmp: The maximum current to be supplied by the selected PMU. One of four possible hardware ranges. This value must be in the specified I_range, as
following:

- Possible hardware range:

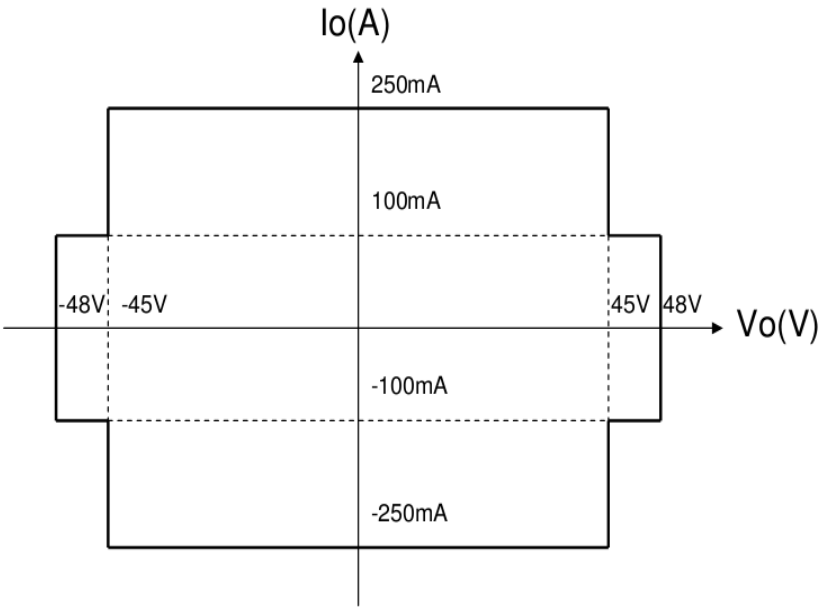
I_range	I_clamp value Minimum	I_clamp value Maximum	Remark
@1uA	-1.00uA	1.00uA	(I_clamp value must 1.00uA * 5%)
@10uA	-10.0uA	10.0uA	(I_clamp value must 10.0uA * 5%)
@100uA	-100uA	100uA	(I_clamp value must 100uA * 5%)
@1mA	-1.00mA	1.00mA	(I_clamp value must 1.00mA * 5%)
@10mA	-10.0mA	10.0mA	(I_clamp value must 10.0mA * 5%)
@100mA	-100mA	100mA	(I_clamp value must 100mA * 5%)
@250mA	-250mA	250mA	I_clamp value must > 250mA * 2%

Variables and constants are allowed
Type: CURRENT or double
Unit: A, pA, nA, uA, mA
No entry: keep current value

ON/OFF: The state of the specified PMU output buffer. There are two possible states:
ON: Turns on PMU by marking PMU utility relay after conditions setup.
OFF: Turns off PMU by breaking PMU utility relay after conditions setup.
Type: integer
No entry: keep current state
[Notice]: Force ON and Force OFF procedure
Force ON : Relay closes first and force setting value
Force OFF : Force setting value first and Relay open

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 2mS @1uA, 500uS for all other ranges.

** MXPMU operating V/I curve **



4.9.4.2 Explanation

This statement sets the PMU in voltage force mode, and programs the respective voltage/current conditions.

4.9.4.3 Example

FORCE_V_PMU (PMU, 5V, @6V, @ 100mA, 100mA,ON, 4mS); **FORCE_V_PMU** (PMU0, 0V, @6V, @100mA, 100mA, OFF, 1mS);

4.9.4.4 Relative

JUDGE_I_PMU , PARAL_INC , PIN_INC

4.9.5 JUDGE_V_PMU

This statement prompts the execution of the current loading voltage measurement (IFVM) or the voltage measurement (VMM) by PMU to generate pass or fail judgment.

4.9.5.1 Format

Parameter list:

```
JUDGE_V_PMU (PMU_number , judge_mode, V_min, V_max, wait_time
                [, ave_cnt, ave_mode, sampling_interval] [, ave_min, ave_max ]);
```

Parameter comment:

PMU_number: "PMU0~31"
PMU: if outside of "PIN_INC" , or "PARAL_INC", then all 32 PMUs will be programmed . If inside of "PIN_INC", or "PARAL_INC", then only the PMU that belong to the pin group of "PIN_INC" or "PARALLEL_INC" or "PARAL_INC" will be programmed .
Type: integer
No entry: Default = PMU

judge_mode: IFVM (The current loading voltage measurement is executed.)
VMM (The voltage measurement only is executed.)

V_min: Refer to the minimum voltage value is used for comparison to generate the pass or fail judgment during the current measurement test.
Omitting the parameter is possible, but the comparison will be made only regarding V_max. When this happens, omitting V_max description is not allowed.
This is the lower limit value.
Variables and constants are allowed
Type: VOLTAGE or double
Unit: V, pV, nV, uV, mV
No entry: only high limit

V_max: Refer to the maximum voltage value is used for comparison to generate the pass or fail judgment during the current measurement test.
Omitting the parameter is possible, but the comparison will be made only regarding V_min. When this happens, omitting V_min description is not allowed.
This is the upper limit value.
Variables and constants are allowed
Type: VOLTAGE or double
Unit: V, pV, nV, uV, mV
No entry: only low limit

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 500uS

ave_cnt: The average count
No entry: Default = 1
ave_mode: The average mode

AVE: average

RMS: root mean square
Type: int
No entry: Default = AVE

sampling_interval: The sampling interval is between each measurement.
**no_entry: Default = 50uS (minimum value)

ave_min: The low limit of each measurement; if any measurement is smaller than this value, then the judge result will be set to fail

ave_max: The high limit of each measurement; if any measurement larger than this value, then the judge result will be set to fail

4.9.5.2 Explanation

This statement prompts the execution of the current loading voltage measurement (IFVM) or the voltage measurement (VMM) by PMU to generate pass or fail judgment.

4.9.5.3 Example

```
JUDGE_V_PMU (PMU, IFVM, 2.4V, 5V, 2mS);
JUDGE_V_PMU (PMU, IFVM, 2.4V, 5V, 2mS, 32, AVE, 50uS, , );
```

4.9.5.4 Relative

FORCE_I_PMU , PARAL_INC , PIN_INC

4.9.6 MEAS_V_PMU

Perform a PMU voltage measurement, and save the measured voltage value to system buffer memory. Read the measured result from the "RESULT_PIN" statement.

4.9.6.1 Format

Parameter list:

```
MEAS_V_PMU (PMU_number, judge_mode, wait_time[, ave_cnt, ave_mode,
                sampling_interval]) ;
```

Parameter comment:

PMU_number: "PMU0~31"
PMU: if outside of "PIN_INC" , or "PARAL_INC", then all 32 PMUs will be programmed . If inside of "PIN_INC", or "PARAL_INC", then only the PMU that belong to the pin group of "PIN_INC" or "PARALLEL_INC" or "PARAL_INC" will be programmed .
Type: integer
No entry: Default = PMU

judge_mode: IFVM (The current loading Voltage measurement is executed)
VMM (The voltage measurement only is executed)

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 500uS

ave_cnt: The average count.
No entry: Default = 1

ave_mode: The average mode.

AVE: average

RMS: root mean square
Type: int
No entry: Default = AVE

sampling_interval: The sampling interval is between each measurement.
**no_entry: Default = 50uS (minimum value)

4.9.6.2 Explanation

This statement prompts the execution of voltage measurement by PMU and stores the measured value as the system variable. The statement "FORCE_V_PMU " must be executed in advance.

4.9.6.3 Example

```
MEAS_V_PMU (PMU, IFVM, 1mS, 16, AVE, 50uS);
imeas = RESULT_PIN(S0, VEE);
JUDGE_VARIABLE (S0, imeas, 1uA, 10uA, "rst");
```

4.9.6.4 Relative

RESULT_PIN, JUDGE_VARIABLE , PARAL_INC , PIN_INC

4.9.7 PARAL_INC

This is a program flow control statement. It provides a compact test loop based on the reserved word PIN_DEFS as the loop increment. PIN_DEFs are used to represent each pin_group in the loop, one at a time.

4.9.7.1 Format

Parameter list:

```
PARAL_INC (pin_name)   {
        statement;
        .
        .
        statement;
}
```

Parameter comment:

pin_name: Pin_def and pin_group can be used.

No entry: illegal

statement: Any valid tester or C statement.

4.9.7.2 Explanation

This is a program flow control statement. It provides a compact test loop based on the reserved word PIN_DEFS as the loop increment. PIN_DEFs are used to represent each pin_group in the loop, one at a time.

4.9.7.3 Example

```
PARAL_INC(allsegpin)
    {
        FORCE_V_PMU(PMU,5V,@12V,@1mA,1mA,ON,2mS);
        LCD_RELAY_ON(PIN_DEFS,PMU,1mS);
        MEAS_I_PMU(PMU, 2mS,50, AVE, 50uS );
        FORCE_V_PMU(PMU,0V,@12V,@1mA,1mA,OFF,2mS);
        LCD_RELAY_OFF(PIN_DEFS,PMU,1mS);

    }
    FORCE_V_PMU(PMU, 0V, @6V, @10mA, 10mA, OFF, 2mS);
    a=0;
PIN_INC(allsegpin)
{
ISEG[a]=RESULT_PIN(S0,PIN_DEF);
a++;
}
```

4.9.8 PIN_INC

This is a program flow control statement. It provides a compact test loop based on the reserved word PIN_DEF as the loop increment. PIN_DEF is used to represent each pin in the loop, one at a time.

PMU0~15 service ate-channel 0~319, PMU16~31 service ate-channel 320~607.

4.9.8.1 Format

Parameter list:

```
PIN_INC (pin_name   [, resource])   {
        statement;
        .
        .
        statement;      }
```

Parameter comment:

pin_name: Pin_def and pin_group can be used.
No entry: illegal

resource : PMU : To avoid using same PMU channel at one time.
LOOP_BY_PIN : Always keep one pin_name at one time, test the loop-pin. And loop count will be total pin numbers.
**no_entry: PMU

statement: Any valid tester or C statement.

Dec File

```
//e.g;   4 site, per-site 2 pin
P0=   0 : 16 : 24 : 32 = 0 = IO ;
P1=   1 : 17 : 18 : 19 = 1 = IO ;
```

Pln File

```
PIN_INC(P0+P1, LOOP_BY_PIN)
{           //1st loop : channel 0/16/24/32
…           //2nd loop : 1/17/18/19
}

PIN_INC(P0+P1)
{           //1st loop : channel 0/1/18/19/24
…           //2nd loop : channel 16/17
            //3rd loop : channel 32
}
```

4.9.8.2 Example

```
PIN_INC (inputs) {
        RELAY_ON (PIN_DEF, PMU, 1mS);
        JUDGE_V_PMU( PMU, VMM, 2.4V, 5V, 2mS )
        RELAY_OFF (PIN_DEF, PMU, 1mS);
    }

FORCE_V_PMU (PMU, 0V, @6V, @1mA, 1mA, ON, 2mS);
PIN_INC (ost_pins, LOOP_BY_PIN ) {
        RELAY_OFF (PIN_DEF, PDCL, 1mS);
        RELAY_ON (PIN_DEF, PMU, 1mS);
        JUDGE_ V_PMU (PMU, -100uA, -1.2V, -0.2V, 2mS);
        RELAY_OFF (PIN_DEF, PMU, 1mS);
    }



//20190109: to test one time 1-pin/per-site     //for PPMU example
PIN_INC( VolPins, LOOP_BY_PIN ) {
        RELAY_ON(PIN_DEF, PPMU_PDCL, 2mS);
        FORCE_I_PPMU(PIN_DEF, -2mA, @2mA,   ,-1.5V, 6V, 2mS);
        JUDGE_V_PPMU(PIN_DEF, 2.5V,   , 2mS);
        FORCE_V_PPMU(PIN_DEF,0V,   ,@2mA,   ,2mS);
        RELAY_OFF(PIN_DEF, PPMU_PDCL, 1mS);
        }
```

4.9.9 READ_PIN_INC

This is a program flow control statement, especially for read the measured result.

4.9.9.1 Format

Parameter list:

```
READ_PIN_INC (pin_name){
        Value = RESULT_PIN( socket_no, pin_name) ;
        Value = RESULT_DPS(socket_no, DPS_name ) ;
        Value = RESULT_PREF(socket_no, PREF_name) ;
        .statement
}
```

Parameter comment:

pin_name: Pin_def and pin_group can be used.
No entry: illegal

socket_no : Specifies the socket can be selected.
S0: socket No. 0
S1: socket No. 1
S2: socket No. 2
S3: socket No. 3
Type: integer
No entry: illegal

statement: Any valid C statement.

4.9.9.2 Explanation

This is a program flow control statement, especially for read the measured result. It provides a compact test loop based on the reserved word PIN_DEF as the pin loop increment, and the reserved word SOCKET as the site loop increment. PIN_DEF is used to represent each pin in the loop, and SOCKET is used to represent each socket in the socket loop, one at a time.

4.9.9.3 Example

```
//== 2-site program ==//
double v_ary[12];   //array_size: 2site* 6pin
int lp=0; //init
READ_PIN_INC (Iil_p4mA)     //Iil_p4mA= CLR+G1+G2+SL+SR+CLK;
{
lp++;
v_ary[SOCKET]= RESULT_PIN(SOCKET, PIN_DEF);
printf("**** Sel-pins iil_leak_test (Socket_%d -- pin_%d) :
iil leakage current= %8.4fuA\n", SOCKET, lp, v_ary[SOCKET]*1.0e6);
DEBUG_BREAK();
}
```

4.10 PPMU Control Statement

4.10.1 FORCE_V_PPMU

This statement sets the Per-pin PMU in voltage force mode, and programs the Voltage conditions.

4.10.1.1 Format

Parameter List:

```
FORCE_V_PPMU(pin_name, f_volt, v_range, I_range, I_clamp, wait_time) ;
```

Parameter comment:

pin_name: pin_def, pin_group (only suit for IO-Pins)
**no_entry: illegal

f_volt: -2V to +6V
variables & constants allowed
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: keep current value

V_range: no use (spare)
**no_entry: depend on voltage(auto-range)

I_range: @2uA, @20uA, @200uA, @2mA, @32mA
// type: integer
@2uA: -2uA to +2uA
@20uA: -20uA to +20uA
@200uA: -200uA to +200uA
@2mA: -2mA to +2mA
@32mA: -32mA to +32mA
**no_entry: keep current range

I_calmp: no use (spare)
(MXLPC PPMU not support I-clamp function)

wait_time specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 500uS

4.10.1.2 Example

```
RELAY_ON(leak_pins,PPMU_PDCL, 1mS);
FORCE_V_PPMU(leak_pins, 5V,   ,@2uA,   , 3mS);
```

```
JUDGE_I_PPMU(leak_pins, -1uA, 1uA, 3mS);
```

4.10.2 FORCE_V_PPMU_X

This statement sets the Per-pin PMU in voltage force mode, and programs the Voltage conditions; and set up PPMU level-set memory in the same time.
// (to avoid LEVELS statement change force value)..

Release version : **CRAFT_3380_2.50-SP3**

4.10.2.1 Format

Parameter List:

```
FORCE_V_PPMU_X(pin_name, f_volt, v_range, I_range, I_clamp, wait_time,
lvl_no) ;
```

Parameter comment:

pin_name: pin_def, pin_group (only suit for IO-Pins)
**no_entry: illegal

f_volt: -2V to +6V
variables & constants allowed
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: keep current value

V_range: no use (spare)
**no_entry: depend on voltage(auto-range)

I_range: @2uA, @20uA, @200uA, @2mA, @32mA
// type: integer
@2uA: -2uA to +2uA
@20uA: -20uA to +20uA
@200uA: -200uA to +200uA
@2mA: -2mA to +2mA
@32mA: -32mA to +32mA
**no_entry: keep current range

I_calmp: no use (spare)
(MXLPC PPMU not support I-clamp function)

wait_time specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 500uS

lvl_no: a level set number 1 ~ 249 (0, 250~255: system use)
variabales & constants allowed
type: integer
**no_entry: illegal

4.10.2.2 Example

```
RELAY_ON(leak_pins,PPMU_PDCL, 1mS);
FORCE_V_PPMU_X(leak_pins, 5V,    ,@2uA,    , 3mS, iil_leak_lvl);
LEVELS(iil_leak_lvl, 3mS);    //this will keep PPMU force-value un-changed !
...
JUDGE_I_PPMU(leak_pins, -1uA, 1uA, 3mS);
```

4.10.3 FORCE_I_PPMU

this statement sets the Per-pin PMU in current force mode, and programs the current conditions.

4.10.3.1 Format

Parameter List:

```
FORCE_I_PPMU(pin_name, f_curent, I_range, v_range, pv_clamp, nv_clamp,
wait_time) ;
```

Parameter comment:

pin_name: pin_def, pin_group (only suit for IO-Pins)
**no_entry: illegal

f_current: -32mA to +32mA
variables & constants allowed
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry: keep current value

I_range: @2uA, @20uA, @200uA, @2mA, @32mA
// type: integer
@2uA: -2uA to +2uA
@20uA: -20uA to +20uA
@200uA: -200uA to +200uA
@2mA: -2mA to +2mA
@32mA: -32mA to +32mA
**no_entry: keep current range

V_range: no use (spare)
**no_entry: depend on voltage(auto-range)

pv_clamp: Positive-Voltage Clamp
This values must be in the specified V_range, as follows:
Range : 0V to +6V
variables & constants allowed
type: VOLTAGE or double

unit: V, pV, nV, uV, mV
**no_entry: keep current value

nv_clamp: Negative-Voltage Clamp
This values must be in the specified V_range, as follows:
Range : -2V to +4V
variables & constants allowed
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: keep current value
Notice : pv_clamp must > nv_clamp (driver will check this)

wait_time: pecify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 500uS

4.10.3.2 Explanation

When we use PPMU to set FIMV(force current measure voltage) mode, the following settings should be avoid in the same time.

i. PPMU channel is FIMV mode and wothout any loading in the LB.
ii. Current range is @2uA, the smallest range in PPMU.
iii. The driving/sink current is small than 60nA.

All of that reasons, PPMU voltage will not go to a clamp value. To measure the voltage value could be 0.7~0.8V.

Because component of PPMU has this characteristic, setting current must be > 60nA. If not doing this(current >60nA), the measure voltage value will be a very small one. And it have the risk of pseudo pass issue while the measure value will not go to a user setting clamp value.

Other if JUDGE_PAT is following FORCE_I_PPMU, please notice to add LEVELS before JUDGE_PAT. Following example of “PPMU and pattern”.

4.10.3.3 Example

```
RELAY_ON(os_pins, PPMU_PDCL, 1mS);
FORCE_I_PPMU(os_pins, -100uA, @200uA,    ,2V, -2V ,3mS);
JUDGE_V_PPMU(os_pins, -1.2V, -0.2V, 2mS);
```

PPMU and pattern

```
RELAY_ON(os_pins, PPMU_PDCL, 1mS);
FORCE_I_PPMU(os_pins, -100uA, @200uA,    ,2V, -2V ,3mS);
JUDGE_V_PPMU(os_pins, -1.2V, -0.2V, 2mS);
FORCE_V_PPMU(os_pins, 0V,    ,@200uA,    , 3mS);
RELAY_OFF(os_pins, ALL, 1mS);

RELAY_ON(os_pins, PDCL, 1mS);
SET_LEVELN(os_lvl, IO_ALLPINS, 0V , 1.8V , -1.2V, -0.2V, 200uA, -200uA, 3V);
```

```
LEVELS(os_lvl,2mS);
JUDGE_PAT(OS,__OS);
```

4.10.4 FORCE_I_PPMU_X

this statement sets the Per-pin PMU in current force mode, and programs the current conditions. And set up PPMU level-set-memory in the same time(to avoid LEVELS statement change force value).

Release version : **CRAFT_3380_2.50-SP3**

4.10.4.1 Format

Parameter List:

FORCE_I_PPMU_X(pin_name, f_curent, I_range, v_range, **pv**_clamp, **nv**_clamp, wait_time, lvl_no) ;

Parameter comment:

pin_name: pin_def, pin_group (only suit for IO-Pins)
**no_entry: illegal

f_current: -32mA to +32mA
variables & constants allowed
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry: keep current value

I_range: @2uA, @20uA, @200uA, @2mA, @32mA
// type: integer
@2uA: -2uA to +2uA
@20uA: -20uA to +20uA
@200uA: -200uA to +200uA
@2mA: -2mA to +2mA
@32mA: -32mA to +32mA
**no_entry: keep current range

V_range: no use (spare)
**no_entry: depend on voltage(auto-range)

pv_clamp: Positive-Voltage Clamp
This values must be in the specified V_range, as follows:
Range : 0V to +6V
variables & constants allowed
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: keep current value

nv_clamp: Negative-Voltage Clamp

This values must be in the specified V_range, as follows:
Range : -2V to +4V
variables & constants allowed
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: keep current value
Notice : pv_clamp must > nv_clamp (driver will check this)

wait_time: pecify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 500uS

lvl_no: a level set number 1 ~ 249 (0, 250~255: system use)
variabales & constants allowed
type: integer
**no_entry: illegal

4.10.4.2 Example

```
RELAY_ON(os_pins, PPMU_PDCL, 1mS);
FORCE_I_PPMU_X(os_pins, -100uA, @200uA,    ,2V, -2V ,3mS, os_lvl);
LEVELS(os_lvl,2mS);
…
JUDGE_V_PPMU(os_pins, -1.2V, -0.2V, 2mS);
```

PPMU and pattern

```
RELAY_ON(os_pins, PPMU_PDCL, 1mS);
FORCE_I_PPMU_X(os_pins, -100uA, @200uA,    ,2V, -2V ,3mS, os_lvl);
JUDGE_V_PPMU(os_pins, -1.2V, -0.2V, 2mS);
FORCE_V_PPMU_X(os_pins, 0V,    ,@200uA,    , 3mS, os_lvl);
RELAY_OFF(os_pins, ALL, 1mS);

RELAY_ON(os_pins, PDCL, 1mS);
SET_LEVELN(os_lvl, IO_ALLPINS, 0V , 1.8V , -1.2V, -0.2V, 200uA, -200uA, 3V);
LEVELS(os_lvl,2mS);
JUDGE_PAT(OS,__OS);
```

4.10.5 JUDGE_I_PPMU

Perform a PPMU current measurement and compare the value against specified limits

4.10.5.1 Format

Parameter List:

JUDGE_I_PPMU(pin_name, I_min, I_max, wait_time [, ave_cnt, ave_mode, sampling_interval , rpt_min, rpt_max] [, **meas_value**]);

Parameter comment:

pin_name: pin_def, pin_group (only suit for IO-Pins)
**no_entry: illegal

I_min: lower limit value, variables & constants allowed
range: -32mA~32mA ;
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry: only high limit

I_max: upper limit value, variables & constants allowed
range: -32mA~32mA ;
type: CURRENT or double
unit: A, pA, nA, uA, mA
**no_entry: only low limit

wait_time; specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 6mS

ave_cnt: t he average count
**no_entry: Default = 1

ave_mode: the average mode
AVE : average
RMS : root mean square
type: int
**no_entry: Default = AVE

sampling_interval: the sampling interval between each measurement
**no_entry: Default= 10uS (minimum value)

rpt_min: the low limit of each measurement ; if any measurement smaller than this value, then the judge result will be set to fail

rpt_max: the high limit of each measurement ; if any measurement larger than this value, then the judge result will be set to fail
if rpt_min & rpt_max no_entry : skip the rpt_min/rpt_max judgement

meas_value: return the measured-values
type double or CURRENT array
e.g. CARRAY_2DIM <double> meas_value[site_no][pin_count];
**no_entry: means not return measured-values
// only return the active-sites measured-values

4.10.5.2 Example

```
JUDGE_I_PPMU (data_bus, -10uA, 10uA, 2mS);
JUDGE_I_PPMU (data_bus, -1uA, 1uA, 1mS, 8, AVE, , , );
JUDGE_I_PPMU (data_bus, -1uA, 1uA, 1mS, 8, AVE, , , ,meas_value);
```

4.10.6 JUDGE_V_PPMU

Perform a PPMU Voltage measurement and compare the value against specified limits.

4.10.6.1 Format

Parameter List:

JUDGE_V_PPMU(pin_name, V_min, V_max, wait_time [, ave_cnt, ave_mode, sampling_interval , rpt_min, rpt_max] [, **meas_value**]);

Parameter comment:

pin_name: pin_def, pin_group (only suit for IO-Pins)
**no_entry: illegal

V_min: lower limit value, variables & constants allowed
range : -2V~6V
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: only high limit

V_max: upperer limit value, variables & constants allowed
range : -2V~6V
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: only low limit

wait_time; specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 500uS

ave_cnt: the average count
**no_entry: Default = 1

ave_mode: the average mode
AVE : average
RMS : root mean square
type: int
**no_entry: Default = AVE

sampling_interval: the sampling interval between each measurement
**no_entry: Default = 10uS (minimum value)

rpt_min: the low limit of each measurement ; if any measurement smaller than this value, then the judge result will be set to fail

rpt_max: the high limit of each measurement ; if any measurement larger than this value, then the judge result will be set to fail

meas_value: return the measured-values

type double or CURRENT array
e.g. CARRAY_2DIM <double> meas_value[site_no][pin_count];
**no_entry: means not return measured-values
// only return the active-sites measured-values

4.10.6.2 Example

```
JUDGE_V_PPMU(outputs, 2.4V, 5V, 2mS);
```

4.10.7 MEAS_I_PPMU

Perform a PPMU current measurement and save the measured value to system buffer. Read the result by the "RESULT_PIN" statement .

4.10.7.1 Format

Parameter List:

```
MEAS_I_PPMU(pin_name, wait_time [, ave_cnt, ave_mode, sampling_interval]
[,meas_value]);
```

Parameter comment:

pin_name : pin-def or pin_group
variables & constants allowed
type: interger
**no_entry: illegal

wait_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 400uS

ave_cnt: the average count
**no_entry: Default = 1

ave_mode: the average mode
AVE : average
RMS : root mean square
type: int
**no_entry: Default = AVE

sampling_interval: the sampling interval between each measurement
**no_entry: Default = 10uS (minimum value)

meas_value: return the measured-values
type double or CURRENT array
e.g. CARRAY_2DIM<double> meas_value(site_no , pin_count);
**no_entry: means not return measured-values
// only return the active-sites measured-values

4.10.7.2 Example

```
CARRAY_DIM<int>   Lp(1000);
CARRAY_2DIM<double>   Imeas(4 , 32);   //suppose 4-sites & 32-pins

MEAS_I_PPMU( leak_pins,2mS,   ,16, AVE,50uS);
SOCKET_INC(FRZ_OFF)
{
```

```
        Lp= 0;    //init
        PIN_INC(leak_pins, FRZ_OFF)
        {
              Imeas[SOCKET][Lp] = RESULT_PIN(SOCKET, PIN_DEF);
              Lp++;
        }
}
```

```
CARRAY_2DIM<double>    Imeas(4 , 1);    //suppose 4-sites & 1-pins
      MEAS_I_PPMU( clk, 2mS, 16, AVE, 50uS);
      SOCKET_INC(FRZ_OFF)
      {
            Imeas[SOCKET][0] = RESULT_PIN(SOCKET, clk);
            JUDGE_VARIABLE (SOCKET,    Imeas[SOCKET][0], -1uA, 1uA, "clk
       current measurement ");
      }
```

```
CARRAY_2DIM<double>    Imeas(4 , 1); //suppose 4-sites & 1-pins

MEAS_I_PPMU( clk, 2mS, 16, AVE, 50uS, Imeas);
SOCKET_INC(FRZ_OFF)
{
      JUDGE_VARIABLE (SOCKET,    Imeas[SOCKET][0], -1uA, 1uA, "clk current
measurement ");
}
```

4.10.8 MEAS_V_PPMU

Perform a reference power supply Voltage measurement and save the measured value to system buffer. Read the result by the "RESULT_PIN" statement .

4.10.8.1 Format

Parameter List:

```
MEAS_V_PPMU(pin_name, wait_time[, ave_cnt, ave_mode, sampling_interval] [, meas_value]) ;
```

Parameter comment:

pin_name : pin-def or pin_group.
variables & constants allowed
type: interger
**no_entry: illegal

wait_time: specify the waiting time after conditions setup
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: Default = 400uS

ave_cnt: the average count
**no_entry: Default = 1

ave_mode: the average mode
AVE : average
RMS : root mean square
type: int
**no_entry: Default = AVE

sampling_interval: the sampling interval between each measurement
**no_entry: Default = 10uS (minimum value)

meas_value: return the measured-values
type double or CURRENT array
e.g. CARRAY_2DIM<double> meas_value(site_no , pin_count);
**no_entry: means not return measured-values
// only return the active-sites measured-values

4.10.8.2 Example

```
MEAS_V_PPMU(output, 1mS);
MEAS_V_PPMU(output, 1mS, 16, AVE, 50uS);
```

```
CARRAY_2DIM<double> Vmeas (4, 32); //suppose 4-site, 32pins
MEAS_V_PPMU(output, 1mS, 16, AVE, 50uS, Vmeas);
```

4.11 Input & Output Condition Setting

4.11.1 SET_LEVELN

Set the IO pin levels for the specified pins and level set number.

4.11.1.1 Format

Parameter List:

```
SET_LEVELN(lvl_no, pin_name, vil, vih, vol,voh, iol, ioh, vref [, cll, clh ] [, vppmu, ippmu,
i_range ]);
```

Parameter comment:

lvl_no: level set number from 1 to 249 (0, 250~255: system use)
Variables and constants are allowed.
Type integer
No entry: illegal

pin_name: Pin_def and pin_group which only suit for IO-Pins can be used.
No entry: illegal

vil: -2.0V to 5.9V
Variables and constants are allowed.
Type VOLTAGE or double
Unit V, pV, nV, uV, mV
No entry: keep current value

vih: -1.9V to 6.0V
Variables and constants are allowed.
Type VOLTAGE or double
Unit V, pV, nV, uV, mV
No entry: keep current value

vol: -2.0V to 6.0V
Variables and constants are allowed.
Type VOLTAGE or double
Unit V, pV, nV, uV, mV
No entry: keep current value

voh: -2.0V to 6.0V
Variables and constants are allowed.
Type VOLTAGE or double
Unit V, pV, nV, uV, mV
No entry: keep current value

iol: Define current value is from Tester to DUT. (Sink current)
cope of setting is from 0mA to +12mA.
Variables and constants are allowed.
Type CURRENT or double
Unit A, pA, nA, uA, mA

No entry: keep current value

ioh: Define current value from DUT to Tester. (Source current)
Scope of setting is from -12mA to 0mA.
Variables and constants are allowed.
Type CURRENT or double
Unit A, pA, nA, uA, mA
No entry: keep current value

vref: -1.75V to 5.75V
Variables and constants are allowed.
Type VOLTAGE or double
Unit V, pV, nV, uV, mV
No entry: keep current value

cll: -2.0V to 5V
Variables and constants are allowed.
Type VOLTAGE or double
NO entry:-2.0V

clh: -2.0V to 6V
For practical reasons, as well as hardware constraints, clh > cll +300mV.
(2 pins share the same value)
Variables and constants are allowed.
Type VOLTAGE or double
Unit V, pV, nV, uV, mV
NO entry :6.0V

4.11.1.2 Explanation

The instruction sets up the high and low voltage values of the driver output, the voltage value used by the comparator to compare the high and low levels of the DUT output waveform, the current value for the device loading, the reference voltage, and the terminal voltage.

Notice: If device keeps CONT-mode (continue mode), that user can't change IO status while in CONT-mode. It means that high-speed running, user can't change the timing/level/format/pattern.

4.11.1.3 Example

```
func_1MHZ_test()
{
   FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;

   FORCE_V_PREF(   Vdd, 5V, @6V, @100mA, 100mA, NORM, ON, 1mS);

   DATALOG_MSG="Func 1MHZ test";

   SET_LEVELN(func_lvl, IO_ALLPINS,vil,vilh,vol,voh,0uA,0uA,0V);
   LEVELS(func_lvl, 3mS);

   PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);

   PIN_MODE(          CLK,    RZ, EDGE,     MASK, IO_NRZ, OFF);

   RELAY_ON( IO_ALLPINS, PDCL, 3mS);

   //Judge pattern programming example
   TEST_NO(20);

   JUDGE_PAT(func_pat, __func_pat);//Judge Pattern Module
}
```

Set IO pins level value into system register.

4.11.1.4 Relative

LEVELS

4.11.2 LEVELS

This statement selects the pre-loaded level set.

4.11.2.1 Format

Parameter list:

LEVELS (level_set_no, wait_time) ;

Parameter comment:

level_set_no: level set number from 1 to 240
Variables and constants are allowed.
Type integer
No entry: illegal

wait_time: Specify the wait time after conditions setup.
Variables and constants are allowed.
Type TIME or double
Unit S, pS, nS, uS, mS
No entry: Default = 3mS

4.11.2.2 Explanation

This statement is simply a compound statement which brackets a list of SET_LEVELN statement, which actually load the hardware. Select the pre-loaded level set.

Notice: If device keeps CONT-mode (continue mode), that user can't change IO status while in CONT-mode. It means that high-speed running; user can't change the timing/level/format/pattern.

4.11.2.3 Example

```
func_1MHZ_test()
{
   FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;

   FORCE_V_PREF(   Vdd, 5V, @6V, @100mA, 100mA, NORM, ON, 1mS);

   DATALOG_MSG="Func 1MHZ test";

   SET_LEVELN(func_lvl, IO_ALLPINS,vil,vilh,vol,voh,0uA,0uA,0V);
   LEVELS(func_lvl, 3mS);

   PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);

   PIN_MODE(          CLK,    RZ, EDGE,     MASK, IO_NRZ, OFF);

   RELAY_ON( IO_ALLPINS, PDCL, 3mS);

   //Judge pattern programming example
   TEST_NO(20);

   JUDGE_PAT(func_pat, __func_pat);//Judge Pattern Module
}
```

Execute “ func_lvl ”level set

4.11.2.4 Relative

SET_LEVELN

4.11.3 INIT_DRIVE_STATE

This statement specifies the IO pin initial status of the driver.

4.11.3.1 Format

Parameter list:

```
INIT_DRIVE_STATE (pin_name, drv_init, io_init) ;
```

Parameter comment:

pin_name: Pin_def and pin_group which only suit for IO-Pins can be used.
No entry: illegal

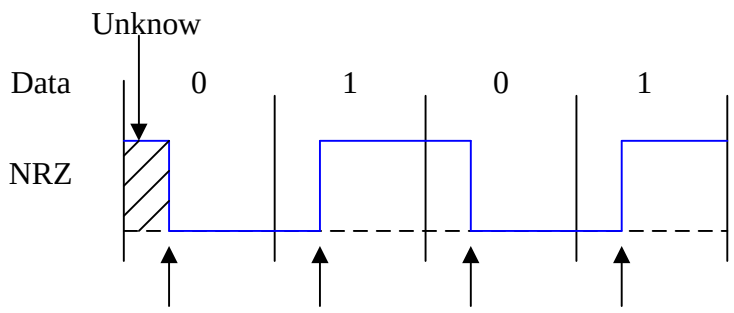
drv_init: There are three modes could be selected.
LOW: Sets the initial status of the driver output at LOW.
HIGH: Sets the initial status of the driver output at HIGH.
NORM: Follow the previous vector state.
No entry: Default = NORM.

io_init: There are three models could be selected.
ON: Sets the initial status of the driver at ON.
OFF: Sets the initial status of the driver at OFF.
NORM: Folow the previous vector state.

No entry: Default = NORM

4.11.3.2 Explanation

This statement specifies the IO pin initial status of the driver for unknown area.
(Especially only for input pin driver NRZ, DNRZ format.)



4.11.3.3 Example

```
func_1MHZ_test()
{
   FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;

   FORCE_V_PREF(   Vdd, 5V, @6V, @100mA, 100mA, NORM, ON, 1mS);

   DATALOG_MSG="Func 1MHZ test";

   SET_LEVELN(func_lvl, IO_ALLPINS,vil,vilh,vol,voh,0uA,0uA,0V);
   LEVELS(func_lvl, 3mS);

   PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);

   PIN_MODE(          CLK,    RZ, EDGE,     MASK, IO_NRZ, OFF);

   INIT_DRIVE_STATE(IO_ALLPINS,LOW,ON); // initial io all pin to low

   RELAY_ON( IO_ALLPINS, PDCL, 3mS);

   //Judge pattern programming example
   TEST_NO(20);

   JUDGE_PAT(func_pat, __func_pat);//Judge Pattern Module
}
```

4.11.4 JUDGE_STATE

Test the low speed static conditions of the pin cards.

4.11.4.1 Format

Parameter list:

```
JUDGE_STATE (pin_name, state);
```

Parameter comment:

pin_name: Pin_def and pin_group which only suit for IO-Pins can be used.
No entry: illegal

state: HIGH – test for high (> Voh)
LOW – test for low (< Vol)
FLOAT – test for float (< Voh and > Vol)
VALID – test for valid level (> Voh or < Vol)
No entry: illegal

4.11.4.2 Explanation

Judge PASS or FAIL by comparing the data with Voh and Vol.
Test the low speed static conditions of the pin cards.

4.11.4.3 Example

```
JUDGE_STATE (rst_pin, HIGH);
```

4.11.5 RATEN

This statement load rate value to memory for the specified time set number.

4.11.5.1 Format

Parameter list:

RATEN (time_set_no, period_value) ;

Parameter comment:

ts_no: Select time set number is from 1 to 15,
or name as defined in TIME_NAME_DEF.
Variables and constants are allowed.
Type integer

No entry: illegal

period_value: Select period value is from 10ns to 5.0mS. The period value less than 40nS should be select DBL_2X or DBL mode in SET_JUDGE_MODE(). And the value less than 20nS should be select DBL_2X mode in SET_JUDGE_MODE(). Or runtime error will occur.

Variables and constants are allowed.
Type TIME or double
Unit S, pS, nS, uS, mS
No entry: illegal
Setting resolution: 0.625nS

4.11.5.2 Explanation

3380 has 15 sets of ts_no that named as TS1~TS15. Load rate value to memory for the specified time set number.

Notice: If device keeps CONT-mode (continue mode), that user can't change IO status while in CONT-mode. It means that high-speed running; user can't change the timing/level/format/pattern.

4.11.5.3 Example

```
func_1MHZ_test()
{
   FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;

   FORCE_V_PREF(   Vdd, 5V, @6V, @100mA, 100mA, NORM, ON, 1mS);

   DATALOG_MSG="Func 1MHZ test";

   RATEN(TM1, 1uS);
   MARKERN(TM1, CLK ,250nS,750nS, ,0nS, ,850nS, );
   MARKERN(TM1, CTRL,0nS, , ,0nS, ,850nS, );
   MARKERN(TM1, OUTPUTS ,0nS, , ,0nS, ,950nS, );

   SET_LEVELN(func_lvl, IO_ALLPINS,vil,vilh,vol,voh,0uA,0uA,0V);
   LEVELS(func_lvl, 3mS);

   PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);

   PIN_MODE(          CLK,   RZ, EDGE,     MASK, IO_NRZ, OFF);

   RELAY_ON( IO_ALLPINS, PDCL, 3mS);

   //Judge pattern programming example
   TEST_NO(20);

   JUDGE_PAT(func_pat, __func_pat);//Judge Pattern Module
}
```

4.11.5.4 Relative

MARKERN , PIN_MODE

4.11.6 MARKERN

Load marker value to memory for the specified pins and time set number.

4.11.6.1 Format

Parameter list:

MARKERN (ts_no, pin_name, DT0, DT1, DT2, DT3, DT4, CT0, CT1) ;

Parameter comment:

ts_no: Select number from 1 to 15, or name as defined in TIME_NAME_DEF.
Variables and constants are allowed.
Type integer
No entry: illegal
pin_name: Pin_def and pin_group which only suit for IO-Pins can be used.
No entry: illegal
DT0/DT1: Drive marker
DT2: SBC marker
DT3/DT4:IO marker
CT0/CT1: Strobe marker;
Setting range: 0nS ~ (2cycle - 10nS);
Setting resolution: 0.625nS

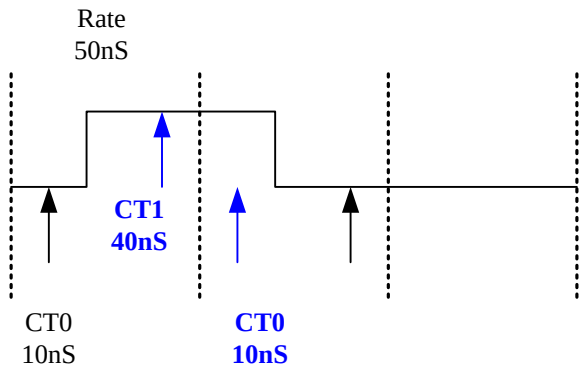
4.11.6.2 Explanation

Load marker value to memory for the specified pins and time set number.

[Notice]: If device keeps CONT-mode (continue mode), that user can't change IO status while in CONT-mode. It means that high-speed running; user can't change the timing/level/format/pattern.

- Window strobe timing limitation: CT0 marker has to >= 30nS from CT1 marker, otherwise pseudo pass.

The example of the window setting may lead pseudo pass



```
Item_A()
      {
        PIN_MODE(Outputs, NRZ, WINDOW, ENABLE, IO_NRZ, OFF);
        RATEN( TS1, 50nS);
        MARKERN(TS1, Outputs, 20nS, 30nS, , 0nS, , 10nS, 40nS);    //this setting may
      lead    pseudo pass

        JUDGE_PAT(func1_st, func1_sp);
      }
```

4.11.6.3 Example

```
func_1MHZ_test()
{
   FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;

   FORCE_V_PREF(    Vdd, 5V, @6V, @100mA, 100mA, NORM, ON, 1mS);

   DATALOG_MSG="Func 1MHZ test";

   RATEN(TM1, 1uS);
   MARKERN(TM1, CLK ,250nS,750nS, ,0nS, ,850nS, );
   MARKERN(TM1, CTRL,0nS, , ,0nS, ,850nS, );
   MARKERN(TM1, OUTPUTS ,0nS, , ,0nS, ,950nS, );

   SET_LEVELN(func_lvl, IO_ALLPINS,vil,vilh,vol,voh,0uA,0uA,0V);
   LEVELS(func_lvl, 3mS);

   PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);

   PIN_MODE(          CLK,    RZ, EDGE,     MASK, IO_NRZ, OFF);

   RELAY_ON( IO_ALLPINS, PDCL, 3mS);

   //Judge pattern programming example
   TEST_NO(20);

   JUDGE_PAT(func_pat, __func_pat);//Judge Pattern Module
}
```

4.11.6.4 Relative

RATEN, PIN_MODE

4.11.7 FRUN_CLK

Load free-run clock marker value to memory for the specified IO pins.

4.11.7.1 Format

Parameter list:

```
FRUN_CLK( pin_name, period_value, rising_edge, falling_edge);
```

Parameter comment:

pin_name: pin_def (each pin can use as a free-run-clock,
per 8 pin shares one free-run-clock resource)
**no_entry: illegal

period_value: period value 5ns to 10mS
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: illegal

rising_edge: Setting range: 0nS~(1cycle –1.25nS);
**no_entry: illegal

falling_edge: Setting range: 0nS~(1cycle –1.25nS);
**no_entry: illegal

4.11.7.2 Explanation

Load free-run clock marker value to memory for the specified IO pins ;
in the same time, we should use "FCLK" format in the PIN_MODE statement.

[Notice]: If device keeps CONT-mode (continue mode), that user can't change IO status while in CONT-mode. It means that high-speed running; user can't change the timing/level/format/pattern.

4.11.7.3 Example

```
1. Mode 0
//==================================//
PIN_MODE(CLK, FCLK, EDGE, MASK, IO_ON, OFF); // CLK:pin name
FRUN_CLK(CLK, 5uS, 0S, 2.5uS);
DATALOG_MSG="FREE_RUN_CLK_CHECK_MODE_0";
FRUN_CLK_MODE (CLK_PIN , 0 , 0 , ON );//set mode 0 and turn on
DEBUG_BREAK();
JUDGE_PAT(func_test,__ func_test);//free run clk output control by pattern(start and stop)
FRUN_CLK_MODE (CLK_PIN , 0 , 0 , OFF );// turn OFF
```

4.11.7.4 Relative

PIN_MODE , FRUN_CLK_MODE

4.11.8 FRUN_CLK_MODE

User can use the statement to control free run clock mode.

4.11.8.1 Format

Parameter list:

```
FRUN_CLK_MODE( pin_name, turn_on_mode, frz_mode, ON/OFF);
```

Parameter comment:

pin_name: pin_def (each pin can use as a free-run-clock
per 8 pin shares one free-run-clock resource)
**no_entry: illegal

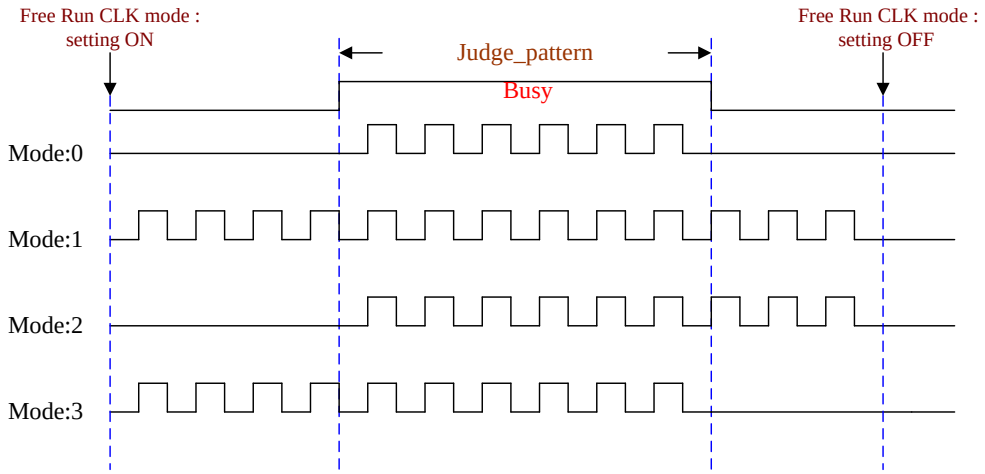
turn_on_mode:

0: free run clk output controlled by pattern burst (BUSY) (system default)

1: free run clk output controlled by statement setting and always running.
It will be stopped by statement setting:
FRUN_CLK_MODE set "OFF"

2: free run clk output controlled by pattern burst (BUSY) and continue run.
It will be stopped by statement setting:
FRUN_CLK_MODE set "OFF"

3: free run clk output controlled by statement setting and turn off while busy done
type: int
**no_entry: illegal



frz_mode:
0: fclk still running while freeze command find in cycle

1: fclk will be hold if freeze command find in cycle (system default)
type: int
**no_entry: illegal

ON/OFF:

OFF: turn off ----> restore the system default value
ON : trun-on "turn_on_mode" & "frz_mode" setting
**no_entry: illegal

4.11.8.2 Example

```
//1. Mode 0
//=================================//
PIN_MODE(CLK, FCLK, EDGE, MASK, IO_ON, OFF); // CLK:pin name
FRUN_CLK(CLK, 5uS, 0S, 2.5uS);
DATALOG_MSG="FREE_RUN_CLK_CHECK_MODE_0";
FRUN_CLK_MODE (CLK_PIN , 0 , 0 , ON );//set mode 0 and turn on
DEBUG_BREAK();
JUDGE_PAT(func_test,__ func_test);//free run clk output control by pattern(start and
stop)
FRUN_CLK_MODE (CLK_PIN , 0 , 0 , OFF );// turn OFF
```

```
//2. Mode 1
//=================================//
PIN_MODE(CLK, FCLK, EDGE, MASK, IO_ON, OFF); // CLK:pin name
FRUN_CLK(CLK, 5uS, 0S, 2.5uS);
DATALOG_MSG="FREE_RUN_CLK_CHECK_MODE_1";
FRUN_CLK_MODE (CLK_PIN , 1 , 0 , ON );//set mode 1 and turn on
//"CLK output start"
DEBUG_BREAK();
JUDGE_PAT(func_test,__ func_test);
FRUN_CLK_MODE (CLK_PIN , 1 , 0 , OFF );// turn OFF and "CLK stop"
```

```
//3. Mode 2
//=================================//
PIN_MODE(CLK, FCLK, EDGE, MASK, IO_ON, OFF); // CLK:pin name
FRUN_CLK(CLK, 5uS, 0S, 2.5uS);
DATALOG_MSG="FREE_RUN_CLK_CHECK_MODE_2";
FRUN_CLK_MODE (CLK_PIN , 2 , 0 , ON );//set mode 2 and turn on
DEBUG_BREAK();
JUDGE_PAT(func_test,__ func_test);//free run CLK output start when pattern start
FRUN_CLK_MODE (CLK_PIN , 2 , 0 , OFF );// turn OFF and "CLK stop"
```

```
//4. Mode 3
//=================================//
PIN_MODE(CLK, FCLK, EDGE, MASK, IO_ON, OFF); // CLK:pin name
FRUN_CLK(CLK, 5uS, 0S, 2.5uS);
DATALOG_MSG="FREE_RUN_CLK_CHECK_MODE_3";
FRUN_CLK_MODE (CLK_PIN , 3 , 0 , ON );//set mode 3 and turn on
//"CLK output start"
DEBUG_BREAK();
JUDGE_PAT(func_test,__ func_test);//free run CLK output STOP when pattern STOP
FRUN_CLK_MODE (CLK_PIN , 2 , 0 , OFF );// turn OFF
```

4.11.9 PIN_MODE

The statement setup the IO pins driver format, IO format, compare mode, and load mode.

4.11.9.1 Format

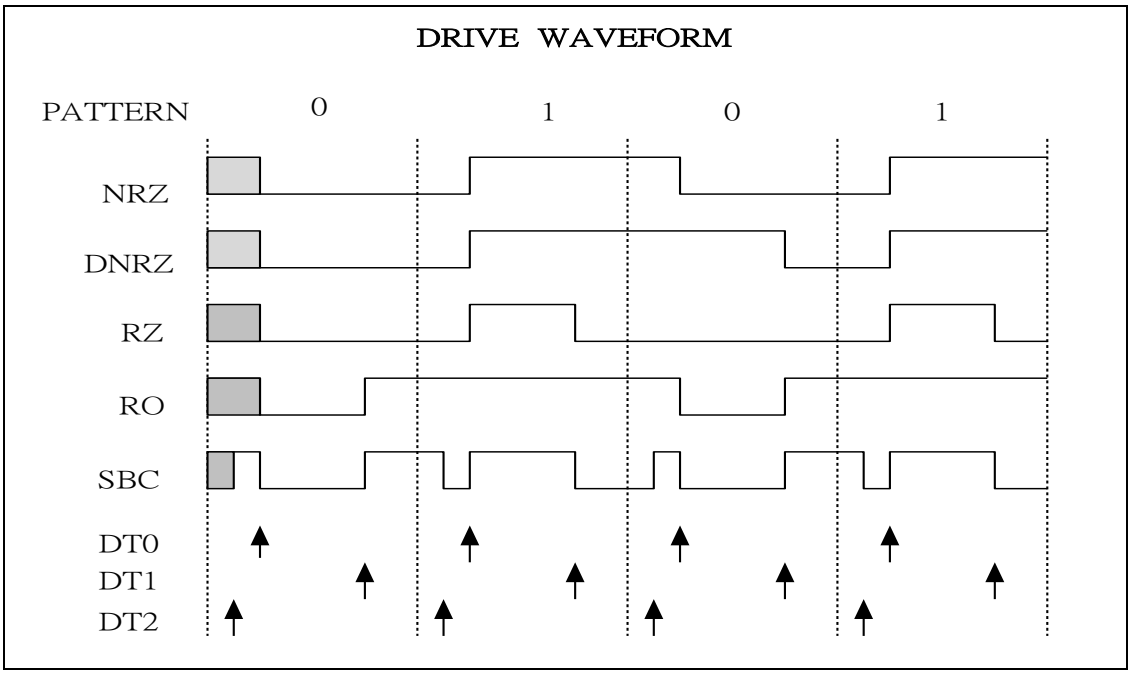
Parameter list:

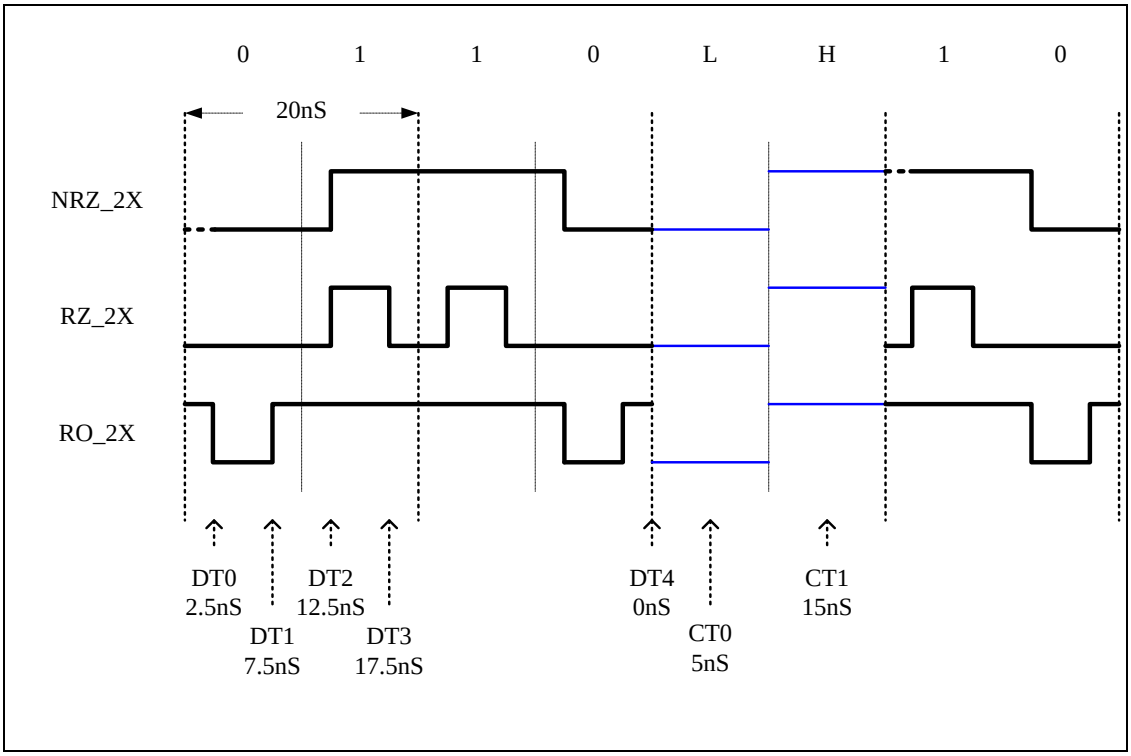
PIN_MODE (pin_name, d_format, c_mode, c_mask, io_format, load_mode);

Parameter comment:

pin_name: pin_def, pin_group (only suit for IO-Pins)
no_entry: illegal

d_format: NRZ, DNRZ, RZ, RO, SBC, FCLK, F0 and F1
NRZB, DNRZB, RZB, ROB, SBCB, FCLKB;
OSC,OSCB : oscillator clock output
(F0: Force Low, F1: Force High)
FCLK : free-run-clock format
type: integer
no_entry: keep current format

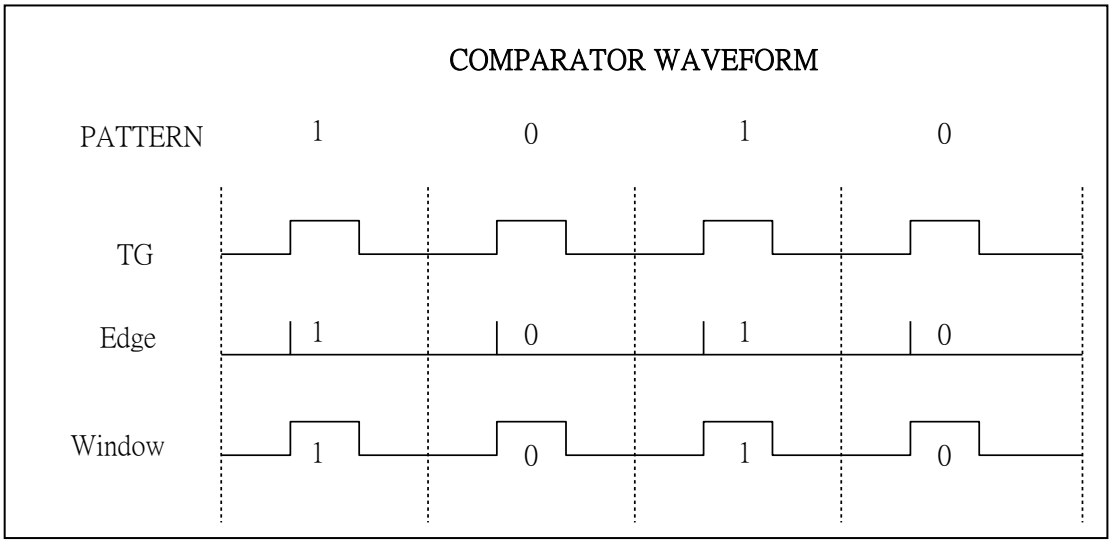




c_mode: One of two possible comparator strobe modes. These modes are set prior to the pattern execution starts, and it will not change during the pattern execution.

EDGE: Strobe occurs only at the first marker and are edge triggered. Only for NORM/DBL use.

WINDOW: Strobe occurs for the duration of an internal pulse (a window),generated between T1 and T2 strobe markers.
Type integer
No entry: keep current mode



c_mask: MASK: master mask on
ENABLE: master mask off
Type integer
No entry: keep current state

io_format: IO_NRZ, IO_RZ, IO_RO, IO_ON, IO_OFF.
Type integer
No entry: keep current format

load_mode: NORM, ON, OFF ;
NORM: load is on when driver is off, and off when driver is on.
ON: load is always on whether driver is on or off.
OFF: load is always off whether driver is on or off.
Type integer
No entry: keep current mode

4.11.9.2 Explanation

The statement setup the IO pins driver format, IO format, compare mode, and load mode.

[Notice]: If device keeps CONT-mode (continue mode), that user can't change IO status while in CONT-mode. It means that high-speed running; user can't change the timing/level/format/pattern.

4.11.9.3 Example

```
func_1MHZ_test()
{
   FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;

   FORCE_V_PREF(   Vdd, 5V, @6V, @100mA, 100mA, NORM, ON, 1mS);

   DATALOG_MSG="Func 1MHZ test";

   RATEN(TM1, 1uS);
   MARKERN(TM1, CLK ,250nS,750nS, ,0nS, ,850nS, );
   MARKERN(TM1, CTRL,0nS, , ,0nS, ,850nS, );
   MARKERN(TM1, OUTPUTS ,0nS, , ,0nS, ,950nS, );

   SET_LEVELN(func_lvl, IO_ALLPINS,vil,vilh,vol,voh,0uA,0uA,0V);
   LEVELS(func_lvl, 3mS);

   PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF)

   PIN_MODE(          CLK,   RZ, EDGE,     MASK, IO_NRZ, OFF);

   RELAY_ON( IO_ALLPINS, PDCL, 3mS);

   //Judge pattern programming example
   TEST_NO(20);

   JUDGE_PAT(func_pat, __func_pat);//Judge Pattern Module
}
```

4.11.9.4 Relative

MARKERN , RATEN

4.12 Relay Setting

4.12.1 SET_CBIT

This section explains how to set the value of the user relay register.

4.12.1.1 Format

Parameter list:

```
SET_CBIT(pin_name , ON/OFF, wait_time) ;
```

Parameter comment:

Pin_name: pin_def, pin_group(pin type = UR)
**no_entry: illegal

ON/OFF: ON (relay turn on)
OFF (relay turn off)
No entry: illegal

wait_time: Specify the waiting time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 500uS

<Declaration files setting>

1. Type:UR
2. Channel assign:
   ➔Normal: 0-63

   ➔ If using SPBUS (only for 3380 & 3380P) serial control function and MAX 4820 device User can use another relay control signal limit to 128 channels
   Channel assign: 2000~~3024(1024 Serial URC). (Appendix: How to use 4820)

4.12.1.2 Explanation

The SET_CBIT statement writes the value of the 64 bits of the User Relay Control Pins.

The statement transfers the values from program variables into hardware register basically.

System provide Relay_5V maximum current "4A" for relay used

If user use MXBUS serial URC to control maxim-4820 IC, then :
The URC channel of 1st max4820 : 2000~2007
The URC channel of 2nd max4820 : 2008~2015

Reference data : APN_B0035

4.12.1.3 Example

<Example1> Pin_def and pin_group can be used
<Declaration file>

```
PIN_LIST    ( LS299_1_Site_DUT_BOARD )//Pin list name max.=32 characters.
{
//PIN NAME max.=64 characters.
//pin name =                    ATE channel                    = dut    = type ;//
//          -->   S0
   SEL0       = 150   =   1    = IN   ;//IN
   OE1        = 148   =   2    = IN   ;//IN
   OE2        = 146   =   3    = IN   ;//IN
   QG         = 144   =   4    = IO   ;//IO
   QE         = 142   =   5    = IO   ;//IO
   QC         = 140   =   6    = IO   ;//IO
   QA         = 138   =   7    = IO   ;//IO
   OA         = 136   =   8    = OUT ;//OUT
   CLR        = 134   =   9    = IN   ;//IN

   Gnd        =       =   10   = GND ;//GND pin didn't any ate channel

   SR         = 133   =   11   = IN   ;//IN
   CLK        = 135   =   12   = IN   ;//IN
   QB         = 137   =   13   = IO   ;//IO
   QD         = 139   =   14   = IO   ;//IO
   QF         = 141   =   15   = IO   ;//IO
   QH         = 143   =   16   = IO   ;//IO
   OH         = 145   =   17   = OUT ;//OUT
   SL         = 147   =   18   = IN   ;//IN
   SEL1       = 149   =   19   = IN   ;//IN
   RL1        =   0   =        = UR;
   RL2        =   1   =        = UR;

   Vdd        =   9   =   20   = PREF;
   Vddd       =       =   20   = DPS ;

   TT         = 132   =        = TRG ;
}
```

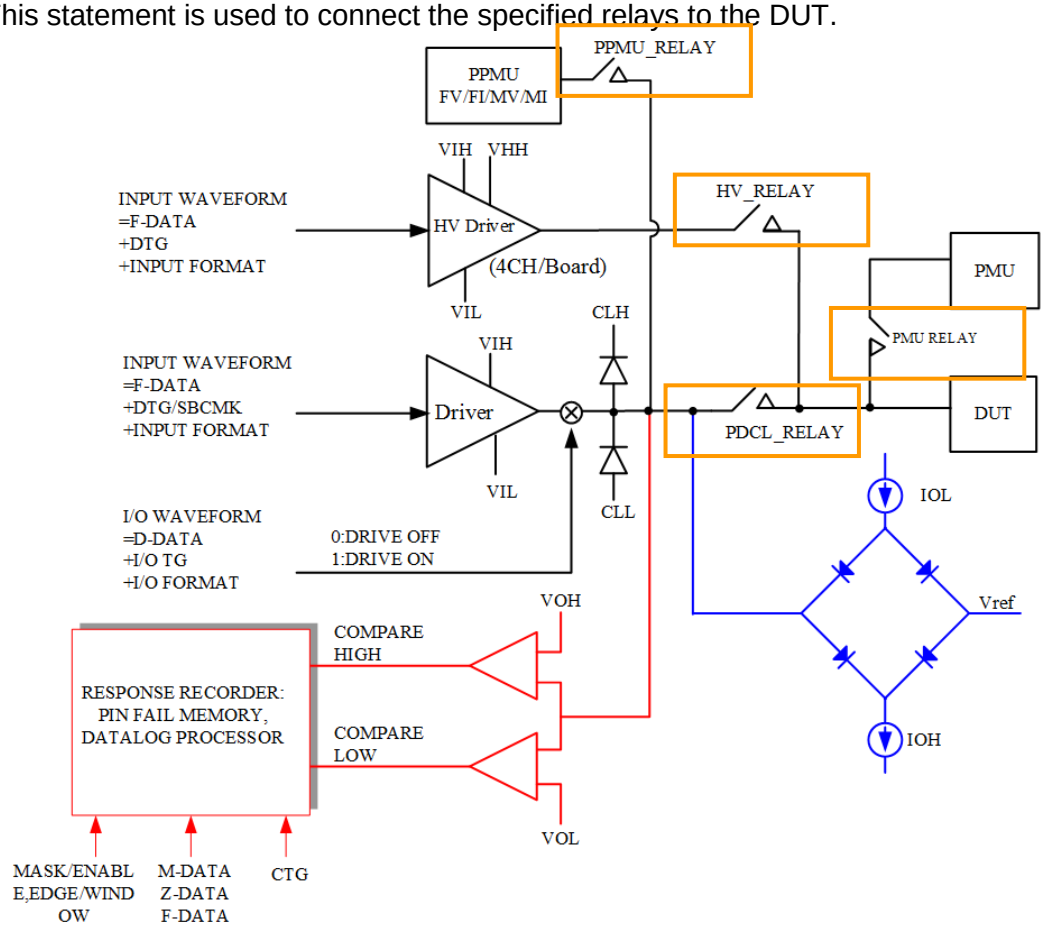
Set Relay channel and Type

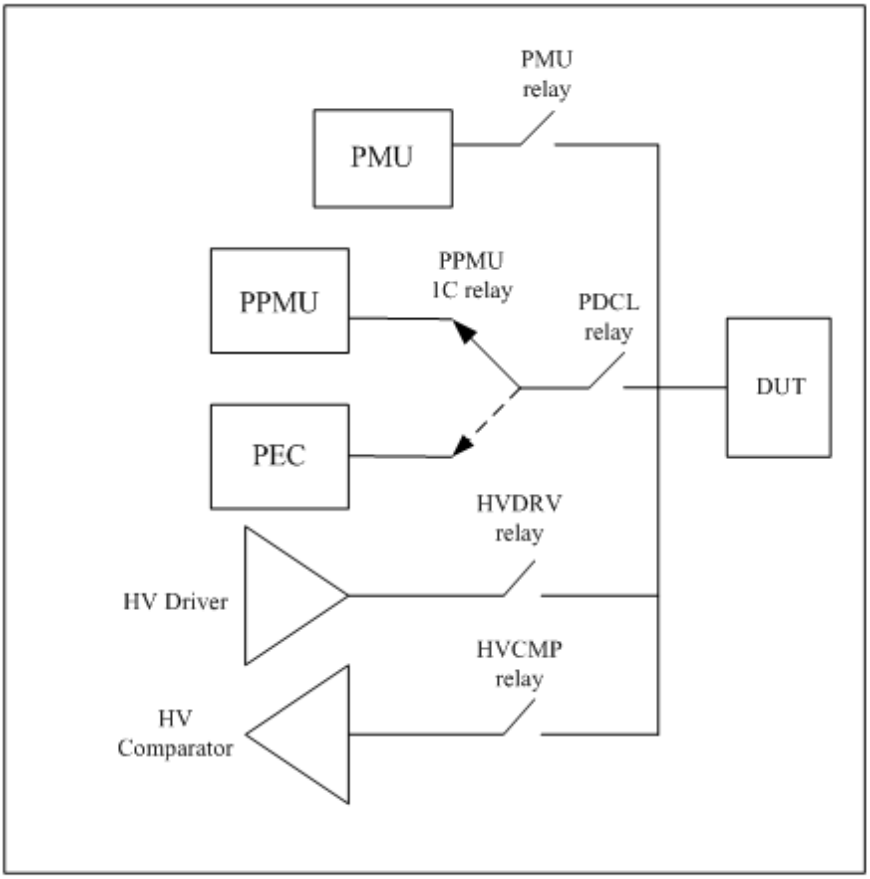
<pln file>

```
func_1MHZ_test()
{
   FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;

   FORCE_V_PREF(   Vdd, 5V, @6V, @100mA, 100mA, NORM, ON, 1mS);
   SET_CBIT(RL1+RL2, ON ,2mS);
   DATALOG_MSG="Func 1MHZ test";

   RATEN(TM1, 1uS);
   MARKERN(TM1, CLK ,250nS,750nS, ,0nS, ,850nS, );
   MARKERN(TM1, CTRL,0nS, , ,0nS, ,850nS, );
   MARKERN(TM1, OUTPUTS ,0nS, , ,0nS, ,950nS, );

   SET_LEVELN(func_lvl, IO_ALLPINS,vil,vilh,vol,voh,0uA,0uA,0V);
   LEVELS(func_lvl, 3mS);

   PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);

   PIN_MODE(          CLK,    RZ, EDGE,     MASK, IO_NRZ, OFF);

   RELAY_ON( IO_ALLPINS, PDCL, 3mS);

   //Judge pattern programming example
   TEST_NO(20);

   JUDGE_PAT(func_pat, __func_pat);//Judge Pattern Module
}
```

4.12.2 RELAY_ON

This statement is used to connect the specified relays to the DUT.





MXLPC PEC

4.12.2.1 Format

Parameter list:

RELAY_ON (pin_name, relay_resource, wait_time) ;

Parameter comment:

pin_name: The pin being connected to the system resources.
Pin_def and pin_group which only suitable for IO-Pins can be used.
No entry: illegal

relay_resource:
PMU (select the pin PMU relays of the specified pins in pin_name.)
PDCL (the relay of IO Pin Driver, Comparator and active Load)
PPMU_PDCL (the PPMU mode-bit & PDCL relay of IO pin)// only for MXLPC
HVDRV_CMP (select the pin HV relays of the specified pins in pin_name.) // only for MXLPC
HVCMP(select the pin HC relays of the specified pins in pin_name)// only for MXLPC
type: int
No entry: illegal

wait_time: Specify the wait time after conditions setup.

Variables and constants are allowed.
Type: TIME or double
Unit:: S, pS, nS, uS, mS
No entry: Default = 500uS

4.12.2.2 Explanation

The statement establishes the necessary connections between the pin under test and the selected tester resources. These connections are performed via relays and / or switches.

high-voltage pins : 4 channel/broad (ATE channel could use 0,16,32,48)

PDCL ON	→ PDCL relay ON & PPMU mode-bit OFF
PPMU_PDCL ON	→ PDCL relay ON & PPMU mode-bit ON
PDCL OFF	→ PDCL relay OFF & PPMU mode-bit OFF
PPMU_PDCL OFF	→ PDCL relay OFF & PPMU mode-bit OFF
HVDRV_CMP ON	→ HVDRV relay ON & HVCMP relay ON
HVCMP ON	→ HVCMP relay ON
HVDRV_CMP OFF	→ HVDRV relay OFF & HVCMP relay OFF
HVCMP OFF	→ HVCMP relay OFF

4.12.2.3 Example

<Example1>

```
func_1MHZ_test()
{
   FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;

   FORCE_V_PREF(    Vdd, 5V, @6V, @100mA, 100mA, NORM, ON, 1mS);
   SET_CBIT(RL1+RL2, ON ,2mS);
   DATALOG_MSG="Func 1MHZ test";

   RATEN(TM1, 1uS);
   MARKERN(TM1, CLK ,250nS,750nS, ,0nS, ,850nS, );
   MARKERN(TM1, CTRL,0nS, , ,0nS, ,850nS, );
   MARKERN(TM1, OUTPUTS ,0nS, , ,0nS, ,950nS, );

   SET_LEVELN(func_lvl, IO_ALLPINS,vil,vilh,vol,voh,0uA,0uA,0V);
   LEVELS(func_lvl, 3mS);

   PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);

   PIN_MODE(          CLK,    RZ, EDGE,     MASK, IO_NRZ, OFF);

   RELAY_ON( IO_ALLPINS, PDCL, 3mS);

   //Judge pattern programming example
   TEST_NO(20);

   JUDGE_PAT(func_pat, __func_pat);//Judge Pattern Module
```

```
}
```

<Example2>

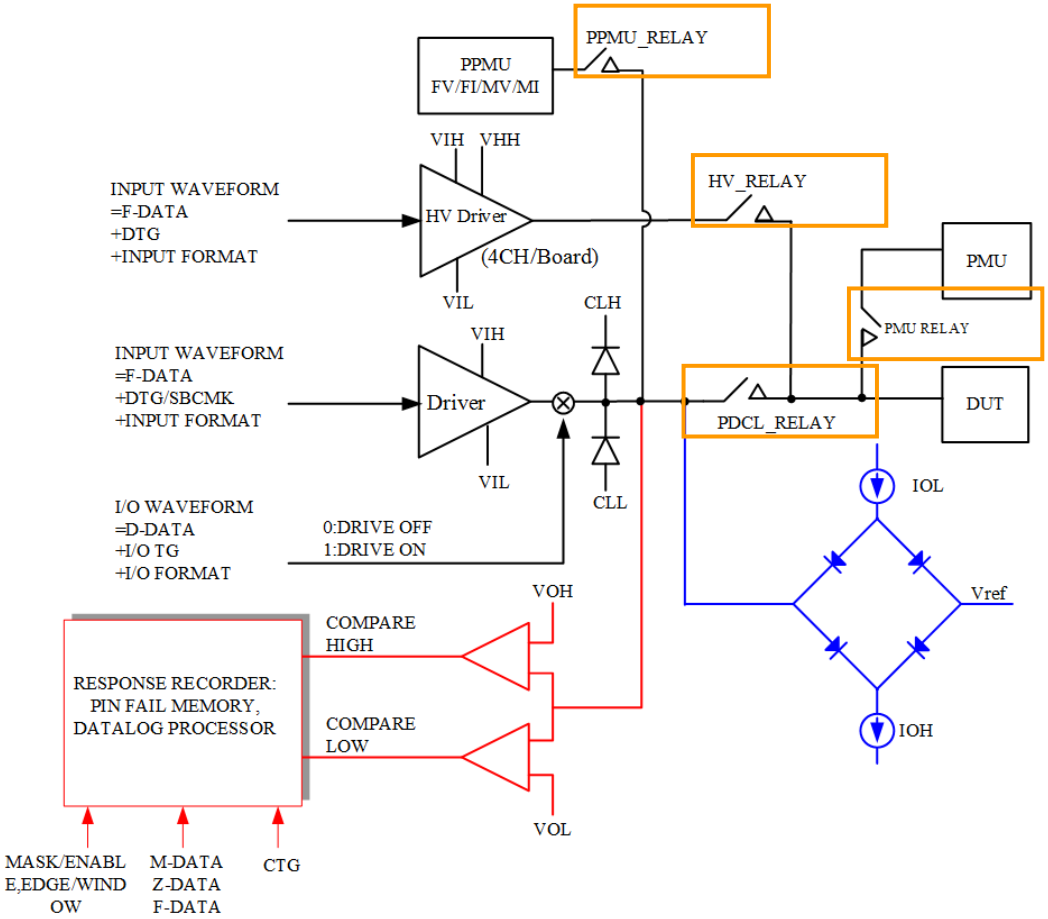
```
        FORCE_V_DPS ( Vddd, 0V, @8V, @100mA, 100mA, NORM, ON, 1mS);
        FORCE_V_PREF(   Vdd, 0V, @6V, @100mA, 100mA, NORM, ON, 1mS);

        LEVELS(os_lvl,5mS);

        PIN_MODE(ALLPINS,F0,EDGE,MASK,IO_ON,OFF);

        RELAY_ON(ALLPINS,PDCL,1mS);

        TEST_NO(10);

        PIN_INC(ALLPINS)
          {
           RELAY_OFF(PIN_DEF,PDCL,1mS);
           RELAY_ON(PIN_DEF,PMU,1mS);
           FORCE_I_PMU(PMU,-100uA,@100uA,@6V,2V,ON,1mS);
           JUDGE_V_PMU(PMU,IFVM,-1.2V,-0.2V,2mS);
           FORCE_I_PMU(PMU,0uA,@100uA,@6V,2V,OFF,1mS);
           RELAY_OFF(PIN_DEF,PMU,1mS);
           RELAY_ON(PIN_DEF,PDCL,1mS);
          }
     power_down();
```

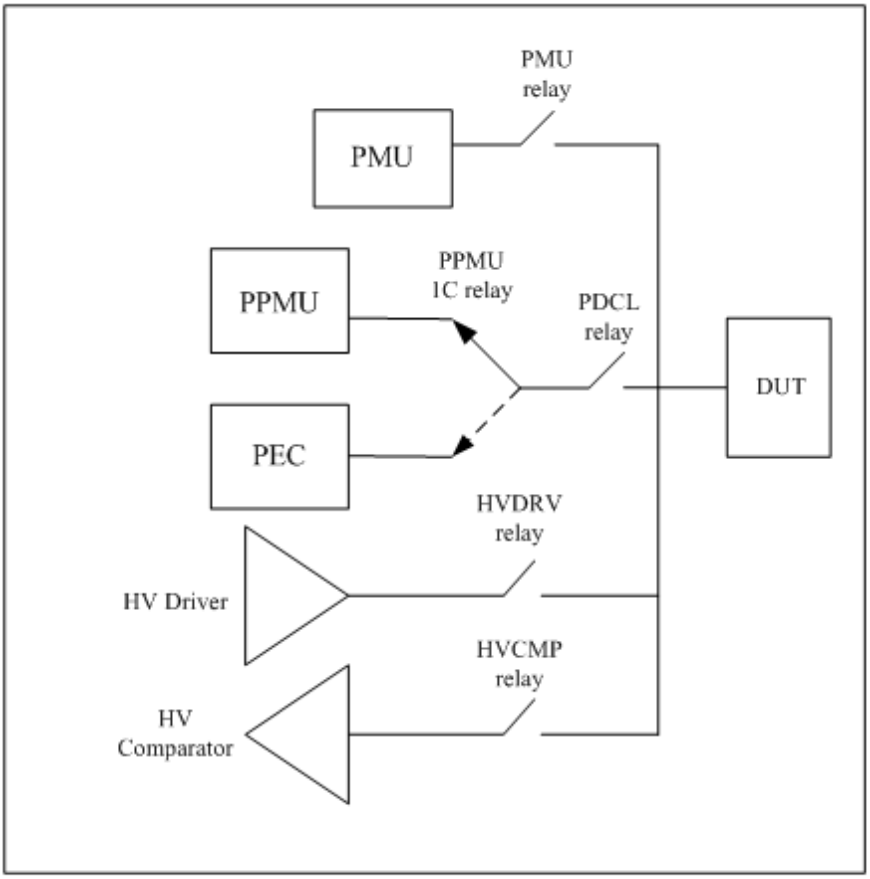
<Example3>

```
          FORCE_V_DPS ( Vddd,0V, @8V, @100mA, 100mA, NORM, ON, 1mS);
          FORCE_V_PREF(   Vdd,0V, @6V, @100mA, 100mA, NORM, ON, 1mS);

          LEVELS(os_lvl,5mS);

          PIN_MODE(IO_OS_PINS,F0,EDGE,MASK,IO_ON,OFF);

          RELAY_ON(IO_OS_PINS,PDCL,1mS);

          TEST_NO(10);
          RELAY_OFF(ODD_PINS,PDCL,1mS);
          RELAY_ON(ODD_PINS,OST,1mS);
          JUDGE_OS(ODD_PINS,-200uA,-1.2V,-0.2V,2mS); //JUDGE_OS current
setting from 100uA~200uA

          RELAY_OFF(ODD_PINS,OST,1mS);
          RELAY_ON(ODD_PINS,PDCL,1mS);
```

4.12.3 RELAY_OFF

This statement is used to switch-off the specified relays from the DUT.





MXLPC PEC

4.12.3.1 Format

Parameter list:

RELAY_OFF (pin_name, relay_resource, wait_time) ;

Parameter comment:
pin_name: The pin being connected to the system resources.
Pin_def and pin_group which only suitable for IO-Pins can be used.
Type: PINS_TYPE
No entry: illegal

relay_resource: The instrument to be connected to the pin under test. Valid entries are:
PMU (select the pin PMU relays of the specified pins in pin_name.)
PDCL (the relay of IO Pin Driver, Comparator and active Load)
PPMU_PDCL(the PPMU mode-bit & PDCL relay of IO pin)// only for MXLPC
HVDRV_CMP(select the pin HV relays of the specified pins in pin_name.) // only for MXLPC
HVCMP(select the pin HC relays of the specified pins in pin_name)// only for MXLPC
ALL (Disconnect all PDCL & PMU & PPMU relays in the specified pin_name.)

Type: int
No entry: illegal

wait_time: Specify the wait time after conditions setup.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Default = 500uS

4.12.3.2 Explanation

This statement removes the connections between the pin under test and the selected tester resources. These connections being removed were performed via relay and / or switches.

high-voltage pins : 4 channel/broad (ATE channel could use 0,16,32,48)

PDCL ON	→ PDCL relay ON & PPMU mode-bit OFF
PPMU_PDCL ON	→ PDCL relay ON & PPMU mode-bit ON
PDCL OFF	→ PDCL relay OFF & PPMU mode-bit OFF
PPMU_PDCL OFF	→ PDCL relay OFF & PPMU mode-bit OFF
HVDRV_CMP ON	→ HVDRV relay ON & HVCMP relay ON
HVCMP ON	→ HVCMP relay ON
HVDRV_CMP OFF	→ HVDRV relay OFF & HVCMP relay OFF
HVCMP OFF	→ HVCMP relay OFF

4.12.3.3 Example

```
        FORCE_V_DPS ( Vddd, 0V, @8V, @100mA, 100mA, NORM, ON, 1mS);
        FORCE_V_PREF(   Vdd, 0V, @6V, @100mA, 100mA, NORM, ON, 1mS);

        LEVELS(os_lvl,5mS);

        PIN_MODE(ALLPINS,F0,EDGE,MASK,IO_ON,OFF);

        RELAY_ON(ALLPINS,PDCL,1mS);

        TEST_NO(10);

        PIN_INC(ALLPINS)
          {
           RELAY_OFF(PIN_DEF,PDCL,1mS);
           RELAY_ON(PIN_DEF,PMU,1mS);
           FORCE_I_PMU(PMU,-100uA,@100uA,@6V,2V,ON,1mS);
           JUDGE_V_PMU(PMU,IFVM,-1.2V,-0.2V,2mS);
           FORCE_I_PMU(PMU,0uA,@100uA,@6V,2V,OFF,1mS);
           RELAY_OFF(PIN_DEF,PMU,1mS);
           RELAY_ON(PIN_DEF,PDCL,1mS);
          }
      power_down();
```

4.13 Oscillator Function Test

4.13.1 JUDGE_FREQ

Perform a frequency measurement and compare the value to the specified limits. Save the measured value and status. Read the measured result and status by "RESULT_PIN" and "STATUS_PIN" statements. Maximum Frequency measure 400 MHz.

4.13.1.1 Format

Parameter list:

JUDGE_FREQ (pin_name, start_addr, stop_addr, ignore_stop_addr, F_min, F_max , freq_range [, divide_count, time_out, TFTM_plus_count]);

Parameter comment:

pin_name: Use the pin_def. Each pin can do Time measurement and per 8 pin shares one Time measurement.

No entry: illegal

start_addr: Start address of the test pattern program.
Describe the Relative start address of the pattern to be executed.
The description is not necessary to start from the top.
Must be represented by a global label [+/- variable or constant] or module name: local_label [+/- variable or constant].
[] Indicates an optional entry.
Type: integer
No entry: CPU trigger mode (Asynchronous measurement).

stop_addr: End address of the test pattern program.
Describe the Relative end address of the pattern to be executed.
The description is not necessary to stop at the end.
Must be represented by a global label [+/- variable or constant] or module name: local_label [+/- variable or constant].
[] Indicates an optional entry.
Type: integer
No entry: CPU trigger mode (Asynchronous measurement)

ignore_stop_addr: Must be represented by a global label [+/- variable or constant] or module name: local_label [+/- variable or constant].
[] Indicates an optional entry.
Type: integer
No entry: ignore pattern result

F_min: Specify the test low limit.
Type: FREQUENCY or double

Unit: HZ, KHZ, MHZ
No entry: only high limit

F_max: Specify the test high limit.
Type: FREQUENCY or double
Unit: HZ, KHZ, MHZ
No entry: only low limit

freq_range: Specifies the input frequency range.

@400MHZ : divide counter = 262144, error=+-15.26ppm
timeout = 3S

@400MHz Release version : **CRAFT_3380_1.20**

@100MHZ : divide counter = 65536, error=+-15.26ppm
timeout = 3S

@10MHZ: divide counter = 16384, error=+-6.11ppm
timeout = 3S

@5MHZ : divide counter = 8192, error=+-6.11ppm
timeout = 3S

@1MHZ : divide counter = 1024, error=+-9.77ppm
timeout = 3S

@100KHZ : divide counter = 128, error=+-7.82ppm
timeout = 3S

@10KHZ : divide counter = 16, error=+-6.25ppm
timeout = 3S

@1KHZ : divide counter = 1, error=+-10ppm
timeout = 3S

freq_range	Suggested input freq range	divided count(cycle)	Supposed input period	Error (ppm)	exhsust time	default time out
@400MHZ	100M~400MHz	262144	2.5nS	+-15.26ppm	0.656mS	3S
@100MHZ	10M~100MHz	65536	10nS	+-15.26ppm	0.656mS	3S
@10MHZ	5M~10MHz	16384	100nS	+-6.11ppm	1.385mS	3S
@5MHZ	1M~5MHz	8192	200nS	+-6.11ppm	1.639mS	3S
@1MHZ	100K~1MHz	1024	1uS	+-9.77ppm	1.025mS	3S
@100KHZ	10K~100KHz	128	10uS	+-7.82ppm	1.29mS	3S
@10KHZ	1K~10KHz	16	100uS	+-6.25ppm	1.7mS	3S
@1KHZ	0~1KHz	1	1mS	+-10ppm	2mS	3S

divide_count: There are 19 kinds of pre-scale value that can be selected;
(If not equal to following 19 values, then F/W will set a nearly value)

→ 2^0 = 1
2^1 = 2
2^2 = 4
2^(3+0) = 8
2^(3+1) = 16
2^(3+2) = 32
2^(3+3) = 64
2^(3+4) = 128
. . . . . .
2^(3+14) = 131072

2^(3+15) = 262144

timeout: Define the waiting time is allowed for measurement from 1uS to 42S. Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: 3 S

TFTM_plus_count: always 0(HW not support yet)
No entry: 0

4.13.1.2 Explanation

- Perform a frequency measurement and compare the value to the specified limits. Save the measured value and status. Read the measured result and status by "RESULT_PIN" and "STATUS_PIN" statements.
  Measure status:
  - 0: measurement valid
  - 1: pattern failed
  - 2: counter overflow

  "measure status" can be read from statement "STATUS_PIN";

- If "freq_range" & "divide_count" are programmed simultaneously, then show warning S/W bypass "freq_range" setting;

- If "freq_range" & "divide_count" are empty, then show the error message.

- S/W run-time-check as following:

  10MHZ < F_min/F_max < 100MHZ → freq_range should be @100MHZ
  Otherwise pc show the warning message

  5MHZ < F_min/F_max < 10MHZ → freq_range should be @10MHZ
  Otherwise pc show the warning message

  1MHZ < F_min/F_max < 5MHZ → freq_range should be @5MHZ
  Otherwise pc show the warning message

  100KHZ < F_min/F_max < 1MHZ → freq_range should be @1MHZ
  Otherwise pc show the warning message

  10KHZ < F_min/F_max < 100KHZ → freq_range should be @100KHZ
  Otherwise pc show warning message

  1KHZ < F_min/F_max < 10KHZ → freq_range should be @10KHZ

- How to calculate error (ppm): The tester used 100MHz cycle base to calculate the divider result. Therefore the error base was 10ns. Take @ 100 KHz for example. The calculator @ 100 KHz range error ppm as follows.

$$Error_ppm = \frac{\pm 10nS}{(Divided_count)\times(Input_Period)}\times 1,000,000$$

$$= \frac{\pm 10nS}{128\times 10uS}\times 1,000,000 = \pm 7.82\,ppm$$

1. Example in using a wrong range: If we want to measure a 2 KHz (0.5mS duration) signal, but we chose @100MHz range to measure it. Due to in @100MHz range that the divider count was 65536, therefore to cumulate the enough cycle to calculate frequency the test time was as follows.

   $$0.5mS\times 65536 = 327.68mS$$

   That 327.68mS is over default time out. The situation can't measure frequency in the time out 3S.

   Another is right example, we use freq_range @ 10 KHz (divider count was 16).

   $$0.5mS\times 16 = 0.08mS$$

4.13.1.3 Example

```
SET_LEVELN(pin_lvl,FFR,vil,vih,0.5V,0.5V,,,,,);
LEVELS(pin_lvl,2mS);
WAIT(10mS);
RELAY_ON(FFR,PDCL,2mS);
JUDGE_FREQ (FFR, , , , 1.5MHZ, 2MHZ , @5MHZ);
```

4.13.2 MEAS_FREQ

Perform a frequency measurement and save the measured value and status. Read the measured result and status by "RESULT_PIN" and "STATUS_PIN" statements. Maximum Frequency measure 400 MHz.

4.13.2.1 Format

Parameter list:

```
MEAS_FREQ (pin_name, start_addr, stop_addr, ignore_addr, freq_range
                [, divide_count, timeout, TFTM_plus_count ] );
```

Parameter comment:

pin_name: Use the pin_def. Each pin can do Time measurement and per 8 pin shares one Time measurement.
No entry: illegal

start_addr: Start address of the test pattern program.
Describe the Relative start address of the pattern to be executed.
The description is not necessary to start from the top.

Must be represented by a global label [+/- variable or constant] or module name: local_label [+/- variable or constant].
[] Indicates an optional entry.
Type: integer
No entry: CPU trigger mode (Asynchronous measurement).

stop_addr: End address of the test pattern program.
Describe the Relative end address of the pattern to be executed.
The description is not necessary to stop at the end.

Must be represented by a global label [+/- variable or constant] or module name: local_label [+/- variable or constant].
[] Indicates an optional entry.
Type: integer
No entry: CPU trigger mode (Asynchronous measurement)

ignore_stop_addr: Must be represented by a global label [+/- variable or constant] or module name: local_label [+/- variable or constant].
[] Indicates an optional entry.
(start_addr~ ignore_addr : ignore)
Type: integer
No entry: ignore pattern result

freq_range: Specifies the input frequency range.
@400MHZ : divide counter = 262144, error=+-15.26ppm
timeout = 3S
@400MHz Release version : **CRAFT_3380_1.20**

@100MHZ : divide counter = 65536, error=+-15.26ppm
timeout = 3S
@10MHZ : divide counter = 16384, error=+-6.11ppm
timeout = 3S
@5MHZ : divide counter = 8192, error=+-6.11ppm
timeout = 3S
@1MHZ : divide counter = 1024, error=+-9.77ppm
timeout = 3S
@100KHZ : divide counter = 128, error=+-7.82ppm
timeout = 3S
@10KHZ : divide counter = 16, error=+-6.25ppm
timeout = 3S
@1KHZ : divide counter = 1, error=+-10ppm
timeout = 3S

freq_range	Suggested input freq range	divided count(cycle)	Supposed input period	Error (ppm)	exhsust time	default time out
@400MHZ	100M~400MHz	262144	2.5nS	+-15.26ppm	0.656mS	3S
@100MHZ	10M~100MHz	65536	10nS	+-15.26ppm	0.656mS	3S
@10MHZ	5M~10MHz	16384	100nS	+-6.11ppm	1.385mS	3S
@5MHZ	1M~5MHz	8192	200nS	+-6.11ppm	1.639mS	3S
@1MHZ	100K~1MHz	1024	1uS	+-9.77ppm	1.025mS	3S
@100KHZ	10K~100KHz	128	10uS	+-7.82ppm	1.29mS	3S
@10KHZ	1K~10KHz	16	100uS	+-6.25ppm	1.7mS	3S
@1KHZ	0~1KHz	1	1mS	+-10ppm	2mS	3S

divide_count: There are 19 kinds of pre-scale value can be selected;

→ 2^0 = 1
2^1 = 2
2^2 = 4
2^(3+0) = 8
2^(3+1) = 16
2^(3+2) = 32
2^(3+3) = 64
2^(3+4) = 128
. . . . . .
2^(3+14) = 131072
2^(3+15) = 262144

timeout: Define the waiting time is allowed for the measurement from1uS to 42S.
Variables and constants are allowed.
Type: TIME or double

Unit: S, pS, nS, uS, mS
No entry: Maximum timeout

TFTM_plus_count: always 0(HW not support yet)
No entry: 0

4.13.2.2 Explanation

- Perform a frequency measurement and save the measured value and status. Read the measured result and status by "RESULT_PIN" and "STATUS_PIN" statements.

Measure status:

0: measurement valid

1: pattern failed

2: time out

"measure status" can be read from statement "STATUS_PIN";

- If "freq_range" & "divide_count" are programmed simultaneously, then show warning S/W bypass "freq_range" setting.
- If "freq_range" & "divide_count" are empty, then show the error message.
- How to calculate error (ppm): The tester used 100MHz cycle base to calculate the divider result. Therefore the error base was 10ns. Take @ 100 KHz for example. To calculate @ 100 KHz range error ppm as follows.

$$Error_ppm = \frac{\pm 10nS}{(Divided_count)\times(Input_Period)}\times 1,000,000$$

$$= \frac{\pm 10nS}{128\times 10uS}\times 1,000,000 = \pm 7.82\,ppm$$

2. Example in using a wrong range: If we want to measure a 2 KHz (0.5mS duration) signal, but we chose @100MHz range to measure it. Due to in @100MHz range that the divider count was 65536, therefore to cumulate the enough cycle to calculate frequency the test time was as follows.

$$0.5mS\times 65536 = 327.68mS$$

That 327.68mS is over default time out. The situation can't measure frequency in the time out 3S.

Another is right example, we use freq_range @ 10 KHz (divider count was 16).

$$0.5mS\times 16 = 0.08mS$$

4.13.2.3 Example

```
SET_LEVELN (pin_lvl,FFR,vil,vih,0.5V,0.5V,,,,,);
LEVELS (pin_lvl, 2mS);
WAIT (10mS);
RELAY_ON (FFR, PDCL, 2mS);
MEAS_FREQ (FFR, osc_st, osc_sp, , @1MHZ);
freq_value = RESULT_PIN( S1, FFR );  /* pattern trigger: triggered by RPM micro-instruction */
MEAS_FREQ (FFR, , , , @100MHZ);
freq_value = RESULT_PIN( S1, FFR );  /* CPU trigger: triggered by CPU */
```

4.13.3 JUDGE_TIME

Perform a time measurement and compare the value to the specified limits. Save the measured value and status. Read the measured result and status by "RESULT_PIN" and "STATUS_PIN" statements.
// measure time resolution : 10nS

4.13.3.1 Format

Parameter list:

JUDGE_TIME (pin_name, meas_type, start_addr, stop_addr. ignore_stop_addr,T_min, T_max, Timeout **[**, TFTM_plus_count **]**);

Parameter comment:

pin_name: Use the pin_def. Each pin can do Time measurement and per 8 pin shares one Time measurement.
No entry: illegal

meas_Type: Specifies the measure mode.
CYCLE: cycle time mode
HIGH_PULSE: high pulse mode
LOW_PULSE: low pulse mode
Type: integer
No entry: CYCLE mode

start_addr: Start address of the test pattern program.
Describe the Relative start address of the pattern to be executed.
The description is not necessary to start from the top.
Must be represented by a global label [+/- variable or constant] or module name: local_label [+/- variable or constant].
[] indicates an optional entry.
Type: integer
No entry: CPU trigger mode (Asynchronous measurement).

stop_addr: End address of the test pattern program.
Describe the Relative end address of the pattern to be executed.
The description is not necessary to stop at the end.
Must be represented by a global label [+/- variable or constant] or module name: local_label [+/- variable or constant] .
[] indicates an optional entry.
Type: integer
No entry: CPU trigger mode (Asynchronous measurement)

ignore_stop_addr: Must be represented by a global label [+/- variable or constant] or module
name: local_label [+/- variable or constant].
[] indicates an optional entry.
Type: integer
No entry: ignore pattern result

T_min: The test low limit.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: only high limit

T_max: The test high limit.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: only low limit
Timeout: Define the waiting time is allowed for the measurement from 1uS to 60S.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Maximum Timeout

Timeout : Define the waiting time is allowed for the measurement from 1uS to 60S.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: 3S

TFTM_plus_count: always 0(HW not support yet)
No entry: 0

4.13.3.2 Explanation

Perform a time measurement and compare the value to the specified limits. Save the measured value and status. Read the measured result and status by "RESULT_PIN" and "STATUS_PIN" statements.
Measure status: 0: measurement valid
1: pattern failed
2: time out
"measure status" can be read from statement "STATUS_PIN";

4.13.3.3 Example

```
SET_LEVELN(pin_lvl,FFR,vil,vih,0.5V,0.5V,,,,,);
LEVELS(pin_lvl,2mS);
WAIT(10mS);
RELAY_ON(FFR,PDCL,2mS);
JUDGE_TIME (FFR, CYCLE, tone_st, tome_sp, , 1uS, 1.1uS, ,200uS);
```

4.13.4 MEAS_TIME

Perform a timing measurement and save the measured value and status. Read the measured result and status by "RESULT_PIN" and "STATUS_PIN" statements.
// measure time resolution : 10nS

4.13.4.1 Format

Parameter list:

MEAS_TIME (pin_name, meas_type, starat_addr, stop_addr, ignore_addr, timeout **[**, TFTM_plus_count **]**);

Parameter comment:

pin_name: Use the pin_def. Each pin can do Time measurement and per 8 pin shares one Time measurement.
No entry: illegal

meas_Type: Specify the measure mode.
CYCLE: cycle mode
HIGH_PULSE: high pulse mode
LOW_PULSE: low pulse mode
Type: integer
No entry: CYCLE mode

start_addr: Start address of the test pattern program.
Describe the Relative start address of the pattern to be executed.
The description is not necessary to start from the top.
Must be represented by al global label [+/- variable or constant] or module name: local_label [+/- variable or constant].
[ ] indicates an optional entry.
Type: integer
No entry: CPU trigger mode (Asynchronous measurement)

stop_addr: End address of the test pattern program.
Describe the Relative end address of the pattern to be executed.
The description is not necessary to stop at the end.
Must be represented by al global label [+/- variable or constant] or module name: local_label [+/- variable or constant].
[ ] indicates an optional entry.
Type: integer
No entry: CPU trigger mode

ignore_addr: Must be represented by a global label [+/- variable or constant] or module name: local_label [+/- variable or constant].
[ ] indicates an optional entry.
(start_addr~ ignore_addr : ignore)
Type: integer
No entry: ignore pattern result

timeout: Define the waiting time is allowed for the measurement from 1uS to 60S.
Variables and constants are allowed.
Type: TIME or double
Unit: S, pS, nS, uS, mS
No entry: Maximum timeout

TFTM_plus_count: always 0(HW not support yet)
No entry: 0

4.13.4.2 Explanation

Perform a timing measurement and save the measured value and status. Read the measured result and status by "RESULT_PIN" and "STATUS_PIN" statements.
Measure status: 0: measurement valid
1: pattern failed
2: time out
"measure status" can be read from statement "STATUS_PIN" ;

4.13.4.3 Example

```
SET_LEVELN(pin_lvl,FFR,vil,vih,0.5V,0.5V,,,,,);
LEVELS(pin_lvl,2mS);
WAIT(10mS);
RELAY_ON(FFR,PDCL,2mS);
MEAS_TIME (FFR, CYCLE, tome_st, tone_sp, , 650uS);
time_value = RESULT_PIN( S1, FFR); /* pattern trigger: triggered by RPM u_instruction
*/

MEAS_TIME (tone_out, H_PULSE, , , ,1mS, 0);
Time_value = RESULT_PIN (S1, tone_out);    /* CPU trigger: triggered by CPU */
```

4.13.5 STATUS_PIN

The statement extracts the result(measured value)after the MEAS_TIME/ MEAS_FREQ /JUDGE_TIME / JUDGE_FREQ statement executed.

4.13.5.1 Format

Parameter list:

```
Status = STATUS_PIN (socket_no, pin_name ) ;
```

Parameter comment:

Status : Specifies the variable to store the measuring status.
The possible returned values are :
0 : Test pattern pass and no time out
1 : Test pattern failed.
2 : time out happened
3 : Test pattern failed and time out happened

socket_no: Specifies the socket can be selected.
S0: socket No. 0
S1: socket No. 1
S2: socket No. 2
S3: socket No. 3
Type: integer
No entry: illegal

pin_name: Specify the pin name by pin_def function.
No entry: illegal

4.13.5.2 Explanation

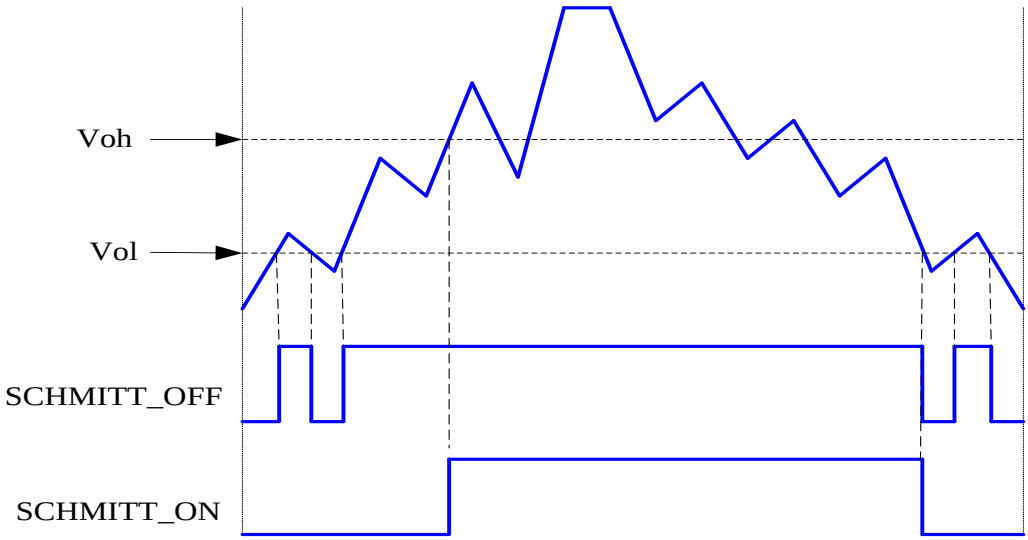
The statement extracts the result (the measured value) after the MEAS_TIME / MEAS_FREQ /JUDGE_TIME / JUDGE_FREQ statements are executed.

4.13.5.3 Example

```
MEAS_TIME(p15, CYCLE, , , , 1mS);
Meas_status = STATUS_PIN(S0, p15);
```

4.13.6 SET_TFTM_MODE

Set Time Frequency Trigger mode.



4.13.6.1 Format

Parameter list:

```
SET_TFTM_MODE(pin_name, trigger_mode );
```

Parameter comment:

pin_name : pin_def, pin_group
**no_entry: illegal

trigger_mode : SCHMITT_OFF → trigger source from PEL signal
SCHMITT_ON → trigger source from PEL/PEH signal (as Schmitt Trigger)
**no_entry : SCHMITT_OFF;

4.13.6.2 Example

```
SET_TFTM_MODE( clk, SCHMITT_ON );
JUDGE_PAT(func_st, func_sp);
JUDGE_FREQ (osc_out, , , , 1.5KHZ, 2KHZ , @10KHZ);
```

4.13.7 OSC_struct

Osc data structure

4.13.7.1 Example

```
struct OSC_struct {
        double      osc_freq; // expected oscllator output frequency
        double      spec;     // expected oscllator output spec
        double      time_out; // search time out
        double      fclk;     // caculated hw fclk rate
        int         mdata;    // caculated hw M rate
        int         ndata;    // caculated hw N rate
        int         pdata;    // caculated hw P rate
        int         status;   // return the caculated result status
};
```

4.13.8 GET_OSC_CLK_DATA

the behavior of oscillator is like a oscillator clock. The statement will return the expected oscillator clock setting data; this data can be used in "OSC_CLK" statement.

4.13.8.1 Format

Parameter list:

status= GET_OSC_CLK_DATA(osc_freq, osc_spec, time_out, osc_data);

Parameter comment:

status:	true: the osc clock data search successful false: search fail type: int
osc_freq:	oscillator frequency value 1Hz to 200MHz variables & constants allowed type: FREQUENCY or double unit: KHZ, MHZ, HZ **no_entry: illegal
osc_spec:	expected oscillator ouput error spec; unit: PPM type: double **no_entry: illegal
osc_data:	osc clock data , return variable type: OSC_struct

4.13.8.2 Explanation

The oscillator Output frequency is:

1.1 FCLKout= FCLKin *M / N / P

Where :M=1~32, N=1~32, P=1~32768

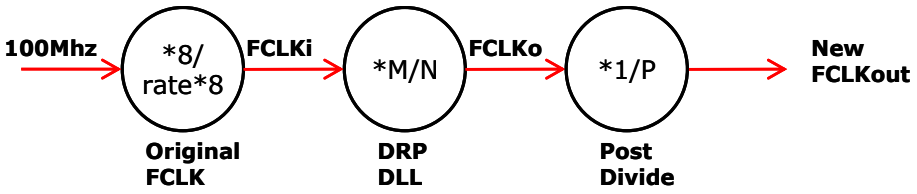
1.2 FPGA DCM :

FCLKi range : 1.0Mhz ~ 200Mhz

FCLKo range: 32.0Mhz ~ 200Mhz

1.3 Bypass on each section.

The dataflow diagram:



4.13.8.3 Example

```
OSC_struct    osc_clk;
int    status;

   PIN_MODE(clk, OSC, EDGE, MASK, IO_ON, OFF);
   // status= GET_OSC_CLK_DATA( 24.545454MHZ, 10PPM, 10S,    osc_clk);    //setup
   in START_UP

   if(osc_clk.status == PASS)    OSC_CLK( clk, osc_clk);
   else
   printf("OSC_CLK statement execution error \n");

   OSC_CLK_LOCK(clk);

   //turn-off oscillator clock output
   PIN_MODE(clk, F0, EDGE, MASK, IO_ON, OFF);
```

Relative:
PIN_MODE

4.13.9 SET_OSC_CLK

The statement will setup a free running clock, the behavior like a oscillator clock.

4.13.9.1 Format

Parameter list:

```
SET_OSC_CLK(pin_name, osc_data);
```

Parameter comment:

pin_name: pin_def (each pin can use as a oscilator-clock
per **16** pin shares one oscilator-clock resource)
**no_entry: illegal

osc_data: type **OSC_struct**,
come from "GET_OSC_CLK_DATA" statement or
manual setting.

4.13.9.2 Example

```
//== OSC data structure ==//
struct OSC_struct {
        double      osc_freq;    // expected oscllator output frequency
        double      spec; // expected oscllator output spec
        double      time_out;    // search time out
        double      fclk;        // caculated hw fclk rate
        int         mdata;       // caculated hw M rate
        int         ndata;       // caculated hw N rate
        int         pdata;       // caculated hw P rate
        int         status;      // return the caculated result status
        double      calc_value;  //returned value after caculating   //20120529
        double      calc_error;  //the error between "osc_freq" and "calc_value"
};

OSC_struct   osc_clk;
   int   status;
      PIN_MODE(clk, OSC, EDGE, MASK, IO_ON, OFF);
      // status= GET_OSC_CLK_DATA( 24.545454MHZ, 10PPM, 10S,   osc_clk);   //setup in
START_UP
      if(osc_clk.status == PASS)   SET_OSC_CLK( clk, osc_clk);
      else

      printf("SET_OSC_CLK statement execution error \n");
      OSC_CLK_LOCK(clk);

      //turn-off oscillator clock output
      PIN_MODE(clk, F0, EDGE, MASK, IO_ON, OFF);
```

Relative:

PIN_MODE

4.13.10 OSC_CLK_LOCK

The statement will triiger the oscillator clock.

4.13.10.1 Format

Parameter list:

```
status=OSC_CLK_LOCK( pin_name);
```

Parameter comment:

status	: true	:: translate complete	(default time out 5S)
	: false	:: translate incomplete	(default time out 5S)

pin_name: pin_def (each pin can use as a oscilator-clock
per 16 pin shares one oscilator-clock resource)
**no_entry: illegal

4.13.10.2 Example

```
OSC_struct   osc_clk;
  int   status;
    PIN_MODE(clk, OSC, EDGE, MASK, IO_ON, OFF);
    // status= GET_OSC_CLK_DATA( 24.545454MHZ, 10PPM, 10S,   osc_clk);   //setup
    in START_UP

    if(osc_clk.status == PASS)   OSC_CLK( clk, osc_clk);
    else
    printf("OSC_CLK statement execution error \n");

    OSC_CLK_LOCK(clk);

    //turn-off oscillator clock output
    PIN_MODE(clk, F0, EDGE, MASK, IO_ON, OFF);
```

Relative:
PIN_MODE

4.13.11 OSC_CLK_MODE

User can use the statement to control oscillator clock mode.

4.13.11.1 Format

Parameter list:

```
OSC_CLK_MODE( pin_name, ON/OFF);
```

Parameter comment:

pin_name: pin_def (each pin can use as a oscillator-clock
per **16** pin shares one oscillator-clock resource)
**no_entry: illegal

ON/OFF:
OFF: turn off ----> restore the system default value
// oscillator clk output controlled **by pattern burst (BUSY) (system default)**
// oscillator clk will be hold if freeze command find in cycle **(system default)**

ON : trun-on "turn_on_mode" & "frz_mode" setting
// oscillator clk output controlled **by statement setting and always running**
// oscillator clk **still running** while freeze command find in cycle
**no_entry: illegal

4.13.11.2 Example

```
        SET_OSC_CLK( clk, 100nS, 25nS, 75nS);
        OSC_CLK_LOCK( clk );
    OSC_struct   osc_clk;
    int   status;
        PIN_MODE(clk, OSC, EDGE, MASK, IO_ON, OFF);
        OSC_CLK_MODE( clk, ON);
        // status= GET_OSC_CLK_DATA( 24.545454MHZ, 10PPM, 10S,   osc_clk);
    //setup in START_UP
        if(osc_clk.status == PASS)   SET_OSC_CLK( clk, osc_clk);
        else        printf("SET_OSC_CLK statement execution error \n");
        OSC_CLK_LOCK(clk);
        ……..
        ……..
        ……..
        //turn-off oscillator clock output
        PIN_MODE(clk, F0, EDGE, MASK, IO_ON, OFF);
        OSC_CLK_MODE( clk, OFF);   //reset
```

Relative:
PIN_MODE

4.14 TMU Control Statement

4.14.1 TMU Pin declare

Pin type : TMU
Channel assign : per pin TMU

[Notice]:

1. A Group has 32 TMU PINS for any site. (Channel % 32=0 ~ Channel %32=31)
   Example: channel0~Channel 31, channel 32~channel 63….

3. When any pin of a group is declared as TMU pin, the other pin will as TMU pins used.

4. TMU pins can't be as norm IO pin.

5. TMU pin-type & pin_group can not be assigned in the same IO_ALLPINS

6. The recommended MXLPC board using the front 0 ~ 31 pins to do HSTMU function, try not to use 32 ~ 63 pins, otherwise the test speed will be limited to 25MHZ in DBL mode or 50MHZ in DBL_2X mode.

Example_1 (correct):

```
PIN_LIST     (tmu_example_correct) {
 //==*name = ATE channel = dut# = type    *==//
      //   S0     :     S1
p0      = 0     :     1      = 1      =TMU; //TMU PIN
p1      = 2     :     3      = 2      =TMU; //TMU PIN
p2      = 4     :     5      = 3      =TMU; //TMU PIN
p3      = 6     :     7      = 4      =TMU; //TMU PIN
p4      = 8     :     28     = 5      =TMU; //TMU PIN
p5      = 10    :     29     = 6      =TMU; //TMU PIN
p6      = 12    :     30     = 7      =TMU; //TMU PIN
p7      = 14    :     31     = 8      =TMU; //TMU PIN
p8      = 32    :     40     = 9      =IO;
p9      = 33    :     41     = 10     =IO;
p10     = 34    :     42     = 11     =IO;
p11     = 35    :     43     = 12     =IO;
p12     = 36    :     44     = 13     =IO;
p13     = 37    :     45     = 14     =IO;
p14     = 38    :     46     = 15     =IO;
p15     = 39    :     47     = 16     =IO;
}
```

Group

Example_2 (incorrect):

```
PIN_LIST     (tmu_example_incorrect) {
 //==*name = ATE channel = dut# = type    *==//
      //   S0
p0      = 0       = 1       =TMU; //TMU PIN
p1      = 2       = 2       =TMU; //TMU PIN
p2      = 4       = 3       =TMU; //TMU PIN
p3      = 6       = 4       =TMU; //TMU PIN
p4      = 8       = 5       =IO;   //incorrect
p5      = 10      = 6       =TMU; //TMU PIN
p6      = 12      = 7       =TMU; //TMU PIN
p7      = 14      = 8       =IO;   //incorrect
p8      = 32      = 9       =IO;
p9      = 33      = 10      =IO;
p10     = 34      = 11      =IO;
p11     = 35      = 12      =IO;
p12     = 36      = 13      =IO;
p13     = 37      = 14      =IO;
p14     = 38      = 15      =IO;
p15     = 39      = 16      =IO;
}
```

Group

4.14.2 SET_TMU_WORK_MODE

Setting TMU Work mode include trigger mode, log length and capture type etc…
Setting VOL/VOH shouldn't be close, waveform log are always : L → Z → H → L → …
Otherwise the result will be a fail value from driver of ATE, the turning point "Z" must be existed.

4.14.2.1 Format

```
SET_TMU_WORK_MODE( pin_name, trigger_mode
        [, _TFTMmode, tftm_mode ]
        [, _DELay, ts_no, delay_time ]
        [,_LOGlength, 1K | 2K | 4K | 8K ]
        [, _CAPTuretype, capture_type ]
        );
```

Parameter list:
_TFTMmode, _DELay, _CAPTuretype, _LOGlength : each parameter Key-Word

pin_name: pin-def or pin-group (only support **even** ate-channel)
PIN TYPE is "**TMU**".
**no_entry: illegal

trigger_mode **:** NORM (CPU low-speed trigger)
UIPP (by pattern u-instruction **TFTM** trigger)
**no_entry : illegle

[, _TFTMmode, tftm_mode**]:** no_entry: SCHMITT_OFF(default) or keep current
_TFTMmode (_TFTM and _TFTMMODE are available), key word.
tftm_mode :

SCHMITT_OFF :
If _CAPTuretype is "**PEL**", trigger by VOL event and record PEL/ PEH data.
If _CAPTuretype is "**PEH**", trigger by VOH event and record PEL/ PEH data.
If _CAPTuretype is "**ALL**", trigger by VOL/VOH event and record PEL/ PEH data.
If _CAPTuretype is "**CONT"**, continue record PEL/PEH data per 625ps period.

SCHMITT_ON :
If _CAPTuretype is "**PEL**"/ "**PEH**"/ "**ALL**", trigger by [from VOL to VOH] or [from VOH to VOL] event and record PEL/ PEH data.
If _CAPTuretype is "**CONT"**, continue record PEL/PEH data per 625ps period.

Relative: SET_TFTM_MODE

[, _DELay, ts_no, delay_time]: no_entry: 0nS (no delay trigger) (default) or keep current
_DELay (_DEL and _DELAY are available), key word.
ts_no: Timing Set (TS1~TS15)
Delay_time: Trigger Time Delay marker for UIPP mode;
Setting resolution: **10nS**
**no_entry: 0nS

[, _LOGlength, log_length]: no_entry: 1K(default) or keep current
_LOGlength (_LOG and _LOGLENGTH are available), key word.
log_length:

For per pin of log length is 1K. In next table, X means that don't use TMU function.

Pin channel	Ch 0	Ch 1	Ch 2	Ch 3	Ch 4	Ch 5	Ch 6	Ch 7
Log length	1K	1K	1K	1K	1K	1K	1K	1K
	2K	X	2K	X	2K	X	2K	X
	4K	X	X	X	4K	X	X	X
	8K	X	X	X	X	X	X	X

[, _CAPTuretype, capture_type]: no_entry: **ALL**(default) or keep current

_CAPTuretype (_CAPT and _CAPTURETYPE are available)
, key word.

Capture_type :

PEL : event log, only log PEL; ignore PEH event)

[Notice]:

NO support

CYCLE / HIGH_PULSE / LOW_PULSE / RISING
FALLING / DUTY measure

PEH : event log, only log PEH; ignore PEL event)

[Notice]:

NO support

CYCLE / HIGH_PULSE / LOW_PULSE / RISING
FALLING / DUTY measure

ALL : event log, (both PEL, PEH)

CONT : continue log PEL, PEH DATA (non-event log)

4.14.2.2 Example

```
Tmu_example()
{
    SET_LEVELN(func_lvl, Allpins, -0.5V, 6V, 0V, 5.5V, 1mA, -1mA, 5V, -1.5V, 6.0V);
    LEVELS(func_lvl, 3mS);
    RATEN(TS1, 1000nS);
    MARKERN(TS1, Allpins, 0nS, 100nS, 0nS, 0nS, 0nS, 40nS, 80nS);
    RELAY_ON( Allpins, PDCL, 1mS);
    PIN_MODE( Allpins, RZ, EDGE, MASK, IO_NRZ, OFF);

    SET_TMU_WORK_MODE( TMUpins, UIPP
                        , _DELAY, TS1, 0nS
                    //[, _TFTMmode, tftm_mode ] no_entry: keep current
                    //[, _DELay, ts_no, delay_time ] no_entry: keep current
                    //[, _LOGlength, Log_length ] no_entry: keep current
                    //[, _CAPTuretype, capture_type ] no_entry: keep current

                    );
    JUDGE_PAT(tmu_uipp_st, tmu_uipp_sp);
```

```
    //==case-1==//
    JUDGE_TMU_RESULT( _NORM, TMUpins, _JUDGE, RISING, 0nS, 5nS );

    //==case-2==//
    JUDGE_TMU_RESULT( _NORM, TMUpins, _JUDGE, FALLING, 0nS, 5nS );

    //==case-3==//
    JUDGE_TMU_RESULT( _NORM, p0, _JUDGE, DUTY, 0.095, 0.105, _WINDOW,
5nS, );

    //==case-4==//
    JUDGE_TMU_RESULT( _NORM, p0, _JUDGE, HIGH_PULSE,    95nS, 105nS );

    //==case-5==//
    JUDGE_TMU_RESULT( _NORM, p0, _JUDGE, LOW_PULSE, 895nS, 905nS );

    //==case-6==//
    JUDGE_TMU_RESULT( _NORM, TMUpins, _JUDGE, CYCLE, 0.995uS, 1.005uS );
}
```

<Test Pattern>

```
    SET_DEC_FILE "3360_tmu_atechk.dec"

    HEADER        p8,p9,p10,p11,p12,p13,p14,p15;



    RPM_PATTERN (tmu_uipp_pat) {
     tmu_uipp_st::
            *00000000* TS1, TFTM;//TFTM:TMU Pattern trig
            *11111111* ;
            *11111111* ;
            *11111111* ;
            *11111111* ;
            *11111111* ;
            *11111111* ;
            *11111111* ;
            *11111111* ;
            *11111111* ;
            *11111111* ;
            *11111111* ;
            *11111111* ;
     tmu_uipp_sp::
            *00000000* ;
    }
```

4.14.3 JUDGE_TMU_RESULT

Perform a timing judgment and save the measured value and status.

4.14.3.1 Format

```
JUDGE_TMU_RESULT (      [ _NORM, pin_name,
                        [, _JUDge, meas_type, low_limit, high_limit
                              [, _RMeas, return_norm_measured_value]
                              [, _RStatus, return_norm_status ]
                              [, _WINdow, window_low_limit, window_high_limit ]
                        ]+
                    ]
                    [ _DIFF, start_pin, stop_pin
                        [, _JUDge, start_type, stop_type, low_limit, high_limit
                              [, _RMeas, return_diff_measured_value]
                              [, _RStatus, return_diff_status ]
                              [, _WINdow, window_low_limit, window_high_limit ]
                        ]+
                    ]*
                    [, _RACTivesite, return_active_site ]
                    );
```

Parameter list:
pin_name: pin-def or power start_pin
PIN TYPE is "**TMU**".
**no_entry: illegal

[, _JUDge, meas_type, low_limit, high_limit]:no_entry: illegal
_JUDge (_JUD and _JUDGE are available), **key word.**
Meas_type:
CYCLE
HIGH_PULSE
LOW_PULSE
DUTY
RISING
FALLING

low_limit: no_entry: only high limit
The low test limit
Type: TIME or double
Unit: S, pS, nS, uS, mS

high_limit: no_entry: only low limit
The high test limit
Type: TIME or double
Unit: S, pS, nS, uS, mS

start_pin:pin_def (only support single-pin) **//Single-Pin**
**no_entry: illegal

stop_pin:pin_def (only support single-pin) **//Single-Pin**
**no_entry: illegal

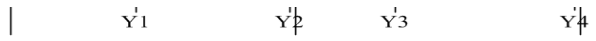
[, _JUDge, start_type, stop_type, low_limit, high_limit]:
no_entry: illegal

_JUDge (_JUD and _JUDGE are available), key word.
start_type:
RISING
FALLING

stop_type:
RISING
FALLING

low_limit: no_entry: only high limit
The low test limit
Type: TIME or double
Unit: S, pS, nS, uS, mS

high_limit: no_entry: only low limit
The high test limit
Type: TIME or double
Unit: S, pS, nS, uS, mS



Start pin (Pin X) type	Stop pin (Pin Y) type	Measure Result
RISING	RISING	Y1-X1
RISING	FALLING	Y2-X1
FALLING	RISING	Y3-X2
FALLING	FALLING	Y2-X2

Start pin (Pin Y) type	Stop pin (Pin X) type	Measure Result
RISING	RISING	X3-Y1
RISING	FALLING	X2-Y1
FALLING	RISING	X3-Y2
FALLING	FALLING	X4-Y2

[, _RMeas, return_norm_measured_value]:

_RMeas (_RM and _RMEAS are available), key word.

Return_norm_measured_value: 2-dimension array (declare by **CARRAY_2DIM**)

return_measured_value [0] [ ]: Site-0 // array size: [site][pin]

return_measured_value [1] [ ]: Site-1 // array size: [site][pin]

Ex: TMUpins = p0+p1+p2;
CARRAY_2DIM<double> duty_value (2, 3);
p0➔ duty_value[site][0]
p1➔ duty_value[site][1]
p2➔ duty_value[site][2]

[Notice]:

1. If log-nothing, then the return_norm_measured_value[] will set **-1.0**
2. Only support Return_norm_measured_value declared by **CARRAY_2DIM** statement

Relative: **CARRAY_2DIM**

[, _RStatus, return_norm_status]:

_RStatus (_RS and _RSTATUS are available), key word.

Return_norm_status : 2-dimension array, // array size: [site][pin]
0: measurement vaild
1: measurement failed

[Notice]:

6 If log-nothing, then the return_norm_status [] will set **-0.0**
7 Only support Return_norm_measured_value declared by **CARRAY_2DIM** statement

[, _RMeas, return_diff_measured_value]:

_RMeas (_RM and _RMEAS are available), key word.

return_diff_measured_value: 2-dimension array2-dimension array (declare by **CARRAY_2DIM**), // array size: [site][1]

return_diff_measured_value[0][0] : Site-0
return_diff_measured_value[1][0] : Site-1
CARRAY_2DIM<double> array_name(site, pin)

[, _RStatus, return_diff_status]:

_RSstatus **(_RS and _RSTATUS are available)**, **key word.**

return_diff_status: 2-dimension array, // array size: [site][1] 0: measurement vaild

1: measurement failed

CARRAY_2DIM<int> array_name(site, 1)

[, **_WIN**dow, window_low_limit, window_high_limit]:

_WINdow **(_WIN and _WINDOW are available)**, **key word.**

window_low_limit: 0nS ~ 1S

can detect the minimum time interval

Type: double, TIME

**no_entry: only high_window

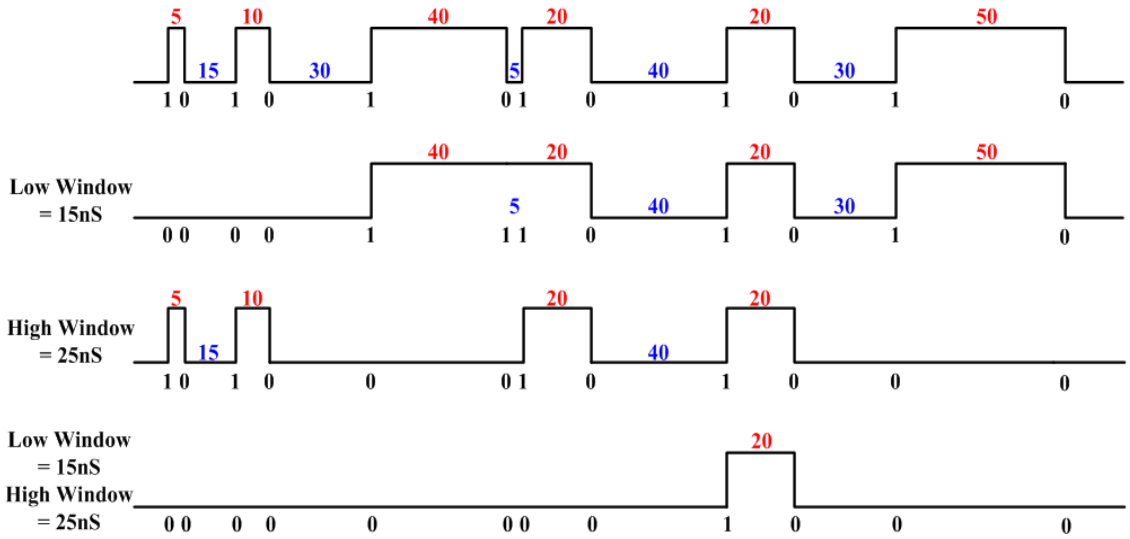
window_high_limit: 0nS ~ 1S

can detect 之 maximum time interval

Type: double, TIME

**no_entry: only low_window

[Notice]: windowing use software filter



[, **_RACT**ivesite, return_active_site]:

_RACTivesite **(_RACT and _RACTIVESITE are available)**

, key word.

return_active_site: 1-dimension array(array size :MAX site)

ex: 2-site

return_active_site [0]: Site-0 (1/0: active-site / none-active-site)

return_active_site [1]: Site-1 (1/0: active-site / none-active-site)

4.14.3.2 Example

```
Tmu_example_1()//norm measure
{
//================array declare================//
CARRAY_2DIM<double> duty_value (2, 15);
CARRAY_2DIM<double> cycle_value (2, 15);
int active_site[4]={0,0,0,0};
SET_LEVELN(func_lvl, Allpins, -0.5V, 6V, 0V, 5.5V, 1mA, -1mA, 5V, -1.5V, 6.0V);
 LEVELS(func_lvl, 3mS);
 RELAY_ON( Allpins, PDCL, 1mS);

 SET_TMU_WORK_MODE (TMUpins, UIPP
                        , _DELAY, TS1, 0nS
                    // [, _TFTMmode, tftm_mode ] no_entry: keep current
                    // [, _LOGlength, Log_length] no_entry: keep current
                    // [, _CAPTuretype, capture_type] no_entry: keep current
                    );
 JUDGE_PAT(tmu_uipp_st, tmu_uipp_sp);//TMU Pattern trig
 //==case-1==//(CYCLE measure)
 JUDGE_TMU_RESULT (_NORM, TMUpins, _JUDGE, CYCLE, 0.9e-6, 1.1e-6,
_WINDOW, 5nS,
                        ,_RMeas, cycle_value
                        );
 //==case-2==//(DUTY measure)
 JUDGE_TMU_RESULT (_NORM, TMUpins, _JUDGE, DUTY, 0.45, 0.55,_RMeas,
duty_value);

 //==case-2==//(multi-judge)
 JUDGE_TMU_RESULT (_NORM, TMUpins
                        , _JUDGE, CYCLE, 0.9e-6, 1.1e-6
                        , _WINDOW, 5nS,
                        ,_RMeas, cycle_value
                        , _JUDGE, DUTY, 0.45, 0.55
                        ,_RMeas, duty_value
                        ,_RACT , active_site
                        );
 for(int ssite=0; ssite < 2; ssite ++){
        if (active_site [ssite]= =0) continue;
        JUDGE_VARIABLE (ssite, cycle_value[ssite][0], 1.0mS, 2.0mS, "cycle time");
//judge p0 value only
  }
```

<Test Pattern>

```
SET_DEC_FILE "3380_tmu_atechk.dec"

HEADER        p8,p9,p10,p11,p12,p13,p14,p15;



RPM_PATTERN (tmu_uipp_pat) {
 tmu_uipp_st::
        *00000000* TS1, TFTM;//TFTM:TMU Pattern trig
        *11111111* ;
        *11111111* ;
        *11111111* ;
        *11111111* ;
        *11111111* ;
```

```
                *11111111* ;
                *11111111* ;
                *11111111* ;
                *11111111* ;
                *11111111* ;
                *11111111* ;
     tmu_uipp_sp::
                *00000000* ;
     }
```

```
    Tmu_example_2()//different(diff) measure
    {
    //================array declare================//
       double return_diff_measured_value[4][2];

       double return_diff_measured_value_1[4][2];

     int active_site[4]={0,0,0,0};
    SET_LEVELN(func_lvl, Allpins, -0.5V, 6V, 0V, 5.5V, 1mA, -1mA, 5V, -1.5V, 6.0V);
     LEVELS(func_lvl, 3mS);
     RELAY_ON( Allpins, PDCL, 1mS);
     SET_TMU_WORK_MODE (TMUpins, UIPP
                                , _DELAY, TS1, 0nS
                              // [, _TFTMmode, tftm_mode ] no_entry: keep current
                              // [, _LOGlength, Log_length] no_entry: keep current
                              // [, _CAPTuretype, capture_type] no_entry: keep current
                              );
     JUDGE_PAT(tmu_uipp_st, tmu_uipp_sp);
     //==case-1==// (Rising-Rising)
     JUDGE_TMU_RESULT (_DIFF, p0, p1,
                              _JUDGE, RISING, RISING, 20e-9, 40e-9,
                              _RMeas, return_diff_measured_value
                              ,_RACT , active_site

                               );
     //==case-2==// (Rising-Falling)
     JUDGE_TMU_RESULT (_DIFF, p0, p1,
                              _JUDGE, RISING, FALLING, 20e-9, 40e-9,
                              _RMeas, return_diff_measured_value_1
                              ,_RACT , active_site
                               );
     //==case-3==// (multi_judge)
     JUDGE_TMU_RESULT (_DIFF, p0, p1,
                              _JUDGE, RISING, RISING, 20e-9, 40e-9
                              ,_RMeas, return_diff_measured_value
                              ,_JUDGE, RISING, FALLING, 20e-9, 40e-9
                              ,_RMeas, return_diff_measured_value_1
                               );
        for(int ssite=0; ssite < 4; ssite ++){
            if (active_site [ssite]= =0) continue;
            JUDGE_VARIABLE (ssite, return_diff_measured_value[ssite][0], 10nS, 20nS,
"DIFF");
      }
   }
```

```
    Tmu_example_3()//norm + diff measure
    {
    //================array declare================//
       double return_diff_measured_value[2][2];

       double return_diff_measured_value_1[2][2];

      CARRAY_2DIM<double> duty_value (2, 15);
      CARRAY_2DIM<double> cycle_value (2, 15);


    SET_LEVELN(func_lvl, Allpins, -0.5V, 6V, 0V, 5.5V, 1mA, -1mA, 5V, -1.5V, 6.0V);
     LEVELS(func_lvl, 3mS);
     RELAY_ON( Allpins, PDCL, 1mS);

     SET_TMU_WORK_MODE (TMUpins, UIPP
                                  , _DELAY, TS1, 0nS
                                // [, _TFTMmode, tftm_mode ] no_entry: keep current
                                // [, _LOGlength, Log_length] no_entry: keep current
                                // [, _CAPTuretype, capture_type] no_entry: keep current
                                );
     JUDGE_PAT(tmu_uipp_st, tmu_uipp_sp);

     JUDGE_TMU_RESULT (_NORM, TMUpins
                                      , _JUDGE, CYCLE, 0.9e-6, 1.1e-6
                                      , _WINDOW, 5nS,
                                      ,_RMeas, cycle_value
                                      , _JUDGE, DUTY, 0.45, 0.55
                                      ,_RMeas, duty_value
                                  _DIFF, p0, p1
                                      ,_JUDGE, RISING, RISING, 20e-9, 40e-9
                                      , _WINDOW, 5nS,
                                      , _RMeas, return_diff_measured_value
                                      _JUDGE, RISING, FALLING, 20e-9, 40e-9
                                      , _RMeas, return_diff_measured_value_1
                                  );
        for (int ssite=0; ssite < 2; ssite ++){
             if (active_site [ssite]= =0) continue;
             JUDGE_VARIABLE (ssite, cycle_value[ssite][0], 1.0mS, 2.0mS, "cycle time");
//judge p0 value only        JUDGE_VARIABLE (ssite,
return_diff_measured_value[ssite][0], 10nS, 20nS, "DIFF");
       }
}
```

4.14.4 MEAS_TMU_RESULT

Perform a timing measure and save the measured value and status.

4.14.4.1 Format

```
MEAS_TMU_RESULT ( [ _NORM, pin_name,
                        [, _MEAStype, meas_type
                        [, _RMeas, return_norm_measured_value]
                        [, _RMultiMeas,
measure_cnt ,return_norm_multi_measured_value]
                        [, _RStatus, return_norm_status ]
                        [, _WINdow, window_low_limit, window_high_limit ]
                        ]+
                  ]
                  [ _DIFF, start_pin, stop_pin
                        [,_MEAStype, start_type, stop_type
                        [, _RMeas, return_diff_measured_value]
                        [, _RMultiMeas,
measure_cnt ,return_norm_multi_measured_value]
                        [, _RStatus, return_diff_status ]
                        [, _WINdow, window_low_limit, window_high_limit ]
                        ]+
                  ]*
                        [, _RACTivesite, return_active_site ]
                  );
```

Parameter list:

pin_name: pin-def or start_pin
PIN TYPE is "**TMU**".
**no_entry: illegal

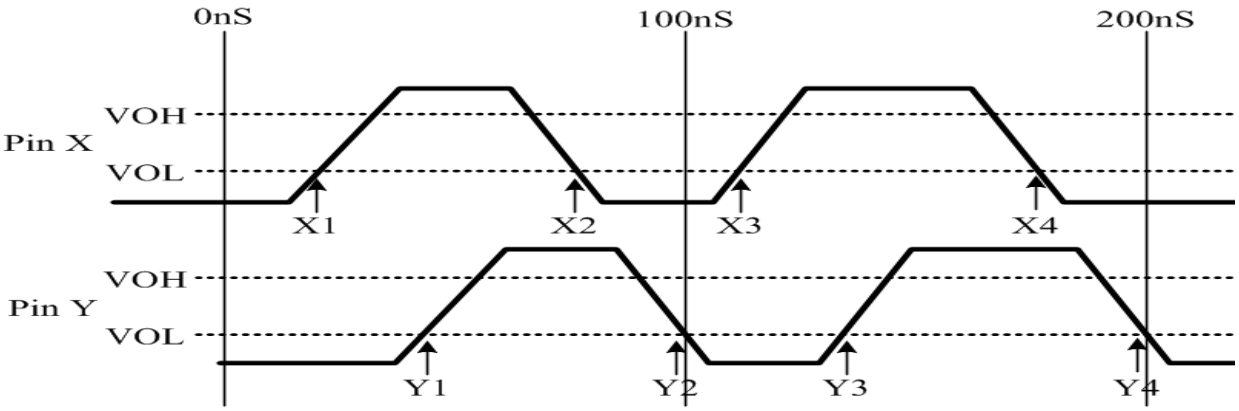
[,_MEAStype, meas_type]:no_entry: illegal
_MEAStype (_MEAS and _MEASTYPE are available), key word.
Meas_type:
CYCLE
HIGH_PULSE
LOW_PULSE
DUTY
RISING
FALLING

[,_MEAStype, start_type, stop_type]:
no_entry: illegal
_MEAStype (_MEAS and _MEASTYPE are available), key word.
start_type:
RISING
FALLING

stop_type:
RISING
FALLING

low_limit: no_entry: only high limit
The low test limit
Type: TIME or double
Unit: S, pS, nS, uS, mS

high_limit: no_entry: only low limit
The high test limit
Type: TIME or double
Unit: S, pS, nS, uS, mS



Start pin (Pin X) type	Stop pin (Pin Y) type	Measure Result
RISING	RISING	Y1-X1
RISING	FALLING	Y2-X1
FALLING	RISING	Y3-X2
FALLING	FALLING	Y2-X2

Start pin (Pin Y) type	Stop pin (Pin X) type	Measure Result
RISING	RISING	X3-Y1
RISING	FALLING	X2-Y1
FALLING	RISING	X3-Y2
FALLING	FALLING	X4-Y2

[, _RMeas, return_norm_measured_value]:

_RMeas (_RM and _RMEAS are available), key word.

Return_norm_measured_value: 2-dimension array (declare by **CARRAY_2DIM**)

return_measured_value [0] [ ]: Site-0 // array size: [site][pin]

return_measured_value [1] [ ]: Site-1 // array size: [site][pin]

Ex: TMUpins = p0+p1+p2;
CARRAY_2DIM<double> duty_value (2, 3);

p0➔ duty_value[site][0]
p1➔ duty_value[site][1]
p2➔ duty_value[site][2]

[Notice]:

1. If log-nothing, then the return_norm_measured_value [] will set **-1.0**
2. Only support Return_norm_measured_value declared by **CARRAY_2DIM** statement
   Type_Name **of CARRAY_2DIM** use double

Relative: **CARRAY_2DIM**

[, _RMultiMeas, measure_cnt ,return_norm_multi_measured_value]:

_RMultiMeas, **key word**
measure_cnt : set measure n data , n<8*1024 , int value
return_norm_multi_measured_value: 3-dimension array (declare by **CARRAY_3DIM**)

[Notice]:

return_norm_multi_measured_value [0] [0][0]: Site0 , pin0, data0
// array size: [site][pin][measure_cnt]
return_norm_multi_measured_value [1] [1][1]: Site1 , pin1, data1
// array size: [site][pin][measure_cnt]

[, _RStatus, return_norm_status]:

_RStatus (_RS and _RSTATUS are available), key word.
Return_norm_status : 2-dimension array, // array size: [site][pin]
0: measurement vaild
1: measurement failed

[Notice]:

1. If log-nothing, then the return_norm_status [] will set **-0.0**
2. Only support Return_norm_measured_value declared by **CARRAY_2DIM** statement
   Type_Name **of CARRAY_2DIM** use double

Relative: **CARRAY_2DIM**

[, _RMeas, return_diff_measured_value]:

_RMeas (_RM and _RMEAS are available), key word.
return_diff_measured_value: 2-dimension array2-dimension array (declare by **CARRAY_2DIM**), // array size: [site][1]

return_diff_measured_value[0][0] : Site-0
return_diff_measured_value[1][0] : Site-1
CARRAY_2DIM<double> array_name(site, pin)

[, _RStatus, return_diff_status]:

_RStatus (_RS and _RSTATUS are available), key word.
return_diff_status: 2-dimension array, // array size: [site][1] 0: measurement vaild
1: measurement failed
CARRAY_2DIM<int> array_name(site, 1)

[, _WINdow, window_low_limit, window_high_limit]:

_WINdow (_WIN and _WINDOW are available), key word.

window_low_limit: 0nS ~ 1S

can detect 之 minimum time interval

Type: double, TIME

**no_entry: only high_window

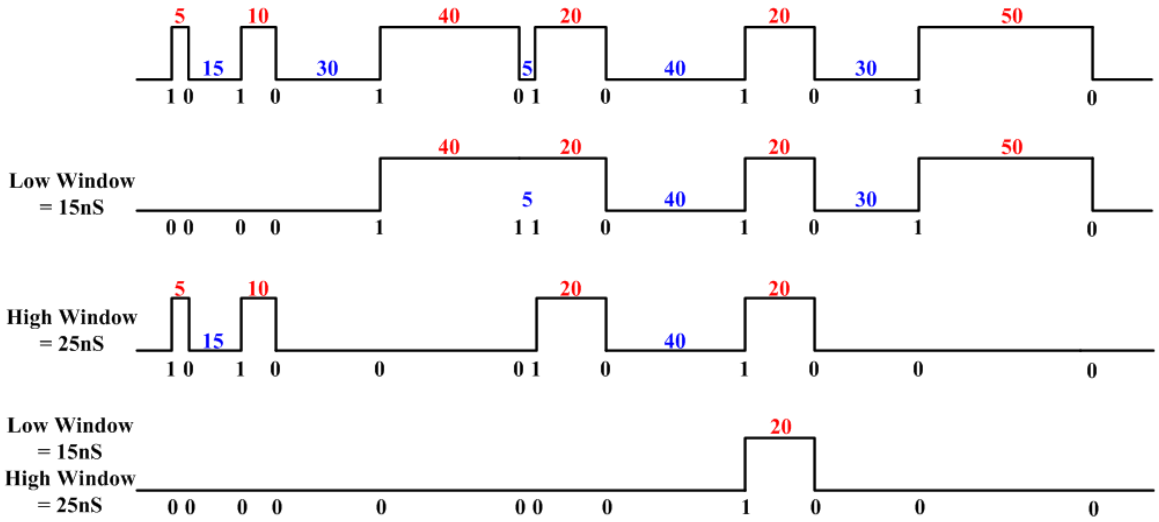
window_high_limit: 0nS ~ 1S

can detect 之 maximum time interval

Type: double, TIME

**no_entry: only low_window

[Notice]: windowing use software filter



[, _RACTivesite, return_active_site]:

_RACTivesite (_RACT and _RACTIVESITE are available)
, key word.

return_active_site: 1-dimension array(array size :MAX site)

Ex: 2-site

return_active_site [0]: Site-0 (1/0: active-site / none-active-site)

return_active_site [1]: Site-1 (1/0: active-site / none-active-site)

4.14.4.2 Example

```
Tmu_example()
    {
    //================array declare================//
    CARRAY_2DIM<double> duty_value (2, 15);
    CARRAY_2DIM<double> cycle_value (2, 15);
    int active_site[4]={0,0,0,0};
    SET_LEVELN (func_lvl, Allpins, -0.5V, 6V, 0V, 5.5V, 1mA, -1mA, 5V, -1.5V, 6.0V);
     LEVELS (func_lvl, 3mS);
     RELAY_ON (Allpins, PDCL, 1mS);

     SET_TMU_WORK_MODE (TMUpins, UIPP        // TMUpins=p0+p1+p2;
```

```
                                , _DELAY, TS1, 0nS
                              //[, _TFTMmode, tftm_mode ]
                              //[, _LOGlength, Log_length ]
                              //[, _CAPTuretype, capture_type ]
                              );
      JUDGE_PAT (tmu_uipp_st, tmu_uipp_sp);

      //==case-1==//(CYCLE measure)
      MEAS_TMU_RESULT (_NORM, TMUpins, _MEAS, CYCLE, _WINDOW, 5nS,    ,
                              _RMeas, cycle_value , _RACT , active_site);

      //==case-2==//(DUTY measure)
      MEAS_TMU_RESULT (_NORM, TMUpins, _MEAS, DUTY, _RMeas,
duty_value ,_RACT , active_site);

     //==case-3==//multi-measure
      MEAS_TMU_RESULT (_NORM, TMUpins
                              , _MEAS, CYCLE
                              , _WINDOW, 5nS,
                              ,_RMeas, cycle_value
                              , _MEAS, DUTY
                              , _RMeas, duty_value
                              ,_RACT , active_site
                              );
      for(int ssite=0; ssite < 2; ssite ++){
              if (active_site [ssite]= =0) continue;
              JUDGE_VARIABLE (ssite, cycle_value[ssite][0], 1.0mS, 2.0mS, "cycle time");
//judge p0 value only
       }
    }
```

<Test Pattern>

```
      SET_DEC_FILE "3360_tmu_atechk.dec"

      HEADER        p8,p9,p10,p11,p12,p13,p14,p15;



      RPM_PATTERN (tmu_uipp_pat) {
       tmu_uipp_st::
                *00000000* TS1, TFTM;//TFTM:TMU Pattern trig
                *11111111* ;
                *11111111* ;
                *11111111* ;
                *11111111* ;
                *11111111* ;
                *11111111* ;
                *11111111* ;
                *11111111* ;
                *11111111* ;
                *11111111* ;
                *11111111* ;
       tmu_uipp_sp::
                *00000000* ;
      }
```

```
Tmu_example()
```

```
    {
    //================array declare================//
    CARRAY_2DIM<double> duty_value (2, 15);
    CARRAY_2DIM<double> cycle_value (2, 15);
    int active_site[4]={0,0,0,0};
    double return_diff_measured_value[4][2] ;

    SET_LEVELN (func_lvl, Allpins, -0.5V, 6V, 0V, 5.5V, 1mA, -1mA, 5V, -1.5V, 6.0V);
     LEVELS (func_lvl, 3mS);
     RELAY_ON (Allpins, PDCL, 1mS);
     SET_TMU_WORK_MODE (TMUpins, UIPP        // TMUpins=p0+p1+p2;
                                  , _DELAY, TS1, 0nS
                                //[, _TFTMmode, tftm_mode ]
                                //[, _LOGlength, Log_length ]
                                //[, _CAPTuretype, capture_type ]
                                );
     JUDGE_PAT (tmu_uipp_st, tmu_uipp_sp);
     MEAS_TMU_RESULT (_NORM, TMUpins
                                        , _MEAS, CYCLE
                                        , _WINDOW, 5nS,
                                        ,_RMeas, cycle_value
                                        , _MEAS, DUTY
                                        , _RMeas, duty_value
                                _DIFF, p0 , p1
                                        ,_MEAS, RISING, RISING
                                        ,_RMeas, return_diff_measured_value
                                ,_RACT , active_site
                                );
     for(int ssite=0; ssite < 2; ssite ++){
            if (active_site [ssite]= =0) continue;
            JUDGE_VARIABLE (ssite, cycle_value[ssite][0], 1.0mS, 2.0mS, "cycle time");
//judge p0 value only        JUDGE_VARIABLE (ssite,
return_diff_measured_value[ssite][0], 10nS, 20nS, "DIFF");
      }
   }
```

```
CARRAY_3DIM<double> Output_Cycle(2,2,1024);    //(Site,Pin,Data)

Meas_loop_CYCLE()
{
     RELAY_ON(Leak_Digital_PINS, PDCL, 1mS);
     RELAY_ON(A_OUT + B_OUT, PDCL, 1mS);
     FORCE_V_MLDPS(VDDA,5V, @6V, @500mA, 200mA, NORM, ON , 1mS);
     PIN_MODE(IO_ALLPINS_1,NRZ,EDGE,ENABLE,IO_NRZ,OFF);
     SET_LEVELN( func_lvl, Leak_Digital_PINS,
0.0V,5.0V,2.5V,2.5V,0.0mA,0.0mA,0.0V, , );
     LEVELS(func_lvl, 3mS);

     JUDGE_PAT(WG_OutPut,__WG_OutPut); //sinewave start to output
     WAIT(10mS);
     SET_LEVELN( func_lvl, A_OUT + B_OUT, 0.0V,1.5V,2.0V,3.0V,0.0mA,0.0mA,0.0V, , );

     SET_TMU_WORK_MODE (A_OUT + B_OUT, NORM, _LOGlength, 1K,
_CAPTuretype, ALL, _TFTM, SCHMITT_ON ); //start to log(CPU trigger)

     WAIT(10mS); //user care delay
```

```
    MEAS_TMU_RESULT( _NORM, A_OUT + B_OUT
                        , _MEAS, CYCLE        , _RMultiMeas, 3, Output_Cycle
                        ); //return 3 data

    SOCKET_INC(FRZ_OFF)
    {
      for(int i=0;i<3;i++)
      {
            JUDGE_VARIABLE(SOCKET,
Output_Cycle[SOCKET][0][i],-10000nS,10000nS,"Cycle_A1");     //(Site,Pin,Data)
          JUDGE_VARIABLE(SOCKET,
Output_Cycle[SOCKET][1][i],-10000nS,10000nS,"Cycle_B1");     //(Site,Pin,Data)
       }
     }
     power_off();

}
```

4.14.5 READ_TMU_LOG_MEM_BLOCK

Read TMU log memory,include PEL/PEH waveform event-log memory data .

4.14.5.1 Format

READ_TMU_LOG_MEM_BLOCK(pin_name, st_event, sp_event, PEL_data, PEH_data
[, _RPINData, return_pin_data]);

Parameter list:
pin_name: pin-def or start_pin
PIN TYPE is "**TMU**".
**no_entry: illegal

st_event : read start event : 0 to 1023*16(setting by SET_TMU_WORK_MODE) //max: 16 event log in 10nS, so resolution 625pS
sp_event: read stop event : 0 to 1023*16(setting by SET_TMU_WORK_MODE)

[Notice]: If sp_event KEEP_CURRENT (empty), then sp_event will be the final pointer of TMU log memory.

PEL_data: 3-dimension array, PEL_data **[**Site**] [**Pin**] [**Event**]**

PEL_data **[**Site**] [**Pin**] [**Event**]**.status: to read back data 0 or 1
0: < Vol
1: > Vol
-1: last-data
Type: integer
**no_entry: illegal

PEL_data **[**Site**] [**Pin**] [**Event**]**.timing: PEL waveform event log data, If input NULL, then not to read
last-data: return -1.0
Type: double, TIME

PEH_data: 3-dimension array, PEH_data **[**Site**] [**Pin**] [**Event**]**

PEH_data **[**Site**] [**Pin**] [**Event**]**.status: to read back data 0 or 1
1: < Voh
0: > Voh
-1: last-data
Type: integer
**no_entry: illegal

PEH_data **[**Site**] [**Pin**] [**Event**]**.timing: PEH waveform event log data if input NULL, then not to read
last-data : return -1.0
Type: double, TIME

[, _RPINData, return_pin_data]:

_RPINDate (_RPIND and _RPINDATE are available)

return_pin_data: 3-dimension array, PIN_data **[**Site**] [**Pin**] [**Event**]**

PIN_data **[**Site**] [**Pin**] [**Event**]**.status: to read back data 0 or 1

0: < Vol

1: > Vol & < Voh

2: > Voh

3: error

-1: last-data

Type: integer

**no_entry: illegal

PIN_data **[**Site**] [**Pin**] [**Event**]**.timing: pin waveform event log data

If input NULL, then not to read

last-data : return -1.0

Type: double, TIME

[Notice]:

1. Only support PIN_data, PEH_data and PEL_data declared by **CARRAY_3DIM** statement
   Type_Name of **CARRAY_3DIM** use TMU_struct

Relative: **CARRAY_3DIM**

PEL.status 1 1 1 0 1 1 1 0

4.14.5.2 Example

```
tmu_example
{
   //===============array declare============//
   CARRAY_3DIM<TMU_struct>      PEL_data(2,3,1024);
   CARRAY_3DIM<TMU_struct>      PEH_data(2,3,1024);
   CARRAY_3DIM<TMU_struct>      PIN_data(2,3,1024);
   int ii,jj;

     SET_LEVELN (func_lvl, Allpins, -0.5V, 6V, 0V, 5.5V, 1mA, -1mA, 5V, -1.5V, 6.0V);
      LEVELS (func_lvl, 3mS);
      RELAY_ON (Allpins, PDCL, 1mS);

      SET_TMU_WORK_MODE (TMUpins, UIPP         // TMUpins=p0+p1+p2;
                                  , _DELAY, TS1, 0nS
                                //[, _TFTMmode, tftm_mode ]
                                //[, _LOGlength, Log_length ]
                                //[, _CAPTuretype, capture_type ]
                                );
      JUDGE_PAT (tmu_uipp_st, tmu_uipp_sp);
     READ_TMU_LOG_MEM_BLOCK( TMUpins, 0, 1023, PEL_data, PEH_data ) ;
     for(ii=0; ii<3; ii++){
            for(jj=0; jj<1024; jj++){
            printf("PIN[ii] ,PEL_EVENT[jj] timing:%f , status:%d\n",
            PEL_data[0][ii][jj].timing, PEL_data[0][ii][jj].status);

            printf("PIN[ii] ,PEH_EVENT[jj] timing:%f , status:%d\n",
            PEH_data[0][ii][jj].timing, PEH_data[0][ii][jj].status);
            }
      }
}
```

4.14.6 TMU_DATA_ARITH

Calculate TMU dynamic characteric data.

4.14.6.1 Format

```
TMU_DATA_ARITH (data_Site_cnt, data_Pin_cnt, data_start, data_stop, PEL_data,
PEH_data
            [, _MEASType, meas_type, _RMeas, return_meas_data
                [, _WINdow, window_low_limit, window_high_limit ]
            ]+) ;
```

Parameter list:

data_Site_cnt: ALL Site count

data_Pin_cnt: ALL pin count

PEL_data : intput data
type TMU_struct; //TMU_struct : CRAFT define TMU data structure

PEH_data : input data
type TMU_struct; //TMU_struct : CRAFT define TMU data structure

,**_MEAS**type, meas_type**:**no_entry: illegal
_MEAStype **(_MEAS and _MEASTYPE are available)**, **key word.**
Meas_type:
CYCLE
HIGH_PULSE
LOW_PULSE
DUTY
RISING
FALLING

, **_RM**eas, return_meas_data**:** no_entry: illegal
_RMeas **(_RM and _RMEAS are available)**, **key word.**
Return_norm_measured_value: 2-dimension array (declare by **CARRAY_2DIM**)

return_measured_value [0] [ ]: Site-0 // array size: [site][pin]

return_measured_value [1] [ ]: Site-1 // array size: [site][pin]

Ex: TMUpins = p0+p1+p2;
CARRAY_2DIM<double> duty_value (2, 3);
p0➔ duty_value[site][0]
p1➔ duty_value[site][1]
p2➔ duty_value[site][2]

[Notice]:

1. If log-nothing, then the return_norm_measured_value [] will set **-1.0**
2. Only support Return_norm_measured_value declared by **CARRAY_2DIM** statement

Type_Name **of CARRAY_2DIM** use double
Relative: **CARRAY_2DIM**

[, _WINdow, window_low_limit, window_high_limit]:

_WINdow **(_WIN and _WINDOW are available)**, **key word.**

window_low_limit: 0nS ~ 1S
can detect 之 minimum time interval

Type: double, TIME
**no_entry: only high_window

window_high_limit: 0nS ~ 1S
can detect 之 maximum time interval
Type: double, TIME
**no_entry: only low_window

[Notice]: windowing use software filter

4.14.6.2 Example

```
tmu_example
{
   //===============array declare============//
   CARRAY_3DIM<TMU_struct>      PEL_data(2,3,1024);
   CARRAY_3DIM<TMU_struct>      PEH_data(2,3,1024);
   CARRAY_3DIM<TMU_struct>      PIN_data(2,3,1024);
   CARRAY_2DIM<TMU_struct>      meas_cycle(2,3);
   CARRAY_2DIM<TMU_struct>      meas_duty(2,3);
   int ii,jj,ssite,pin_num;

     SET_LEVELN (func_lvl, Allpins, -0.5V, 6V, 0V, 5.5V, 1mA, -1mA, 5V, -1.5V, 6.0V);
      LEVELS (func_lvl, 3mS);
      RELAY_ON (Allpins, PDCL, 1mS);

      SET_TMU_WORK_MODE (TMUpins, UIPP        // TMUpins=p0+p1+p2;
                                  , _DELAY, TS1, 0nS
                                //[, _TFTMmode, tftm_mode ]
                                //[, _LOGlength, Log_length ]
                                //[, _CAPTuretype, capture_type ]
                                );
      JUDGE_PAT (tmu_uipp_st, tmu_uipp_sp);
     READ_TMU_LOG_MEM_BLOCK( TMUpins, 0, 1023, PEL_data, PEH_data ) ;
     TMU_DATA_ARITH (2, 3, 0, 1023, PEL_data, PEH_data
             , _MEASType, CYCLE      , _RMeas, meas_cycle
             , _MEASType, DUTY       , _RMeas, meas_duty
             );

     for(ssite=0; ssite<2; ssite++){
       for(pin_num=0; pin_num<3; pin_num++){
             JUDGE_VARIABLE (ssite, meas_cycle [ssite][pin_num], 1.0mS, 2.0mS, "cycle
time");
         }
     }
}
```

4.14.7 CARRAY_3DIM

Create a three-dimension array

4.14.7.1 Format

```
CARRAY_3DIM <Type_Name> Name(Size, Size, Size) ;
```

Parameter list:
Type_Name: array data type
Keyword: double, float, int, **TMU_struct** etc…
<TMU_struct>

```
struct TMU_struct
{
        double      timing;
        int         status;
};
```

Name: array name
Size: array size
Type: int

4.14.7.2 Example

```
//==========array type "int"===========//
CARRAY_3DIM<int> array_int (10, 20, 30);

//==========array type "double"===========//
CARRAY_3DIM<double> array_double (10, 20, 30);

//==========array type "TMU_struct"===========//
CARRAY_3DIM< TMU_struct > array_tmu (10, 20, 30);
```

4.14.8 CARRAY_2DIM

Create a two-dimension array

4.14.8.1 Format

```
CARRAY_2DIM <Type_Name> Name(Size, Size) ;
```

Parameter list:
Type_Name: array data type
Keyword: double, float, int, **TMU_struct** etc…
<TMU_struct>

```
struct TMU_struct
{
        double      timing;
        int         status;
};
```

Name: array name
Size: array size
Type: int

4.14.8.2 Example

```
//==========array type "int"===========//
CARRAY_2DIM<int> array_int (10, 20);

//==========array type "double"===========//
CARRAY_2DIM<double> array_double (10, 20);

//==========array type "TMU_struct"===========//
CARRAY_2DIM< TMU_struct > array_tmu (10, 20);
```

4.14.9 CARRAY_DIM

Create a 1-dimension array

4.14.9.1 Format

```
CARRAY_DIM <Type_Name> Name(Size) ;
```

Parameter list:
Type_Name: array data type
Keyword: double, float, int, **TMU_struct** etc…
<TMU_struct>

```
struct TMU_struct
{
        double      timing;
        int         status;
};
```

Name: array name
Size: array size
Type: int

4.14.9.2 Example

```
//==========array type "int"===========//
CARRAY_DIM<int> array_int (20);

//==========array type "double"===========//
CARRAY_DIM<double> array_double (20);

//==========array type "TMU_struct"===========//
CARRAY_DIM< TMU_struct > array_tmu (20);
```

4.15 Function Test

4.15.1 JUDGE_PAT

This statement executes the specific test pattern program to carry out the function test.

4.15.1.1 Format

Parameter list:

```
JUDGE_PAT (start_addr, stop_addr [, stop_plus_count, ignore_start_addr, ignore_stop_addr,
           Ignore_stop_plus_count, fail_count_limit, log_start_address, log_plus_count ]) ;
```

Parameter comment:

start_addr: Specifies the start address (the Relative address) of the test pattern program.
SPM addr is 16M to 32M-1.
A SPM address from the pattern file.
Must be represented by a global label [+/- variable or constant] or module name: local_label [+/- variable or constant] .

Type: integer
No entry: illegal

stop_addr: Specifies the end address(the Relative address) of the test pattern program.
SPM addr is 16M to 32M-1.
A SPM address from the pattern file.
Must be represented by a global label [+/- variable or constant] or module name: local_label [+/- variable or constant].
Type: integer
No entry: stop by micro-instruction or the module last address.

stop_plus_count: 0 <= stop_plus_count <= 64k-1
Type: integer
No entry: 0

ignore_start_addr: Specifies the ignore start address(absolute address) of the test pattern program.
SPM address is 16M to 32M-1.
A SPM address from the pattern file must be represented by a global label [+/- variable or constant] or module name: local_label [+/- variable or constant] .
Type: integer
No entry: ignore off

ignore_stop_addr: Specifies the ignore end address(absolute address) of the test pattern program.
SPM address is 16M to 32M-1.
A SPM address from the pattern file must be represented by a

global label [+/- variable or constant] or module name: local_label [+/- variable or constant] .
Type: integer
No entry: ignore off

ignore_stop_plus_count: 0 <= ignore_count < = 128
Type: integer
No entry: 0

fail_count_limit: 1 <= fail_count < = 128
fail_counter increment as vector fail unless it is ignored.
Type: integer
No entry: 1

[Notice]: fail_count_limit not support "APAS MODE"

log_start_address: Specifies the log start address of the test pattern program.
Set log start address.
Must be represented by a global label [+/- variable or constant] or module name: local_label [+/- variable or constant] .
No entry: start_address

log_plus_count : 0 <= log_plus_count < = 64k-1
Type: integer
No entry: 0

4.15.1.2 Explanation

Execute a pre-loaded pattern set, and return fail if any pin failed any vector.

4.15.1.3 Example

```
RATEN(TM1, 1uS);
MARKERN(TM1,        CLK, 250nS, 750nS, , 0nS,    , 850nS,    );
MARKERN(TM1,       CTRL,     0nS,        ,    , 0nS,    , 850nS,    );
MARKERN(TM1, OUTPUTS,     0nS,        ,    , 0nS,    , 950nS,    );
SET_LEVELN(func_lvl, IO_ALLPINS,vil,vilh,vol,voh,0uA,0uA,0V);
LEVELS(func_lvl, 3mS);
PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE(           CLK,    RZ, EDGE,     MASK, IO_NRZ, OFF);
RELAY_ON( IO_ALLPINS, PDCL, 3mS);

//Judge pattern programming example
TEST_NO(20);
JUDGE_PAT(func_pat, __func_pat);//Judge Pattern Module
TEST_NO(21);
JUDGE_PAT(sfr_st, sfr_sp);//Judge Global Label
TEST_NO(22);
JUDGE_PAT(func_pat:sfr_st, func_pat:sfr_sp);//Judge Local Label
TEST_NO(23);
JUDGE_PAT(func_pat+2, func_pat+4);//Judge +Cycle count
```

Relative:
RATEN , MARKERN , SET_LEVELN , LEVELS, PIN_MODE, RELAY_ON

4.15.2 PAT_CONTINUE

Continue a pattern execution suspended by the JCONT u-instruction.
All pin stop status is the last vector.

4.15.2.1 Format

Parameter list:

```
PAT_CONTINUE( ) ;
```

Parameter comment:
None

4.15.2.2 Explanation

Continue a pattern execution is suspended by the JCONT u-instruction.

4.15.2.3 Example

```
//u_in : JCONT
DATALOG_MSG="IDD Dynamic Test and u_in : JCONT";
TEST_NO(50);
JUDGE_PAT(jcont1_pat, __jcont1_pat);//JCONT

DATALOG_MSG="Judge I mode";
TEST_NO(51);
JUDGE_I_DPS ( Vddd, idd.min, idd.max, 2mS);
JUDGE_I_PREF(  Vdd, idd.min, idd.max, 2mS);
PAT_CONTINUE();
```

Pattern Loop stops and finish .

Relative:
Pattern Micro-instruction

4.15.3 LOAD_PAT

Load the compiled pattern file into the hardware pattern memory.

4.15.3.1 Format

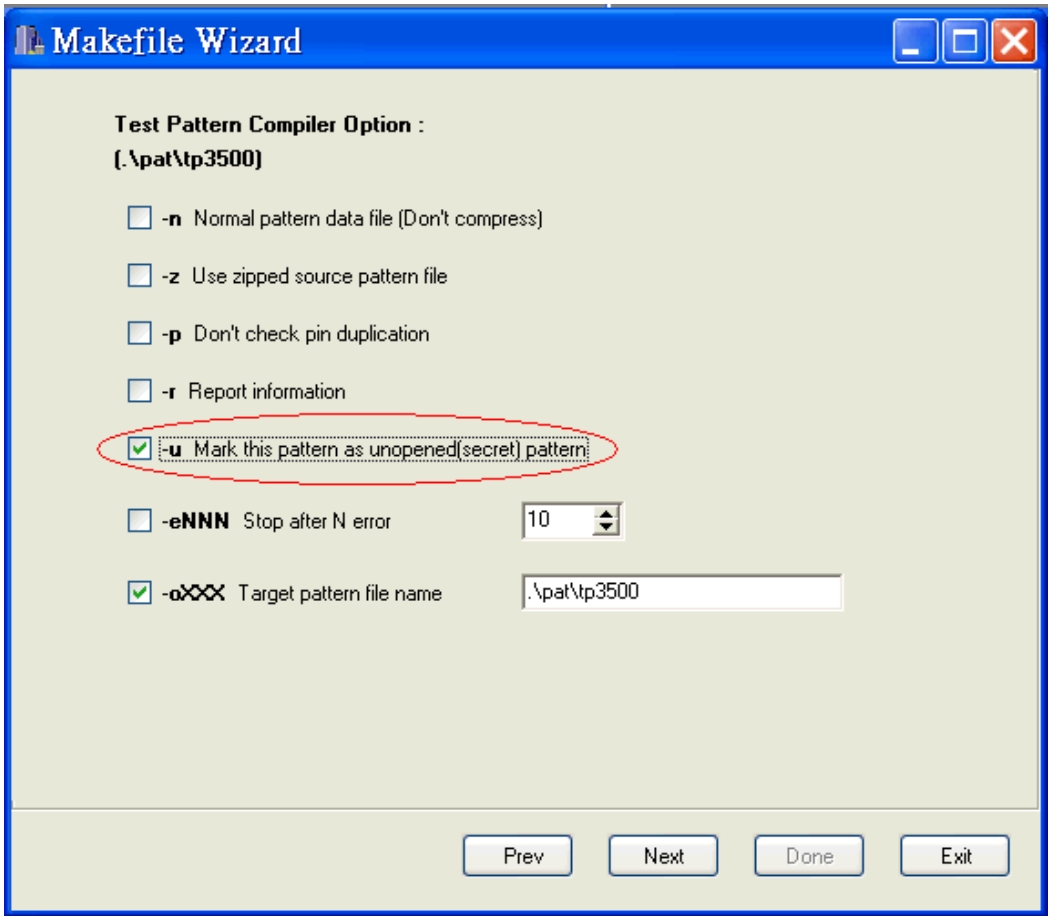
Parameter list:

```
LOAD_PAT ("./pattern_object_file_name" [, SPM_start_address]) ;
```

Parameter comment:

pattern_object_file_name: Maximum: an array of 32 characters.
No entry: illegal
Filename Extension: ppo, ppos (secret)

[Notice]: The ppos file is protected can not read back from ATE.
READ_MEM_PAT and VECTOR_EXTRACT statement are invalid.
When compiler pattern source, use "–u" parameter to create a ppos (secret) file.



SPM_start_address: Specifies the beginning address of loaded_pattern.
No entry: continue the last vector address of last pattern.

4.15.3.2 Explanation

Load the compiled pattern file into the hardware pattern memory. These patterns are bound together into a single binary file; optionally, the user may select a single binary segment by specifying the original source file, intended to be loaded.

4.15.3.3 Example

```
LOAD_PAT (". / PAT / ls299pat.ppo" );
LOAD_PAT (".\\ PAT \\ ls299pat.ppo" );    /*because "\" is escape sign in c-language*/
```

```
LOAD_PAT (". / PAT / ls299pat.ppos" );

LOAD_PAT (".\\ PAT \\ ls299pat.ppos" );
```

4.15.3.4 Relative

PAT_SECURITY

4.15.4 VECTOR_EXTRACT

This statement enable the vector extract and save the pattern.

4.15.4.1 Format

Parameter list:

```
VECTOR_EXTRACT(socket_no , start_addr, stop_addr, stop_plus_count, ve_start, ve_stop,
                    ve_burst_limit, pattern_name, ve_status );
```

Parameter comment:

socket_no : S0 : socket No. 0
S1 : socket No. 1
S2 : socket No. 2
S3 : socket No. 3
Type : integer
**no_entry: illegal

start_addr: SPM addr is 16M to 32M-1 a SPM address from the pattern file. Must be represented by a global label [+/- variable or constant] or module name:local_label [+/- variable or constant] .
type: integer
**no_entry: illegal

stop_addr: SPM addr is 16M to 32M-1 a SPM address from the pattern file. Must be represented by a global label [+/- variable or constant]. or module name:local_label [+/- variable or constant] .
type: integer
**no_entry: stop by micro-instruction or the module last address

stop_plus_count: 0 <= stop_plus_count <= 64k-1
type: integer
**no_entry: 0

ve_start : vector extract start address.
**no_entry: the same as the start_addr.

ve_stop : vector extract stop address ,
**no_entry: the same as the stop_addr.

ve_burst_limit : vector extract burst limit ,
**no_entry: 10 .

pattern_name : the file name of the extract pattern .

ve_status : 0 : vector extract pass
1: Vectors extract fail

4.15.4.2 Explanation

This statement enables the vector extract and save the pattern.

4.15.4.3 Example

```
VECTOR_EXTRACT( S0,romce_st, romce_sp, 0, ve_st, ve_sp, 100, "romce.ve", ve_status);
```

4.15.5 SET_JUDGE_MODE

Set judge mode of "JUDGE_PAT" statement.

4.15.5.1 Format

Parameter list:

```
SET_JUDGE_MODE(judge_mode) ;
```

Parameter comment:

judge_mode : NORM : (maximum speed 25MHz)
DBL : double speed (maximum speed 50MHz),
pattern limitation : only even cycle can use micro-instruction,
others as NORM coding
DBL_2X : (maximum speed 100 MHz)
pattern limitation : only 4*i th(i=0,1,2,3….) cycle can use micro-instruction,
others as NORM & DBL coding.
type: integer
default : NORM
**no_entry: NORM

4.15.5.2 Explanation

1. DBL mode pattern limitation: only even cycle can use micro-instruction ; DBL_2X mode pattern limitation: only 4*I th(i=0,1,2,3….) cycle can use micro-instruction.
2. DBL_2X: User has to put statement of SET_JUDGE_MODE before RATEN and MARKERN.
3. DBL_2X: The pins of pattern must be defined in statement of MARKERN.
4. Using K_SET or Z_SET, we have to avoid change IO symbol fastly.

For example:

```
*1H *;
*H1 *;
*0L *;
*L0 *;
```

If pattern symbols is a error combination, system will check status like next:

```
SPM_PATTERN   (DBL_2X_K_SET, DBL_2X, K_SET)
{
      *ZZ*   // ← Compile error
      *ZX*   // ← Compile error
      *K1*
      *K0*
}
```

5. DBL_2X Pattern Set

	NORM_SET (Default)		Z_SET		K_SET	
Pattern Symbol Combination	0	0	0	0	0	0
	0	1	0	1	0	1
	0	L	0	L	0	L
	0	H	0	H	0	H
	0	X	0	X	0	X
	1	0	1	0	1	0
	1	1	1	1	1	1
	1	L	1	L	1	L
	1	H	1	H	1	H
	1	X	1	X	1	X
	L	0	L	0	L	0
	L	1	L	1	L	1
	L	L	L	L	L	L
	L	H	L	H	L	H
	L	X	L	X	L	X
	H	0	H	0	H	0
	H	1	H	1	H	1
	H	L	H	L	H	L
	H	H	H	H	H	H
	H	X	H	X	H	X
	X	0	X	0	X	0
	X	1	X	1	X	1
	X	L	X	L	X	L
	X	H	X	H	X	H
	X	X	X	X	X	X
	Z	Z	Z	Z	K	K
	V	V	Z	0	K	0
			Z	1	K	1
			Z	X	K	X
			0	Z	0	K
			1	Z	1	K
			X	Z	X	K

4.15.5.3 Example

```
//SET_JUDGE_MODE ( DBL );
//SET_JUDGE_MODE ( DBL_2X );
SET_JUDGE_MODE ( NORM);
```

Example of DBL_2X:

```
SET_DEC_FILE "3380P_SN74AVC16245.dec"
HEADER        DIR,OE,A1,A2,A3,A4,A5,A6,A7,A8,B1,B2,B3,B4,B5,B6,B7,B8;

SPM_PATTERN (func_run_DBL_2X, DBL_2X)
{
                    * 1000000000XXXXXXXX *TS1;    //follow 4.14.5.2 Explanation
point 1)
                    * 1010000000HLLLLLLL *;
                    * 1001000000LHLLLLLL *;
                    * 1000100000LLHLLLLL *;
                    * 1000010000LLLHLLLL *;
                    * 1000001000LLLLHLLL *;
                    * 1000000100LLLLLHLL *;
                    * 1000000010LLLLLLHL *;
                    * 1000000001LLLLLLLH *;
                    * 1000000010LLLLLLHL *;
                    * 1000000100LLLLLHLL *;
                    * 1000001000LLLLHLLL *;
                    * 1000010000LLLHLLLL *;
                    * 1000100000LLHLLLLL *;
                    * 1001000000LHLLLLLL *;
                    * 1010000000HLLLLLLL *;
                    * 1001111111LHHHHHHH *;
                    * 1010111111HLHHHHHH *;
                    * 1011011111HHLHHHHH *;
                    * 1011101111HHHLHHHH *;
                    * 1011110111HHHHLHHH *;
                    * 1011111011HHHHHLHH *;
                    * 1011111101HHHHHHLH *;
                    * 1011111110HHHHHHHL *;
                    * 1011111101HHHHHHLH *;
                    * 1011111011HHHHHLHH *;
                    * 1011110111HHHHLHHH *;
                    * 1011101111HHHLHHHH *;
                    * 1011011111HHLHHHHH *;
                    * 1010111111HLHHHHHH *;
                    * 1001111111XXXXXXXX *;
                    * 00HLLLLLLL10000000 *;
                    * 00LHLLLLLL01000000 *;
                    * 00LLHLLLLL00100000 *;
                    * 00LLLHLLLL00010000 *;
                    * 00LLLLHLLL00001000 *;
                    * 00LLLLLHLL00000100 *;
                    * 00LLLLLLHL00000010 *;
                    * 00LLLLLLLH00000001 *;
                    * 00LLLLLLHL00000010 *;
                    * 00LLLLLHLL00000100 *;
                    * 00LLLLHLLL00001000 *;
```

```
                    * 00LLLHLLLL00010000 *;
                    * 00LLHLLLLL00100000 *;
                    * 00LHLLLLLL01000000 *;
                    * 00HLLLLLLL10000000 *;
                    * 00LHHHHHHH01111111 *;
                    * 00HLHHHHHH10111111 *;
                    * 00HHLHHHHH11011111 *;
                    * 00HHHLHHHH11101111 *;
                    * 00HHHHLHHH11110111 *;
                    * 00HHHHHLHH11111011 *;
                    * 00HHHHHHLH11111101 *;
                    * 00HHHHHHHL11111110 *;
                    * 00HHHHHHLH11111101 *;
                    * 00HHHHHLHH11111011 *;
                    * 00HHHHLHHH11110111 *;
                    * 00HHHLHHHH11101111 *;
                    * 00HHLHHHHH11011111 *;
                    * 00HLHHHHHH10111111 *;
                    * 00XXXXXXXX01111111 *;
                    * 1100000000XXXXXXXX *;
                    * 1100000000XXXXXXXX *;
                    * 1100000000XXXXXXXX *;
}
```

```
func_dbl_2x_test()
{
      FORCE_V_MLDPS(Vcca, 1.8V, @12V, @500mA, 500mA, NORM, ON, 2mS);
      SET_LEVELN(func_DBL_2X_lvl, Allpins, 0.00V, 1.8V, 0.65V, 1.1V, , , , , );
      LEVELS(func_DBL_2X_lvl,5mS);
      WAIT(5mS);
      PIN_MODE(Allpins, NRZ, EDGE, ENABLE, IO_NRZ, NORM);
      PIN_MODE(ctrl,       NRZ, EDGE, MASK, IO_NRZ, NORM);
      RELAY_ON(Allpins,PDCL,3mS);
      // The group of Allpins = DIR+OE + A1+A2+A3+A4+A5+A6+A7+A8
B1+B2+B3+B4+B5+B6+B7+B8;

      SET_JUDGE_MODE (DBL_2X); //follow 4.14.5.2 Explanation point 2)

      RATEN(TS1,10nS);
      MARKERN(TS1, Allpins, 0nS, , , 0nS, 0nS, 8.125nS, 10nS);
       //follow 4.14.5.2 Explanation point 3)

      JUDGE_PAT(func_run_DBL_2X,__func_run_DBL_2X);
      SET_JUDGE_MODE (NORM);

      SET_LEVELN( rst_lvl, Allpins, 0.0V,0.5V,0.0V,0.0V,0.0mA,0.0mA,0.0V, -2V,
6.5V);
      LEVELS(rst_lvl, 3mS);
      RELAY_OFF(Allpins,PDCL,1mS);
      FORCE_V_MLDPS(Vcca, 0.0V, @12V, @500mA, 500mA, NORM, OFF, 2mS);
}
```

4.16 SOCKET Control function

4.16.1 GET_ACTIVE_SOCKET

GET_ACTIVE_SOCKET – CRAFT Statement

Get current active sockets, the maximum sites is for 32.

4.16.1.1 Format

Parameter list:

```
Active_site = GET_ACTIVE_SOCKET ( ) ;
```

Parameter comment:

Active_site : return current active sites

Bit wise, bit0 --> Site_0

bit1 --> Site_1

…

Bit31 --> Site_31

Type : unsigned int

4.16.1.2 Example

```
unsigned int   Active_site;
Active_site = GET_ACTIVE_SOCKET ( );
Printf("current active sockets = %x \n", Active_site );
```

4.16.1.3 Relative

GET_ACTIVE_SOCKET_EX

4.16.2 DISABLE_SOCKET

This statement isolates the socket with the No. which is specified.

4.16.2.1 Format

Parameter list:

```
DISABLE_SOCKET (socket_no[ | socket_no ]);
```

Parameter comment:

socket_no:

S0: socket No. 0
S1: socket No. 1
S2: socket No. 2
S3: socket No. 3
Type: integer
No entry: illegal

B: Site_Var : variable, Bit wise,
bit0 --> Site_0
bit1 --> Site_1
…
bit1023 --> Site_1023
Type : unsigned int

**no_entry: illegal

4.16.2.2 Example

```
Example 1:
// disable Site 0 and Site 1

DISABLE_SOCKET (S0 | S1) ;
DISABLE_SOCKET (0x3);

//==========================================================//

Example 2:

unsigned int Active_site;
Active_site = GET_ACTIVE_SOCKET ( ) ;
DISABLE_SOCKET(B:~Active_site);
```

4.16.3 ENABLE_SOCKET

This statement connects the socket with the No. which is specified.

4.16.3.1 Format

Parameter list:

```
ENABLE_SOCKET (socket_no[ | socket_no ]);
```

Parameter comment:

socket_no:

S0: socket No. 0
S1: socket No. 1
S2: socket No. 2
S3: socket No. 3
Type: integer
No entry: illegal

B:Site_Var : variable, Bit wise,
bit0 --> Site_0
bit1 --> Site_1
…
bit1023 --> Site_1023
Type : unsigned int
**no_entry: illegal

4.16.3.2 Example

```
//Example 1:

// Enable Site 0 and Site 1

DISABLE_SOCKET (S0 | S1) ;
DISABLE_SOCKET (0x3);

//==========================================================//

//Example 2:

unsigned int Active_site;
Active_site=0x3;
ENABLE_SOCKET(B:Active_site);
```

4.16.4 SOCKET_SET

This function is used to set data or test individually for one socket among multiple sockets tested in parallel mode.

4.16.4.1 Format

Parameter list:

```
SOCKET_SET    ( socket_no [ ,FRZ_ON / FRZ_OFF ] )   {
      Statement;
      …
}
```

Parameter comment:

socket_no :

JUDGE_RESULT_PASS : previous JUDGE_RESULT_MS[] pass & active_sites
JUDGE_RESULT_FAIL : previous JUDGE_RESULT_MS[] fail & active_sites

S0 : socket No. 0
S1 : socket No. 1
…
S1023 : socket No. 1023
Type : integer
**no_entry: illegal

B:Site_Var : variable, Bit wise,
bit0 --> Site_0
bit1 --> Site_1
…
bit1023 --> Site_1023
Type : unsigned int
**no_entry: illegal

FRZ_ON / FRZ_OFF : FRZ_ON → CRAFT will do "freeze non-active sites" in the socket-loop.
FRZ_OFF → CRAFT do nothing, just socket-loop
**no entry: default FRZ_ON

statement: Any valid tester or C statement. (can't use “DISABLE_SOCKET”, “DISABLE_FAIL_SOCKET”)

4.16.4.2 Explanation

[Notice]: If device keeps CONT-mode (continue mode), only allow “FRZ_OFF”. Otherwise CRAFT will pop-up error message (may induce pseudo-pass).

4.16.4.3 Example

Example 1 :

```
RATEN(TS1, 500ns);
MARKERN(TS1, data, 20ns,40ns, , , ,100ns,) ; // 100 ns is set to ct0 for both sites //

SOCKET_SET (S1, FRZ_OFF) //set site 1
  {
    MARKERN (TS1, data, , , , , ,120ns, ); //120 ns is set to ct0 for data pins of socket
    no.1 //
  }

JUDGE_PAT(fun1,__fun1);        /* The test will be executed with ct1 of 100 ns for data
                                pins of socket no.0 and ct0 of 120 ns for the data pins of socket
                                no.1. */
```

Example 2 :// use bitwise

```
RATEN(TS1, 500ns);
MARKERN(TS1, data, 20ns,40ns, , , ,100ns,) ; // 100 ns is set to ct0 for both sites //

SOCKET_SET (0x2, FRZ_OFF) //set site 1
  {
    MARKERN (TS1, data, , , , , ,120ns, ); //120 ns is set to ct0 for data pins of socket
    no.1 //
  }

JUDGE_PAT(fun1,__fun1);        /* The test will be executed with ct1 of 100 ns for data
                                pins of socket no.0 and ct0 of 120 ns for the data pins of socket
                                no.1. */
```

4.16.5 SOCKET_INC

This is a program flow control statement.

4.16.5.1 Format

Parameter list:

```
SOCKET_INC   [( FRZ_ON | FRZ_OFF [ , FRZ_SOCKET_DISABLE ] )] {
            statement;
            .
            .
            statement;     }
```

Parameter comment:

FRZ_ON / FRZ_OFF : FRZ_ON → CRAFT will do "freeze non-active sites" in the socket-loop.
FRZ_OFF → CRAFT do nothing, just socket-loop
no entry: default FRZ_ON

FRZ_SOCKET_ENABLE / FRZ_SOCKET_DISABLE :

FRZ_SOCKET_ENABLE → CRAFT will do "enable non-active sites in HW register" in the socket-loop.

FRZ_SOCKET_DISABLE → CRAFT will do "disable non-active sites in HW register" in the socket-loop. Release version : **CRAFT_3380_1.40**
**no entry: default FRZ_SOCKET_ENABLE

statement: Any valid tester or C statement. (can't use "DISABLE_SOCKET", "DISABLE_FAIL_SOCKET")

4.16.5.2 Explanation

This is a program flow control statement. It provides a compact test loop based on the reserved word SOCKET as the loop increment.

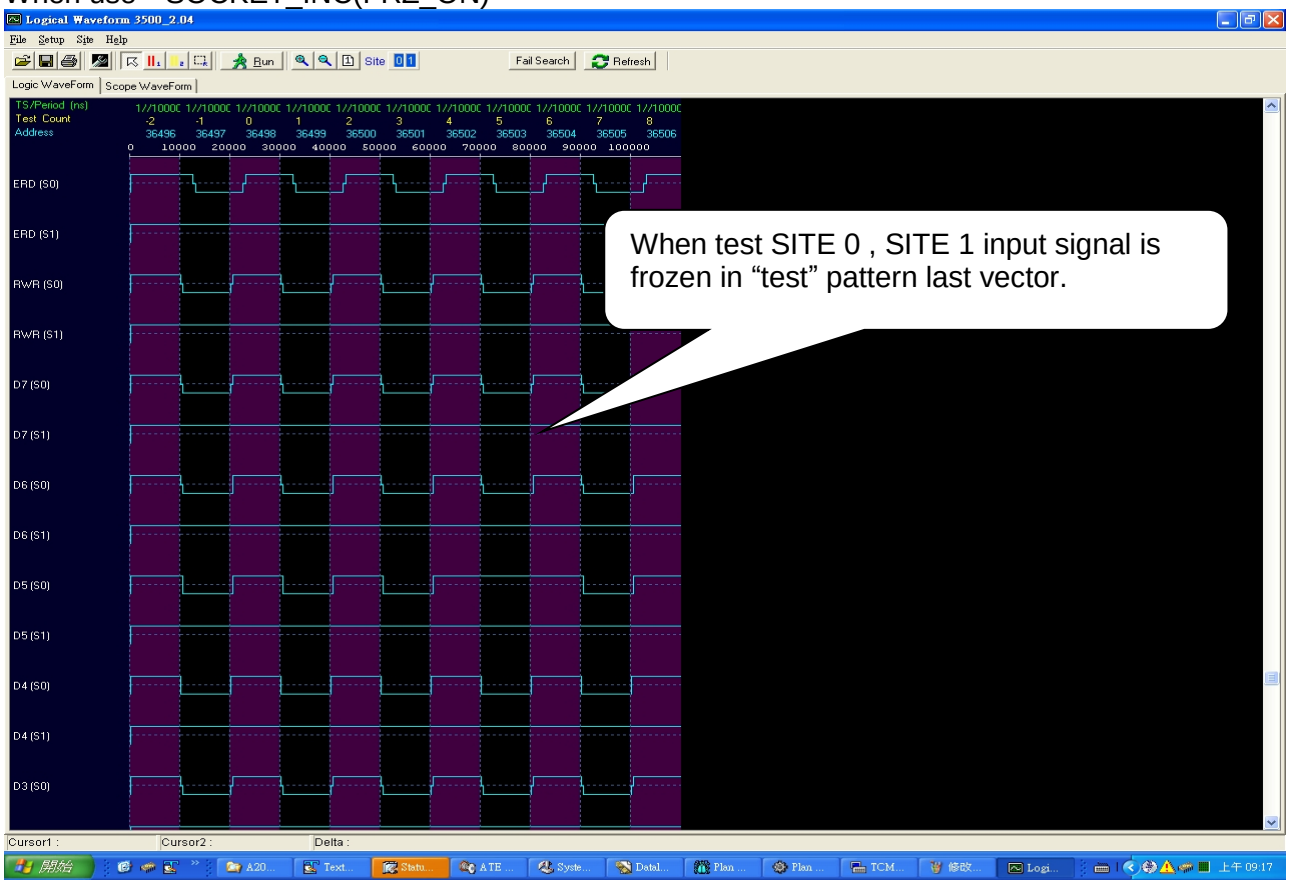
[Notice]: If device keeps CONT-mode (continue mode), only allow "FRZ_OFF". Otherwise CRAFT will pop-up error message (may induce pseudo-pass).

4.16.5.3 Example

```
JUDGE_PAT (test,__test);
MEAS_V_PREF(V0,IFVM,2mS, 16, AVE, 50uS);
```

```
SOCKET_INC(FRZ_ON)
{
   v0[1][SOCKET] = RESULT_PREF(SOCKET, V0);
   DATALOG_MSG = "VR_V0_1";
   TEST_NO(2301);
   JUDGE_VARIABLE(SOCKET, v0[1][SOCKET], V0_1_MIN, V0_1_MAX,"");
   JUDGE_PAT(test2,__test2);
   MEAS_V_PREF(V0,IFVM,2mS, 16, AVE, 50uS);
   v0[2][SOCKET] =RESULT_PREF(SOCKET, V0);
   JUDGE_VARIABLE(SOCKET, v0[2][SOCKET], V0_S_MIN,V0_S_MAX,"");
}
```

When use SOCKET_INC(FRZ_ON)




```
JUDGE_PAT (test,__test);
MEAS_V_PREF(V0,IFVM,2mS, 16, AVE, 50uS);
SOCKET_INC(FRZ_OFF)
{
   v0[1][SOCKET] = RESULT_PREF(SOCKET, V0);
   DATALOG_MSG = "VR_V0_1";
   TEST_NO(2301);
   JUDGE_VARIABLE(SOCKET, v0[1][SOCKET], V0_1_MIN, V0_1_MAX,"");
    JUDGE_PAT(test2,__test2);
   MEAS_V_PREF(V0,IFVM,2mS, 16, AVE, 50uS);
   v0[2][SOCKET] =RESULT_PREF(SOCKET, V0);
   JUDGE_VARIABLE(SOCKET, v0[2][SOCKET], V0_S_MIN,V0_S_MAX,"");
}
```

When use SOCKET_INC(FRZ_OFF)

4.16.6 POWER_DOWN_FAIL_SITE

It provides a power down control for Multi site test.

4.16.6.1 Format

Parameter list:

```
POWER_DOWN_FAIL_SITE([ mode ] )
{
 ……..     //cout<<endl<<"You want do power down action."<<endl;
}
```

Parameter comment:

Mode :

MODE_A: NO Entry (Default)

If fail site power channel were duplicated to declare for other site, and the other site is "PASS".
POWER_DOWN_FAIL_SITE" statement will not be executed "

MODE_B:

If fail site power channel were duplicated to declare for other site, and the other site is "PASS".
POWER_DOWN_FAIL_SITE procedures will always be executed except Power statement.

Notice: When these sites (use the same power) fail, all of POWER_DOWN_FAIL_SITE procedures will be executed.

4.16.6.2 Explanation

It is usually called at test item end of multi-site test. Any valid statements can be used in the body of this function. This function is then followed by a default power down function (including level, DPS & PREF power, and relay off).

4.16.6.3 Example

```
PIN_LIST    ( LS299_4_Sites_DUT_BOARD )//Pin list name max.=32 characters.
{
  SEL0      = 150 :  50 : 322 : 194  =  1   = IN   ;
  OE1       = 148 :  48 : 320 : 196  =  2   = IN   ;
  OE2       = 146 :  46 : 318 : 198  =  3   = IN   ;
  OA        = 136 :  36 : 308 : 208  =  8   = OUT  ;
  CLR       = 134 :  34 : 306 : 210  =  9   = IN   ;
  SR        = 133 :  33 : 305 : 213  =  11  = IN   ;
  CLK       = 135 :  35 : 307 : 211  =  12  = IN   ;
  OH        = 145 :  45 : 317 : 201  =  17  = OUT  ;
  SL        = 147 :  47 : 319 : 199  =  18  = IN   ;
  SEL1      = 149 :  49 : 321 : 197  =  19  = IN   ;
  VDD       =   0  : 0   :   1 :   1  =  20  = DPS ;
}
```

DPS channel were duplicated to declare

```
//multi site (4 site)
POWER_DOWN_FAIL_SITE()
{
        SET_LEVELN(rst_lvl,ALL_PINS_IO,0V,0.5V,0V,0V,0uA,0uA,0V,-3V,8V);
        FORCE_V_DPS(VDD,0V,@8V,@1A,800mA,NORM ,OFF,1mS);
        RELAY_OFF(ALL_PINS_IO,ALL,1mS);
}

Example_item()
{
  POWER_DOWN_FAIL_SITE(MODE_A);

  POWER_DOWN_FAIL_SITE(MODE_B);
}
```

If S0 is PASS and S1 FAIL POWER_DOWN_FAIL_SITE will not be executed.

If S0 is PASS and S1 FAIL POWER_DOWN_FAIL_SITE will still be executed except Power statement

4.16.7 PLAN_RESULT_MS

User can use it to read back Plan result (PASS or FAIL) in any time.
And use result to check something that you want.
PLAN_RESULT_MS is default "PASS" until Fail is happened. It will always be set "FAIL".

4.16.7.1 Format

Parameter list:

```
PLAN_RESULT_MS[SOCKET_NUMBER]
```

Parameter comment:
SOCKET_NUMBER: use "S0" , "S1" , "S2" ," S3" ……. or "0", "1", "2", "3" ……

4.16.7.2 Example

```
FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
RELAY_ON( IO_ALLPINS, PDCL, 3mS);
LEVELS(func_lvl, 3mS);
PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE(          CLK,   RZ, EDGE,     MASK, IO_NRZ, OFF);
JUDGE_PAT(TEST_1, __TEST_1);

If(PLAN_RESULT_MS[S0]==FAIL)
{
      JUDGE_PAT(retest_1, __ retest_1);
}
If(JUDGE_RESULT_MS[S0]==PASS)   PLAN_RESULT_MS[S0]=PASS;
```

4.16.8 JUDGE_RESULT_MS

User can use it to read back JUDGE result (PASS or FAIL) .
And use result to check something that you want.
JUDGE_RESULT_MS is default "PASS" until JUDGE happen "fail". It will be set "FAIL".
But "Non-active site" will always be set "PASS".

4.16.8.1 Format

Parameter list:

```
JUDGE_RESULT_MS[SOCKET_NUMBER]
```

Parameter comment:
SOCKET_NUMBER: use "S0" , "S1" , "S2" ," S3" ……. or "0", "1", "2", "3" ……

4.16.8.2 Example

```
FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
RELAY_ON( IO_ALLPINS, PDCL, 3mS);
LEVELS(func_lvl, 3mS);
PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE(          CLK,   RZ, EDGE,     MASK, IO_NRZ, OFF);
JUDGE_PAT(TEST_1, __TEST_1);

If(PLAN_RESULT_MS[S0]==FAIL)
{
      JUDGE_PAT(retest_1, __ retest_1);
}
If(JUDGE_RESULT_MS[S0]==PASS)   PLAN_RESULT_MS[S0]=PASS;
```

4.16.9 BACKUP_PLAN_RESULT

The statement backup the program plan result that are used in following statement "FAIL_REJUDGE".
[Notice]: The following "FAIL_REJUDGE" must in the same function code, otherwise the backup plan result are useless

4.16.9.1 Format

Parameter list:

```
BACKUP_PLAN_RESULT( ) ;
```

Parameter comment:
None.

4.16.9.2 Example

```
FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
RELAY_ON( IO_ALLPINS, PDCL, 3mS);
LEVELS(func_lvl, 3mS);
PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE(          CLK,   RZ, EDGE,     MASK, IO_NRZ, OFF);

BACKUP_PLAN_RESULT();

JUDGE_PAT(TEST_1, __TEST_1);

FAIL_REJUDGE(3)
{
      JUDGE_PAT(retest_1, __ retest_1);
}
```

4.16.10 FAIL_REJUDGE

This is a program flow control statement. It provides a compact re-test loop that if the "PLAN" result is fail.

(It provides a compact test loop based on the reserved word **RJ_CNT** as the loop increment) (Notice : when there are more than one judge statements, only the last judge statement is usefull ; that means only last judge result will be **u**pdated to plan result) .

4.16.10.1 Format

Parameter list:

```
FAIL_REJUDGE( [ max_cnt ] [ ,FRZ_ON / FRZ_OFF ])   {
        statement;
        .
        .
        statement;
}
```

Parameter comment:

max_cnt : fail retest maximum count
Type : integer
**no_entry: Default = 1

FRZ_ON / FRZ_OFF : FRZ_ON → CRAFT will do "freeze non-active sites" in the socket-loop.
(also **Ignore the board Fail**, means always Pass)

FRZ_OFF → CRAFT do nothing, just socket-loop
(restore the board fail setting)
**no entry: default FRZ_ON

statement: Any valid tester or C statement (can't use statements : SOCKET_SET , SOCKET_INC)

4.16.10.2 Explanation

[Notice]: If CONT-mode, only allow "FRZ_OFF"; otherwise CRAFT will pop-up error message (may induce pseudo-pass).

4.16.10.3 Example

```
FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
RELAY_ON( IO_ALLPINS, PDCL, 3mS);
LEVELS(func_lvl, 3mS);
PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE(           CLK,    RZ, EDGE,     MASK, IO_NRZ, OFF);

BACKUP_PLAN_RESULT();

JUDGE_PAT(TEST_1, __TEST_1);

FAIL_REJUDGE(3)
{
      JUDGE_PAT(retest_1, __ retest_1);
}
```

4.16.11 ENABLE_SOCKET_EX

This statement connect the socket with the No. specified.specially support > 32 Sites condition.

4.16.11.1 Format

Parameter list:

```
ENABLE_SOCKET_EX(socket_no [ | socket_no ]);
```

Parameter comment:

socket_no :

JUDGE_RESULT_PASS : previous JUDGE_RESULT_MS[] pass & active_sites
JUDGE_RESULT_FAIL : previous JUDGE_RESULT_MS[] fail & active_sites
S0 : socket No. 0 (& active_sites)
S1 : socket No. 1
…
S31 : socket No. 31

B: site_var : variable, site number array
Type : **LUINT**
site_var[] : 0 --> none active
1 --> active
**no_entry: illegal

4.16.11.2 Example

```
// typedef   bitset<DF_MAX_Site_Count>   LUINT;    //defined in   cpin.h
LUINT   enable_site;
  int   i;
      for(i=0; i<64; i++)   enable_site[i]=0;   //init
      enable_site[0]=1;
      enable_site[63]=1;
      ENABLE_SOCKET_EX( B: enable_site ) ;   // enable site0 & site63
```

4.16.12 DISABLE_SOCKET_EX

This statement connect the socket with the No. specified.specially support > 32 Sites condition.

4.16.12.1 Format

Parameter list:

```
DISABLE_SOCKET_EX(socket_no [ | socket_no ]);
```

Parameter comment:

socket_no :

JUDGE_RESULT_PASS : previous JUDGE_RESULT_MS[] pass & active_sites
JUDGE_RESULT_FAIL : previous JUDGE_RESULT_MS[] fail & active_sites
S0 : socket No. 0 (& active_sites)
S1 : socket No. 1
…
S31 : socket No. 31

B: site_var : variable, site number array
Type : **LUINT**
site_var[] : 0 --> none active
1 --> active
**no_entry: illegal

4.16.12.2 Example

```
// typedef   bitset<DF_MAX_Site_Count>   LUINT;    //defined in   cpin.h
LUINT   disable_site;
  int   i;
     for(i=0; i<64; i++)   disable_site[i]=0;   //init
     disable_site[0]=1;
     disable_site[63]=1;
     DISABLE_SOCKET_EX( B: disable_site ) ;   // enable site0 & site63
```

4.16.13 SOCKET_SET_EX

This function is used to set data or test individually for sockets among multiple sockets tested in parallel. specially support > 32 Sites condition.

4.16.13.1 Format

Parameter list:

```
//only when (socket_no & active_sites)== true, then execution
SOCKET_SET_EX(socket_no[,FRZ_ON/FRZ_OFF])
{
        Statement ;
        …
}
```

Parameter comment:

socket_no :

JUDGE_RESULT_PASS : previous JUDGE_RESULT_MS[] pass & active_sites

JUDGE_RESULT_FAIL : previous JUDGE_RESULT_MS[] fail & active_sites

S0 : socket No. 0 (& active_sites)

S1 : socket No. 1

…

S31 : socket No. 31

B: site_var : variable, site number array

Type : **LUINT**

site_var[] : 0 --> none active

1 --> active

**no_entry: illegal

FRZ_ON / FRZ_OFF : FRZ_ON → CRAFT will do "freeze non-active sites" in the socket-loop.

(also **Ignore the board Fail**, means always Pass)

FRZ_OFF → CRAFT do nothing, just socket-loop

(restore the board fail setting)

**no entry: default FRZ_ON

statement: Any valid tester or C statement.

(Can't use "DISABLE_SOCKET", "DISABLE_FAIL_SOCKET")

4.16.13.2 Explanation

[Notice]: If device keeps CONT-mode (continue mode), only allow “FRZ_OFF”. Otherwise CRAFT will pop-up error message (may induce pseudo-pass).

4.16.13.3 Example

```
            RATEN(TS1, 500ns);
            MARKERN(TS1, data, 20ns,40ns, , , ,100ns,) ;
                                    // 100 ns is set to ct0 for both sites //
                                    // respective socket no.0 and no.1. //

            SOCKET_SET_EX ( S1,FRZ_OFF )   {     // Points socket no.1 */
            MARKERN(TS1, data, , , , , ,120ns, ); // 120 ns is set to ct0 for data pins of socket
no.1
        }

JUDGE_PAT( , , , , , );  /* The test will be executed with ct1 of 100 ns for data pins of socket
                            no.0 and ct0 of 120 ns for the data pins of socket no.1. */

   LUINT   site_no;
            for(i=0; i<64; i++)   site_no[i]= 0;
            site_no[63]= 1;
            SOCKET_SET_EX ( B:site_no)   //only enable site63
      {
         JUDGE_PAT(….);
      }
```

4.16.14 GET_ACTIVE_SOCKET_EX

Get current active sockets, specially support > 32 Sites condition.

4.16.14.1 Format

Parameter list:

```
LUINT Active_site =GET_ACTIVE_SOCKET_EX();
```

Parameter comment:

Active_site[] : return current active sites
0 --> none active
1 --> active

4.16.14.2 Example

```
LUINT   Active_site;
Active_site = GET_ACTIVE_SOCKET_EX ( ) ;
printf("Site0 = %d \n", Active_site[0]);     // if Active_site[0]== 0 --> means site0 none active
                                             // if Active_site[0]== 1 --> means site0 active
```

4.16.14.3 Relative

GET_ACTIVE_SOCKET

4.16.15 POWER_DOWN_FAIL_SITE_X

It provides a power down control for Multi site test.
When all active sites are pass, software will control the subroutine of POWER_DOWN_FAIL_SITE_X that don't have any site get in. POWER_DOWN_FAIL_SITE is controlled with hardware that will cause much testing time.
Release version : CRAFT_3380_2.10

4.16.15.1 Format

Parameter list:

```
POWER_DOWN_FAIL_SITE_X([ mode ] )
{

 ……..     //cout<<endl<<"You want do power down action."<<endl;

}
```

Parameter comment:

MODE_A: NO Entry (Default)

Multi-site use different power channel that fail site will be truned off with the subroutine.

If fail site power channel were duplicated to declare for other site, and the other site is "PASS". POWER_DOWN_FAIL_SITE of statement will not be executed.

MODE_B:

If fail site power channel were duplicated to declare for other site, and the other site is "PASS".

POWER_DOWN_FAIL_SITE procedures will always be executed except Power statement.

[Notice]: When these sites (use the same power) fail, all of **POWER_DOWN_FAIL_SITE_X** procedures will be executed.

4.16.15.2 Explanation

It is usually called at test item end of multi-site test. Any valid statements can be used in the body of this function. This function is then followed by a default power down function (including level, DPS & PREF power, and relay off).

4.16.15.3 Example

```
PIN_LIST    ( LS299_4_Sites_DUT_BOARD )//Pin list name max.=32 characters.
{
  SEL0      = 150 :  50 : 322 : 194  =  1   = IN   ;
  OE1       = 148 :  48 : 320 : 196  =  2   = IN   ;
  OE2       = 146 :  46 : 318 : 198  =  3   = IN   ;
  OA        = 136 :  36 : 308 : 208  =  8   = OUT  ;
  CLR       = 134 :  34 : 306 : 210  =  9   = IN   ;
  SR        = 133 :  33 : 305 : 213  =  11  = IN   ;
  CLK       = 135 :  35 : 307 : 211  =  12  = IN   ;
  OH        = 145 :  45 : 317 : 201  =  17  = OUT  ;
  SL        = 147 :  47 : 319 : 199  =  18  = IN   ;
  SEL1      = 149 :  49 : 321 : 197  =  19  = IN   ;
  VDD       =   0  : 0   :   1 :   1  =  20  = DPS  ;
}
```

DPS channel were duplicated to declare for other site.

```
//multi site (4 site)
POWER_DOWN_FAIL_SITE_X ()
{
        SET_LEVELN(rst_lvl,ALL_PINS_IO,0V,0.5V,0V,0V,0uA,0uA,0V,-3V,8V);
        FORCE_V_DPS(VDD,0V,@8V,@1A,800mA,NORM ,OFF,1mS);
        RELAY_OFF(ALL_PINS_IO,ALL,1mS);
}


Example_item()

{

    POWER_DOWN_FAIL_SITE_X(MODE_A);

    POWER_DOWN_FAIL_SITE_X(MODE_B);

}
```

If S0 is PASS and S1 FAIL POWER_DOWN_FAIL_SITE will not be executed.

If S0 is PASS and S1 FAIL POWER_DOWN_FAIL_SITE will still be executed except Power statement

4.17 Measure Result Getting

4.17.1 RESULT_PIN_MS

The statement extracts the result(measured value)after the MEAS_TIME/ MEAS_FREQ /MEAS_I_PMU/MEAS_V_PMU/ MEAS_I_PPMU/MEAS_V_PPMU statement executed.

4.17.1.1 Format

Parameter list:

```
RESULT_PIN_MS(pin_name, value_ms ) ;
```

Parameter comment:

pin_name: pin_def
**no_entry: illegal

value_ms: Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value
Type: array

4.17.1.2 Explanation

The statement extracts the result (the measured value) after the MEAS_TIME/ MEAS_FREQ /MEAS_I_PMU/MEAS_V_PMU/ MEAS_I_PPMU/MEAS_V_PPMU statement is executed.

4.17.1.3 Example

```
TEST_NO(80);
JUDGE_PAT(measure1_ft:clr_st, measure1_ft:m_ft_cpu_sp);
MEAS_TIME(OH, CYCLE, , , ,1mS);//Measure Time

double   value_ms[4];   //suppose 4-site
double   high_limit, low_limit;

RESULT_PIN_MS(OH, value_ms );
JUDGE_VARIABLE_MS(value_ms, high_limit, low_limit, "OH test");
```

4.17.2 MIN_RESULT_PIN_MS

The statement extracts the result(measured value)after the MEAS_TIME/ MEAS_FREQ /MEAS_I_PMU/MEAS_V_PMU/ MEAS_I_PPMU/MEAS_V_PPMU statement executed.

4.17.2.1 Format

Parameter list:

```
MIN_RESULT_PIN_MS(pin_name, value_ms ) ;
```

Parameter comment:

pin_name: pin_def
**no_entry: illegal

value_ms: Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value
Type: array

4.17.2.2 Example

```
TEST_NO(80);
JUDGE_PAT(measure1_ft:clr_st, measure1_ft:m_ft_cpu_sp);
MEAS_TIME(OH, CYCLE, , , ,1mS);//Measure Time

double   value_ms[4];   //suppose 4-site
double   high_limit, low_limit;

MIN_RESULT_PIN_MS(OH, value_ms );
JUDGE_VARIABLE_MS(value_ms, high_limit, low_limit, "OH test");
```

4.17.2.3 Relative

MEAS_V_DPS , MEAS_I_DPS , JUDGE_VARIABLE

4.17.3 MAX_RESULT_PIN_MS

The statement extracts the result(measured value)after the MEAS_TIME/ MEAS_FREQ /MEAS_I_PMU/MEAS_V_PMU/ MEAS_I_PPMU/MEAS_V_PPMU statement executed.

4.17.3.1 Format

Parameter list:

```
MAX_RESULT_PIN_MS(pin_name, value_ms ) ;
```

Parameter comment:

pin_name: pin_def
**no_entry: illegal

value_ms: Specifies the array to store the measuring results. The description is given in the form of the current type or the voltage, return the measured value
Type: array

4.17.3.2 Example

```
TEST_NO(80);
JUDGE_PAT(measure1_ft:clr_st, measure1_ft:m_ft_cpu_sp);
MEAS_TIME(OH, CYCLE, , , ,1mS);//Measure Time

double   value_ms[4];   //suppose 4-site
double   high_limit, low_limit;

MAX_RESULT_PIN_MS(OH, value_ms );
JUDGE_VARIABLE_MS(value_ms, high_limit, low_limit, "OH test");
```

4.17.3.3 Relative

MEAS_V_DPS , MEAS_I_DPS , JUDGE_VARIABLE

4.17.4 RESULT_PIN

This statement extracts the result of measure statement execution.

4.17.4.1 Format

Parameter list:

```
Value = RESULT_PIN (socket_no, pin_name ) ;
```

Parameter comment:

Value: Specifies the variable to store the measuring results. The description is given in the form of CURRENT or VOLTAGE or TIME or FREQUENCY or RESISTANCE
type variable.

socket_no: Specifies the socket can be selected.
S0: socket No. 0
S1: socket No. 1
S2: socket No. 2
S3: socket No. 3
Type: integer
No entry: illegal

pin_name: Refer to the device pin No. enter and specify only one pin No. or pin name for the device. The pin group cannot be used.
No entry: illegal

4.17.4.2 Explanation

The statement extracts the result (the measured value) after the MEAS_OSC_TIME/ MEAS_OSC_FREQ /MEAS_I_PMU/MEAS_V_PMU statement is executed. Support PPMU measure.

4.17.4.3 Example

```
// TIME measure_time[4];
TEST_NO(80);
JUDGE_PAT(measure1_ft:clr_st, measure1_ft:m_ft_cpu_sp);
MEAS_TIME(OH, CYCLE, , , ,1mS);//Measure Time
SOCKET_INC(FRZ_OFF)
{
        measure_time[SOCKET]=RESULT_PIN(SOCKET,OH);
        JUDGE_VARIABLE ( SOCKET, measure_time[SOCKET], 1.9uS, 2.1uS, "Measure
time" );
  }
```

4.17.5 READ_PIN_INC

This statement is a program flow control statement, especially for reading the measured result.

4.17.5.1 Format

Parameter list:

```
READ_PIN_INC (pin_name)   {

        Value = RESULT_PIN (socket_no, PIN_DEF) ;

        Value = RESULT_DPS (socket_no, PIN_DEF) ;

        Value = RESULT_PREF (socket_no,Pin_DEF) ;

        .statement

}
```

Parameter comment:

pin_name: Pin_def and pin_group can be used.
No entry: illegal

socket_no: Specifies the socket can be selected.
S0: socket No. 0
S1: socket No. 1
S2: socket No. 2
S3: socket No. 3
Type: integer
No entry: illegal

statement: Any valid C statement.

4.17.5.2 Explanation

This is a program flow control statement, especially for reading measured result. It provides a compact test loop based on the reserved word PIN_DEF as the pin loop increment, and the reserved word SOCKET as the site loop increment. PIN_DEF is used to represent each pin in the loop, and SOCKET is used to represent each socket in the socket loop, one at a time.

4.17.5.3 Example

```
TEST_NO(14);
PIN_INC(INPUTS) //INPUTS=A_PIN+B_PIN;
{
	FORCE_V_PMU (PMU, 5.5V, @6V, @10uA, 5uA, ON, 1mS);
	RELAY_ON( PIN_DEF, PMU, 1mS);
	MEAS_I_PMU(PMU, 2mS, 16, AVE, 50uS );
	RELAY_OFF( PIN_DEF, PMU, 3mS);
	FORCE_V_PMU (PMU, 5.5V, @6V, @10uA, 5uA, OFF, 1mS);
}


READ_PIN_INC (inputs) //read each socket measured result

{

	RESULT_PIN (SOCKET, PIN_DEF);

	printf ("MEAS_PMU value (Socket_%d) : %f\n", SOCKET, value);

	JUDGE_VARIABLE (SOCKET, meas_value, -1uA, 1uA, "input_leakage");
}
```

4.17.6 MAX_RESULT_PIN

The statement extracts the result (the measured value) after the MEAS_OSC_TIME/ MEAS_OSC_FREQ /MEAS_I_PMU/MEAS_V_PMU statement is executed.

4.17.6.1 Format

Parameter list:

```
Value = MAX_RESULT_PIN (socket_no, pin_name) ;
```

Parameter comment:
Value: Specifies the variable to store the measuring results. The description is given in
the form of current type or voltage type variable.
socket_no: S0 : socket No. 0
S1: socket No. 1
S2: socket No. 2
S3: socket No. 3
Type: integer
No entry: illegal

pin_name: pin-def
Variables & constants are allowed.
Type: integer
No entry: illegal

4.17.6.2 Explanation

The statement extracts the result (the measured value) after the MEAS_OSC_TIME/ MEAS_OSC_FREQ /MEAS_I_PMU/MEAS_V_PMU statement is executed.

4.17.6.3 Example

```
TEST_NO(14);
PIN_INC(INPUTS) //INPUTS=A_PIN+B_PIN;
{
	FORCE_V_PMU (PMU, 5.5V, @6V, @10uA, 5uA, ON, 1mS);
	RELAY_ON( PIN_DEF, PMU, 1mS);
	MEAS_I_PMU(PMU, 2mS, 16, AVE, 50uS );
	RELAY_OFF( PIN_DEF, PMU, 3mS);
	FORCE_V_PMU (PMU, 5.5V, @6V, @10uA, 5uA, OFF, 1mS);
}
// CURRENT meas_value_A[4] , meas_value_B[4] ;
SOCKET_INC(FRZ_OFF)
{
	meas_value_A[SOCKET]= MAX_RESULT_PIN(SOCKET, A_PIN);
	meas_value_B[SOCKET]= MAX_RESULT_PIN(SOCKET, B_PIN);
	JUDGE_VARIABLE (SOCKET, meas_value_A[SOCKET], -1uA, 1uA, "Vee_current");
	JUDGE_VARIABLE (SOCKET, meas_value_B[SOCKET], -1uA, 1uA, "Vee_current");
}
```

4.17.7 MIN_RESULT_PIN

The statement extracts the minimum result (the measured value) after the MEAS_OSC_TIME /MEAS_OSC_FREQ /MEAS_I_PMU/MEAS_V_PMU statement is executed.

4.17.7.1 Format

Parameter list:

```
Value = MIN_RESULT_PIN (socket_no, pin_name) ;
```

Parameter comment:

Value: Specifies the variable to store the measuring results. The descriptionis given in the form of current type or voltage type variable.

socket_no: S0: socket No. 0
S1: socket No. 1
S2: socket No. 2
S3: socket No. 3
Type: integer
No entry: illegal

pin_name: pin-def
Variables & constants are allowed.
Type: integer
No entry: illegal

4.17.7.2 Example

```
TEST_NO(14);
PIN_INC(INPUTS) //INPUTS=A_PIN+B_PIN;
{
        FORCE_V_PMU (PMU, 5.5V, @6V, @10uA, 5uA, ON, 1mS);
        RELAY_ON( PIN_DEF, PMU, 1mS);
        MEAS_I_PMU(PMU, 2mS, 16, AVE, 50uS );
        RELAY_OFF( PIN_DEF, PMU, 3mS);
        FORCE_V_PMU (PMU, 5.5V, @6V, @10uA, 5uA, OFF, 1mS);
}
// CURRENT meas_value_A[4] , meas_value_B[4] ;
SOCKET_INC(FRZ_OFF)
{
        meas_value_A[SOCKET]= MIN_RESULT_PIN(SOCKET, A_PIN);
        meas_value_B[SOCKET]= MIN_RESULT_PIN(SOCKET, B_PIN);
        JUDGE_VARIABLE (SOCKET, meas_value_A[SOCKET], -1uA, 1uA, "Vee_current");
        JUDGE_VARIABLE (SOCKET, meas_value_B[SOCKET], -1uA, 1uA, "Vee_current");
}
```

4.17.8 JUDGE_VARIABLE

Make a Pass/fail decision by comparing the value of a program variable to the specified limits.

4.17.8.1 Format

Parameter list:

```
JUDGE_VARIABLE (socket_no, var, lower_limit, upper_limit [, "string" ]);
```

Parameter comment:

socket_no: Specifies the socket can be selected.
S0: socket No. 0
S1: socket No. 1
S2: socket No. 2
S3: socket No. 3
Type: integer
No entry: illegal

var: Variable with the value to be tested.
Type: TIME, CURRENT, VOLTAGE, FREQUENCY or double
No entry: illegal

lower _limit: Judge PASS or FAIL by comparing the data with the lower limit. Variable and constant are allowed.
Type: TIME, CURRENT, VOLTAGE, FREQUENCY or double
No entry: only high limit

upper limit: Judge PASS or FAIL by comparing the data with the upper limit. Variable and constant are allowed
Type: TIME, CURRENT, VOLTAGE, FREQUENCY or double
No entry: only low limit

string: Only used to be printed by datalog.
Maximum array of 256 characters.
No entry: do not print any string.

4.17.8.2 Explanation

Make a Pass/fail decision by comparing the value of a program variable to the specified limits.

4.17.8.3 Example

```
TEST_NO(14);
PIN_INC(INPUTS) //INPUTS=A_PIN+B_PIN;
{
	FORCE_V_PMU (PMU, 5.5V, @6V, @10uA, 5uA, ON, 1mS);
	RELAY_ON( PIN_DEF, PMU, 1mS);
	MEAS_I_PMU(PMU, 2mS, 16, AVE, 50uS );
	RELAY_OFF( PIN_DEF, PMU, 3mS);
	FORCE_V_PMU (PMU, 5.5V, @6V, @10uA, 5uA, OFF, 1mS);
}

CURRENT meas_value_A[4] , meas_value_B[4] ;

SOCKET_INC(FRZ_OFF)
{
	meas_value_A[SOCKET]= RESULT_PIN(SOCKET, A_PIN);
	meas_value_B[SOCKET]= RESULT_PIN(SOCKET, B_PIN);
	JUDGE_VARIABLE (SOCKET, meas_value_A[SOCKET], -1uA, 1uA, "Vee_current");
	JUDGE_VARIABLE (SOCKET, meas_value_B[SOCKET], -1uA, 1uA, "Vee_current");
}
```

4.17.9 JUDGE_VARIABLE_MS

Make a Pass/fail decision by comparing the value of a program variable against specified limits.

4.17.9.1 Format

Parameter list:

```
JUDGE_VARIABLE_MS(var, lower_limit, upper_limit    [, "string" ]);
```

Parameter comment:

Var : array with the value to be tested
type: TIME, CURRENT, VOLTAGE, FREQUENCY or double array
array size = test plan site-no
e.g. var[0] → Site-0 value, var[1] → Site-1 value, ...
**no_entry: illegal

lower _limit: variable and constant allowed
type: TIME, CURRENT, VOLTAGE, FREQUENCY or double
**no_entry: only high limit

upper_limit: variable and constant allowed
type: TIME, CURRENT, VOLTAGE, FREQUENCY or double
**no_entry: only low limit

string: Only used to be printed by datalog.
Maximum array of 256 characters.
**no_entry: do not print any string .

4.17.9.2 Example

```
double   vdd_value[4];   //suppose 4-site

JUDGE_VARIABLE_MS( vdd_value, 2.7V, 3.0V, "output vdd value");
```

4.17.10 RESULT_PIN_EX

The statement extracts the result(measured value)after the MEAS_TIME/ MEAS_FREQ /MEAS_I_PMU/MEAS_V_PMU statement executed.
release version : **CRAFT_3380_2.40-SP7**

4.17.10.1 Format

Parameter list:

```
Value = RESULT_PIN_EX (socket_no, pin_name ) ;
```

Parameter comment:

Value: Specifies the variable to store the measuring results. The description is given in the form of CURRENT or VOLTAGE or TIME or FREQUENCY or RESISTANCE
Type variable.

socket_no: Specifies the socket can be selected.
S0: socket No. 0
S1: socket No. 1
…..
Type: integer
No entry: illegal

pin_name: Refer to the device pin No. enter and specify only one pin No. or pin name for the device. The pin group cannot be used.
No entry: illegal

4.17.10.2 Explanation

The statement extracts the result (the measured value) after the MEAS_TIME/ MEAS_ FREQ /MEAS_I_PMU/MEAS_V_PMU statement is executed. Support PPMU measure.

4.17.10.3 Example

```
double meas_value[32], min_value[32], max_value[32];    //support 32-sites

MEAS_I_PPMU( SEL0,2mS,16, AVE,50uS);    //SEL0; Pin-Share channel
SOCKET_INC(FRZ_OFF)
{
    meas_value[SOCKET]= RESULT_PIN_EX(SOCKET,SEL0);
    JUDGE_VARIABLE (SOCKET, meas_value, -1uA, 1uA, "float_leak_test by SEL0 ");
    min_meas[SOCKET]= MIN_RESULT_PIN_EX(SOCKET,SEL0);
    JUDGE_VARIABLE (SOCKET, min_meas, -1uA, 1uA, "float_leak_test by min_meas ");
    max_meas[SOCKET]= MAX_RESULT_PIN_EX(SOCKET,SEL0);
    JUDGE_VARIABLE (SOCKET, max_meas, -1uA, 1uA, "float_leak_test by max_meas ");
}
```

4.17.11 MIN_RESULT_PIN_EX

The statement extracts the result(measured value)after the MEAS_TIME/ MEAS_FREQ /MEAS_I_PMU/MEAS_V_PMU statement executed.
release version : **CRAFT_3380_2.40-SP7**

4.17.11.1 Format

Parameter list:

```
Value = MIN_RESULT_PIN_EX (socket_no, pin_name ) ;
```

Parameter comment:

Value: Specifies the variable to store the measuring results. The description is given in the form of CURRENT or VOLTAGE or TIME or FREQUENCY or RESISTANCE
Type variable.

socket_no: Specifies the socket can be selected.
S0: socket No. 0
S1: socket No. 1
…..
Type: integer
No entry: illegal

pin_name: Refer to the device pin No. enter and specify only one pin No. or pin name for the device. The pin group cannot be used.
No entry: illegal

4.17.11.2 Explanation

The statement extracts the result (the measured value) after the MEAS_TIME/ MEAS_ FREQ /MEAS_I_PMU/MEAS_V_PMU statement is executed. Support PPMU measure.

4.17.11.3 Example

```
double meas_value[32], min_value[32], max_value[32];    //support 32-sites

MEAS_I_PPMU( SEL0,2mS,16, AVE,50uS);    //SEL0; Pin-Share channel
SOCKET_INC(FRZ_OFF)
{
    meas_value[SOCKET]= RESULT_PIN_EX(SOCKET,SEL0);
    JUDGE_VARIABLE (SOCKET, meas_value, -1uA, 1uA, "float_leak_test by SEL0 ");
    min_meas[SOCKET]= MIN_RESULT_PIN_EX(SOCKET,SEL0);
    JUDGE_VARIABLE (SOCKET, min_meas, -1uA, 1uA, "float_leak_test by min_meas ");
    max_meas[SOCKET]= MAX_RESULT_PIN_EX(SOCKET,SEL0);
    JUDGE_VARIABLE (SOCKET, max_meas, -1uA, 1uA, "float_leak_test by max_meas ");
}
```

4.17.12 MAX_RESULT_PIN_EX

The statement extracts the result(measured value)after the MEAS_TIME/ MEAS_FREQ /MEAS_I_PMU/MEAS_V_PMU statement executed.
release version : **CRAFT_3380_2.40-SP7**

4.17.12.1 Format

Parameter list:

```
Value = MAX_RESULT_PIN_EX (socket_no, pin_name ) ;
```

Parameter comment:

Value: Specifies the variable to store the measuring results. The description is given in the form of CURRENT or VOLTAGE or TIME or FREQUENCY or RESISTANCE
Type variable.

socket_no: Specifies the socket can be selected.
S0: socket No. 0
S1: socket No. 1
…..
Type: integer
No entry: illegal

pin_name: Refer to the device pin No. enter and specify only one pin No. or pin name for the device. The pin group cannot be used.
No entry: illegal

4.17.12.2 Explanation

The statement extracts the result (the measured value) after the MEAS_TIME/ MEAS_ FREQ /MEAS_I_PMU/MEAS_V_PMU statement is executed. Support PPMU measure.

4.17.12.3 Example

```
double meas_value[32], min_value[32], max_value[32];    //support 32-sites

MEAS_I_PPMU( SEL0,2mS,16, AVE,50uS);    //SEL0; Pin-Share channel
SOCKET_INC(FRZ_OFF)
{
    meas_value[SOCKET]= RESULT_PIN_EX(SOCKET,SEL0);
    JUDGE_VARIABLE (SOCKET, meas_value, -1uA, 1uA, "float_leak_test by SEL0 ");
    min_meas[SOCKET]= MIN_RESULT_PIN_EX(SOCKET,SEL0);
    JUDGE_VARIABLE (SOCKET, min_meas, -1uA, 1uA, "float_leak_test by min_meas ");
    max_meas[SOCKET]= MAX_RESULT_PIN_EX(SOCKET,SEL0);
    JUDGE_VARIABLE (SOCKET, max_meas, -1uA, 1uA, "float_leak_test by max_meas ");
}
```

4.18 Scan Function (Option)

4.18.1 SCAN PIN Declaration

Only be ate-channel %4 ==0 can be assigned as scan pin in dec file.

4.18.1.1 Example

SR ate channel: 128, 128 %4=0
OH ate channel: 152, 152 %4=0

```
PIN_LIST    ( LS299_2_Sites_scan )//Pin list name max.=32 characters.
{
//PIN NAME max.=64 characters.
//pin name = ATE channel= dut    = type
//            -->    S0 :    S1
   SEL0        = 150 :    50    =    1      = IN    ;//IN
   OE1         = 148 :    48    =    2      = IN    ;//IN
   OE2         = 146 :    46    =    3      = IN    ;//IN
   QG          = 144 :    44    =    4      = IO    ;//IO
   QE          = 142 :    42    =    5      = IO    ;//IO
   QC          = 140 :    40    =    6      = IO    ;//IO
   QA          = 138 :    38    =    7      = IO    ;//IO
   OA          = 136 :    36    =    8      = OUT ;//OUT//SCAN OUTPUT
   CLR         = 134 :    34    =    9      = IN    ;//IN

   Gnd         =          :          =    10    = GND ;//gnd pin didn't any ate channel

   SR          = 128 :    60    =    11    = IN    ;//IN//SCAN INPUT
   CLK         = 135 :    35    =    12    = IN    ;//IN
   QB          = 137 :    37    =    13    = IO    ;//IO
   QD          = 139 :    39    =    14    = IO    ;//IO
   QF          = 141 :    41    =    15    = IO    ;//IO
   QH          = 143 :    43    =    16    = IO    ;//IO
   OH          = 152 :    52    =    17    = OUT ;//OUT//SCAN OUTPUT
   SL          = 147 :    47    =    18    = IN    ;//IN
   SEL1        = 149 :    49    =    19    = IN    ;//IN

   Vdd         =      9 :    11    =    20    = PREF;
}
```

4.18.1.2 Relative

The instruction of scan function , How to debug scan function

4.18.2 Scan Pattern

4.18.2.1 Format

Parameter list:

SET_DEC_FILE "Device definition file"	Block
HEADER	Block
SCANIN_PINS	Block
SCANOUT_PINS	Block
PATTERN	Block

Parameter comment:

SET_DEC_FILE: The block defines the declaration file name and its' directory.
Only be ate-channel %4 ==0 can be assigned as scan pin in dec file.

HEADER: The block defines declaration that is used by the order of entry and display of the actual binary information, on a pin basis.

SCANIN_PINS: Pin_name :
a name of a single pin defined in the PIN_LIST declaration.
Only constant names from the PIN_LIST are allowed.
"C" type: STRING, 64 characters maximum.

SCANOUT_PINS: Pin_name :
a name of a single pin defined in the PIN_LIST declaration.
Only constant names from the PIN_LIST are allowed.
"C" type: STRING, 64 characters maximum.

PATTERN: This block defines the pattern function and format.
SCAN function only support "SPM memory"

4.18.2.2 Example

```
SET_DEC_FILE "./3360_ls299_pin.dec"

HEADER     CLR,%SEL1,SEL0,%OE1,OE2,%CLK,%SL,SR,%QA,QB,
           QC,     QD,    QE,    QF, QG,    QH,%OA,OH;

SCANIN_PINS    %SR ;
SCANOUT_PINS %OA,%OH ;

           /*C SS GG C SS ABCDEFGH */
           /*L EE 12 L LR QQQQQQQQ */
           /*R LL       K       ABCDEFGH */
           /*    01                      */
SPM_PATTERN (scan_func_pat) {

                    *0 X0 00 X XX LLLLLLLL LL* TS2;
                    *0 X0 00 X XX LLLLLLLL LL*;

                    //CLEAR Mode Check
                    *1 11 XX 1 XX 10101010 10*;//-----load
                    *1 00 00 X XX HLHLHLHL HL*;//-----hold
                    *0 X0 00 X XX LLLLLLLL LL*;//-----Clear1

                    *1 11 XX 1 XX 10101010 10*;//-----load
                    *1 00 00 X XX HLHLHLHL HL*;//-----hold
                    *0 0X 00 X XX LLLLLLLL LL*;//-----Clear2

                    *1 11 XX 1 XX 10101010 10*;//-----load
                    *1 00 00 X XX HLHLHLHL HL*;//-----hold
                    *0 11 XX X XX XXXXXXXX LL*;//-----Clear3


                    *0 X0 00 X XX LLLLLLLL LL*;//-------clear data using clear1
     scan_st::      *1 01 00 1 X1 XXXXXXXX XX*;//Scan Setup, scan hold pin setup

     <SCAN_ST>
     <SR>     *11111111 00000000 11111111 00000000 11111111 00000000
              10101010 10101010 10101010 10101010 10101010 10101010
              11111111 00000000 11111111 00000000 11111111 00000000 *TS1;
     <OA>     *HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL HHHHHHHH
LLLLLLLL
              HLHLHLHL HLHLHLHL HLHLHLHL HLHLHLHL HLHLHLHL
HLHLHLHL
              HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL HHHHHHHH
LLLLLLLL *;
     <OH>     *LLLLLLHH HHHHHHHL LLLLLLLH HHHHHHHL LLLLLLLH
HHHHHHHL
              LLLLLLLH LHLHLHLH LHLHLHLH LHLHLHLH LHLHLHLH LHLHLHLH
              LHLHLHLH HHHHHHHL LLLLLLLH HHHHHHHL LLLLLLLH
HHHHHHHL *;
     <SCAN_SP>
```

Scan pin declare include input pin and output pin

SCAN function only support SPM memory

Scan Pattern

```
    scan_sp::          *0 11 XX X XX XXXXXXXX LL*;//clear data using clear3
}
```

4.18.2.3 Relative

The instruction of scan function , How to debug scan function

4.18.3 SET_SCAN_WORK_MODE

SET_SCAN_WORK_MODE – CRAFT statement

Set SCAN pin work mode. (only SPM pattern memory can use scan function)
(only the channel which the ate-channel %4 ==0 can be assigned as scan pin)

4.18.3.1 Format

Parameter list:

```
SET_SCAN_WORK_MODE( scan_in_pin_name, scan_out_pin_name , mode );
```

Parameter comment:

scan_in_pin_name: pin_def, pin_group (only suit for IO-Pins)
**no_entry; illegal

scan_out_pin_name: pin_def, pin_group (only suit for IO-Pins)
**no_entry; illegal

mode: ON/OFF (turn on/off scan function)`

4.18.3.2 Example

```
SET_SCAN_WORK_MODE(SR, QA, ON);
SET_LEVELN(func_lvl, IO_ALLPINS, 0V, 5V, 0.5V, 2.0V, 150uA, -150uA,
2.5V, , );
LEVELS(func_lvl, 2mS);
RELAY_ON(IO_ALLPINS, PDCL, 1mS);
TEST_NO(30);
JUDGE_PAT(scan_func_pat, __scan_func_pat);    //expect Pass

SET_SCAN_WORK_MODE(SR, QA, OFF);
```

4.18.3.3 Relative

The instruction of scan function , How to debug scan function

4.19 ALPG Function (Option)

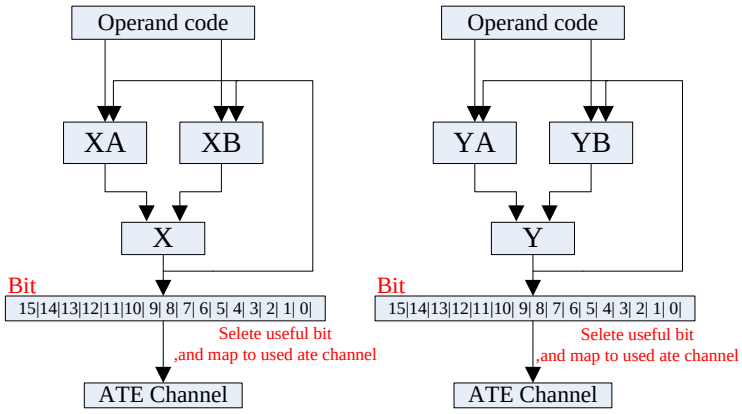
Refer to Example program: CRAFT\TestPlanExamples\ALPG_Example_Program

4.19.1 ALPG_PIN_GROUP

ALPG_PIN_GROUP – CRAFT declaration

Provides normal and advance function for the ALPG pin define for programming in declaration file(*.dec).

< The X/Y Address Generator Block Diagram >



4.19.1.1 Normal

```
<ALPG_ID_0>

ALPG_PIN_GROUP {
        ALPG_X = pin_name operand pin_name operand …;
        ALPG_Y = pin_group_name operand pin_name operand…;
        ALPG_Z = pin_group_name operand pin_group_name operand…;
}

<ALPG_ID_1>

ALPG_PIN_GROUP {
        ALPG_X1 = pin_name operand pin_name operand …;
        ALPG_Y1 = pin_group_name operand pin_name operand…;
        ALPG_Z1 = pin_group_name operand pin_group_name operand…;
}

<ALPG_ID_2>

ALPG_PIN_GROUP {
        ALPG_X2 = pin_name operand pin_name operand …;
        ALPG_Y2 = pin_group_name operand pin_name operand…;
        ALPG_Z2 = pin_group_name operand pin_group_name operand…;
```

}

……..
……..
……..

ALPG_X : ALPG X output ;
max. 16 pin
type: string

ALPG_Y : ALPG Y output ;
max. 16 pin
type: string

ALPG_Z: ALPG Z output ;
max. 16 pin
type: string

pin_name: as defined in PIN_LIST statement which the pin type of the pin_name must be "IN" or "IO".

operand: + > include pin/group in pin group being defined
– > exclude pin/group from pin group being defined

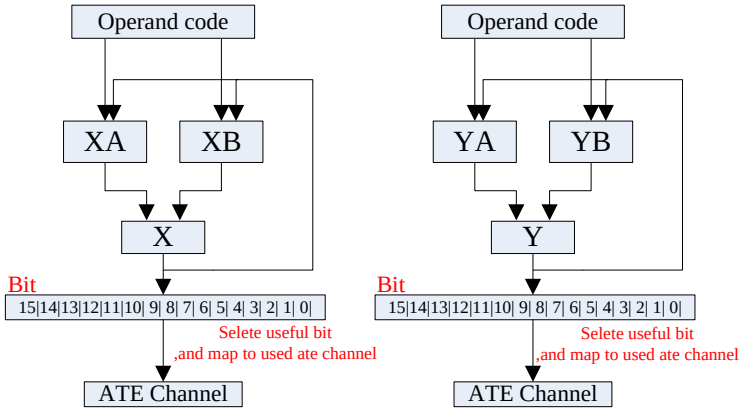
4.19.1.2 Example

```
//ALPG_ID_0
ALPG_PIN_GROUP
{
ALPG_X=A0+A1+A2+A10+A9+A11;//(LSB---->MSB)
ALPG_Y=A8+A14+A13+A12+A7+A6+A5+A4+A3;//(LSB---->MSB)
ALPG_Z=D1+D2+D3+D4+D5+D6+D7+D8;//(LSB---->MSB)
}
//ALPG_ID_1
ALPG_PIN_GROUP
{
ALPG_X1=A0 +A2+A10+A9+A11;//(LSB---->MSB)
ALPG_Y1=A8+A14+A13+A12+A7+A6+A5+A4+A3;//(LSB---->MSB)
ALPG_Z1=D1+D2+D3+D4+D5+D6+D7+D8;//(LSB---->MSB)
}
//ALPG_ID_2
ALPG_PIN_GROUP
{
ALPG_X2=A0+A1 +A10+A9+A11;//(LSB---->MSB)
ALPG_Y2=A8+A14+A13+A12+A7+A6+A5+A4+A3;//(LSB---->MSB)
ALPG_Z2=D1+D2+D3+D4+D5+D6+D7+D8;//(LSB---->MSB)
}
//ALPG_ID_3
ALPG_PIN_GROUP
{
ALPG_X3=A0+A1+A2 +A9+A11;//(LSB---->MSB)
ALPG_Y3=A8+A14+A13+A12+A7+A6+A5+A4+A3;//(LSB---->MSB)
ALPG_Z3=D1+D2+D3+D4+D5+D6+D7+D8;//(LSB---->MSB)
}
```

4.19.1.3 Advance: (Craft 1.01 SP3 or above)

Advance function will provide more varied method to define ALPG X/Y/Z bit number for used pin.

< The X/Y Address Generator Block Diagram >



<ALPG_ID_0>

```
ALPG_PIN_GROUP {
ALPG_X_BIT0          = pin_name [ | pin_name ] ;
…
ALPG_X_BIT15         = pin_name [ | pin_name ] ;
ALPG_Y_BIT0          = pin_name [ | pin_name ] ;
…
ALPG_Y_BIT15         = pin_name [ | pin_name ] ;
ALPG_Z_BIT0          = pin_name [ | pin_name ] ;
…
ALPG_Z_BIT15         = pin_name [ | pin_name ] ;
}
```

<ALPG_ID_1>

```
ALPG_PIN_GROUP {
ALPG_X1_BIT0         = pin_name [ | pin_name ] ;
…
ALPG_X1_BIT15        = pin_name [ | pin_name ] ;
ALPG_Y1_BIT0         = pin_name [ | pin_name ] ;
…
ALPG_Y1_BIT15        = pin_name [ | pin_name ] ;
ALPG_Z1_BIT0         = pin_name [ | pin_name ] ;
…
ALPG_Z1_BIT15        = pin_name [ | pin_name ] ;
}
```

This definition also support multi-channel mapping to the same ALPG X/Y/Z bit number.

```
ALPG_Z_BIT0= p10 | p15 ;        //p10 or p15 mapping to ALPG Z bit0
ALPG_Z_BIT1= p11 | p18 ;        //p11 or p18 mapping to ALPG Z bit1
```

4.19.1.4 Example

```
ALPG_PIN_GROUP // Use ALPG_ID_0
  {
      ALPGa_X_BIT0 = A0 ;
      ALPG_X_BIT1 = A2;
      ALPG_X_BIT2 = A10 ;
      ALPG_X_BIT3 = A9;
      ALPG_X_BIT4= A11 ;

      ALPG_Y_BIT0 = A8;
      ALPG_Y_BIT1 = A14;
      ALPG_Y_BIT2 = A13;
      ALPG_Y_BIT3 = A12;
      ALPG_Y_BIT4 = A7;
      ALPG_Y_BIT5 = A6;
      ALPG_Y_BIT6 = A5;
      ALPG_Y_BIT7 = A4;
      ALPG_Y_BIT8 = A3;

      ALPG_Z_BIT0 = D1;
      ALPG_Z_BIT1 = D2;
      ALPG_Z_BIT2 = D3
      ALPG_Z_BIT3 = D4;
      ALPG_Z_BIT4 = D5;
      ALPG_Z_BIT5 = D6;
      ALPG_Z_BIT6 = D7;
      ALPG_Z_BIT7 = D8;
  }

ALPG_PIN_GROUP // Use ALPG_ID_1
  {
       ALPG_Y1_BIT0=A8 | A0 ; //A8 & A0 map to Bit0
      ALPG_Y1_BIT1=A14 |A1;
       ALPG_Y1_BIT2=A13|A2;
       ALPG_Y1_BIT3=A12|A10;
       ALPG_Y1_BIT4=A7|A9;
       ALPG_Y1_BIT5=A6|A11;
       ALPG_Y1_BIT6=A5;
       ALPG_Y1_BIT7=A4;
  }
```

4.19.1.5 Relative

How to use ALPG
Example program: CRAFT\TestPlanExamples\ALPG_Example_Program

4.19.2 ALPG Pattern

4.19.2.1 Format

Parameter list:

SET_DEC_FILE "Device definition file"	Block
HEADER	Block
PATTERN	Block

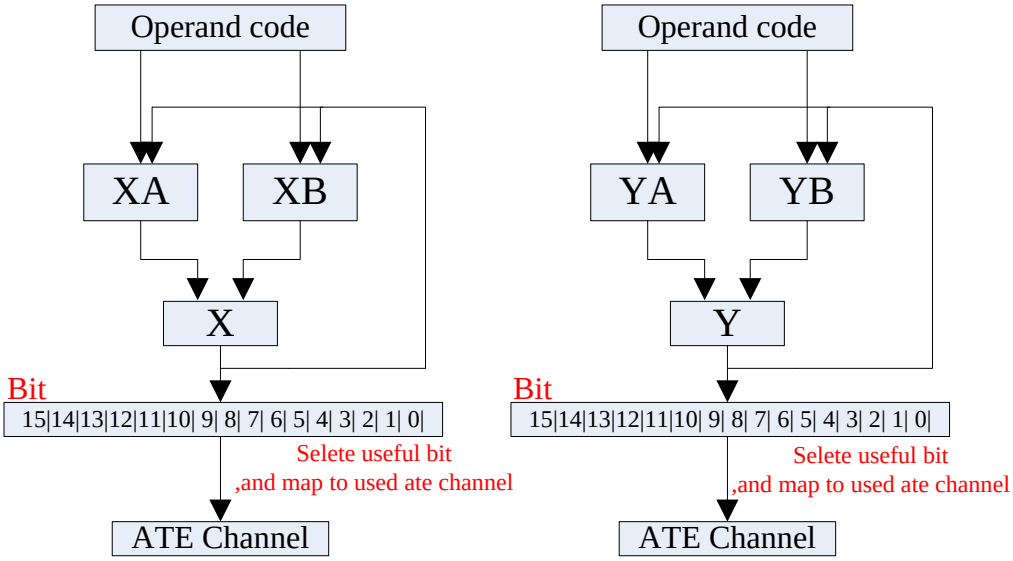
Parameter comment:
SET_DEC_FILE: The block defines the declaration file name and its' directory.

HEADER: The block defines declaration that is used by the order of entry and display of the actual binary information, on a pin basis.

PATTERN: This block defines the pattern function and format.
Use "APM_PATTERN".

4.19.2.2 Example

< The X/Y Address Generator Block Diagram >



```
SET_DEC_FILE "./ALPG_TEST.dec"

HEADER     %CE,WE,OE,%D8,D7,D6,D5,D4,D3,D2,D1,
       %A3,A4,A5,A6,A7,A12,A13,A14,A8,%A11,A9,A10,A2,A1,A0;

                /* CWO DDDDDDDD AAAAAAAAA AAAAAA */
                /* EEE 87654321 345671118 191210 */
                /*                         234   1 0      */
                /*                                  */
APM_PATTERN(Check_Board_ALPG_pat)
{
//                                  SQPG'UI                                          X'UI                  Y'UI                  Z'UI
   CH_ST:* 100 XXXXXXXX 000000000 000000 * TS1;

   W_CH0:* 000 DDDDDDDD BBBBBBBBB AAAAAA * SETX0 511,UISYNC;   [XA:0,X=XA,          YA:0,Y=YA,        ZA:0,ZB:0,Z=ZA==ZB]
           * 000 DDDDDDDD BBBBBBBBB AAAAAA * RPT 63;            [++XA,X=XA,          Y=YA,             ++ZA,        Z=ZA==ZB]
           * 000 DDDDDDDD BBBBBBBBB AAAAAA * JNZ0 .-1;          [XA:0,X=XA,          ++YA,Y=YA,        ZA:0,++ZB,Z=ZA==ZB]
           * 000 DDDDDDDD BBBBBBBBB AAAAAA * RPT 63;            [++XA,X=XA,          Y=YA,             ++ZA,        Z=ZA==ZB]

   R_CH0:* 010 CCCCCCCC BBBBBBBBB AAAAAA * TS1,SETX0 511;   [XA:0,X=XA,          YA:0,Y=YA,        ZA:0,ZB:0,Z=ZA==ZB]
           * 010 CCCCCCCC BBBBBBBBB AAAAAA * RPT 63;            [++XA,X=XA,          Y=YA,             ++ZA,        Z=ZA==ZB]
           * 010 CCCCCCCC BBBBBBBBB AAAAAA * JNZ0 .-1;          [XA:0,X=XA,          ++YA,Y=YA,        ZA:0,++ZB,Z=ZA==ZB]
           * 010 CCCCCCCC BBBBBBBBB AAAAAA * RPT 63;            [++XA,X=XA,          Y=YA,             ++ZA,        Z=ZA==ZB]

   W_CH1:* 000 DDDDDDDD BBBBBBBBB AAAAAA * TS1,SETX0 511;   [XA:0,X=XA,          YA:0,Y=YA,        ZA:0,ZB:0,Z=ZA!=ZB]
           * 000 DDDDDDDD BBBBBBBBB AAAAAA * RPT 63;            [++XA,X=XA,          Y=YA,             ++ZA,        Z=ZA!=ZB]
           * 000 DDDDDDDD BBBBBBBBB AAAAAA * JNZ0 .-1;          [XA:1,X=XA,          ++YA,Y=YA,        ZA:0,++ZB,Z=ZA!=ZB]
           * 000 DDDDDDDD BBBBBBBBB AAAAAA * RPT 63;            [++XA,X=XA,          Y=YA,             ++ZA,        Z=ZA!=ZB]

   r_CH1:* 010 CCCCCCCC BBBBBBBBB AAAAAA * TS1,SETX0 511;   [XA:0,X=XA,          YA:0,Y=YA,        ZA:0,ZB:0,Z=ZA!=ZB]
           * 010 CCCCCCCC BBBBBBBBB AAAAAA * RPT 63;            [++XA,X=XA,          Y=YA,             ++ZA,        Z=ZA!=ZB]
           * 010 CCCCCCCC BBBBBBBBB AAAAAA * JNZ0 .-1;          [XA:0,X=XA,          ++YA,Y=YA,        ZA:0,++ZB,Z=ZA!=ZB]
           * 010 CCCCCCCC BBBBBBBBB AAAAAA * RPT 63;            [++XA,X=XA,          Y=YA,             ++ZA,        Z=ZA!=ZB]

   CH_SP:* 100 XXXXXXXX 000000000 000000 * TS1;
}
```

4.19.2.3 Relative

How to use ALPG
Example program: CRAFT\TestPlanExamples\ALPG_Example_Program

4.19.3 SET_ALPG_WORK_MODE

SET_ALPG_WORK_MODE – CRAFT statement

Set ALPG output mapping of X/Y/Z which get from ALPG_PIN_GROUP declaration, and ALPG imask mode

4.19.3.1 Format

Parameter list:

```
SET_ALPG_WORK_MODE( mask_mode [, XAimask, Xbimask, YAimask,
YBimask,ZAimask, ZBimask] [,ALPG_PG_id]);
```

Parameter comment:

mask_mode : ON : the XMASK/YMASK/ZMASK load with XIMASK/YIMASK/ZIMASK while pattern burst (INIT)
OFF : XMASK/YMASK/ZMASK keep unchanged while pattern burst
**no_entry : default OFF ;

XAimask : ALPG XA mask initial value
0:NO MASK
1:MASK

XBimask : ALPG XB mask initial value
0:NO MASK
1:MASK

YAimask : ALPG YA mask initial value
0:NO MASK
1:MASK

YBimask : ALPG YB mask initial value
0:NO MASK
1:MASK

ZAimask : ALPG ZA mask initial value
0:NO MASK
1:MASK

ZBimask : ALPG ZB mask initial value
0:NO MASK
1:MASK

no_entry: keep current

ALPG_PG_ID: 0 → ALPG_X / ALPG_Y / ALPG_Z
1 → ALPG_X1 / ALPG_Y1 / ALPG_Z1
2 → ALPG_X2 / ALPG_Y2 / ALPG_Z2
…….
…….
…….
No entry:default 0

4.19.3.2 Example

```
TEST_NO(2);DATALOG_MSG="ALPG_Check_Board";
SET_ALPG_WORK_MODE(ON,0x0,0x0,0x0,0x0,0xFFFE,0xFFFE);
JUDGE_PAT(Check_Board_ALPG_pat:CH_ST ,Check_Board_ALPG_pat:CH_SP);
```

4.19.3.3 Relative

How to use ALPG
Example program: CRAFT\TestPlanExamples\ALPG_Example_Program

4.19.4 ALPG Micro Instruction

XA@R	XACNT clear to 0s
XA:data	XACNT load from immediately data
XA=X	XACNT load from XALU
++XA	XACNT +1
--XA	XACNT -1
++XA<	XACNT +1 if carry
--XA<	XACNT -1 if borrow
XB@R	----", 4,0, "0001", 4,39,36),-- 07,AUI[39:36] XBCNT clear to 0s
XB:data	XBCNT load from immediately data
XB=X	XBCNT load from XALU
++XB	XBCNT +1
--XB	XBCNT -1
++XB<	XBCNT +1 if carry
--XB<	XBCNT -1 if borrow
X=XA	XALU XA
X=~XA	XALU !XA
X=2XA	XALU 2XA=(XA+XA)
X=0	XALU 0=(XA-XA)
X=XB	XALU XB
X=~XB	XALU !XB
X=XA+XB	XALU XA+XB
X=XA-XB	XALU XA-XB
XMK:data	XADDR load XA/XB mask via immediately data
XF0	XADDR force XADDR to 0s
XF1	XADDR force XADDR to 1s
YA@R	YACNT clear to 0s

YA:data	YACNT load from immediately data
YA=Y	YACNT load from YALU
++YA	YACNT +1
--YA	YACNT -1
++YA<	YACNT +1 if carry
--YA<	YACNT -1 if borrow
YB@R	YBCNT clear to 0s
YB:data	YBCNT load from immediately data
YB=Y	YBCNT load from YALU
++YB	YBCNT +1
--YB	YBCNT -1
++YB<	YBCNT +1 if carry
--YB<	YBCNT -1 if borrow
Y=YA	YALU YA
Y=~YA	YALU !YA
Y=2YA	YALU 2YA=(YA+YA)
Y=0	YALU 0=(YA-YA)
Y=YB	YALU YB
Y=~YB	YALU !YB
Y=YA+YB	YALU YA+YB
Y=YA-YB	YALU YA-YB
YMK:data	YADDR load YA/YB mask via immediately data
YF0	YADDR force YADDR to 0s
YF1	YADDR force YADDR to 1s
ZA@R	ZACNT clear to 0s
ZA:data	ZACNT load data from immediate data
ZA@S	ZACNT set to 1s
++ZA	ZACNT incrment
--ZA	ZACNT decrement
ZA<<	ZACNT rotate left
ZA>>	ZACNT rotate right
ZA<<0	ZACNT shift left & carry in with 0
ZA0>>	ZACNT shift right & carry in with 0
ZA<<1	ZACNT shift left & carry in with 1
ZA1>>	ZACNT shift right & carry in with 1
ZB=0	ZBCNT clear to 0s
ZB:data	ZBCNT load data from immediate data

ZB=~0	ZBCNT set to 1s
++ZB	ZBCNT incrment
--ZB	ZBCNT decrement
ZB<<	ZBCNT rotate left
ZB>>	ZBCNT rotate right
ZB<<0	ZBCNT shift left & carry in with 0
ZB0>>	ZBCNT shift right & carry in with 0
ZB<<1	ZBCNT shift left & carry in with 1
ZB1>>	ZBCNT shift right & carry in with 1
Z=0	ZBLU all 0s
Z=ZA^ZB	ZBLU ZA xor ZB
Z=ZA	ZBLU ZA
Z=ZB	ZBLU ZB
Z=~0	ZBLU all 1s
Z=~(ZA^ZB)	ZBLU ZA xnor ZB
Z=~ZA	ZBLU !ZA
Z=~ZB	ZBLU !ZB
Z=ZA==ZB	ZDATA ZA=ZB
Z=ZA!=ZB	ZDATA ZF!=ZB
ZMK:data	ZDATA load ZA/ZB mask via immediately data
ZF0	ZDATA force ZDATA to 0s
ZF1	ZDATA force ZDATA to 1s

4.19.5 SET_ALPG_HOLDING_REG

SET_ALPG_HOLDING_REG – CRAFT statement
Set ALPG holding register HA0~HA7 and HB0~HB7

4.19.5.1 Format

Parameter list:

```
SET_ALPG_HOLDING_REG( holding_register, register_value);
```

Parameter comment:

holding_register : HA0~HA7 and HB0~HB7
**no_entry : illegal ;

register_value: 0 <= register_value <= 65535
**no_entry: illegal

4.19.5.2 Example

```
int d0, d1;

   d0 = 123;
   d1 = 456;
   SET_ALPG_HOLDING_REG(HA0+HA6, 10);
   SET_ALPG_HOLDING_REG(HA1, 20);
   SET_ALPG_HOLDING_REG(HA7, 40);
   SET_ALPG_HOLDING_REG(HA2, d0);
   SET_ALPG_HOLDING_REG(HB0, d1);
   SET_ALPG_HOLDING_REG(HB2+HB4, 100);
   SET_ALPG_HOLDING_REG(HB3, 200);
   SET_ALPG_HOLDING_REG(HB5, 400);
   JUDGE_PAT(ALPG_UI_HOLDING,__ALPG_UI_HOLDING);
```

4.19.6 READ_ALPG_REG_COUNTER

READ_ALPG_REG_COUNTER – CRAFT statement
Read ALPG XA/XB/XC/XD/YA/YB/YC/YD/ZA/ZB/ZC/ZD counter value

4.19.6.1 Format

Parameter list:

```
READ_ALPG_REG_COUNTER( register_name, return_value);
```

Parameter comment:

register_name : XA/XB/XC/XD/YA/YB/YC/YD/ZA/ZB/ZC/ZD
**no_entry : illegal ;

register_value: return register counter value
Type: integer
**no_entry: illegal

4.19.6.2 Example

```
int d0,d1,d2,d3;

SET_ALPG_HOLDING_REG(HA0+HA6, 10);
SET_ALPG_HOLDING_REG(HA1, 20);
SET_ALPG_HOLDING_REG(HA7, 40);
SET_ALPG_HOLDING_REG(HB2+HB4, 100);
SET_ALPG_HOLDING_REG(HB3, 200);
SET_ALPG_HOLDING_REG(HB5, 400);

JUDGE_PAT(ALPG_UI_HOLDING,__ALPG_UI_HOLDING);

READ_ALPG_REG_COUNTER(XA, d0);
READ_ALPG_REG_COUNTER(YB, d1);
READ_ALPG_REG_COUNTER(XC, d2);
READ_ALPG_REG_COUNTER(YD, d3);

JUDGE_VARIABLE (S0, d0, 4, 6);   //d0=5
JUDGE_VARIABLE (S0, d1, 9, 11);   //d1=10
JUDGE_VARIABLE (S0, d2, 34, 36);   //d2=35
JUDGE_VARIABLE (S0, d3, 39, 41);   //d3=40
```

4.19.7 READ_ALPG_LOG_MEM

READ_ALPG_LOG_MEM – CRAFT statement
Read ALPG datalog memory

4.19.7.1 Format

Parameter list:

```
READ_ALPG_LOG_MEM(socket _no, return_site_log_cnt, X_log, Y_log,
                  Z_log, PMA_log);
```

Parameter comment:

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

return_site_log_cnt : return site log counter
Type : integer
**no_entry: illegal

X_log : return X address log data
Type : integer array
**no_entry: illegal

Y_log : return Y address log data
Type : integer array
**no_entry: illegal

Z_log : return Z data or address log data
Type : integer array
**no_entry: illegal

PMA_log : return pattern memory address log data
Type : integer array
**no_entry: illegal

4.19.7.2 Example

```
int i,log_cnt;
int x_log[1024],y_log[1024],z_log[1024],pma_log[1024];

    //===== ADDRESS PATTERN =====
    DATALOG_MSG="ALPG_Address";
    TEST_NO(1004);
    SET_ALPG_WORK_MODE(ON, 0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    JUDGE_PAT(Address_ALPG_pat:AD_ST ,Address_ALPG_pat:AD_SP);

    READ_ALPG_LOG_MEM(S0, log_cnt, X_log, Y_log, Z_log, pma_log);
    printf("log_cnt = %d \n",log_cnt);
    for(i=0;i<log_cnt;i++)
    {
        printf("x_log[%4d]=%x \n",i,x_log[i]);
        printf("y_log[%4d]=%x \n",i,y_log[i]);
        printf("z_log[%4d]=%x \n",i,z_log[i]);
        printf("pma_log[%4d]=%x \n",i,pma_log[i]);
    }
```

4.19.8 SET_ALPG_WORK_MODE_X

SET_ALPG_WORK_MODE_X – CRAFT statement
Set ALPG output mapping of X/Y/Z which get from ALPG_PIN_GROUP declaration, and ALPG imask mode

4.19.8.1 Format

Parameter list:

```
SET_ALPG_WORK_MODE_X( mask_mode [, XAimask, XBimask, YAimask,
YBimask,ZAimask, ZBimask, XCimask, XDimask, YCimask,
YDimask,ZCimask, ZDimask] [,ALPG_PG_id]);
```

Parameter comment:

```
mask_mode :  ON    : the XMASK/YMASK/ZMASK load with XIMASK/YIMASK/ZIMASK
                     while pattern burst (INIT)
             OFF   : XMASK/YMASK/ZMASK keep unchanged while pattern burst
             **no_entry : default OFF ;

XAimask : ALPG XA mask initial value
          0:NO MASK
          1:MASK

XBimask : ALPG XB mask initial value
          0:NO MASK
          1:MASK

YAimask : ALPG YA mask initial value
          0:NO MASK
          1:MASK

YBimask : ALPG YB mask initial value
          0:NO MASK
          1:MASK

ZAimask : ALPG ZA mask initial value
          0:NO MASK
          1:MASK

ZBimask : ALPG ZB mask initial value
          0:NO MASK
          1:MASK

XCimask : ALPG XC mask initial value
          0:NO MASK
          1:MASK

XDimask : ALPG XD mask initial value
          0:NO MASK
          1:MASK
```

YCimask: ALPG YC mask initial value
0:NO MASK
1:MASK

YDimask: ALPG YD mask initial value
0:NO MASK
1:MASK

ZCimask: ALPG ZC mask initial value
0:NO MASK
1:MASK

ZDimask: ALPG ZD mask initial value
0:NO MASK
1:MASK

no_entry: keep current

ALPG_PG_ID: 0 → ALPG_X / ALPG_Y / ALPG_Z
1 → ALPG_X1 / ALPG_Y1 / ALPG_Z1
2 → ALPG_X2 / ALPG_Y2 / ALPG_Z2
…….
…….
…….
No entry:default 0

4.19.8.2 Example

```
TEST_NO(2);DATALOG_MSG="ALPG_Check_Board";
SET_ALPG_WORK_MODE_X(ON,0x0,0x0,0x0,0x0,0xFFFE,0xFFFE,
                                    0x0,0x0,0x0,0x0,0xFFFE,0xFFFE );
JUDGE_PAT(Check_Board_ALPG_pat:CH_ST ,Check_Board_ALPG_pat:CH_SP);
```

4.19.8.3 Relative

How to use ALPG
Example program: CRAFT\TestPlanExamples\ALPG_Example_Program

4.20 AWG/Digitizer (Option)

4.20.1 AWI Pins Declaration

AWI Pin must be declared in Declaration file (*.dec).

4.20.1.1 Pogopin Definition

MXAWI PoGo Pin Define									
Pogo Pin No	define	define	define	define	define	define	define	define	Pogo Pin No
	a	b	c	d	e	f	g	h	
1	AG	AG	AG	TC	AG	AG	AG	AG	1
2	WD0+	AG	WD0-	AG	NC	AG	NC	NC	2
3	AG	NC	AG	NC	AG	External Trigger0	NC	NC	3
4	WD2+	AG	WD2-	AG	NC	AG	NC	NC	4
5	AG	NC	AG	NC	AG	External Trigger2	NC	NC	5
6	WD4+	AG	WD4-	AG	NC	AG	NC	NC	6
7	AG	NC	AG	NC	AG	External Trigger4	NC	NC	7
8	WD6+	AG	WD6-	AG	NC	AG	NC	NC	8
9	AG	NC	AG	NC	AG	External Trigger6	NC	NC	9
10	WG8_F+	AG	WG8_F-	AG	AGS8	AG	NC	NC	10
11	AG	NC	AG	NC	AG	External Trigger8	NC	NC	11
12	WG10_F+	AG	WG10_F-	AG	AGS10	AG	NC	NC	12
13	AG	NC	AG	NC	AG	External Trigger10	NC	NC	13
14	WG12_F+	AG	WG12_F-	AG	AGS12	AG	NC	NC	14
15	AG	NC	AG	NC	AG	External Trigger12	NC	NC	15
16	WG14_F+	AG	WG14_F-	AG	AGS14	AG	NC	NC	16
17	AG	NC	AG	NC	AG	External Trigger14	NC	NC	17

4.20.1.2 TYPE: WG or WD

.

4.20.1.3 Channel Assign

WD+ : channel num=slot*64 + (0, 2, 4, 6)
WG+ : channel num=slot*64 + (8, 10, 12, 14)

Example:
AWI slot6: WD0+ (384) , WD2+ (386)
: WD4+ (388) , WD6+ (390)
: WG8+ (392) , WG10+ (394)
: WG12+ (396), WG14+ (398)

4.20.1.4 Example

```
PIN_LIST( AD976ar_3380 )
        {
        /*name = ATE channel = dut# = type    */
          WG0   = 392        = 1   = WG    ;   //TIP
          DB15 = 160        = 6   = IO    ;   //IO
          DB14 = 128        = 7   = IO    ;   //IO
          DB13 = 152        = 8   = IO    ;   //IO
          DB12 = 184        = 9   = IO    ;   //IO
          DB11 = 168        = 10 = IO    ;   //IO
          DB10 = 208        = 11 = IO    ;   //IO
          DB9   = 224        = 12 = IO    ;   //IO
          DB8   = 240        = 13 = IO    ;   //IO
          DB7   = 248        = 15 = IO    ;   //IO
          DB6   = 232        = 16 = IO    ;   //IO
          DB5   = 216        = 17 = IO    ;   //IO
          DB4   = 192        = 18 = IO    ;   //IO
          DB3   = 200        = 19 = IO    ;   //IO
          DB2   = 176        = 20 = IO    ;   //IO
          DB1   = 144        = 21 = IO    ;   //IO
          DB0   = 136        = 22 = IO    ;   //IO
          R_C   = 134        = 24 = IO    ;   //IO
          Vdig = 0          = 28 = DPS   ;   /*use DPS0   */
          Vana = 1          = 27 = DPS   ;   /*use DPS1   */
          AGnd =            = 2   = GND   ;   /*GND pin didn't any ate channel */
          DGnd =            = 14 = GND   ;   /*GND pin didn't any ate channel */
        }
```

```
PIN_LIST( DAC7741_3380 )
        {
        /*name = ATE channel = dut# = type    */
          WD0   = 384     = 11   = WD    ;   //TIP
          DB15 = 207     = 35   = IO    ;   //IO
          DB14 = 206     = 34   = IO    ;   //IO
          DB13 = 205     = 33   = IO    ;   //IO
          DB12 = 204     = 32   = IO    ;   //IO
          DB11 = 203     = 31   = IO    ;   //IO
          DB10 = 202     = 30   = IO    ;   //IO
```

```
        DB9   = 201    = 29   = IO    ;   //IO
        DB8   = 200    = 28   = IO    ;   //IO
        DB7   = 199    = 27   = IO    ;   //IO
        DB6   = 198    = 22   = IO    ;   //IO
        DB5   = 197    = 21   = IO    ;   //IO
        DB4   = 196    = 20   = IO    ;   //IO
        DB3   = 195    = 19   = IO    ;   //IO
        DB2   = 194    = 18   = IO    ;   //IO
        DB1   = 193    = 17   = IO    ;   //IO
        DB0   = 192    = 16   = IO    ;   //IO
        RST   = 224    = 39   = IO    ;   //IO
        R_W   = 227    = 42   = IO    ;   //IO
        CS    = 226    = 41   = IO    ;   //IO
        LDAC = 225     = 40   = IO    ;   //IO
        Vp15 = 5       = 3    = PREF ;   /*use PREF   5   */
        Vn15 = 4       = 2    = PREF ;   /*use PREF   4   */
        Vp5   = 12     = 38   = PREF ;   /*use PREF 12   */
        AGnd =         = 6    = GND   ;   /*GND pin didn't any ate channel */
        DGnd =         = 37   = GND   ;   /*GND pin didn't any ate channel */
    }
```

4.20.1.5 Relative

PIN_LIST Declaration

4.20.2 AWI_HEADER – CRAFT declaration

The AWI_HEADER declaration is used to the order of entry and display of the actual binary Information in Pattern file .

4.20.2.1 Format

Parameter list:

```
AWI_HEADER          [%]pin_name, [%]pin_name, …;
```

Parameter comment:

Pin_name : a name of a single pin defined in the PIN_LIST declaration.
Only the WG or WD pin-type constant names from the PIN_LIST are allowed.
"C" type: STRING, 64 characters maximum.

% : a seperator, indicating the compiler to use one space between the previous pin and next pin, in all of the output from the system, such as Pattern Editor Utility .

the AWI pin pattern symbol : (Pattern trigger)
#P01 → means the AWI pin use waveform_set 1 by pattern trigger
…
#P31 → means the AWI pin use waveform_set 31 by pattern trigger
the AWI pin pattern symbol : (External trigger)
#E01 → means the AWI pin use waveform_set 1 by external trigger
…
#E31 → means the AWI pin use waveform_set 31 by external trigger

4.20.2.2 Example

```
SET_DEC_FILE "./ad976ar_pin.dec"

HEADER
DB15,DB14,DB13,DB12,%DB11,DB10,DB9,DB8,%DB7,DB6,DB5,DB4,%DB3,DB2,
DB1,DB0,%R_C;
AWIHEADER WG0;

RPM_PATTERN(func_pat8)
{
            /* DDDD DDDD DDDD DDDD R */
            /* BBBB BBBB BBBB BBBB / */
            /* 1111 1198 7654 3210 C */
            /* 5432 10               */
sin8_st ::  *XXXX XXXX XXXX XXXX 1      * TS1 ;
            *XXXX XXXX XXXX XXXX 1  #P16 * ;
            *XXXX XXXX XXXX XXXX 1      * RPT 2048;
            *XXXX XXXX XXXX XXXX 1      * ;
            *XXXX XXXX XXXX XXXX 1      * ;
            *ZZZZ ZZZZ ZZZZ ZZZZ 0  #P16 * UISYNC;
            *ZZZZ ZZZZ ZZZZ ZZZZ 0      * RPT 1022;
```

```
                    *ZZZZ ZZZZ ZZZZ ZZZZ 0            * ;
sin8_sp ::    *XXXX XXXX XXXX XXXX 1            * ;
}
```

```
SET_DEC_FILE "./dac7741_pin.dec"

HEADER
DB15,DB14,DB13,DB12,%DB11,DB10,DB9,DB8,%DB7,DB6,DB5,DB4,%DB3,DB2,
DB1,DB0,
            %RST,R_W,CS,LDAC;
AWIHEADER WD0;


RPM_PATTERN(func_pat1)
{
                    /* DDDD DDDD DDDD DDDD RRCL    */
                    /* BBBB BBBB BBBB BBBB S/SD    */
                    /* 1111 1198 7654 3210 TW A    */
                    /* 5432 10                     C    */
    ramp1_st ::   *XXXX XXXX XXXX XXXX 1110          * TS1 ;
                    *XXXX XXXX XXXX XXXX 0110          * ;
                    *0000 0000 0000 0000 1001          * RPT 100;
                    *0000 0000 0000 0000 1001 #P16 * UISYNC;
                    *0000 0000 0000 0001 1001          * ;
                    *0000 0000 0000 0010 1001          * ;
                    *0000 0000 0000 0011 1001          * ;
                    *0000 0000 0000 0100 1001          * ;
                    *0000 0000 0000 0101 1001          * ;
                    *0000 0000 0000 0110 1001          * ;
                    *0000 0000 0000 0111 1001          * ;
                    *0000 0000 0000 1000 1001          * ;
                    *0000 0000 0000 1001 1001          * ;
                    *0000 0000 0000 1010 1001          * ;
                    *0000 0000 0000 1011 1001          * ;
}
```

4.20.2.3 Relative

Pattern Files

4.20.3 LOAD_AWG_PAT

LOAD_AWG_PAT – CRAFT statement

Load the waveform generater pattern file into the hardware pattern memories.

Each waveform_set has "rpt_mode" setting indenpendently

4.20.3.1 Format

Parameter list:

```
LOAD_AWG_PAT(pin_name, "./awg_pattern_file_name",
waveform_set_no,rpt_mode,rpt_cnt) ;
```

Parameter comment:

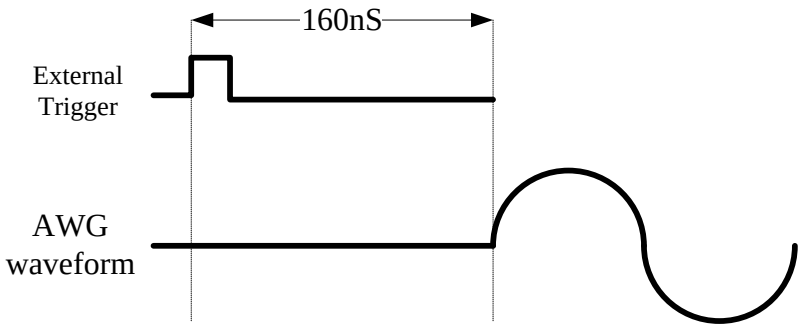
pin_name :pin_def, pin_group (WG PIN)
**no_entry; illegal

awg_pattern_file_name :max 32 chars
**no_entry: illegal

waveform_set_no :waveform set number, 1~31
**no_entry: illegal

rpt_mode :NORM //equal RPT 1
RPT
CONT
STEP_TRIG
**no_entry:default NORM

rpt_cnt : repeat count



AWG will delay 160nS, then output the waveform , when AWI receive the external trigger.

4.20.3.2 Example

```
SET_DEC_FILE "./ad976ar_pin.dec"

HEADER
DB15,DB14,DB13,DB12,%DB11,DB10,DB9,DB8,%DB7,DB6,DB5,DB4,%DB3,DB2,
DB1,DB0,%R_C;
AWIHEADER WG0;

RPM_PATTERN(func_pat8)
{
            /* DDDD DDDD DDDD DDDD R */
            /* BBBB BBBB BBBB BBBB / */
            /* 1111 1198 7654 3210 C */
            /* 5432 10                */
sin8_st ::  *XXXX XXXX XXXX XXXX 1      * TS1 ;
              *XXXX XXXX XXXX XXXX 1  #P16 * ;
              *XXXX XXXX XXXX XXXX 1      * RPT 2048;
              *XXXX XXXX XXXX XXXX 1      * ;
              *XXXX XXXX XXXX XXXX 1      * ;
              *ZZZZ ZZZZ ZZZZ ZZZZ 0  #P16 * UISYNC;
              *ZZZZ ZZZZ ZZZZ ZZZZ 0      * RPT 1022;
              *ZZZZ ZZZZ ZZZZ ZZZZ 0      * ;
sin8_sp ::  *XXXX XXXX XXXX XXXX 1       * ;
}
```

```
  LOAD_AWG_PAT(WG0,"sin_25v.awi", 16 , RPT, 13);
  SET_WG_WORK_MODE(WG0, SINGLE_END, @RNG3, 0.0V, R50, OFF, ON);

SET_CAPTURE_MEM_MODE(DB_PINS,PEL,ON,64K,1024,LOG_FAIL_ONLY,ON);
  WAIT(Delay);

  CRAFT_STATEMENT
  {
    JUDGE_PAT(sin8_st,sin8_sp);
    PLAN_RESULT_MS[0]=PASS;
    for(i=0; i<1024; i++)
    {
        READ_CAPTURE_MEM(S0,DB_PINS,i,pel);
        printf("%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d\n", pel[0], pel[1],
pel[2], pel[3], pel[4], pel[5], pel[6], pel[7], pel[8], pel[9], pel[10], pel[11], pel[12], pel[13],
pel[14], pel[15]);
        data_f[i]=0;
        for(j=0,w=15; w>=0; w--,j++)
        {
            data_f[i] = data_f[i] | (pel[j]<<w);
        }
        data_val[i] = ((double)(data_f[i] ^ 0x8000) - 32768.0) / 32768.0 * 10.0;
    }

    AWI_DATA_AC_ARITH(data_val,1024, thd, snr, snd, sfdr,thd_n);
```

Load “sin_25v.awi” pattern to waveform set “16 ”

4.20.3.3 WG Text Output Format

Sine:

```
VERSION: 1.02
BOARD_NAME: LXAWI
AWG_WAVE: sine
AWG_SAMPLING_RATE: 100000.00000
AWG_SAMPLING_PERIOD: 0.0000100000                    //Reference Only
AWG_OUTPUT_MODE: SINGLE_END | DIFFERENTIAL
AWG_DIFF_DATA_RATIO: X1 | X2      // If AWG_OUTPUT_MODE is DIFFERENTIAL, this
comment will be existed.
AWG_SAMPLES_NO: 200
OUTPUT_FREQUENCY: 500.000000
AWG_FILTER: OFF
OUTPUT_CYCLES_NO: 1                                  //Reference Only
AWG_OFFSET:   0.00                                   //Reference Only
OUTPUT_AMPLITUDE: 0.50                               //Reference Only
AWG_SAMPLING_PHASE: 0                                //Reference Only
AWG_VRNG: @RNG3                                      //Reference Only
COMMENT: (null)
AWG_PATTERN (sine) {
0.000000000000
0.015704916500
...
...
}
```

Ramp:

```
VERSION: 1.02
BOARD_NAME: LXAWI
AWG_WAVE: ramp
AWG_START: 0.000000
AWG_STOP: 3.000000
AWG_STAIRS: 1024
AWG_PER_STAIR: 0.002929688
AWG_REPEAT_PER_STAIR: 1
AWG_RAMP_PERIOD: 0.0010000000
AWG_FILTER: OFF
AWG_VRNG: @RNG4       //Reference Only
COMMENT: (null)
AWG_PATTERN (ramp) {
-2.500000000000
-2.497070312000
...
...
}
```

4.20.4 SET_WG_WORK_MODE

SET_WG_WORK_MODE – CRAFT statement

Set STAWI waveform Generator work mode

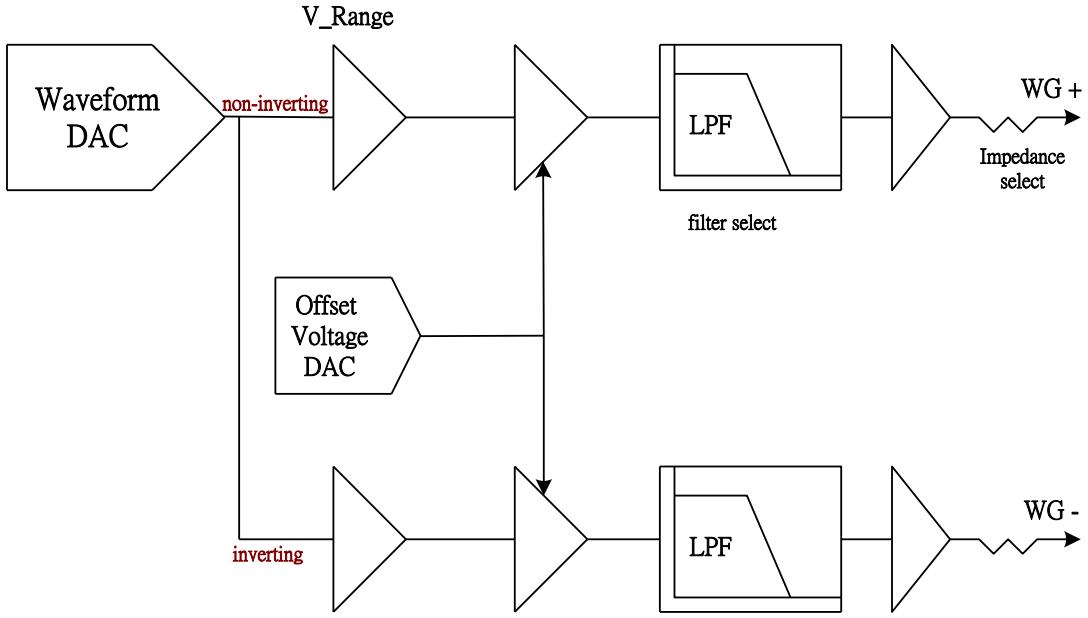


Fig. Waveform Generater

4.20.4.1 Format

Parameter list:

SET_WG_WORK_MODE(pin_name,output_mode,v_range, focus_offset_voltage, impedence_select, filter_select, relay [, cont_stop_mode]);

Parameter comment:

pin_name: pin_def, pin_group (WG PIN TYPE)
**no_entry; illegal

output_mode: SINGLE_END : single end output (just use WG+)

v_range : @RNG0 : -10V ~ -5V //(LV: -2.5V~+2.5V, & offset -7.5V)
@RNG1 : -7.5V ~ -2.5V //(LV: -2.5V~+2.5V, & offset -5.0V)
@RNG2 : -5V ~ 0V //(LV: -2.5V~+2.5V, & offset -2.5V)
@RNG3 : -2.5V ~ 2.5V //(LV: -2.5V~+2.5V, & offset 0V)
@RNG4 : -0.1V ~ 5.1V //(LV: -2.5V~+2.5V, & offset +2.5V)
@RNG5 : 2.5V ~ 7.5V //(LV: -2.5V~+2.5V, & offset +5.0V)
@RNG6 : 5V ~ 10V //(LV: -2.5V~+2.5V, & offset +7.5V)
@RNG7 : -10V ~ 10V //(HV: -10V~+10V, & offset 0V)
@USER : user define offset voltage (Focus Calibration)
type: integer

**no_entry : illegal
//if user never setup the WG Focus calibration, then @USER will be equal to @RNG7

focus_offset_voltage: Focus offset voltage (-7.5V ~ 7.5V)
the focus offset voltage must match the Maint-Utility WD Focus calibration setting.
when "v_range" is not @USER, the value is no meaning.
type: double / VOLTAGE

impedence_select :
R50 : 50 Ohm
R10 : from the OP
filter_select:
R0 : 20KHz Low Pass filter
OFF :

relay: ON – turns on WG output relay.
OFF – turns off WG output relay.
type: integer
**no_entry: keep current state

cont_stop_mode: MODE_A → Continue-mode stop by pattern end
MODE_B → Continue-mode stop by statement "WG_PAT_STOP"
**no_entry: MODE_A
PS: WG output waveform settling time : 1uS

4.20.4.2 Example

```
  LOAD_AWG_PAT(WG0,"sin_25v.awi", 16 , RPT, 13);
  SET_WG_WORK_MODE(WG0, SINGLE_END, @RNG3, 0.0V, R50, OFF, ON);

SET_CAPTURE_MEM_MODE(DB_PINS,PEL,ON,64K,1024,LOG_FAIL_ONLY,ON);
  WAIT(Delay);
  CRAFT_STATEMENT
  {
    JUDGE_PAT(sin8_st,sin8_sp);
    PLAN_RESULT_MS[0]=PASS;
    for(i=0; i<1024; i++)
    {
        READ_CAPTURE_MEM(S0,DB_PINS,i,pel);
        printf("%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d\n", pel[0], pel[1],
pel[2], pel[3], pel[4], pel[5], pel[6], pel[7], pel[8], pel[9], pel[10], pel[11], pel[12], pel[13],
pel[14], pel[15]);
        data_f[i]=0;
        for(j=0,w=15; w>=0; w--,j++)
        {
            data_f[i] = data_f[i] | (pel[j]<<w);
        }
        data_val[i] = ((double)(data_f[i] ^ 0x8000) - 32768.0) / 32768.0 * 10.0;
    }
     AWI_DATA_AC_ARITH(data_val,1024, thd, snr, snd, sfdr,thd_n);
```

```
SET_DEC_FILE "./ad976ar_pin.dec"

HEADER
DB15,DB14,DB13,DB12,%DB11,DB10,DB9,DB8,%DB7,DB6,DB5,DB4,%DB3,DB2,
DB1,DB0,%R_C;
AWIHEADER WG0;

RPM_PATTERN(func_pat8)
{
            /* DDDD DDDD DDDD DDDD R */
            /* BBBB BBBB BBBB BBBB / */
            /* 1111 1198 7654 3210 C */
            /* 5432 10                */
sin8_st ::  *XXXX XXXX XXXX XXXX 1      * TS1 ;
              *XXXX XXXX XXXX XXXX 1  #P16 * ;
              *XXXX XXXX XXXX XXXX 1      * RPT 2048;
              *XXXX XXXX XXXX XXXX 1      * ;
              *XXXX XXXX XXXX XXXX 1      * ;
              *ZZZZ ZZZZ ZZZZ ZZZZ 0  #P16 * UISYNC;
              *ZZZZ ZZZZ ZZZZ ZZZZ 0      * RPT 1022;
              *ZZZZ ZZZZ ZZZZ ZZZZ 0      * ;
sin8_sp ::  *XXXX XXXX XXXX XXXX 1      * ;
}
```

Trigger “P16 “

4.20.4.3 Relative

LOAD_AWG_PAT , SET_CAPTURE_MEM_MODE , READ_CAPTURE_MEM ,AWI_DATA_AC_ARITH

4.20.5 SET_WD_WAVEFORM_SET

SET_WD_WAVEFORM_SET – CRAFT statement
Set STAWI digitizer waveform set setting

4.20.5.1 Format

Parameter list:

```
SET_WD_WAVEFORM_SET(Pin_name,      Waveform_set_no,
Sampling_rate,Phase_delay, Start_addr, Log_count );
```

Parameter comment:

Waveform_set_no : 1~31
Sampling_rate: period value 4uS to 5mS(resolution 10nS)
variables & constants allowed
type: TIME or double
unit: S, pS, nS, uS, mS
**no_entry: illegal

Phase_delay : 0 ~ sampling_period
Start_addr : 0 ~ 256K
Log_count : 0 ~ 256K
in pattern trigger, only when digitizer total log "Log_count" data, then the DGT complete the pattern-trigger mode.

4.20.5.2 Example

```
        SET_WD_WORK_MODE(WD0, @RNG3, 0.0V, OFF, OFF, OFF, ON);
        SET_WD_WAVEFORM_SET(WD0, WD_waveform_set, 10.0e-6, 9.0e-6,
0x0, log_cnt_temp);
       JUDGE_PAT(ramp1_st,ramp1_sp);

      READ_WD_DATA(S0, WD0, WD_waveform_set, log_cnt_temp, data_tmp);

        AWI_DATA_DC_ARITH(data_tmp, 4096, -1*(DAC_FSR/2), DAC_FSR, Inl,
                                Dnl, gain_error, offset_error ) ;
```

```
SET_DEC_FILE "./dac7741_pin.dec"

HEADER
DB15,DB14,DB13,DB12,%DB11,DB10,DB9,DB8,%DB7,DB6,DB5,DB4,%DB3,DB2
,DB1,DB0,
            %RST,R_W,CS,LDAC;
AWIHEADER WD0;


RPM_PATTERN(func_pat1)
{
                    /* DDDD DDDD DDDD DDDD RRCL    */
```

```
                /* BBBB BBBB BBBB BBBB S/SD    */
                /* 1111 1198 7654 3210 TW A    */
                /* 5432 10                     C    */
ramp1_st ::   *XXXX XXXX XXXX XXXX 1110          * TS1 ;
                *XXXX XXXX XXXX XXXX 0110          * ;
                *0000 0000 0000 0000 1001        * RPT 100;
                *0000 0000 0000 0000 1001 #P16 * UISYNC;
                *0000 0000 0000 0001 1001        * ;
                *0000 0000 0000 0010 1001        * ;
                *0000 0000 0000 0011 1001        * ;
                *0000 0000 0000 0100 1001        * ;
                *0000 0000 0000 0101 1001        * ;
                *0000 0000 0000 0110 1001        * ;
                *0000 0000 0000 0111 1001        * ;
                *0000 0000 0000 1000 1001        * ;
                *0000 0000 0000 1001 1001        * ;
                *0000 0000 0000 1010 1001        * ;
                *0000 0000 0000 1011 1001        * ;
```

4.20.5.3 Relative

SET_WD_WORK_MODE , READ_WD_DATA , AWI_DATA_DC_ARITH

4.20.6 SET_WD_WORK_MODE

SET_WD_WORK_MODE – CRAFT statement
Set STAWI waveform digitizer work mode

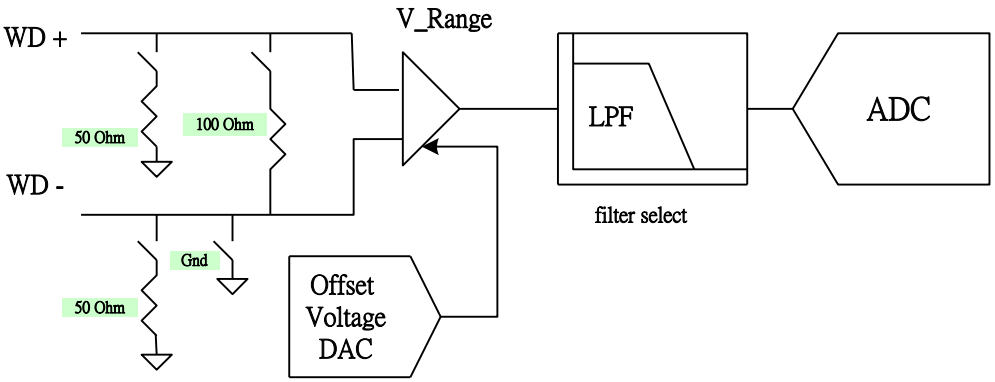


Fig. Waveform Digitizer

4.20.6.1 Format

Parameter list:

SET_WD_WORK_MODE(pin_name, v_range, focus_offset_ voltage, Pos_impedence_select, Neg_ impedence_select, filter_select, relay);

Parameter comment:

v_range : @RNG0 : -5V ~ 0V
@RNG1 : -2.5V ~ 2.5V
@RNG2 : 0V ~ 5V
@RNG3 : -10V ~ 10V
@RNG3 : -1V ~ 4V
@USER : user define offset voltage (Focus Calibration)
type: integer
**no_entry : illegal

focus_offset_voltage: when v_range != @USER, the setting is useless
Pos_impedence_select :
R50 : WD+ 50 Ohm relay
R100 : between WD+ and WD- relay
OFF : relay off
Neg_impedence_select :
R50 : WD- 50 Ohm relay
R0 : WD- Gnd relay
OFF : relay off
filter_select:
R0 : 20K HZ Low pass filter
R1 : 50K HZ Low pass filter
OFF : all relay off

relay: ON – turns on WD output relay.
OFF – turns off WD output relay.
type: integer
**no_entry: keep current state

4.20.6.2 Example

```
SET_WD_WORK_MODE(WD0, @RNG3, 0.0V, OFF, OFF, OFF, ON);
SET_WD_WAVEFORM_SET(WD0, WD_waveform_set, 10.0e-6, 9.0e-6, 0x0,
log_cnt_temp);
JUDGE_PAT(ramp1_st,ramp1_sp);
READ_WD_DATA(S0, WD0, WD_waveform_set, log_cnt_temp, data_tmp);
AWI_DATA_DC_ARITH(data_tmp, 4096, -1*(DAC_FSR/2), DAC_FSR, Inl,
                                Dnl, gain_error, offset_error ) ;
```

```
SET_DEC_FILE "./dac7741_pin.dec"

HEADER
DB15,DB14,DB13,DB12,%DB11,DB10,DB9,DB8,%DB7,DB6,DB5,DB4,%DB3,DB2
,DB1,DB0,
            %RST,R_W,CS,LDAC;
AWIHEADER WD0;


RPM_PATTERN(func_pat1)
{
                    /* DDDD DDDD DDDD DDDD RRCL    */
                    /* BBBB BBBB BBBB BBBB S/SD    */
                    /* 1111 1198 7654 3210 TW A    */
                    /* 5432 10                     C    */
    ramp1_st ::   *XXXX XXXX XXXX XXXX 1110          * TS1 ;
                    *XXXX XXXX XXXX XXXX 0110          * ;
                    *0000 0000 0000 0000 1001          * RPT 100;
                    *0000 0000 0000 0000 1001 #P16 * UISYNC;
                    *0000 0000 0000 0001 1001          * ;
                    *0000 0000 0000 0010 1001          * ;
                    *0000 0000 0000 0011 1001          * ;
                    *0000 0000 0000 0100 1001          * ;
                    *0000 0000 0000 0101 1001          * ;
                    *0000 0000 0000 0110 1001          * ;
                    *0000 0000 0000 0111 1001          * ;
                    *0000 0000 0000 1000 1001          * ;
                    *0000 0000 0000 1001 1001          * ;
                    *0000 0000 0000 1010 1001          * ;
                    *0000 0000 0000 1011 1001          * ;
}
```

4.20.6.3 Relative

SET_WD_WAVERORM_SET , READ_WD_DATA , AWI_DATA_DC_ARITH

4.20.7 READ_WD_DATA

READ_WD_DATA -- CRAFT statement
Read digitizer acquire memory raw data

4.20.7.1 Format

Parameter list:

```
READ_WD_DATA(socket _no, pin_name, waveform_set_no, log_cnt, dgt_data);
```

Parameter comment:

socket_no :
S0 : socket No. 0
S1 : socket No. 1
…
S1023 : socket No. 1023
Type : integer
**no_entry: illegal

pin_name : pin_name (only one pin)
**no_entry: illegal

data : to specify the target DUT number to be read and read data from AQM, then save it as double type data in the array variable .

4.20.7.2 Example

```
SET_WD_WORK_MODE(WD0, @RNG3, 0.0V, OFF, OFF, OFF, ON);
SET_WD_WAVEFORM_SET(WD0, WD_waveform_set, 10.0e-6, 9.0e-6, 0x0,
log_cnt_temp);
JUDGE_PAT(ramp1_st,ramp1_sp);
READ_WD_DATA(S0, WD0, WD_waveform_set, log_cnt_temp, data_tmp);
AWI_DATA_DC_ARITH(data_tmp, 4096, -1*(DAC_FSR/2), DAC_FSR, Inl,
                                  Dnl, gain_error, offset_error ) ;
```

4.20.7.3 Relative

SET_WD_WAVERORM_SET , READ_WD_DATA , AWI_DATA_DC_ARITH , SET_WD_WORK_MODE

4.20.8 AWI_DATA_AC_ARITH

AWI_DATA_AC_ARITH- CRAFT statement
For Calculating THD,SNR,SND,SFDR and THD+N and return value

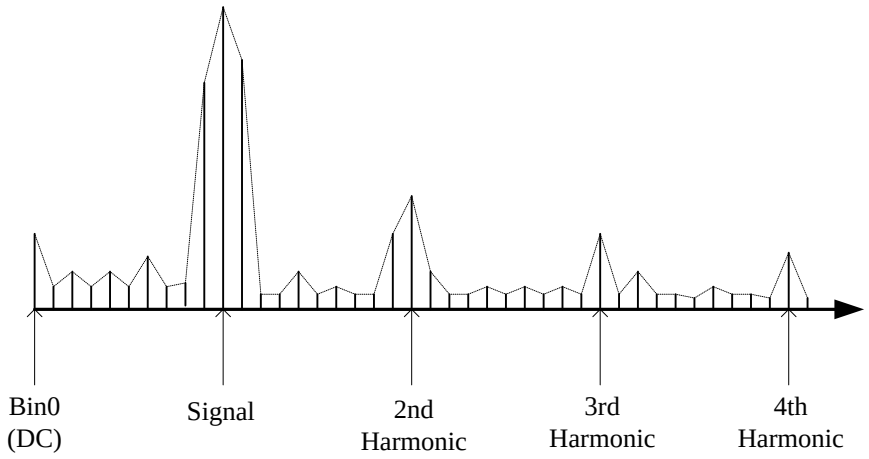
4.20.8.1 Format

Parameter list:

```
AWI_DATA_AC_ARITH(T_data, point_count, THD_value, SNR_value, SND_value, SFDR_value,
THD_n_value [, DC_value, SIGNAL_value, HARMONIC_value, Noise_value, weighting_window ]
      [, main_lobe_width, side_lobe_width ] ) ;
```

Parameter comment:

T_data : Time domain data,
Type : double array
Point_count : effective point number, must be 2^n
THD_value : Total Harmonic Distorsion (unit: db)
SNR _value : Signal to Noise Ratio (unit: db)
SND _value : Signal to noise plus Distortion (unit: db)
SFDR _value : Spurious Free Dynamic Range (unit: db)
THD_n_value : %THD+n (unit: %)
DC_value : (unit: Vrms)
SIGNAL_value : (unit: Vrms)
HARMONIC_value : (unit: Vrms)
Noise_value : (unit: Vrms)
weighting_window : HAN (Hanning window)
HAM (Hamming window)
REC (Rectangular window, default)
main_lobe_width : default 2
side_lobe_width : default 1



if main lobe width = 0, then signal = bin10 ;
if main lobe width = 1, then signal = bin9+bin10+bin11 ;

if side lobe width = 0, then 2nd-harmonic = bin20 ;
if side lobe width = 1, then 2nd-harmonic = bin19+bin10+bin21 ;

Caculation Equation :

Signal_value = $H_{in(rms)}$

THD_value = 20 * Log $\dfrac{\sqrt{\sum_{2}^{N} H_i^{\,2}}}{H_{in(rms)}}$

THD_n_value = $\dfrac{\sqrt{\sum_{2}^{N} H_i^{\,2}}}{\sqrt{\sum_{1}^{N} H_i^{\,2} + \sum_{1}^{M} N_i^{\,2}}}$

SND_value = 20 * Log $\dfrac{H_{in(rms)}}{\sqrt{\sum_{1}^{N} H_i^{\,2} + \sum_{1}^{M} N_i^{\,2}}}$

H_i : *Harmonic*
N_i : *Noise*

4.20.8.2 Example

```
   LOAD_AWG_PAT(WG0,"sin_25v.awi", 16 , RPT, 13);
   SET_WG_WORK_MODE(WG0, SINGLE_END, @RNG3, 0.0V, R50, OFF, ON);

SET_CAPTURE_MEM_MODE(DB_PINS,PEL,ON,64K,1024,LOG_FAIL_ONLY,ON);
   WAIT(Delay);
   CRAFT_STATEMENT
   {
      JUDGE_PAT(sin8_st,sin8_sp);
      PLAN_RESULT_MS[0]=PASS;
      for(i=0; i<1024; i++)
      {
            READ_CAPTURE_MEM(S0,DB_PINS,i,pel);
            printf("%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d\n", pel[0], pel[1],
pel[2], pel[3], pel[4], pel[5], pel[6], pel[7], pel[8], pel[9], pel[10], pel[11], pel[12], pel[13],
pel[14], pel[15]);
            data_f[i]=0;
            for(j=0,w=15; w>=0; w--,j++)
            {
                  data_f[i] = data_f[i] | (pel[j]<<w);
            }
            data_val[i] = ((double)(data_f[i] ^ 0x8000) - 32768.0) / 32768.0 * 10.0;
      }
        AWI_DATA_AC_ARITH(data_val,1024, thd, snr, snd, sfdr,thd_n);
```

4.20.8.3 Relative

LOAD_AWG_PAT , SET_CAPTURE_MEM_MODE , READ_CAPTURE_MEM

4.20.9 AWI_DATA_DC_ARITH

AWI_DATA_DC_ARITH- CRAFT statement
This function is designed to calculate AWI static characteric data (INL/DNL/Gain Error/Offset Error).

4.20.9.1 Format

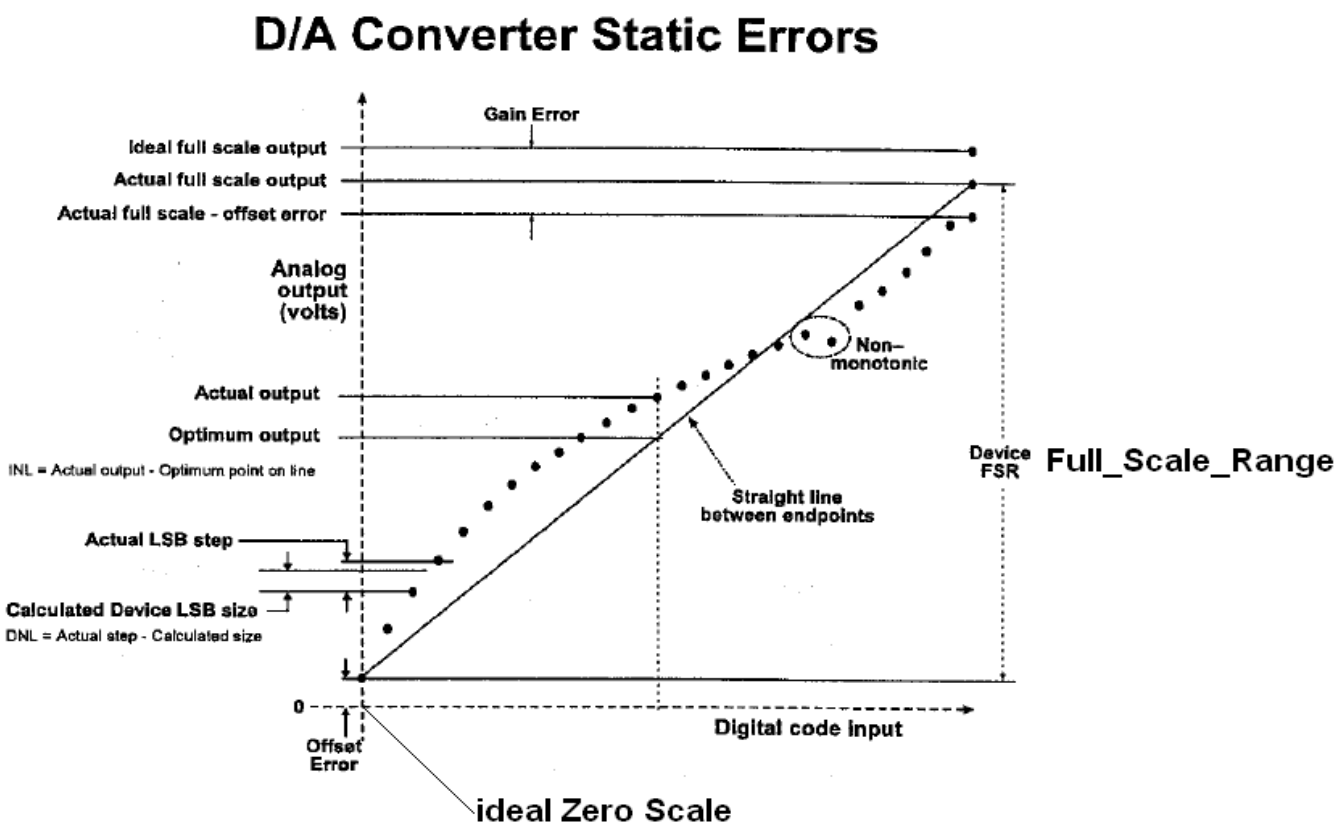
Parameter list:

```
AWI_DATA_DC_ARITH(T_data, T_data_cnt, ideal_zero_scale_value, Full_Scale_Range, INL_value,
                  DNL_value, gain_error, offset_error ) ;
```

Parameter comment:

T_data : Time domain data, //input
Type : double array
T_data_cnt : data count //input
ideal_zero_scale_value : type double //input
Full_Scale_Range : type double //input

INL_value : //return double array
DNL_value : //return double array
gain_error : type double
offset_error : type double



Caculation Equation :

```
// n = T_data_cnt ;
Offset_error = T_data[0] - ideal_zero_scale_value ;
Gain_error = (T_data[n-1] - T_data[0]) - Full_Scale_Range ;
```

$$\text{DeviceLSB} = \frac{V[n-1] - V[0]}{n-1}$$ **Where: n is the total data count**

$$\text{DNL}(i) = \frac{V[i+1] - V[i]}{\text{DeviceLSB}} - 1.0$$ **Where: i = 0,1,2,.., n-2**

$$\text{INL}(i) = \frac{V[i] - (\text{DeviceLSB} * i + V[0])}{\text{DeviceLSB}}$$ **Where: i = 0,1,2,.., n-1**

4.20.9.2 Example

```
SET_WD_WORK_MODE(WD0, @RNG3, 0.0V, OFF, OFF, OFF, ON);
SET_WD_WAVEFORM_SET(WD0, WD_waveform_set, 10.0e-6, 9.0e-6, 0x0,
log_cnt_temp);
JUDGE_PAT(ramp1_st,ramp1_sp);
READ_WD_DATA(S0, WD0, WD_waveform_set, log_cnt_temp, data_tmp);
AWI_DATA_DC_ARITH(data_tmp, 4096, -1*(DAC_FSR/2), DAC_FSR, Inl,
                  Dnl, gain_error, offset_error ) ;
```

4.20.9.3 Relative

SET_WD_WAVERORM_SET , READ_WD_DATA , AWI_DATA_DC_ARITH

SET_WD_WORK_MODE

4.20.10 AWI_SPECTRUM

AWI_SPECTRUM -- CRAFT function

The function is designed to transfer time-domain to freq-domain .

4.20.10.1 Format

Parameter list:

```
AWI_SPECTRUM(T_data, point_count, F_data [ ,weighting_window ]);
```

Parameter comment:

T_data : Time domain data

Type : double array

Point_count : must be 2^n

F_data : Frequency domain $|Amplitude|^2$

Type : double array, array size = point_count.

Weighting_window : HAN (Hanning window, default)

HAM (Hamming window)

REC (Rectangular window)

4.20.10.2 Example

```
double T_data[1024],F_data[1024];

AWI_SPECTRUM(T_data,1024,F_data);
```

4.20.11 WG_PAT_STOP

WG_PAT_STOP– CRAFT statement

stop WG waveform output in WG WORK MODE_B(continue mode)

4.20.11.1 Format

Parameter list:

```
WG_PAT_STOP( wg_name ) ;
```

Parameter comment:
wg_name : pin_name or pin_group (pin_type = WG)

4.20.11.2 Example

```
LOAD_AWG_PAT(WG0,"sin_25v.awi", 16 , RPT, 13);
SET_WG_WORK_MODE(WG0, SINGLE_END, @RNG3, 0.0V, R50, OFF,
ON,MODE_B);
WAIT(Delay);
JUDGE_PAT(sin8_st,sin8_sp);
WG_PAT_STOP(WG0);
```

Use the statement to stop WG output waveform in MODE_B

4.20.11.3 Relative

LOAD_AWG_PAT , SET_WG_WORK_MODE

4.20.12 LOAD_FOCUS_CAL_FILE

LOAD_FOCUS_CAL_FILE – CRAFT statement
Load the focus calibration data file into CRAFT .
TDR release version : **CRAFT_3380_2.00**

4.20.12.1 Format

Parameter list:

```
LOAD_FOCUS_CAL_FILE( ID, " cal_file_name " [    , "load_board_id" ]) ;
```

Parameter comment:

ID : LXAWI
NI4461
TDR : AC Load Board calibration. If user chooses TDR that we must fill load_board_id in statement. The other is not necessary.
Eg: LXAWI & NI4461.
TDR_ADV : Advanced AC Load Board calibration

**no_entry: illegal

[Notice]: 1. **TDR** and **TDR ADV** need to use the tool to generate TDR
2. Before we use **TDR** tool, need to follow next lists.

I. The position of LOAD_FOCUS_CAL_FILE is before MARKERN.
II. User could put LOAD_FOCUS_CAL_FILE in START_UP() first.
III. If need use Logic Waveform, please set up "Programming mode".

3. The difference between TDR and TDR_ADV
TDR : Remove common-trace-delay. Removing common trace delay. will make the DUT output pin of a strobe timing displacement value
TDR_ADV : Keep common-trace-delay. If the load board trace and delay is too long. We need to check Timing Dead-Zoom (2T-10nS).

cal_file_name : max 64 chars
**no_entry: illegal

load_board_id : max 64 chars

1. User can bypass some char by "?". e.g. "3380-IO512-0102-RT" could be "3380-IO512-????-??"
2. The load_board_id must fully match load_board_id in focus-cal-file.
3. Otherwise system will be error and exit test program in run-time.

**no_entry: illegal

4.20.12.2 Example

```
LOAD_FOCUS_CAL_FILE (LXAWI, ". / dbs / focus_cal_file.txt" );
LOAD_FOCUS_CAL_FILE (LXAWI, ".\\ dbs \\ focus_cal_file.txt" );
                    /*because "\" is escape sign in c-language*/
```

```
START_UP()
{
    LOAD_FOCUS_CAL_FILE (TDR, ". / focus_cal_file.txt" ," 3380-IO512-0102-RT");
    RATEN(TS1, 1000nS);
    MARKERN(TS1, IO_ALLPINS,0nS,10nS, ,0nS,10nS,900nS,1000nS);
}
```

4.20.13 AWI_SPECTRUM_EX

The function is designed to transfer time-domain to freq-domain specially for None 2^n data points. Release version : **CRAFT_3380_1.30**

4.20.13.1 Format

Parameter list:

```
AWI_SPECTRUM_EX(T_data, point_count, F_data, transfer_mode [ ,weighting_window ] );
```

Parameter comment:

T_data : Time domain data

Type : double array

Point_count : 2^n → use FFT

None 2^n → must use DFT

F_data : Frequency domain $|Amplitude|^2$

Type : double array, array size = point_count.

Weighting_window : HAN (Hanning window, default)

HAM (Hamming window)

REC (Rectangular window)

Transfer_mode : FFT

DFT (None 2^n point count must assign this mode)

**no_entry: illegal

4.20.13.2 Example

```
CARRAY_DIM ouble T_data[1000],F_data[1000];
AWI_SPECTRUM_EX(T_data,1000,F_data, HAN, DFT);
```

4.20.14 AWI_DATA_AC_ARITH_EX

This function is designed to calculate AWI dynamic characteric data specially for None 2^n points. Release version : **CRAFT_3380_1.30**

4.20.14.1 Format

Parameter list:

AWI_DATA_AC_ARITH_EX(T_data, point_count, THD_value, SNR_value, SND_value, SFDR_value, THD_n_value, transfer_mode [, FBin_st, FBin_sp] [, DC_value, SIGNAL_value, HARMONIC_value, Noise_value, weighting_window] [, main_lobe_width, side_lobe_width]) ;

Parameter comment:

T_data : Time domain data,
Type : double array
Point_count : effective point number, 2^n → use FFT
None 2^n → must use DFT
THD_value : Total Harmonic Distorsion (unit: db)
SNR _value : Signal to Noise Ratio (unit: db)
SND _value : Signal to noise plus Distortion (unit: db)
SFDR _value : Spurious Free Dynamic Range (unit: db)
THD_n_value : %THD+n (unit: %)
Transfer_mode : FFT
DFT (None 2^n point count must assign this mode)
**no_entry: illegal

//20190125:
FBin_st : type integer
**no_entry: Default = 1
variables & constants allowed

FBin_sp : type integer
**no_entry: Default = Last_FBin (calculated from T_data)
variables & constants allowed

If parameters, FBIN_st and FBIN_sp are filled in , the returen values("DC_value"," SIGNAL_value"," HARMONIC_value"," Noise_value" and "SFDR_value") will be fixed to correct one.

Formula Difference :

✧ DC_value

Without FBin_st & FBin_sp : get the average value in frequency domain
With FBin_st & FBin_sp : get the average value in time domain

✧ SIGNAL_value、HARMONIC_value、Noise_value:

Without FBin_st & FBin_sp : After FFT，normalize FFT result。
With FBin_st & FBin_sp : Return back to VRMS from FFT and then do the normalization.

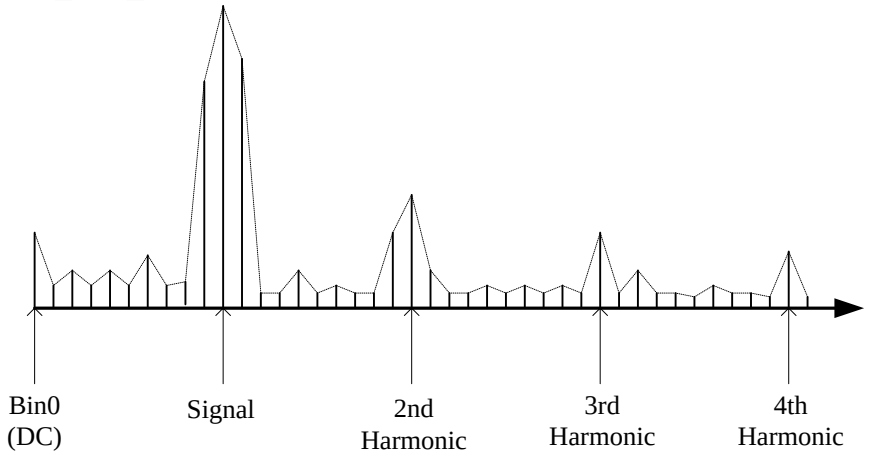
✧ SFDR_value

Without FBin_st & FBin_sp : 20 $\boldsymbol{log_{10}} \frac{除主bin外的Max\ \ bin}{主bin}$

With FBin_st & FBin_sp : 10 $\boldsymbol{log_{10}} \frac{主bin}{除主bin外的Max\ \ bin}$

//----------

DC_value : (unit: Vrms)
SIGNAL_value : (unit: Vrms)
HARMONIC_value : (unit: Vrms)
Noise_value : (unit: Vrms)
weighting_window : HAN (Hanning window)
HAM (Hamming window)
REC (Rectangular window, default)
main_lobe_width : default 2
side_lobe_width : default 1



if main lobe width = 0, then signal = bin10 ;
if main lobe width = 1, then signal = bin9+bin10+bin11 ;

if side lobe width = 0, then 2nd-harmonic = bin20 ;
if side lobe width = 1, then 2nd-harmonic = bin19+bin10+bin21 ;

Caculation Equation :

$$\text{Signal_value} = H_{in(rms)}$$

$$\text{THD_value} = 20 * \text{Log}\frac{\sqrt{\sum_{2}^{N} H_i^{\,2}}}{H_{in(rms)}}$$

$$\text{THD_n_value} = \frac{\sqrt{\sum_{2}^{N} H_i^{\,2}}}{\sqrt{\sum_{1}^{N} H_i^{\,2} + \sum_{1}^{M} N_i^{\,2}}}$$

$$\text{SND_value} = 20 * \text{Log}\frac{H_{in(rms)}}{\sqrt{\sum_{1}^{N} H_i^{\,2} + \sum_{1}^{M} N_i^{\,2}}}$$

H_i : *Harmonic*

N_i : *Noise*

4.20.14.2 Example

```
CARRAY_DIM double   data[1024];   //

CARRAY_DIM   double   thd,snr,snd,sfdr,thd_n;

   CARRAY_DIM   double   dc, signal,harmonic,noise;

   AWI_DATA_AC_ARITH_EX(data,1000, thd, snr, snd, sfdr, thd_n, DFT);

   AWI_DATA_AC_ARITH_EX(data,1000, thd, snr, snd, sfdr, thd_n, DFT, dc,
signal, harmonic, noise, REC, 2,1);
```

4.21 Capture Memory

4.21.1 SET_CAPTURE_MEM_MODE

4.21.1.1 Explanation

Set capture memory log mode. When user set the capture mode, the original log function(capture PEL/PEH/PFAIL in the same time) will be disabled.

4.21.1.2 Format

Parameter list:

SET_CAPTURE_MEM_MODE(pin_name, log_src, ON/OFF ,log_size_rng [, datalog_display_count][,log_mode, turn_over_mode]);

Parameter comment:

pin_name : pin_def, pin_group (assign which pins will capture the log data)
**no_entry: illegal

log_src : PEL : log PEL data
PEH : log PEH data
PFAIL : log PinFail data
PEL_K : log PEL data & controlled by pattern symbol "**K**"
PEH_K : log PEH data & controlled by pattern symbol "**K**"
**no_entry : illegal

ON/OFF : ON : turn on the capture mode.
OFF : turn off the capture mode.
**no_entry : illegal

log_size_rng : (Maximum log size)
8K (any pin can be capture pin)→ log size maximum 8K
16K (each 8pin block maximum 4 capture pin)→ log size maximum 16K
32K (each 8pin block only 2 capture pin)→ log size maximum 32K
64K (each 8pin block only 1 capture pin)→ log size maximum 64K
**no_entry: illegal ;
(driver will check pin_name if match log_size)

log_mode : LOG_CONT : log continue mode
LOG_FAIL_ONLY : log fail only mode (single site / multi site)
LOG_SITE_FAIL_ONLY : log site fail only mode
**no_entry : default is mode ON→LOG_FAIL_ONLY,
When mode OFF→ LOG_SITE_FAIL_ONLY ;

datalog_display_count : Datalog Utility display count value, 1 to 65535
**no_entry: default

when mode ON → display All HW log data,
when mode OFF → 1 ;

turn_over_mode : ON : if log memory pointer already move to last address, then it will turn over and move to address 0 in next log.
OFF : if log memory pointer already move to last address, then it will keep the same address (last address) in next log.
**no_entry: default
when mode ON → ON,
when mode OFF → OFF;

4.21.1.3 Example

```
  LOAD_AWG_PAT(WG0,"sin_25v.awi", 16 , RPT, 13);
  SET_WG_WORK_MODE(WG0, SINGLE_END, @RNG3, 0.0V, R50, OFF, ON);

SET_CAPTURE_MEM_MODE(DB_PINS,PEL,ON,64K,1024,LOG_FAIL_ONLY,ON);
  WAIT(Delay);
  CRAFT_STATEMENT
  {
    JUDGE_PAT(sin8_st,sin8_sp);
    PLAN_RESULT_MS[0]=PASS;
    for(i=0; i<1024; i++)
    {
        READ_CAPTURE_MEM(S0,DB_PINS,i,pel);
        data_f[i]=0;
        for(j=0,w=15; w>=0; w--,j++)
        {
            data_f[i] = data_f[i] | (pel[j]<<w);
        }
        data_val[i] = ((double)(data_f[i] ^ 0x8000) - 32768.0) / 32768.0 * 10.0;
    }
     AWI_DATA_AC_ARITH(data_val,1024, thd, snr, snd, sfdr,thd_n);
```

4.21.1.4 NOTE

1 When using captures memory and set "**log fail only"** mode, **any fail happened** will be log to memory.

Because normal pin fail happened will cause capture pin to get wrong data.
For the reason suggest comparing capture pin **only**.

2. Must be sure capture memory setting is **correct** (for example: "**log size**").

Example (Incorrect example):

Capture Pin.
Set log fail memory “8K”, and log memory is full.

```
376977  *011LLLLLLLLXXXXXXXXXXXXXXXXOOOOXXXXLLLL1XOXXXO10 * TS1 ;
376978  *011LLLLLLLLXXXXXXXXXXXXXXXXOOOOXXXXLLLL1XOXXXO10 * RPT  212  ;
376979  *011LLLLLLLLZZZZZZZZZZZZZZZZOOOOXXXXLLLL1XOXXXO10 *   ;
376980  *011LLLLLLLLXXXXXXXXXXXXXXXXOOOOXXXXLLLL1XOXXXO10 *   ;
376981  *011LLLLLLLLXXXXXXXXXXXXXXXXOOOOXXXXLLLL1XOXXXO10 *   ;
376982  *011LLLLLLLLXXXXXXXXXXXXXXXXOOOOXXXXLLLL1XOXXXO10 *   ;
376983  *011LLLLLLLLXXXXXXXXXXXXXXXXOOOOXXXXLLLL1XOXXXO10 *   ;
376984  *011LLLLLLLLXXXXXXXXXXXXXXXXOOOOXXXXLLLL1XOXXXO10 *   ;
376985  *011LLLLLLLLXXXXXXXXXXXXXXXXOOOOXXXXLLLL1XOXXXO10 *   ;
376986  *011LLLLLLLLXXXXXXXXXXXXXXXXOOOOXXXXLLLL1XOXXXO10 *   ;
376987  *011LLLLLLLLXXXXXXXXXXXXXXXXOOOOXXXXLLLL1XOXXXO10 *   ;
376988  *011LLLLLLLLXXXXXXXXXXXXXXXXOOOOXXXXLLLL1XOXXXO10 *   ;
376989  *011LLLLLLLLXXXXXXXXXXXXXXXXOOOOXXXXLLLL1XOXXXO10 *   ;
376990  *011LLLLLLLLHHHHHHHHHXLLLLLHHOOOOXXXXLLLL1XOXXXO10 *   ;
376991  *011LLLLLLLLHHHHHHHHHXLLLLLHHOOOOXXXXLLLL1XOXXXO10 *   ;
376992  *011LLLLLLLLHHHHHHHHHXLLLLLHHOOOOXXXXLLLL1XOXXXO10 *   ;
376993  *011LLLLLLLLHHHHHHHHHXLLLLLHHOOOOXXXXLLLL1XOXXXO10 *   ;
376994  *011LLLLLLLLHHHHHHHHHXLLLLLHHOOOOXXXXLLLL1XOXXXO10 * STOP  ;
376995  *011LLLLLLLLHHHHHHHHHXLLLLLHHOOOOXXXXLLLL1XOXXXO10 *   ;
376996  *011LLLLLLLLHHHHHHHHHXLLLLLHHOOOOXXXXLLLL1XOXXXO10 *   ;
```

Normal Pin

If fail happened will cause log memory overflow.

4.21.1.5 Relative

READ_CAPTURE_MEM

4.21.2 READ_CAPTURE_MEM

4.21.2.1 Explanation

Read IO capture memory.

4.21.2.2 Format

Parameter list:

```
READ_CAPTURE_MEM(socket _no, pin_name, addr, data) ;
```

Parameter comment:

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

pin_name; pin_def, pin_group,
(only Capture-Mode pins, otherwise the data will be wrong; //Driver will check pin name)
**no_entry: illegal

addr: log index : 0 ~ 8K/16K/32K/64K (depend on SET_CAPTURE_MEM_MODE setting)
type: integer
**no_entry: illegal

data : return pin log memory,
0/1: pass/fail (if log PFail)
0/1: lower than Vol/higher than Vol(if log PEL)
0/1: higher than Voh/lower than Voh (if log PEH)
type: int array

4.21.2.3 Example

```
 // bus_pin = q0+q1+q2+q3+q4+q5+q6+q7 ;
int data[8];

SET_CAPTURE_MEM_MODE( bus_pin, PEL, ON );
JUDGE_PAT(func_st, func_sp);
SET_CAPTURE_MEM_MODE( bus_pin, , OFF );

READ_CAPTURE_MEM(S0, bus_pin, 0, data);
```

4.21.2.4 Relative

SET_CAPTURE_MEM_MODE, READ_LOG_MEM

4.21.3 READ_CAPTURE_MEM_BLOCK

4.21.3.1 Explanation

Read IO capture memory block.

4.21.3.2 Format

Parameter list:

```
READ_CAPTURE_MEM_BLOCK(socket _no, pin_name, st_addr, sp_addr, data) ;
```

Parameter comment:

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

pin_name; pin_def, pin_group,
(only Capture-Mode pins, otherwise the data will be wrong; //Driver will check pin name)
**no_entry: illegal

addr: log index : 0 ~ 8K/16K/32K/64K (depend on SET_CAPTURE_MEM_MODE setting)
type: integer
**no_entry: illegal

data : return pin log memory,
0/1: pass/fail (if log PFail)
0/1: lower than Vol/higher than Vol(if log PEL)
0/1: higher than Voh/lower than Voh (if log PEH)
type: int array

4.21.3.3 Example

return data[] :

Data [0]	Data [1]		Data [7]	Data [8]	Data [9]		Data [15]	...	Data [79]
Addr 0: p0	Addr 0: p1		Addr 0: p7	Addr 1: p0	Addr 1: p1		Addr 1: p7	...	Addr 9: p7

```
CRAFT_STATEMENT
{
   JUDGE_PAT(sin8_st,sin8_sp);
   PLAN_RESULT_MS[0]=PASS;
   READ_CAPTURE_MEM_BLOCK(S0,DB_PINS, 0 , 1023 , temp);

   for(i=0; i<1024; i++)
   {
         data_f[i]=0;
         for(j=0,w=15; w>=0; w--,j++)
         {
               data_f[i] = data_f[i] | (temp [j]<<w);
         }
         data_val[i] = ((double)(data_f[i] ^ 0x8000) - 32768.0) / 32768.0 * 10.0;
   }
```

4.21.3.4 Relative

SET_CAPTURE_MEM_MODE, READ_CAPTURE_MEM, READ_LOG_MEM

4.21.4 READ_CAPTURE_MEM_BLOCK_X

4.21.4.1 Explanation

Read IO capture memory block.

4.21.4.2 Format

Parameter list:

READ_CAPTURE_MEM_BLOCK_X(socket_no, pin_name, [st_addr, sp_addr] | [total_log_cnt] , data) ;

Parameter comment:

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S31 : socket No. 31
Type : integer
**no_entry: illegal

pin_name; pin_def, pin_group,
(only Capture-Mode pins, otherwise the data will be wrong; //Driver will check pin name)
**no_entry: illegal

st_addr/sp_addr: log index : 0 ~ 8K/16K/32K/64K (depend on SET_CAPTURE_MEM_MODE setting)
type: integer
**no_entry: illegal

total_log_cnt: return how many capture memory data log count after judge pattern
type: integer
**no_entry: illegal

data : return pin log memory,
0/1: pass/fail (if log PFail)
0/1: lower than Vol/higher than Vol(if log PEL)
0/1: higher than Voh/lower than Voh (if log PEH)
type: CUSTOM_2DIM_BOOL
**no_entry: illegal

4.21.4.3 Example

if pin_group = p0+p1+p2;
return data[pin_group][per pin data] :

Data [0][0]	Data [0][1]	…	Data [0][7]	Data [0][8]	Data [0][9]	…	Data [0][15]	…	Data [0][79]
p0 data log 0	p0 data log 1	…	p0 data log 7	p0 data log 8	p0 data log 9	…	p0 data log 15	…	p0 data log 79

Data [1][0]	Data [1][1]	…	Data [1][7]	Data [1][8]	Data [1][9]	…	Data [1][15]	…	Data [1][79]
p1 data log 0	p1 data log 1	…	p1 data log 7	p1 data log 8	p1 data log 9	…	p1 data log 15	…	p1 data log 79

Data [2][0]	Data [2][1]	…	Data [2][7]	Data [2][8]	Data [2][9]	…	Data [2][15]	…	Data [2][79]
p2 data log 0	p2 data log 1	…	p2 data log 7	p2 data log 8	p2 data log 9	…	p2 data log 15	…	p2 data log 79

```
  CUSTOM_2DIM_BOOL return_data;
  CUSTOM_2DIM_BOOL return_data_X;
  Int i,j,total_log;

SET_CAPTURE_MEM_MODE(SI+SO,PEL_K,ON,16K,LOG_FAIL_ONLY,ON,FEOP
_ON);
JUDGE_PAT(sin8_st,sin8_sp);

  SOCKET_INC(FRZ_OFF)
  {
     READ_CAPTURE_MEM_BLOCK_X(SOCKET, SI+SO, 0, 13439, return_data) ;
     READ_CAPTURE_MEM_BLOCK_X(SOCKET, SI+SO, total_log,
return_data_X) ;

     Printf("total_log = %d\n", total_log);
     for(i=0;i<2;i++)
     {
        for(j=0; j< total_log; j++)
        {
           printf("return_data_X [%d][%d] = %d\n",i,j, return_data_X [i][j]);
        }
      }
  }
```

4.21.4.4 Relative

SET_CAPTURE_MEM_MODE, READ_CAPTURE_MEM, READ_LOG_MEM

4.21.5 SET_ECM_MODE

4.21.5.1 Explanation

ECM uses dynamic random-access memory(DRAM), set extended capture memory log mode. The pins set in this statement that those pins are able to use ECM function.ECM would capture the status of ECM pin whenever JUDGE_PAT executed.

4.21.5.2 Format

Parameter list:

SET_ECM_MODE(pin_name, log_src, ON/OFF, Condense_mode);

Parameter comment:

pin_name : pin_def, pin_group (assign which pins will capture the log data)

**no_entry: illegal

log_src : **PEL_K** : log PEL data & controlled by pattern symbol "**K**"

PEH_K : log PEH data & controlled by pattern symbol "**K**"

**no_entry : illegal

// Maximum speed : 50MHz (PEL_K / PEH_K);

ON/OFF : ON : turn on the ECM mode.

OFF : turn off the ECM mode.

**no_entry : illegal

Condense_mode : In fixed pins, choose the condense mode more bigger, the test time will be faster.

X32 (Any 1 pin can be use in each 32 pins block)→ log size maximum 32*16M=512M
//The efficiency of data reading is same as 64K mode in Capture Memory

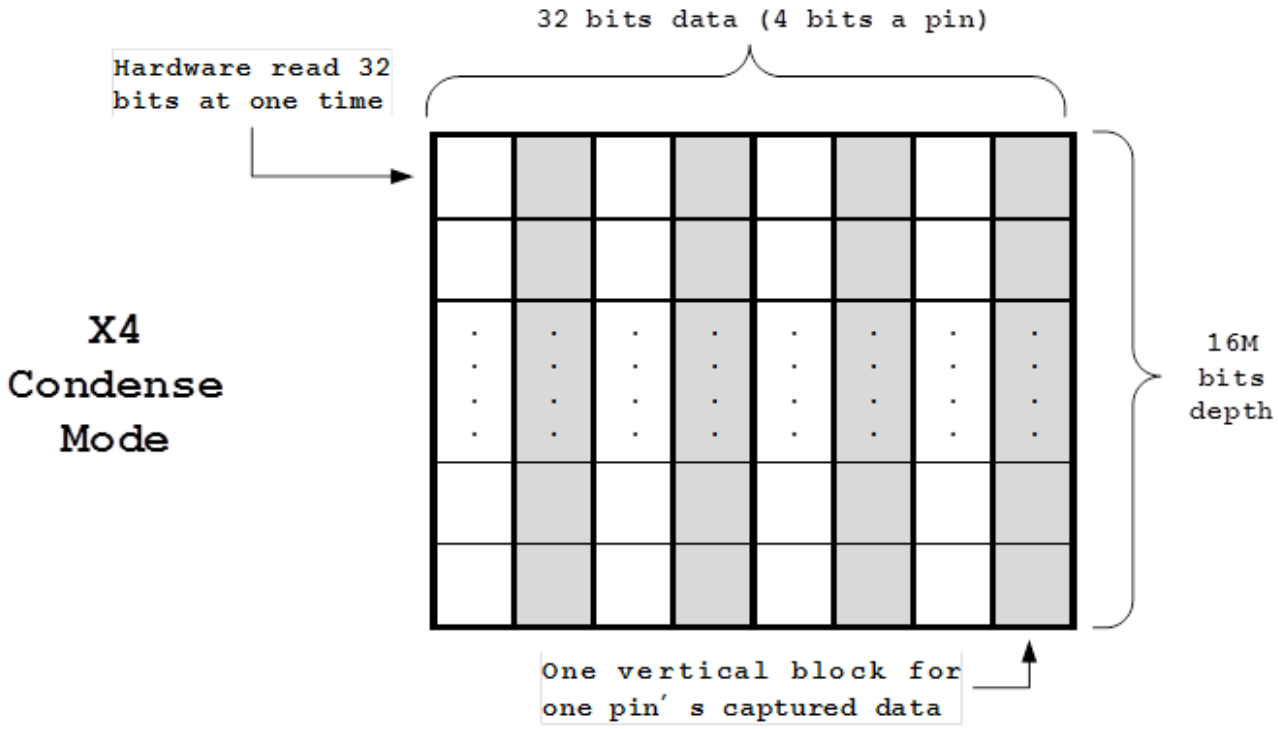
X16 (Any 2 pins can be use in each 32 pins block)→ log size maximum 16*16M=256M
//The efficiency of data reading is same as 32K mode in Capture Memory

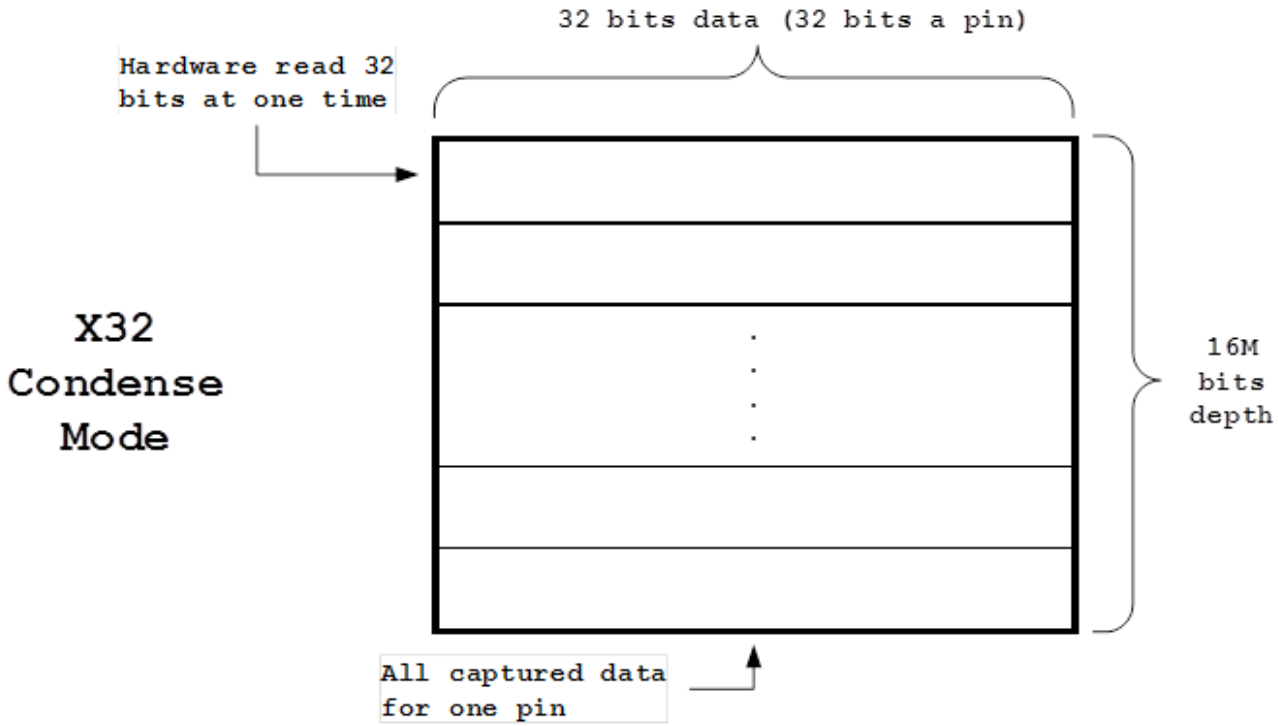
X8 (Any 4 pins can be use in each 32 pins block)→ log size maximum 8*16M=128M
//The efficiency of data reading is same as 16K mode in Capture Memory

X4 (Any 8 pins can be use in each 32 pins block)→ log size maximum 4*16M=64M
//The efficiency of data reading is same as 8K mode in Capture Memory

X1 (per-pin 1-bit; any pin can be capture pin)→ log size maximum 16M

**no_entry: illegal ;





4.21.5.3 Example

```
CUSTOM_3DIM_BOOL    Data_simple;

ecm_func_simple_test()
{
      DATALOG_MSG="LS299_ecm_func_test";
      TEST_NO(900);

      FORCE_VOLT( LS299_Vdd, MLDPS,_FORCE_V, 5.0V, _V_RNG, @6V,
_I_RNG, @500mA, _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY, ON,_DELAY,
3mS);

      SET_LEVELN(func_lvl, LS299_IO_PINS, 0V, 5V, 1V, 3V, 0uA, 0uA, 2V, , );
      LEVELS(func_lvl, 5mS);

      SET_JUDGE_MODE(NORM);
      RATEN(TS2, 1uS);
      MARKERN(TS2, CLK,250nS,750nS, ,0nS,0nS,850nS,850nS);
      MARKERN(TS2, CTRL,0nS,100nS, ,0nS,0nS,850nS,850nS);
      MARKERN(TS2, OUTPUTS,0nS,25nS, ,0nS,0nS,950nS,950nS);

      PIN_MODE(LS299_IO_PINS, NRZ, EDGE, ENABLE, IO_NRZ, NORM);
      PIN_MODE(CLK, RZ, EDGE, MASK, IO_NRZ, OFF);
      RELAY_ON(LS299_IO_PINS, PDCL, 3mS);

      SET_ECM_MODE( OUTPUTS, PEL_K , ON, X1);       // Setting ecm on
      SET_DATALOG_MODE( LOG_CONT, 64, ON);

      JUDGE_PAT( SIMPLE_TEST_64DATAS, __SIMPLE_TEST_64DATAS);

      READ_ECM_BLOCK( OUTPUTS, 0, 63, Data_simple);        // To get captured
datas in array for analysis
      PRINT_ECM_DATA( OUTPUTS, 0, 63);                     // To show datas in
datalog directly
      PRINT_ECM_DATA( OUTPUTS, 0, 63,
"D:\\3380D_DemoBrd_for_ecm_1site\\ReadData\\Read_x1_PEL_K(SIMPLE_TEST_
64DATAS).txt"); // To Show datas in text file

      SET_ECM_MODE( OUTPUTS, PEL_K, OFF, X1 );     // Setting ecm off

}// End of ecm_func_simple_test()
```

ECM Pattern file

```
SET_DEC_FILE "Tutorial_pins.dec"

HEADER       CLR,%SEL0,SEL1,%G1,G2,%CLK,%SL,SR,%QA,QB,QC,
             QD,QE,QF,QG,QH;


SPM_PATTERN (SIMPLE_TEST_64DATAS) {

                 *0 00 00 0 00 XXXXXXXX *TS2;
                 *0 11 00 0 00 XXXXXXXX *;
                 *0 11 00 0 00 XXXXXXXX *;
```

```
            *0 11 00 0 00 XXXXXXXX *;

            *0 11 00 0 00 XXXXXXXX *;//SETX0 125;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;

x1_k_0:     *1 11 11 1 00 10001000 *;//-----load
            *1 11 11 1 00 10001000 *;
            *1 11 11 1 00 10001000 *;
            *1 11 11 1 00 10001000 *;
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HLLLHLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HLLLHLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HLLLHLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HLLLHLLL

            *1 11 11 1 00 01000100 *;//-----load
            *1 11 11 1 00 01000100 *;
            *1 11 11 1 00 01000100 *;
            *1 11 11 1 00 01000100 *;
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LHLLLHLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LHLLLHLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LHLLLHLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LHLLLHLL

            *1 11 11 1 00 00100010 *;//-----load
            *1 11 11 1 00 00100010 *;
            *1 11 11 1 00 00100010 *;
            *1 11 11 1 00 00100010 *;
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLHLLLHL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLHLLLHL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLHLLLHL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLHLLLHL

            *1 11 11 1 00 00010001 *;//-----load
            *1 11 11 1 00 00010001 *;
            *1 11 11 1 00 00010001 *;
            *1 11 11 1 00 00010001 *;
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLHLLLH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLHLLLH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLHLLLH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLHLLLH

            *0 11 00 0 00 XXXXXXXX *;//JNZ0 x1_k_0;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;//SETX1 125;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;
                                    //---------------- Second Range
x1_k_1:     *1 11 11 1 00 11001100 *;//-----load
            *1 11 11 1 00 11001100 *;
            *1 11 11 1 00 11001100 *;
```

```
            *1 11 11 1 00 11001100 *;
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHLLHHLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHLLHHLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHLLHHLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHLLHHLL

            *1 11 11 1 00 00110011 *;//-----load
            *1 11 11 1 00 00110011 *;
            *1 11 11 1 00 00110011 *;
            *1 11 11 1 00 00110011 *;
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLHHLLHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLHHLLHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLHHLLHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLHHLLHH

            *1 11 11 1 00 11001100 *;//-----load
            *1 11 11 1 00 11001100 *;
            *1 11 11 1 00 11001100 *;
            *1 11 11 1 00 11001100 *;
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHLLHHLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHLLHHLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHLLHHLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHLLHHLL

            *1 11 11 1 00 00110011 *;//-----load
            *1 11 11 1 00 00110011 *;
            *1 11 11 1 00 00110011 *;
            *1 11 11 1 00 00110011 *;
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLHHLLHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLHHLLHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLHHLLHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLHHLLHH

            *0 11 00 0 00 XXXXXXXX *;//JNZ1 x1_k_1;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;//SETX2 125;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;
                                    //---------------- Third Range
x1_k_2:     *1 11 11 1 00 11111111 *;//-----load
            *1 11 11 1 00 11111111 *;
            *1 11 11 1 00 11111111 *;
            *1 11 11 1 00 11111111 *;
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
```

```
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    HHHHHHHH

            *0 11 00 0 00 XXXXXXXX *;//JNZ2 x1_k_2;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;//SETX3 125;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;

x1_k_3:     *1 11 11 1 00 00000000 *;//-----load
            *1 11 11 1 00 00000000 *;
            *1 11 11 1 00 00000000 *;
            *1 11 11 1 00 00000000 *;
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL
            *1 00 00 0 00 KKKKKKKK *;//-----hold          //    LLLLLLLL

            *0 11 00 0 00 XXXXXXXX *;//JNZ3 x1_k_3;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;
            *0 11 00 0 00 XXXXXXXX *;

            *0 00 00 0 00 LLLLLLLL *;
            *0 00 00 0 00 LLLLLLLL *;
            *0 00 00 0 00 LLLLLLLL *;
            *0 00 00 0 00 LLLLLLLL *;
}
```

4.21.6 READ_ECM_BLOCK

4.21.6.1 Explanation

Read IO extended capture memory block. Only the pins had set in ECM mode can be read.

4.21.6.2 Format

Parameter list:

READ_ECM_BLOCK(pin_name, st_idx, sp_idx, return_ecm_data) ;

Parameter comment:

pin_name; pin_def, pin_group

(only ECM-Mode pins in SET_ECM_MODE, otherwise the data will be wrong; Driver will check pin name)

**no_entry: illegal

st_idx : (start_index)

based on selected condense mode,the last 1K momery size is prohibition. e.g.: 16M use 0~(16M-1K) ; 64M use 0~(64M-1K)…,etc.

X1: 0 ~ 16M

X4: 0 ~ 64M

X8: 0 ~ 128M

X16: 0 ~ 256M

X32: 0 ~ 512M

type: **integer**

**no_entry: illegal

sp_idx : (stop_index)

based on selected condense mode,the last 1K momery size is prohibition. e.g.: 16M use 0~(16M-1K) ; 64M use 0~(64M-1K)…,etc.

X1: 0 ~ 16M

X4: 0 ~ 64M

X8: 0 ~ 128M

X16: 0 ~ 256M

X32: 0 ~ 512M

type: **integer**

**no_entry: illegal

return_ecm_data :

type: **CUSTOM_3DIM_BOOL** ;

Maximum array size is 128M

// CUSTOM_3DIM_BOOL: CRAFT default 3 dimension bool array,

// CRAFT auto-allocate array size [site_no][pin_count][cycle_cnt];

// cycle_cnt = read_length;

// site_no*pin_count*cycle_cnt must less then 128M

**no_entry: illegal

```
//CASE 1 :
// About CUSTOM_3DIM_BOOL
// 2 site DUT
// data_bus = p0+p1+p2+p3+p4+p5+p6+p7;
// 10 data per pin
    CUSTOM_3DIM_BOOL   return_ecm_data;                    //declare return data
    READ_ECM_BLOCK( data_bus, 0, 9, return_ecm_data) ;   //read from index 0, index 9

                Site 0 p0 data : return_ecm_data [0][0][0~9]
                Site 0 p1 data : return_ecm_data [0][1][0~9]
                Site 0 p2 data : return_ecm_data [0][2][0~9]
                Site 0 p3 data : return_ecm_data [0][3][0~9]
                Site 0 p4 data : return_ecm_data [0][4][0~9]
                Site 0 p5 data : return_ecm_data [0][5][0~9]
```

```
                    Site 0 p6 data : return_ecm_data [0][6][0~9]

                    Site 0 p7 data : return_ecm_data [0][7][0~9]


                    Site 1 p0 data : return_ecm_data [1][0][0~9]

                        …                …

                    Site-1 p7 data : return_ecm_data [1][7][0~9]


  //CASE 2 :

  // About CUSTOM_3DIM_BOOL and returm_ecm_data

  // 4 site DUT

  // data_bus = p0+p32;

      CUSTOM_3DIM_BOOL    return_ecm_data;                //declare  return
data

  //a. at X1 mode read 16M data
     READ_ECM_BLOCK( data_bus, 0, 16M, return_ecm_data) ;

  //b. at X4 mode read 64M data
     for(int i=0;i<4;i++)
     {
         log_addr_offset = i*16M;
         READ_ECM_BLOCK(data_bus, 0+log_addr_offset,
                                 16M+log_addr_offset, return_ecm_data) ;
     }

  //c. at X8 mode read 128M data
     for(int i=0;i<8;i++)
     {
         log_addr_offset = i*16M;
         READ_ECM_BLOCK(data_bus, 0+log_addr_offset,
                                 16M+log_addr_offset, return_ecm_data) ;
     }

  //d. at X16 mode read 256M data
     for(int i=0;i<16;i++)
     {
         log_addr_offset = i*16M;
         READ_ECM_BLOCK(data_bus, 0+log_addr_offset,
                                 16M+log_addr_offset, return_ecm_data) ;
     }

  //e. at X32 mode read 512M data
     for(int i=0;i<32;i++)
```

```
    {
        log_addr_offset = i*16M;
        READ_ECM_BLOCK(data_bus, 0+log_addr_offset,
                                    16M+log_addr_offset, return_ecm_data) ;
    }
```

4.21.6.3 Example:

```
CUSTOM_3DIM_BOOL   Data_simple;

ecm_func_simple_test()
{
    DATALOG_MSG="LS299_ecm_func_test";
    TEST_NO(900);

    FORCE_VOLT( LS299_Vdd, MLDPS,_FORCE_V, 5.0V, _V_RNG, @6V,
_I_RNG, @500mA, _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY, ON,_DELAY,
3mS);

    SET_LEVELN(func_lvl, LS299_IO_PINS, 0V, 5V, 1V, 3V, 0uA, 0uA, 2V, , );
    LEVELS(func_lvl, 5mS);

    SET_JUDGE_MODE(NORM);
    RATEN(TS2, 1uS);
    MARKERN(TS2, CLK,250nS,750nS, ,0nS,0nS,850nS,850nS);
    MARKERN(TS2, CTRL,0nS,100nS, ,0nS,0nS,850nS,850nS);
    MARKERN(TS2, OUTPUTS,0nS,25nS, ,0nS,0nS,950nS,950nS);

    PIN_MODE(LS299_IO_PINS, NRZ, EDGE, ENABLE, IO_NRZ, NORM);
    PIN_MODE(CLK, RZ, EDGE, MASK, IO_NRZ, OFF);
    RELAY_ON(LS299_IO_PINS, PDCL, 3mS);

    SET_ECM_MODE( OUTPUTS, PEL_K , ON, X1);      // Setting ecm on
    SET_DATALOG_MODE( LOG_CONT, 64, ON);

    JUDGE_PAT( SIMPLE_TEST_64DATAS, __SIMPLE_TEST_64DATAS);

    READ_ECM_BLOCK( OUTPUTS, 0, 63, Data_simple);        // To get captured
datas in array for analysis
    PRINT_ECM_DATA( OUTPUTS, 0, 63);                     // To show datas in
datalog directly
    PRINT_ECM_DATA( OUTPUTS, 0, 63,
"D:\\3380D_DemoBrd_for_ecm_1site\\ReadData\\Read_x1_PEL_K(SIMPLE_TEST_
64DATAS).txt"); // To Show datas in text file

    SET_ECM_MODE( OUTPUTS, PEL_K, OFF, X1 );     // Setting ecm off

}// End of ecm_func_simple_test()
```

ECM Pattern file

4.21.7 PRINT_ECM_DATA

4.21.7.1 Explanation

Output IO extended capture memory data to Datalog-window or save to file. Only the pins had set in ECM mode can be print. Datalog window maximum count 1024,but user could output to file, then skip the maximum limitation of datalog.

4.21.7.2 Format

Parameter list:

```
PRINT_ECM_DATA( pin_name, st_idx, sp_idx [, file_name]) ;
```

Parameter comment:

pin_name: pin_def, pin_group

(only ECM-Mode pins in SET_ECM_MODE, otherwise the data will be wrong; Driver will check pin name)

**no_entry: illegal

st_idx : (start_index)

based on selected condense mode,the last 1K momery size is prohibition. e.g.: 16M use 0~(16M-1K) ; 64M use 0~(64M-1K)…,etc.

X1: 0 ~ 16M

X4: 0 ~ 64M

X8: 0 ~ 128M

X16: 0 ~ 256M

X32: 0 ~ 512M

type: **integer**

**no_entry: illegal

sp_idx : (stop_index)

based on selected condense mode,the last 1K momery size is prohibition. e.g.: 16M use 0~(16M-1K) ; 64M use 0~(64M-1K)…,etc.

X1: 0 ~ 16M

X4: 0 ~ 64M

X8: 0 ~ 128M

X16: 0 ~ 256M

X32: 0 ~ 512M

type: **integer**

**no_entry: illegal

file_name : The file path that going to create file and store data

**no_entry: output data to datalog To output to datalog window maximum count 1024, but if output to file, then skip the maximum limitation.

4.21.7.3 Example

```
CUSTOM_3DIM_BOOL    Data_simple;

ecm_func_simple_test()
{
      DATALOG_MSG="LS299_ecm_func_test";
      TEST_NO(900);

      FORCE_VOLT( LS299_Vdd, MLDPS,_FORCE_V, 5.0V, _V_RNG, @6V,
_I_RNG, @500mA, _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY, ON,_DELAY,
3mS);

      SET_LEVELN(func_lvl, LS299_IO_PINS, 0V, 5V, 1V, 3V, 0uA, 0uA, 2V, , );
      LEVELS(func_lvl, 5mS);

      SET_JUDGE_MODE(NORM);
      RATEN(TS2, 1uS);
      MARKERN(TS2, CLK,250nS,750nS, ,0nS,0nS,850nS,850nS);
      MARKERN(TS2, CTRL,0nS,100nS, ,0nS,0nS,850nS,850nS);
      MARKERN(TS2, OUTPUTS,0nS,25nS, ,0nS,0nS,950nS,950nS);

      PIN_MODE(LS299_IO_PINS, NRZ, EDGE, ENABLE, IO_NRZ, NORM);
      PIN_MODE(CLK, RZ, EDGE, MASK, IO_NRZ, OFF);
      RELAY_ON(LS299_IO_PINS, PDCL, 3mS);

      SET_ECM_MODE( OUTPUTS, PEL_K , ON, X1);       // Setting ecm on
      SET_DATALOG_MODE( LOG_CONT, 64, ON);

      JUDGE_PAT( SIMPLE_TEST_64DATAS, __SIMPLE_TEST_64DATAS);

      READ_ECM_BLOCK( OUTPUTS, 0, 63, Data_simple);        // To get captured
datas in array for analysis
      PRINT_ECM_DATA( OUTPUTS, 0, 63);                     // To show datas in
datalog directly
      PRINT_ECM_DATA( OUTPUTS, 0, 63,
"D:\\3380D_DemoBrd_for_ecm_1site\\ReadData\\Read_x1_PEL_K(SIMPLE_TEST_
64DATAS).txt"); // To Show datas in text file

      SET_ECM_MODE( OUTPUTS, PEL_K, OFF, X1 );      // Setting ecm off
      }// End of ecm_func_simple_test()
```

ECM Pattern file

4.22 Shmoo statement

4.22.1 SET_SHMOO_TG

SET_SHMOO_TG – CRAFT statement

4.22.1.1 Explanation

Use the function to set shmoo "Timing" condition

4.22.1.2 Format

Parameter list:

SET_SHMOO_TG (direction, index, TG_type, pin_name, Start, Stop, Delta, Marker, TS_no, delay [, strobe_style, window_width]);

Parameter comment:

direction : X or Y

index : integer, variable is available(0-127)

TG_type : RATE, IO_MARKER, **FREQ_SHMOO**
(FREQ_SHMOO Release version : CRAFT_3380_2.50-SP3)

pin_name : pin name defined in dec file,single pin or pin group

Start : double, variable is available

Stop : double, variable is available

Delta : double, variable is available

Marker : IO_MARKER ➔ DT0, DT1 DT2, DT3, DT4, CT0, CT1

TS_no : TS1~TS15, integer, variable is available

delay : double, variable is available

strobe_style : FIXED or NOFIXED

window_width : double, variable is available

4.22.1.3 Example

```
JUDGE_PAT(fun2, __fun2);
SET_SHMOO_TG(X,0,IO_MARKER, PIN1 ,0nS,30nS,,DT0,TS1,);
SET_SHMOO_TG(Y,0,IO_MARKER, PIN1 ,0nS,30nS,,DT1,TS1,);

RUN_SHMOO(X0,25,Y0,25,,1,POINT_MODE,ON);
```

4.22.1.4 Relative

RUN_SHMOO

4.22.2 SET_SHMOO_LEVEL

SET_SHMOO_LEVEL – CRAFT statement

4.22.2.1 Explanation

Use the function to set shmoo "IO_LEVEL" condition

4.22.2.2 Format

Parameter list:

SET_SHMOO_LEVEL (direction, index, Level_type, pin_name, Start, Stop, Delta, Level_mode, delay);

Parameter comment:

direction : X or Y

index : integer, variable is available(0-127)

Level_type : IO_LEVEL , HV_LEVEL

pin_name : pin name defined in dec file,single pin or pin group

Start : double, variable is available

Stop : double, variable is available

Delta : double, variable is available

Level_mode :

IO_LEVEL => VIL_LEVEL, VIH_LEVEL, VOL_LEVEL, VOH_LEVEL,IOL_LEVEL, IOH_LEVEL, VREF_LEVEL

HV_LEVEL => HV_VIL, HV_VIH, VIHH, HV_VOL, HV_VOH

delay : double, variable is available

4.22.2.3 Example

```
SET_SHMOO_LEVEL(X,0, IO_LEVEL,PIN1,0V,1V, ,VIL_LEVEL,2mS);
SET_SHMOO_LEVEL(Y,0, IO_LEVEL,PIN1,3V,2V, ,VIH_LEVEL,2mS);
RUN_SHMOO(X0,25,Y0,25,,1,POINT_MODE,ON);
```

```
SET_SHMOO_LEVEL(X,0, HV_LEVEL, p0,1.000V, ,100.000mV,    HV_VIL,
5.000mS);
SET_SHMOO_LEVEL(X,1, HV_LEVEL, p0,1.000V, ,100.000mV,    HV_VIH,
5.000mS);
SET_SHMOO_LEVEL(X,2, HV_LEVEL, p0,6.000V, ,200.000mV,    VIHH,
5.000mS);
SET_SHMOO_LEVEL(X,3, HV_LEVEL, p0,0.000V, ,400.000mV,
HV_VOL,    5.000mS);
SET_SHMOO_LEVEL(X,4, HV_LEVEL, p0,0.000V, ,400.000mV,
HV_VOH,    5.000mS);
```

4.22.2.4 Relative

RUN_SHMOO

4.22.3 SET_SHMOO_POWER

SET_SHMOO_POWER – CRAFT statement

4.22.3.1 Explanation

Use the function to set power shmoo condition .For example:DPS , PREF, UVI.

4.22.3.2 Format

Parameter list:

SET_SHMOO_POWER (direction, index, DC_type, force_type ,pin_name, Start, Stop, Delta, delay);

Parameter comment:

direction :	X or Y
index :	integer, variable is available(0-127)
DC_type :	DPS ,PREF, UVI
Force_type:	FORCE_V
pin_name :	pin name defined in dec file,single pin or pin group
Start :	double, variable is available
Stop :	double, variable is available
Delta :	double, variable is available
delay :	double, variable is available

4.22.3.3 Example

```
    JUDGE_PAT(fun_1, __fun_1);
    SET_SHMOO_POWER( X, 1, DPS, FORCE_V ,VDD, 3V, 2V,    , 2mS );
//DPS setting
    SET_SHMOO_LEVEL(X,0, IO_LEVEL,PIN1,3V,2V,    ,VIH_LEVEL,2mS);
//VIH setting
    RUN_SHMOO(X0,25,    ,    ,    , 1 ,POINT_MODE,ON);
//run shmoo for "fun_1" pattern
```

4.22.3.4 Relative

RUN_SHMOO

4.22.4 RUN_SHMOO

RUN_SHMOO – CRAFT statement

4.22.4.1 Explanation

The statement will execute the Shmoo functions, and output the Shmoo plot. RUN_SHMOO adds three parameters before function, after function and datalog on/off. Release version : **CRAFT_3380_1.40**

4.22.4.2 Format

Parameter list:

```
RUN_SHMOO( location, point, location, point, point_delay, Iteration, exec_mode,
per_site_display_flag
                    [, output_path [ , before_judge_function, after_judge_function,
                shmoo_function_datalog ] ] );
```

Parameter comment:

location : X, (means all declared X-axis parameters in SET_SHMOO_TG/SET_SHMOO_LEVEL)
Y, (means all declared Y-axis parameters in SET_SHMOO_TG/SET_SHMOO_LEVEL)
X_n, Y_n, or use '&' to combine multi conditions

point : integer, variable is available
default : 1

point_delay : the delay between each point
double, variable is available

Iteration : integer > 0,
variable is available

exec_mode : POINT_MODE,
STEP_MODE (just for debug using, the function the same as Shmoo-Tool, only work at per_site_display_flag : ON)
no_entry => POINT_MODE

per_site_display_flag : ON (only pisplay each site Shmoo plot)
OFF (only display summary Shmoo plot)

output_path : STDOUT, string, or use '&' to combine
no_entry => STDOUT

NOTE:
SET_SHMOO_TG/LEVEL delta value is empty, RUN_SHMOO have value → from SET_SHMOO decides start/stop value
SET_SHMOO_TG/LEVEL delta have value, RUN_SHMOO have value,too→ SET_SHMOO_TG/LEVEL decides start value, stop value = delta value* point

SET_SHMOO_TG/LEVEL delta have value, RUN_SHMOO is empty (means point 1) → SET_SHMOO_TG/LEVEL decides start value, stop value = delta value* 1

before_judge_function : a function name; RUN_SHMOO follow setup X/Y parameters first, and run thisfunction.
// If don't need before_judge_function, please empty the item
// If before_judge_function includes JUDGE_PAT statement then finish this function, will restore original content of JUDGE_PAT next routine.

after_judge_function : a function name; after RUN_SHMOO, will run this function.
// If don't need after_judge_function, please empty the item

shmoo_function_datalog : OFF (shmoo function datalog off)
ON (shmoo function datalog on)

execution flow setting :
original shmoo flow ➔ Xn -> Ym -> JUDGE_PAT
function shmoo flow ➔ Xn -> Ym -> before_judge_function -> JUDGE_PAT -> after_judge_function

4.22.4.3 Example

```
JUDGE_PAT(fun1_, __fun_1);
SET_SHMOO_TG(X,0, IO_LEVEL,PIN1,0V,1V,,VIL_LEVEL,2mS);//setting X
SET_SHMOO_TG(Y,0, IO_LEVEL,PIN1,3V,2V,,VIH_LEVEL,2mS);//setting Y
RUN_SHMOO(X0,25,Y0,25, ,1,POINT_MODE,ON);//execute
```

```
char   shmoo_file_name[256];
int   point_x,slot_no;

SHMOO_MSG = "DBL_2X-mode pin2pin SPM DT0 Test";
sprintf(shmoo_file_name,"C:\\CRAFT\\Maint\\LOG_FILE\\RAW_DATA_LOG\\slot_%d_D
BL_2X_dt0.txt", slot_no);

SET_SHMOO_TG(X,0,IO_MARKER,
MT_GROUP_PINS ,0nS,period, ,CT0,TS1,0.5mS);
SET_SHMOO_LEVEL(Y,0,
IO_LEVEL,MT_GROUP_PINS,1.1V,0V, ,VOH_LEVEL,3mS);
point_x= period/0.625e-9+1;
RUN_SHMOO(X0,point_x,Y0,23,1mS,1,POINT_MODE,ON, shmoo_file_name);
```

```
void before_judge_function (int x,int y)    // before_judge_function could be naming by user
{
      MessageBox(0,"in"," ",MB_OK);
      JUDGE_VARIABLE(S0,1,100,200,"");
}

void after_judge_function (int x,int y) // after_judge_function could be naming by user
{
        MessageBox(0,"out"," ",MB_OK);
}
....
....
....
RUN_SHMOO(X,21,,,5ms,1,POINT_MODE,ON, , before_judge_function,
after_judge_function,OFF);
```

4.22.4.4 Relative

SET_SHMOO_LEVEL, SET_SHMOO_TG

4.22.5 RUN_SHMOO_EX

RUN_SHMOO – CRAFT statement

4.22.5.1 Explanation

The statement will execute the Shmoo functions, and output the Shmoo plot.
RUN_SHMOO_EX add parameter, judge function(any CRAFT statement or C code combination). Release version : **CRAFT_3380_2.40-SP5**

4.22.5.2 Format

Parameter list:

```
RUN_SHMOO_EX( location, point, location, point, point_delay, Iteration, exec_mode,
per_site_display_flag, judge_function
                    [, output_path [ , before_judge_function, after_judge_function,
                    shmoo_function_datalog ] ] );
```

Parameter comment:

location : X, (means all declarated X-axis parameters in SET_SHMOO_TG/SET_SHMOO_LEVEL)
Y, (means all declarated Y-axis parameters in SET_SHMOO_TG/SET_SHMOO_LEVEL)
X_n, Y_n, or use '&' to combine multi conditions

point : integer, variable is available
default : 1

point_delay : the delay between each point
double, variable is available

Iteration : integer > 0,
variable is available

exec_mode : POINT_MODE,
STEP_MODE (just for debug using, the function the same as Shmoo-Tool, only work at per_site_display_flag : ON)
no_entry => POINT_MODE

per_site_display_flag : ON (only pisplay each site Shmoo plot)
OFF (only display summary Shmoo plot)

judge_function : any CRAFT statement or C code combination

output_path : STDOUT, string, or use '&' to combine
no_entry => STDOUT

NOTE:
SET_SHMOO_TG/LEVEL delta value is empty, RUN_SHMOO have value → from SET_SHMOO decides start/stop value
SET_SHMOO_TG/LEVEL delta have value, RUN_SHMOO have value,too→ SET_SHMOO_TG/LEVEL decides start value, stop value = delta value* point

SET_SHMOO_TG/LEVEL delta have value, RUN_SHMOO is empty (means point 1) → SET_SHMOO_TG/LEVEL decides start value, stop value = delta value* 1

before_judge_function : a function name; RUN_SHMOO follow setup X/Y parameters first, and run thisfunction.
// If don't need before_judge_function, please empty the item
// If before_judge_function includes JUDGE_PAT statement then finish this function, will restore original content of JUDGE_PAT next routine.

after_judge_function : a function name; after RUN_SHMOO, will run this function.
// If don't need after_judge_function, please empty the item

shmoo_function_datalog : OFF (shmoo function datalog off)
ON (shmoo function datalog on)

execution flow setting :
original shmoo flow ➔ Xn -> Ym -> JUDGE_PAT
function shmoo flow ➔ Xn -> Ym -> before_judge_function -> JUDGE_PAT -> after_judge_function

4.22.5.3 Example

```
JUDGE_PAT(fun1_, __fun_1);
SET_SHMOO_TG(X,0, IO_LEVEL,PIN1,0V,1V,,VIL_LEVEL,2mS);//setting X
SET_SHMOO_TG(Y,0, IO_LEVEL,PIN1,3V,2V,,VIH_LEVEL,2mS);//setting Y
RUN_SHMOO(X0,25,Y0,25, ,1,POINT_MODE,ON);//execute
```

```
char   shmoo_file_name[256];
int   point_x,slot_no;

SHMOO_MSG = "DBL_2X-mode pin2pin SPM DT0 Test";
sprintf(shmoo_file_name,"C:\\CRAFT\\Maint\\LOG_FILE\\RAW_DATA_LOG\\slot_%d_
DBL_2X_dt0.txt", slot_no);

SET_SHMOO_TG(X,0,IO_MARKER,
MT_GROUP_PINS ,0nS,period, ,CT0,TS1,0.5mS);
SET_SHMOO_LEVEL(Y,0,
IO_LEVEL,MT_GROUP_PINS,1.1V,0V, ,VOH_LEVEL,3mS);
point_x= period/0.625e-9+1;
RUN_SHMOO(X0,point_x,Y0,23,1mS,1,POINT_MODE,ON, shmoo_file_name);
```

```
void Judge_function(int x,int y)    // int x,int y : SW demand
{
   // any CRAFT statement or C code combination
   …
}
void before_judge_function (int x,int y)    // before_judge_function could be naming by user
{
      MessageBox(0,"in"," ",MB_OK);
      JUDGE_VARIABLE(S0,1,100,200,"");
}

void after_judge_function (int x,int y) // after_judge_function could be naming by user
{
       MessageBox(0,"out"," ",MB_OK);
}
....
....
....
RUN_SHMOO(X,21,,,5ms,1,POINT_MODE,ON, Judge_function, ,
before_judge_function, after_judge_function,OFF);
```

4.22.5.4 Relative

SET_SHMOO_LEVEL, SET_SHMOO_TG

4.22.6 SET_SHMOO_COMPARE

SET_SHMOO_COMPARE – CRAFT statement

4.22.6.1 Explanation

The statement will make a difference between multiple Shmoo plots, and output a Compare Shmoo Plot after executing statement "RUN_SHMOO".

4.22.6.2 Format

Parameter list:

```
SET_SHMOO_COMPARE(compare_mode, compare_socket, compare_times);
```

Parameter comment:

compare_mode : XOR, AND, OR,
SUM → display how many sites are pass

compare_socket : S0~S1023, use ' | ' to combine multi site,
or use "ALL" to selected all site

compare_times : ONCE, only run one time
ALWAYS, run every time
no_entry => ALWAYS

4.22.6.3 Example

```
JUDGE_PAT(fun_2, __fun_2);
SET_SHMOO_TG(X,0, IO_LEVEL,PIN1,0V,1V,,VIL_LEVEL,2mS);
SET_SHMOO_TG(Y,0, IO_LEVEL,PIN1,3V,2V,,VIH_LEVEL,2mS);
SET_SHMOO_COMPARE(AND, S0 | S1 | S6, ONCE);
//compare S0 ,S1 ,S6 result by AND
RUN_SHMOO(X0,25,Y0,25,,1,POINT_MODE,ON);
```

4.22.6.4 Relative

SET_SHMOO_LEVEL, SET_SHMOO_TG, RUN_SHMOO ,
SET_SHMOO_COMPARE

4.22.7 SET_SHMOO_X(Y)

SET_SHMOO_X(Y) – CRAFT statement

4.22.7.1 Explanation

Enable or disable X (Y) axis Shmoo setting condition.

4.22.7.2 Format

Parameter list:

```
SET_SHMOO_X(index,mode)
SET_SHMOO_Y(index,mode)
```

Parameter comment:
Index: interger, variable is available(0-127)

Mode: ENABLE, DISABLE

4.22.7.3 Example

```
JUDGE_PAT(fun_3 , __fun_3);
SET_SHMOO_TG(X,1, IO_LEVEL,PIN2,0V,1V,,VIL_LEVEL,2mS);
SET_SHMOO_TG(X,2, IO_LEVEL,PIN3,0V,1V,,VIL_LEVEL,2mS);
SET_SHMOO_TG(Y,0, IO_LEVEL,PIN1,3V,2V,,VIH_LEVEL,2mS);
SET_SHMOO_TG(Y,1, IO_LEVEL,PIN3,3V,2V,,VIH_LEVEL,2mS);
SET_SHMOO_TG(Y,2, IO_LEVEL,PIN4,3V,2V,,VIH_LEVEL,2mS);
SET_SHMOO_X(2,DISABLE);
SET_SHMOO_Y(2,DISABLE);
RUN_SHMOO(X,25,Y,25,,1,POINT_MODE,ON);// run condition "X1" ,"Y0" ,"Y1"
```

4.22.7.4 Relative

SET_SHMOO_LEVEL, SET_SHMOO_TG, RUN_SHMOO

4.22.8 SET_SHMOO_COOR

SET_SHMOO_COOR – CRAFT statement

4.22.8.1 Explanation

If setting on (" 1 "), X_COOR condition will be printed on first site.

4.22.8.2 Format

Parameter list:

```
SET_SHMOO_COOR( flag);
```

Parameter comment:

Flag:
1: turn on (default)
0: turn off

4.22.8.3 Example

```
SET_SHMOO_COOR(0); //X_COOR condition will be printed on first site.
JUDGE_PAT(fun_2, __fun_2);
SET_SHMOO_LEVEL(X , 0 , IO_LEVEL , IO_ALLPINS , 0V , 5V , ,VIH_LEVEL,5mS);
RUN_SHMOO(X,22, , ,,1,POINT_MODE,ON);
```

```
SHMOO : Socket 0
                    + + + +
 Pass:X  Fail:.     2 4 7 9
                    3 7 1 5 + + + + + + + + + + + + + + + + +
                  + 8 6 4 2 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5
                  0 . . . . . . . . . . . . . . . . . . . . .
                  . 0 1 2 3 1 4 6 9 1 3 6 8 0 3 5 8 0 2 5 7 0
                  0 9 9 8 8 9 2 6 0 4 8 1 5 9 3 7 1 4 8 2 6 0
                  0 5 0 6 1 0 9 7 5 3 1 9 7 5 3 1 0 8 6 4 2 0
                  0
                    m m m m
                  V V V V V V V V V V V V V V V V V V V V V V
                 ----------------------------------------@--
                | . . . . . . . . . . . . X X X X X X X X X X

SHMOO : Socket 1
                 ----------------------------------------@--
                | . . . . . . . . . . . . X X X X X X X X X X
```

SET_SHMOO_COOR(0).
Condition will only be showed on first site.

```
SHMOO : Socket 0
                    + + + +
 Pass:X  Fail:.     2 4 7 9
                    3 7 1 5 + + + + + + + + + + + + + + + + +
                  + 8 6 4 2 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5
                  0 . . . . . . . . . . . . . . . . . . . . .
                  . 0 1 2 3 1 4 6 9 1 3 6 8 0 3 5 8 0 2 5 7 0
                  0 9 9 8 8 9 2 6 0 4 8 1 5 9 3 7 1 4 8 2 6 0
                  0 5 0 6 1 0 9 7 5 3 1 9 7 5 3 1 0 8 6 4 2 0
                  0
                    m m m m
                  V V V V V V V V V V V V V V V V V V V V V V
                 ----------------------------------------@--
                | . . . . . . . . . . . . X X X X X X X X X X
```

SET_SHMOO_COOR(1).
Condition will always be showed on all site.

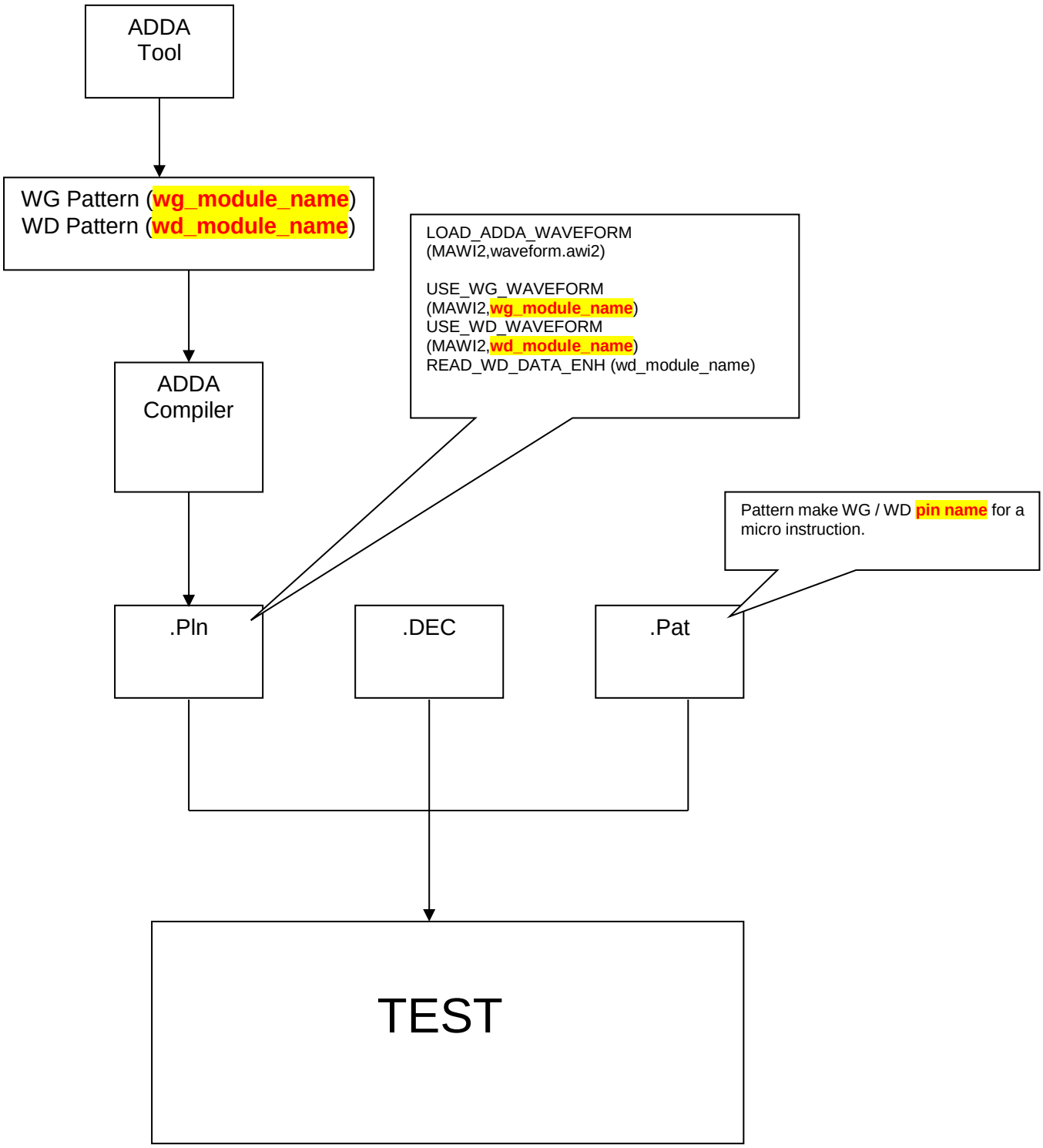
```
SHMOO : Socket 1
                    + + + +
 Pass:X  Fail:.     2 4 7 9
                    3 7 1 5 + + + + + + + + + + + + + + + + +
                  + 8 6 4 2 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5
                  0 . . . . . . . . . . . . . . . . . . . . .
                  . 0 1 2 3 1 4 6 9 1 3 6 8 0 3 5 8 0 2 5 7 0
                  0 9 9 8 8 9 2 6 0 4 8 1 5 9 3 7 1 4 8 2 6 0
                  0 5 0 6 1 0 9 7 5 3 1 9 7 5 3 1 0 8 6 4 2 0
                  0
                    m m m m
                  V V V V V V V V V V V V V V V V V V V V V V
                 ----------------------------------------@--
                | . . . . . . . . . . . . X X X X X X X X X X
```

4.22.8.4 Relative

SET_SHMOO_LEVEL, SET_SHMOO_TG, RUN_SHMOO ,
SET_SHMOO_COMPARE

4.23 MAWI2 Generator / Digitizer

4.23.1 Architecture



4.23.2 AWI2 Pins Declaration

AWI Pin must be declared in Declaration file (*.dec).
Release version : **CRAFT_3380_2.1**

4.23.2.1 Explanation

TYPE: MAWI_WG or MAWI_WD

Channel assign:
WD : channel num=slot*64 + (0, 2, 4, 6)
WG : channel num=slot*64 + (8, 10, 12, 14)

4.23.2.2 Example

```
PIN_LIST     (LPC0_MAWI2_6) {
 //==*name = ATE channel = dut# = type    *==//
      //   S0
p0      =   0 = 1 =IO;
p1      =   1 = 2 =IO;
p2      =   2 = 3 =IO;
p3      =   3 = 4 =IO;
p4      =   4 = 5 =IO;
p5      =   5 = 6 =IO;
p6      =   6 = 7 =IO;
p7      =   7 = 8 =IO;

wg0     = 392 = 9    = MAWI2_WG; // 392 = 64*6 + 8 ; 3380 series slot 6
wg1     = 394 = 10 = MAWI2_WG; // 394 = 64*6 + 10 ; 3380 series slot 6
wg2     = 396 = 11 = MAWI2_WG; // 396 = 64*6 + 12 ; 3380 series slot 6
wg3     = 398 = 12 = MAWI2_WG; // 398 = 64*6 + 14 ; 3380 series slot 6
wd0     = 384 = 13 = MAWI2_WD; // 384 = 64*6 + 0 ; 3380 series slot 6
wd1     = 386 = 14 = MAWI2_WD; // 386 = 64*6 + 2 ; 3380 series slot 6
wd2     = 388 = 15 = MAWI2_WD; // 388 = 64*6 + 4 ; 3380 series slot 6
wd3     = 390 = 16 = MAWI2_WD; // 390 = 64*6 + 6 ; 3380 series slot 6
}
```

4.23.3 Pattern Form

The AWI2 declaration is used to the order of entry and display of the actual binary Information in **Pattern file**. (*.pat)
Release version : **CRAFT_3380_2.1**
Capture trigger of release version : **CRAFT_3380_2.20_SP1**

4.23.3.1 Explanation

Parameter comment:
Module name : define name from awi2 waveform pattern in SPM_PATTERN

U'I(micro instruction) :
MAWI2 pin name : define from *.dec file for continuous trigger use.
MAWI2_WG_STEP: define WG set step trigger, need MAWI2_WG_USE together.
MAWI2_WG_USE: define WG is a label setting, and check setting time of MAWI2_WG_STEP to MAWI2_WG_USE is enough whether or not.
MAWI2_WD_CONV: define WD is a label setting, need MAWI2_WD_CAP together.
MAWI2_WD_CAP: define WD set capture step trigger, and check converter time of MAWI2_WD_CONV to MAWI2_WD_CAP is enough whether or not.

4.23.3.2 Example

Continuous trigger

Example of WG

```
SET_DEC_FILE "./example_pin.dec"

HEADER
DB15,%DB14,DB13,DB12,DB11,%DB10,%DB9,%DB8,DB7,DB6,DB5,%DB4,DB3,%DB2,DB
1,DB0,%CS,RD_WR,RESET,SYNC,DRDY;

SPM_PATTERN(fun_ramp_MCLK_Dev32_cont_trig)
{
        /* D DDDD D D DDDD DD DDD CRRSD */
        /* B BBBB B B BBBB BB BBB SDEYR */
        /* 1 1111 1 9 8765 43 210   |SND */
        /* 5 4321 0                  WECY */
        /*                            RT   */
    st: * X XXXX X X XXXX XX XXX 1001X      * TS1;      //Register Write init
        * X XXXX X X XXXX XX XXX 1001X      * RPT 49;
        * 0 0000 0 0 0000 00 000 1111X      * ;
        * 0 0000 0 0 0000 00 000 1111X      * ;
        * 0 0000 0 0 0000 00 000 1111X      * RPT 98;
        * 0 0000 0 0 0000 00 010 1111X      * RPT 10;
        * 0 0000 0 0 0000 00 010 0111X      * RPT 120; //address      cs=0

        * 0 0000 0 0 0000 00 010 1111X      * RPT 40;   //cs=1
        * 0 0000 0 0 0000 00 010 1111X      * RPT 80;
        * 0 0000 0 0 0000 00 010 0111X      * RPT 120; //Data          cs=0
        * 0 0000 0 0 0000 00 000 1111X      * RPT 10;
        //---------------------------------------------
        * 0 0000 0 0 0000 00 000 1111X      * RPT 100;//Set Sample rate & Filter
        * 0 0000 0 0 0000 00 001 1111X      * RPT 10;
        * 0 0000 0 0 0000 00 001 0111X      * RPT 120;      //Write 0x01 cs=0
```

```
                * 0 0000 0 0 0000 00 001 1111X          * RPT 10;
                * 0 0000 0 0 0000 00 000 1111X          * RPT 100;
                * 0 0000 0 0 0000 11 001 1111X          * RPT 10;        //Write 0x19
                * 0 0000 0 0 0000 11 001 0111X          * RPT 120;       //Write 0x19 cs=0
                * 0 0000 0 0 0000 11 001 1111X          * RPT 10;
                * 0 0000 0 0 0000 00 000 1111X          * RPT 30;
                * 0 0000 0 0 0000 00 000 1111X          * RPT 30;
               //-------------------------------------------
                * 0 0000 0 0 0000 00 000 1111X          * SETM 2048;    //IMATCH setting loop
                * X XXXX X X XXXX XX XXX 1011X          * RPT 30;        //Data Read Start
                * X XXXX X X XXXX XX XXX 1011X          * SETX2 4096 ;// U'I setting loop
                * X XXXX X X XXXX XX XXX 1011X          * ;
                * X XXXX X X XXXX XX XXX 1011X          * RPT 204799;
                * X XXXX X X XXXX XX XXX 1011X          * ;
                * X XXXX X X XXXX XX XXX 1011X          * ;
                * X XXXX X X XXXX XX XXX 1011X          * TS1,RPT 22;

                * X XXXX X X XXXX XX XXX 1011X          * TS1,wg0;
                                                  // Trig by Pin name .define in dec file

                //-----------------------------------------------
                * X XXXX X X XXXX XX XXX 1011L          * TS3,IMATCH; // check IMATCH
                * X XXXX X X XXXX XX XXX 1011X          * TS2;
                * X XXXX X X XXXX XX XXX 1011X          * TS2;
                * X XXXX X X XXXX XX XXX 1011X          * TS1,RPT 2;
                //-----------------------------------------------

                * X XXXX X X XXXX XX XXX 1011X          * TS1,UISYNC;    // ATE signal
            CC: * X XXXX X X XXXX XX XXX 1011X          * TS2,RPT 10;    // loop start
//-----------------------------------------------
                * X XXXX X X XXXX XX XXX 1011X          * TS1,RPT 174;

                * X XXXX X X XXXX XX XXX 0011X          * RPT 7;
                * K KKKK K K KKKK KK KKK 0011X          * UISYNC;        //ATE signal
                * X XXXX X X XXXX XX XXX 0011X          * ;

                * X XXXX X X XXXX XX XXX 1011X          * RPT 7
                * X XXXX X X XXXX XX XXX 1011X          * ;
                * X XXXX X X XXXX XX XXX 1011X          * ;

                * X XXXX X X XXXX XX XXX 0011X          * RPT 7;
                * K KKKK K K KKKK KK KKK 0011X          * ;
                * X XXXX X X XXXX XX XXX 0011X          * ;

                * X XXXX X X XXXX XX XXX 1011X          * RPT 4;
                //-----------------------------------------------
                * X XXXX X X XXXX XX XXX 1011X          * TS2,RPT 10;

                //-----------------------------------------------
                * X XXXX X X XXXX XX XXX 1011X          * JNZ2 CC;       //loop end

                * X XXXX X X XXXX XX XXX 1111X          * ;
                * X XXXX X X XXXX XX XXX 1111X          * RPT 20000;
          sp : * X XXXX X X XXXX XX XXX 1111X          *;

}
```

Continuous trigger

Example of WD

```
SPM_PATTERN (WD_CONT) {
                *0000* TS2;
                *1000* ;
                *0000* ;
                *1000* ;
                *0000* ;
                *1000* ;
                *0000* ;
               *1000* ;
               *0000* TS2, wd0;// Trig by Pin name .define in dec file
                *1000* ;
                *0000* ;
               *1000* ;
               *0000* ;
               *1000* ;
}
```

Step trigger

Example of WG (*.pln)

```
SET_DEC_FILE "./example_pin.dec"

HEADER
DB15,%DB14,DB13,DB12,DB11,%DB10,%DB9,%DB8,DB7,DB6,DB5,%DB4,DB3,%DB2,DB
1,DB0,%CS,RD_WR,RESET,SYNC,DRDY;

SPM_PATTERN(fun_ramp_MCLK_Dev32)
{
            /* D DDDD D D DDDD DD DDD CRRSD */
            /* B BBBB B B BBBB BB BBB SDEYR */
            /* 1 1111 1 9 8765 43 210    |SND */
            /* 5 4321 0                   WECY */
            /*                             RT     */
      st:   * X XXXX X X XXXX XX XXX 1001X        * TS1;          //Register Write init
             * X XXXX X X XXXX XX XXX 1001X        * RPT 49;
             * 0 0000 0 0 0000 00 000 1111X        * ;
             * 0 0000 0 0 0000 00 000 1111X        * ;
             * 0 0000 0 0 0000 00 000 1111X        * RPT 98;
             * 0 0000 0 0 0000 00 010 1111X        * RPT 10;       //address

             * 0 0000 0 0 0000 00 010 0111X        * RPT 120;      //address       cs=0

             * 0 0000 0 0 0000 00 010 1111X        * RPT 40;       //cs=1
             * 0 0000 0 0 0000 00 010 1111X        * RPT 80;

             * 0 0000 0 0 0000 00 010 0111X        * RPT 120;      //Data          cs=0
             * 0 0000 0 0 0000 00 000 1111X        * RPT 10;
            //-------------------------------------------
             * 0 0000 0 0 0000 00 000 1111X        * RPT 100;//Set Sample rate & Filter
             * 0 0000 0 0 0000 00 001 1111X        * RPT 10;       //Write 0x01

             * 0 0000 0 0 0000 00 001 0111X        * RPT 120;      //Write 0x01 cs=0

             * 0 0000 0 0 0000 00 001 1111X        * RPT 10;
             * 0 0000 0 0 0000 00 000 1111X        * RPT 100;
```

```
            * 0 0000 0 0 0000 11 001 1111X        * RPT 10;        //Write 0x19

            * 0 0000 0 0 0000 11 001 0111X        * RPT 120;       //Write 0x19 cs=0
            * 0 0000 0 0 0000 11 001 1111X        * RPT 10;

            * 0 0000 0 0 0000 00 000 1111X        * RPT 30;
            * 0 0000 0 0 0000 00 000 1111X        * RPT 30;
          //-------------------------------------------
            * 0 0000 0 0 0000 00 000 1111X        * SETM 2048;    //IMATCH setting loop
            * X XXXX X X XXXX XX XXX 1011X          * RPT 30;        //Data Read Start

            * X XXXX X X XXXX XX XXX 1011X          * SETX2 4096 ; // U'I setting loop
            * X XXXX X X XXXX XX XXX 1011X          * ;

            * X XXXX X X XXXX XX XXX 1011X          * RPT 204799;
            * X XXXX X X XXXX XX XXX 1011X          * ;
            * X XXXX X X XXXX XX XXX 1011X          * ;

            //-----------------------------------------------
        CC: * X XXXX X X XXXX XX XXX 1011X          * TS1,MAWI2_WG_STEP(wg0);
                                                      //WG step trigger
            * X XXXX X X XXXX XX XXX 1011X          * TS1,RPT 9;     // delay cycle 10
            * X XXXX X X XXXX XX XXX 1011X          * TS1,MAWI2_WG_USE(wg0);
                                                      //WG step use
            //-----------------------------------------------
            * X XXXX X X XXXX XX XXX 1011X          * TS2,RPT 10;    // delay cycle 10
            //-----------------------------------------------
            * X XXXX X X XXXX XX XXX 1011L          * TS3,IMATCH;    // check IMATCH
            * X XXXX X X XXXX XX XXX 1011X          * TS1,RPT 174;
            //-----------------------------------------------
            * X XXXX X X XXXX XX XXX 1011X          * TS1,UISYNC;    //ATE signal

            * X XXXX X X XXXX XX XXX 0011X          * RPT 7;
            * K KKKK K K KKKK KK KKK 0011X            * ;
            * X XXXX X X XXXX XX XXX 0011X          * ;

            * X XXXX X X XXXX XX XXX 1011X          * RPT 7;
            * X XXXX X X XXXX XX XXX 1011X          * ;
            * X XXXX X X XXXX XX XXX 1011X          * ;

            * X XXXX X X XXXX XX XXX 0011X          * RPT 7;
            * K KKKK K K KKKK KK KKK 0011X            * ;
            * X XXXX X X XXXX XX XXX 0011X          * ;
            //-----------------------------------------------
            * X XXXX X X XXXX XX XXX 1011X          * RPT 4;
            //-----------------------------------------------

            * X XXXX X X XXXX XX XXX 1011X          * TS2,RPT 10;
            //-----------------------------------------------
            * X XXXX X X XXXX XX XXX 1011X          * JNZ2 CC;

            * X XXXX X X XXXX XX XXX 1111X          * ;
            * X XXXX X X XXXX XX XXX 1111X          * RPT 20000;
        sp : * X XXXX X X XXXX XX XXX 1111X         *;

}
```

Capture step trigger

Example of WD(*.pln)

```
SET_DEC_FILE "./example_pin.dec"

HEADER     CS,%DIN,%SCLK,%LDAC;

SPM_PATTERN(fun_ramp_2048_1V)
 {
        //----------------------------------------//
        // TS1 = 100 ns                                    //
        // TS2 = DUT setting time                         //
        // TS3 = WD converter rate                         //
        //----------------------------------------//

        /* C D S L */
        /* S I C D */
        /*     N L A */
        /*        K C */

    st: * X X X X * TS3, RPT 20 ;        // WD converter rate int 20 cycle
         * X X X X * SETX1 2048;           // U'I setting loop
         * X X X X * ;

        //-------------------------------------------------------------------
DD:      * 1 0 1 1 * TS1; // loop start , dout level       0 setup
         * 1 0 1 1 * ;       // DIN=0xB333 = +1.000000 V

         * 0 1 1 1 * ; // DB15
         * 0 0 1 1 * ; // DB14
         * 0 1 1 1 * ; // DB13
         * 0 1 1 1 * ; // DB12
         * 0 0 1 1 * ; // DB11
         * 0 0 1 1 * ; // DB10
         * 0 1 1 1 * ; // DB09
         * 0 1 1 1 * ; // DB08
         * 0 0 1 1 * ; // DB07
         * 0 0 1 1 * ; // DB06
         * 0 1 1 1 * ; // DB05
         * 0 1 1 1 * ; // DB04
         * 0 0 1 1 * ; // DB03
         * 0 0 1 1 * ; // DB02
         * 0 1 1 1 * ; // DB01
         * 0 1 1 1 * ; // DB00

         * 1 0 1 1 * ;
         * 1 0 1 0 * ;

         * 1 0 1 1 * TS2;                              //DUT seting time

         * 1 0 1 1 * TS3,MAWI2_WD_CONV(wd0) ; // WD conv check point
         * 1 0 1 1 * TS3,RPT 4;                        // WD conv time delay    5 cycle
         * 1 0 1 1 * TS3,MAWI2_WD_CAP(wd0) ;    // WD capture point

         * 1 0 1 1 * TS3,RPT 4,JNZ1 DD;        // loop end ,WD hold time delay    5 cycle

          //-------------------------------------------------------------------

         * 1 0 1 1 * TS1;

         * 1 0 1 1 * ;
```

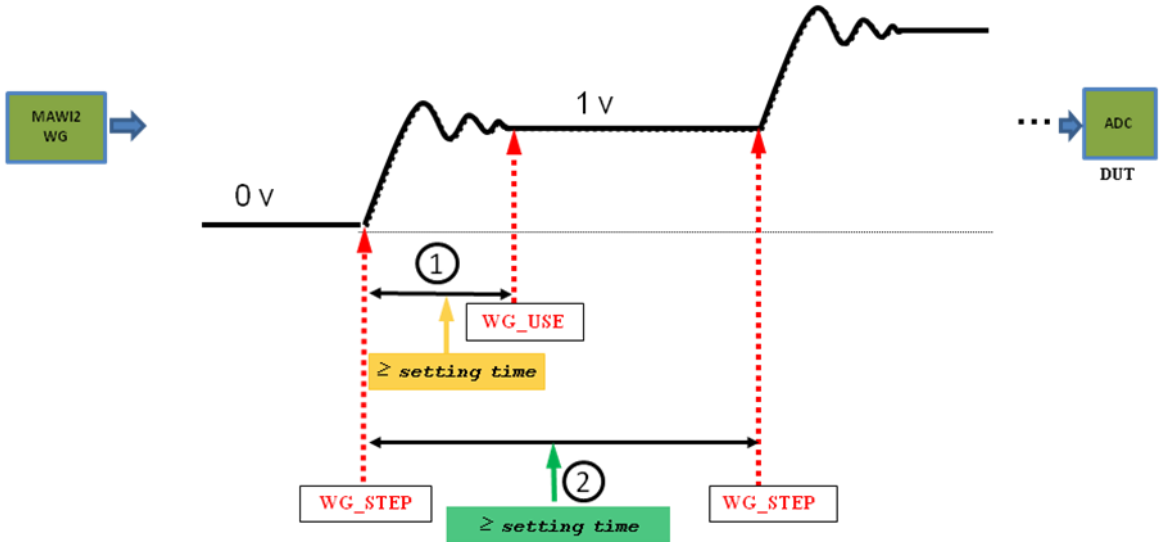
```
        * 1 0 1 1 * TS3, RPT 9 ;                    // WD done time delay 10 cycle
   sp: * 1 0 1 1 *   ;
}
```

4.23.3.3 The limit of setting time

[Setting time of WG]

①check MAWI2_WG_STEP to MAWI2_WG_USE ≧ 1uS

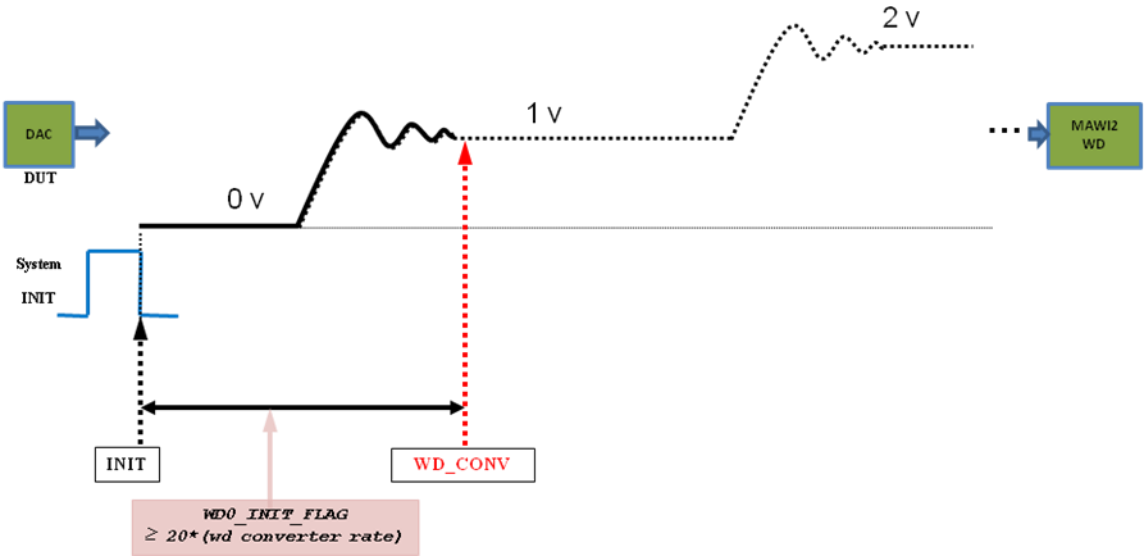
②check MAWI2_WG_STEP to MAWI2_WG_STEP ≧ 1uS



[Initial time of WD]

From system start (the first pattern address) to the first MAWI2_WD_CONV ≧20fs

(One fs is defined by WD converter rate)



[Converter time of WD]

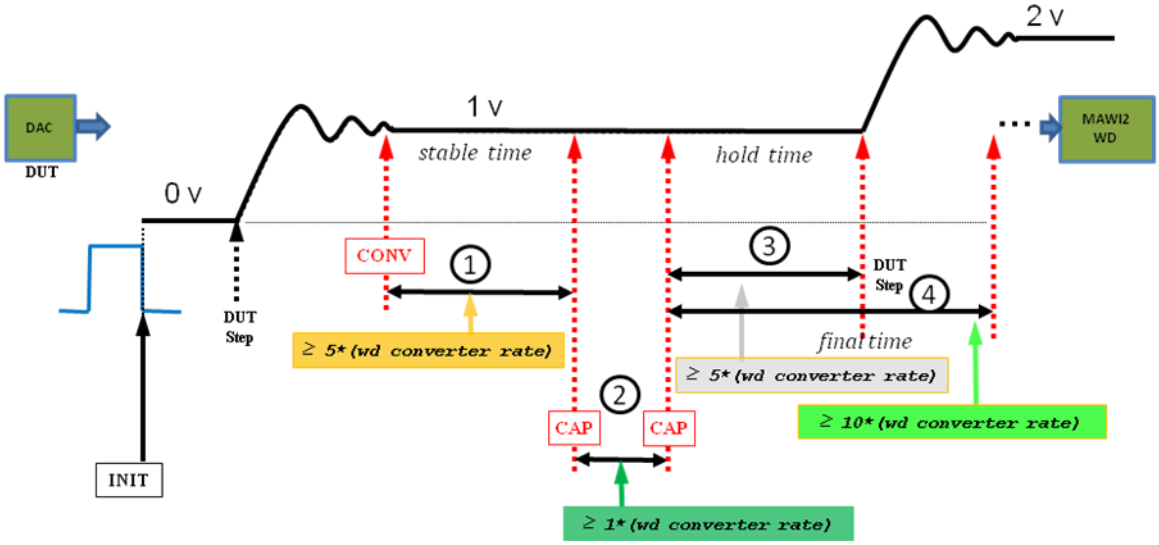
①check stable time of MAWI2_WD_CONV to MAWI2_WD_CAP ≧ 5fs

②check MAWI2_WD_CAP to MAWI2_WD_CAP ≧ 1fs

③check hold time of MAWI2_WD_CAP to next step ≧ 5fs

④check final time of last MAWI2_WD_CAP to pattern end ≧10fs

(One fs is defined by WD converter rate)



4.23.4 LOAD_ADDA_WAVEFORM

LOAD_WG_WAVEFORM – CRAFT statement
Load the waveform generator pattern file into the hardware pattern memories. This is a *.awi2 complier file from AWI2 tool.
Release version : **CRAFT_3380_2.1**

4.23.4.1 Format

```
LOAD_ADDA_WAVEFORM (Brd_type,File_Name);
```

Parameter comment:

Brd_type :MAWI2

**no_entry: illegal

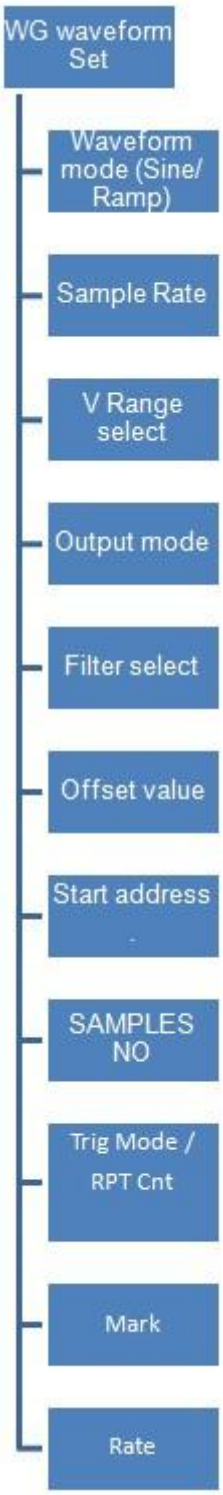
File_Name :awg_pattern_file_name

**no_entry: illegal

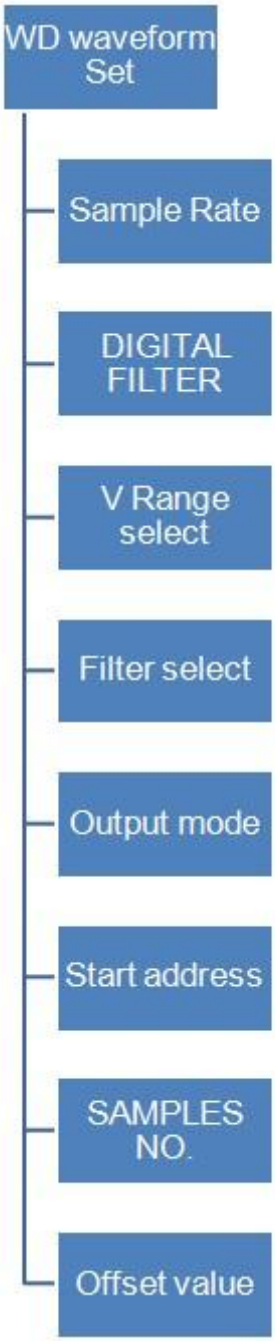
4.23.4.2 Example

```
START_UP()
{
   SET_LEVELN(rst_lvl, IO_ALLPINS, 0.0V,0.5V,0.0V,0.0V,0.0mA,0.0mA,0.0V,
-2V, 7V);
   SET_LEVELN( os_lvl, IO_ALLPINS, 0.0V,0.0V,0.0V,0.0V,0.0mA,0.0mA,0.0V,
-2V, 7V);
   SET_LEVELN(func_lvl, IO_ALLPINS, 0V, 5V, 1V, 3.0V, 0uA, 0uA, 2V, , );

   LEVELS(rst_lvl, 3mS);
   RELAY_OFF( IO_ALLPINS, ALL, 1mS);

   FORCE_VOLT( MLDPS_PINS, MLDPS,_FORCE_V, 0V, _V_RNG, @6V,
_I_RNG, @500mA, _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY,
OFF,_DELAY, 3mS);

   LOAD_PAT("./PAT/dac_PCM5102.ppo");
   LOAD_ADDA_WAVEFORM(MAWI2,"../WD_CAPTURE.awi2");
}
```

4.23.4.3 WG Waveform Set



4.23.4.4 WD Waveform Set



4.23.5 USE_WG_WAVEFORM

USE_WG_WAVEFORM – CRAFT statement

Select WG waveform set. This data is build form AWI tool.

Release version : **CRAFT_3380_2.1**

4.23.5.1 Format

```
USE_WG_WAVEFORM (Brd_type,wg_module_name,delay);
```

Parameter comment:

Brd_type :MAWI2
**no_entry: illegal

wg_module_name : define in AWG Pattern file
**no_entry; illegal

delay : Variables and constants are allowed.
Type: TIME or double
**no_entry; 1mS

4.23.5.2 Example

Example is a AWI2 WG signal to AWI2 WD. User could wire WG and WD together, and copy next example to bulid a sine wave self test. WG send sine wave frequency 1.5K with update rate 384K to WD, and WD receive signal with 48K, 96K, 192K and 384K. Program steps:

1)Loading AWG pattern in START_UP().
2)Use WG waveform setting.
3)Setting WD work mode.
4)Running WG and WD trigger.
5)To read data after getting value.

```
WD_SINE_func_test()
{
   int w,i,j,adc_code,log_cnt;
   double value;
   FILE *stream0;
   char msg[1024];
   double noise,snr,sdr,sndr,mag_fund,dc_offset;
   double THD_value,SNR_value,SND_value, SFDR_value, THD_n_value;
   int Dnl_flag=0,Inl_flag=0,log_cnt_temp=0;
   double max_offset,max_gain,dac_offset,dac_gain;

   DEBUG_BREAK();
```

```
log_cnt = pow(2.0,test_bit);
log_cnt_temp = log_cnt+1;

DEBUG_BREAK();

FORCE_V_DPS(VDD, 5.0V, @8V, @1A, 100mA, NORM, ON, 3mS);

PIN_MODE( CTL_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE(DATA_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);

RATEN(TS1, 100.0nS);
MARKERN(TS1, CTL_PINS,     0.000nS,       0.0nS, ,0nS,0nS, 25.0nS, 50.0nS);
MARKERN(TS1,DATA_PINS,     0.000nS,       0.0nS, ,0nS,0nS, 25.0nS, 50.0nS);

SET_LEVELN(AWI2_lvl, ALLPINS, 0V, 3.3V, 0.8V, 2.4V, 0uA, 0uA, 0V, , );
LEVELS(AWI2_lvl, 10mS);

RELAY_ON(ALLPINS,PDCL,2mS);

WAIT(10mS);

USE_WG_WAVEFORM (MAWI2,WG_1K,2mS);
WG_RELAY_ON (MAWI2, WG0, WG, GND, 2mS);

DEBUG_BREAK();

SOCKET_INC {
        if(AWI2_SW){
         CRAFT_STATEMENT
         {
                USE_WD_WAVEFORM(Without_LPF,5mS);
                WD_RELAY_ON(MAWI2,WD0,WD,GND,5mS);
```

```
                    DEBUG_BREAK();

                    JUDGE_PAT(AD7111_func,__AD7111_func);



                //==***************** Print Data to File !!! *****************==//
                    SOCKET_INC
                    {

READ_WD_DATA_MS(MAWI2,WD0,Without_LPF,data_tmp,8192);

                        sprintf(msg,"AWI2_AD7111_site%d.dat",SOCKET);
                        stream0 = fopen(msg, "w+");
                        for(i=0; i<3125; i++)        fprintf(stream0,"%f
\n",data_tmp[SOCKET][0][i]);
                        fclose(stream0);
                    }

                    SOCKET_INC{

                    AWI_DATA_AC_ARITH(data_tmp[0][0], 3125, THD_value, SNR_value,
SND_value, SFDR_value, THD_n_value) ;

                    JUDGE_VARIABLE (SOCKET, SNR_value, 70,       , "SNR_dB");
                    JUDGE_VARIABLE (SOCKET, THD_value,     , -70, "THD_dB");
                    }
                }

            }
    }

    FORCE_V_DPS(VDD, 0.0V, @8V, @1A, 100mA, NORM, ON, 3mS);
    RELAY_OFF(ALLPINS,PDCL,2mS);
    WG_RELAY_OFF(MAWI2,WG0,WG,GND,5mS);
```

```
   power_down();

}
```

Example of WG step trigger (*.pat)

```
SET_DEC_FILE   ".\PAT\example_pin.dec"

GLOBAL int       os_test_by_PMU       = 1,
                 WG_log_cnt           = 8192,
                 WG_Data_cnt          = 4096,
                 Show_capture_data = 0,
                 Show_data_Value      = 0,
                 RPT_Cont_0           = 120,
                 RPT_Cont_1           = 8;
GLOBAL float     Rate_X               = 32.0,
                 osc_value_4          = 10000000.000;// 10.00Mhz

double           Delay                = 10.0e-3;

int pel[16], peh[16], pfail[16];
FILE *stream0;
char msg[4096];
unsigned int i;
int w,j;
int test_cnt;

/**declaration for IO level set use**/
int              rst_lvl              = 1,
                 os_lvl               = 2,
                 func_lvl             = 3;

int data_fa[2],data_f[2][65536];
double data_val[2][65536];

HW_BIN_DEF
{
   all_pass          =   1;
   fail_os_test      = !3;
   fail_func_AC_test = !4;
   fail_func_DC_test = !5;
}

SW_BIN_DEF
{
   all_pass          =   1;
   fail_os_test      = !3;
   fail_func_AC_test = !4;
   fail_func_DC_test = !5;
}
START_UP()
{
   SET_LEVELN(rst_lvl, IO_ALLPINS, 0.0V,0.5V,0.0V,0.0V,0.0mA,0.0mA,0.0V, -2V, 7V);
   SET_LEVELN(rst_lvl,     DB_PINS, 0.0V,0.0V,0.0V,0.0V,0.0mA,0.0mA,0.0V, -2V, 7V);
   SET_LEVELN( os_lvl, IO_ALLPINS, 0.0V,0.0V,0.0V,0.0V,0.0mA,0.0mA,0.0V, -2V, 7V);
   LEVELS(rst_lvl, 3mS);
   RELAY_OFF( IO_ALLPINS, ALL, 3mS);
   FORCE_VOLT( MLDPS_PINS, MLDPS,_FORCE_V, 0V, _V_RNG, @6V, _I_RNG, @500mA,
         _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY, OFF,_DELAY, 3mS);
```

```
   SET_CBIT(UR_ALLPINS,OFF,3mS);
   test_cnt = 1;

   LOAD_PAT("./PAT/example.ppo");
   LOAD_ADDA_WAVEFORM(MAWI2,"../ramp_test.awi2");


}

test_start()
{
   SET_UPAT_REG(X0, RPT_Cont_0);
   SET_UPAT_REG(X1, RPT_Cont_1);
   LEVELS(rst_lvl, 3mS) ;
   SET_CBIT(POWER_UR,ON,3mS);
   RELAY_OFF( IO_ALLPINS, ALL, 3mS);
}

test_end()
{
   LEVELS(rst_lvl, 3mS) ;
   FORCE_VOLT( MLDPS_PINS, MLDPS,_FORCE_V, 0V, _V_RNG, @6V, _I_RNG, @500mA,
         _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY, OFF,_DELAY, 3mS);

   RELAY_OFF( IO_ALLPINS, ALL, 3mS);
   SET_CBIT(UR_ALLPINS,OFF,3mS);
   test_cnt = test_cnt+1;
}

power_down()
{
   LEVELS(rst_lvl, 3mS) ;

   FORCE_VOLT( MLDPS_PINS, MLDPS,_FORCE_V, 0V, _V_RNG, @6V, _I_RNG, @500mA,
         _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY, OFF,_DELAY, 3mS);
   FORCE_V_PMU(PMU, 0V, @6V, @100mA, 100mA, OFF, 3mS);
   RELAY_OFF(IO_ALLPINS, ALL, 3mS);
   SET_CBIT(UR_ALLPINS,OFF,3mS);
}

RELEASE()
{
   LEVELS(rst_lvl, 3mS) ;
   FORCE_VOLT( MLDPS_PINS, MLDPS,_FORCE_V, 0V, _V_RNG, @6V, _I_RNG, @500mA,
         _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY, OFF,_DELAY, 3mS);
   RELAY_OFF( IO_ALLPINS, ALL, 3mS);
   SET_CBIT(UR_ALLPINS,OFF,3mS);
}

TEST_PRO
{
   BEFORE_TEST: test_start;
      func_WG_step_ramp?#C(all_pass,all_pass): #C(fail_func_AC_test, fail_func_AC_test) ;
   AFTER_TEST: test_end ;
}

func_WG_step_ramp()
{
   // MAWI2 WG ----> sample
```

```
double INL_value[8192],DNL_value[8192],gain_error,offset_error;

double inl_max=-999.0;
double dnl_max=-999.0;
double inl_min=999.0;
double dnl_min=999.0;
int      inl_max_addr, dnl_max_addr, inl_min_addr, dnl_min_addr;

RELAY_OFF( IO_ALLPINS, ALL, 3mS);
SET_CBIT(UR_ALLPINS,OFF,3mS);

SET_CBIT(WG_UR_SINGLE,ON,3mS);

RATEN(TS1, 100nS); //10Mhz
MARKERN(TS1, CTL_PINS,      5nS,     0nS, ,0nS,0nS, 50.0nS, 76nS);
MARKERN(TS1,   DB_PINS,      0nS,     0nS, ,0nS,0nS, 95.0nS, 76nS);

RATEN(TS2, 100nS*Rate_X); //10Mhz
MARKERN(TS2, CTL_PINS,      5nS,     0nS, ,0nS,0nS, 50.0nS*Rate_X,   76nS*Rate_X);
MARKERN(TS2,        DRDY,      5nS,     0nS, ,0nS,0nS, 60.0nS*Rate_X, 0.0nS*Rate_X);
MARKERN(TS2,   DB_PINS,      0nS,     0nS, ,0nS,0nS, 95.0nS*Rate_X,   76nS*Rate_X);

RATEN(TS3, 1500ns);
MARKERN(TS3, CTL_PINS,      5nS,     0nS, ,0nS,0nS, 50nS    , );
MARKERN(TS3,   DB_PINS,      0nS,     0nS, ,0nS,0nS, 0nS     , );
MARKERN(TS3,        DRDY,      5nS,     0nS, ,0nS,0nS, 50.0nS, );

PIN_MODE(CTL_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE( DB_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);

SET_LEVELN(func_lvl, CTL_PINS+DB_PINS, 0.00V, 2.50V, 1.75V, 1.75V, 0uA, 0uA, 0V, , );
SET_LEVELN(func_lvl, DRDY, , , 0.8V, 1.75V, 0uA, 0uA, 0V, , );
SET_LEVELN(func_lvl, MCLK, 0V, 5.0V, 0.1V, 1.5V, 0uA, 0uA, 0V, , );
LEVELS(func_lvl, 5mS);

float     osc_freq_f;
            osc_freq_f = osc_value_4;

double   time_value;
int        pin_no[16] = {0,8};
int        pin_cnt       = 2;
double   osc_freq     = 100.0;
double   osc_spec     = 0.0000001;
double   time_out     = 60.0;
double   fclk_rate    = 0.0;

int        m_data = 0;
int        n_data = 0;
int        p_data = 0;
int        status = 0;
int        d_format[2];
int        ret = 0;

OSC_struct   osc_clk;
FILE *fp;
osc_freq = (double)osc_freq_f;

PIN_MODE(MCLK, OSC, EDGE, ENABLE, IO_ON, OFF);

OSC_CLK_MODE(MCLK, ON);
```

```
ret=0;                                          // calculate & get OSC clk value
while(!ret)
{
        ret = GET_OSC_CLK_DATA(
                osc_freq,//Hz
                osc_spec,
                time_out,
                osc_clk
        );
        osc_spec=osc_spec*5;
        DEBUG_BREAK();
}
fflush(stdout);

if(!osc_clk.status){
      DEBUG_BREAK();
      printf("search fail\n");
      printf("\n");
      return 0;
}

SET_OSC_CLK(MCLK,osc_clk);   // OSC clk output
fflush(stdout);
ret = OSC_CLK_LOCK(MCLK);
RELAY_ON(CTL_PINS+DB_PINS+MCLK, PDCL, 3mS);
WAIT(5mS);

//-----------------------------------
//
//-----------------------------------
USE_WG_WAVEFORM (MAWI2, LV_4096_Vpp1 ,2mS);
WG_RELAY_ON (MAWI2, WG0, WG, GND, 2mS);

//--------------------------
INIT_WG_STATE (MAWI2,WG0);
SET_CAPTURE_MEM_MODE(DB_PINS,PEL_K,ON,8K,
8191,LOG_FAIL_ONLY,ON,FEOP_ON);
WAIT(Delay);

DATALOG_MSG="312_5K_test -- MCLK_10M dev 32";
JUDGE_PAT(fun_ramp_MCLK_Dev32:st,fun_ramp_MCLK_Dev32:sp);     //      10Mhz/32   =
312.5KHz

SOCKET_INC
{
      PLAN_RESULT_MS[SOCKET]=PASS;
      printf("SITE%d\n",SOCKET);

      for(i=0; i< WG_Data_cnt; i++)
      {
            data_f[SOCKET][i]=0;
      }

      for(i=0; i< WG_log_cnt; i++)
      {
         READ_CAPTURE_MEM(SOCKET,DB_PINS,i,pel);
         if(Show_capture_data)
         {
            if(i %2 == 0)
            {
```

```
                printf("%d%d%d%d%_%d%d%d%d%_%d%d%d%d%_%d%d%d%d%_", pel[ 0], pel[ 1], pel[ 2], pel[ 3], pel[ 4], pel[ 5], pel[ 6], pel[ 7], pel[ 8], pel[ 9], pel[10], pel[11], pel[12], pel[13], pel[14], pel[15]);
            }
            else
            {
                printf("%d%d%d%d%_%d%d%d%d%_%d%d%d%d%_%d%d%d%d%\n", pel[ 0], pel[ 1], pel[ 2], pel[ 3], pel[ 4], pel[ 5], pel[ 6], pel[ 7], pel[ 8], pel[ 9], pel[10], pel[11], pel[12], pel[13], pel[14], pel[15]);
            }

        }
        if (i %2 == 0)
        {
            data_fa[SOCKET] = 0;
            for(j=0,w=15; w>=0; w--,j++)
            {
                data_fa[SOCKET] = data_fa[SOCKET] | (pel[j]<<w);
                //printf("%d",pel[j]);

            }
            //printf("\n");
        }
        else
        {

            data_f[SOCKET][(i-1)/2] = data_fa[SOCKET]<<8;
            for(j=0,w=7; w>=0; w--,j++)
            {
                data_f[SOCKET][(i-1)/2] = data_f[SOCKET][(i-1)/2] | (pel[j]<<w);
                //printf("%d",pel[j]);
            }
            //printf("\n");
        }

    }

    for(i=0; i< WG_Data_cnt; i++){
        //printf("Wd_data_result[%d] = %d \n",i,data_f[i]);
        //printf("\n");

        data_val[SOCKET][i] = ((double ((data_f[SOCKET][i]^0x800000)/8388608.0))-1.0)*(3.25*0.5/0.65);
    }

    if(Show_data_Value)
    {
        for(i=0; i< WG_Data_cnt; i++)
        {           printf("%3.3f", data_val[SOCKET][i]);
                    printf("\n");
        }
    }

    AWI_DATA_DC_ARITH(data_val[SOCKET], WG_Data_cnt, -0.5, 0.5, INL_value, DNL_value, gain_error, offset_error ) ;

        for(int gg=0;gg<WG_Data_cnt;gg++)
        {
            //-------------------------------------------------
            if(gg==0)
```

```
            {
                //-----------------------------------------------
                dnl_max = DNL_value[gg];
                dnl_min = DNL_value[gg];
                inl_max = INL_value[gg];
                inl_min = INL_value[gg];
                //-----------------------------------------------
                dnl_max_addr   = gg ;
                dnl_min_addr   = gg ;
                inl_max_addr   = gg ;
                inl_min_addr   = gg ;
                //-----------------------------------------------
            }
            else
            {
                //-----------------------------------------------
                if(DNL_value[gg]>dnl_max)
                {
                    dnl_max        =   DNL_value[gg];
                    dnl_max_addr=   gg ;
                }
                //-----------------------------------------------
                if(DNL_value[gg]<dnl_min)
                {
                    dnl_min        =   DNL_value[gg];
                    dnl_min_addr=   gg ;
                }
                //-----------------------------------------------
                if(INL_value[gg]>inl_max)
                {
                    inl_max        =   INL_value[gg];
                    inl_max_addr=   gg ;
                }
                //-----------------------------------------------
                if(INL_value[gg]<inl_min)
                {
                    inl_min        =   INL_value[gg];
                    inl_min_addr=   gg ;
                }
                //-----------------------------------------------
            }
            //-----------------------------------------------
        }
        //-----------------------------------------------

        //-----------------------------------------------
    DATALOG_MSG="ramp_312_5K_test";

        TEST_NO(10);
        JUDGE_VARIABLE (SOCKET, dnl_max, -10, 10, "DNL_MAX");
        TEST_NO(20);
        JUDGE_VARIABLE (SOCKET, inl_max, -10, 10, "INL_MAX");
        TEST_NO(30);
        JUDGE_VARIABLE (SOCKET, dnl_min, -10, 10, "DNL_MIN");
        TEST_NO(40);
        JUDGE_VARIABLE (SOCKET, inl_min, -10, 10, "INL_MIN");
    TEST_NO(03);

    //==***************** Print Data to File !!! *****************==//
    sprintf(msg,"RAW_312_5K_Site[%d]_%d.dat",SOCKET,test_cnt);
```

```
        stream0 = fopen(msg, "w+");
        for(i=0; i< WG_Data_cnt; i++)        fprintf(stream0,"%f \n",data_val[SOCKET][i]);
        fclose(stream0);

        //==***************** Print Data to File !!! *****************==//
   }

   SET_CAPTURE_MEM_MODE(DB_PINS,PEL_K,OFF,8K);
   WG_RELAY_OFF (MAWI2, WG0, WG, GND, 2mS);
      OSC_CLK_MODE(MCLK, OFF);

   power_down();
}
```

4.23.6 USE_WD_WAVEFORM

USE_SET_WD_WAVEFORM – CRAFT statement

Select WD waveform set.

Release version : **CRAFT_3380_2.1**

4.23.6.1 Format

```
USE_WD_WAVEFORM (Brd_type,wd_module_name, delay);
```

Parameter comment:

Brd_type :MAWI2
**no_entry: illegal

module_name : define in AWD condition file
**no_entry; illegal

delay : Variables and constants are allowed.
Type: TIME or double
**no_entry; 1mS

4.23.6.2 Example

User could send AWI2 WD signal to PCM5102.

```
WD_SINE_func_5M_Without_LPF()
{
   int w,i,j,adc_code,log_cnt;
   double value;
   FILE *stream0;
   char msg[1024];
   double noise,snr,sdr,sndr,mag_fund,dc_offset;
   int Dnl_flag=0,Inl_flag=0,log_cnt_temp=0;
   double max_offset,max_gain,dac_offset,dac_gain;

   DEBUG_BREAK();
   log_cnt = pow(2.0,test_bit);
   log_cnt_temp = log_cnt+1;

   DEBUG_BREAK();

   SET_CBIT(Audio_UR,ON,3mS);
```

```
SET_JUDGE_MODE (DBL);

PIN_MODE(CTL_A_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE(          BCK, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE(          SCK,   RZ, EDGE, ENABLE, IO_NRZ, OFF);

RATEN(TS1, 20.0nS); //50.00Mhz
MARKERN(TS1, CTL_A_PINS,     0.000nS,      0.0nS, ,0nS,0nS, 17.449nS, 19.66nS);
MARKERN(TS1,            BCK,        0.0nS,      0.0nS, ,0nS,0nS, , );
MARKERN(TS1,            SCK,10.0nS, 20.0nS, ,0nS,0nS, , );

SET_LEVELN(func_lvl, ALLPINS_A, 0V, 3.3V, 1.0V, 2.3V, 0uA, 0uA, 0V, , );
LEVELS(func_lvl, 10mS);

RELAY_ON(ALLPINS_A,PDCL,2mS);

SOCKET_INC {

        if(AWI2_SW){

            CRAFT_STATEMENT
            {
                USE_WD_WAVEFORM(MAWI2,Without_LPF,5mS);
                WD_RELAY_ON(MAWI2,WD0,WD,GND,5mS);

                JUDGE_PAT(PCM5102_func_20140604,__PCM5102_func_20140604);
            //==***************** Print Data to File !!! *****************==//
                SOCKET_INC
                {
                        if(SOCKET==0)
                        {

                READ_WD_DATA_MS(MAWI2,WD0,Without_LPF,data_tmp,8192);
```

```
                    sprintf(msg,"Sin_f_12K_amp_0_2V_Sa_5M_Without_LPF.dat");
                        stream0 = fopen(msg, "w+");
                  for(i=0; i<8192; i++)
                      fprintf(stream0,"%f \n",data_tmp[SOCKET][0][i]);
                       fclose(stream0);
                      }
                    }
                //==***************** Print Data to File !!! *****************==//
                    WD_RELAY_OFF(MAWI2,WD0,ALL,ALL,5mS);
              }

          }
   }

   SET_JUDGE_MODE (NORM);
   RELAY_OFF(ALLPINS_A,PDCL,2mS);

   SET_CBIT(UR_ALLPINS,OFF,3mS);

   power_down();
}
```

Example of WD capture step trigger(*.pat)

```
SET_DEC_FILE    ".\PAT\example_pin.dec"

GLOBAL int       WD_Data_cnt        = 2048,
                  AWI2_SW             = 1;

GLOBAL double    Rate_X              = 32; // 6.144Mhz
double dgt_data[2][20000];
double INL_value[2][8192];
double DNL_value[2][8192];
int test_cnt;

int rst_lvl    = 1,
```

```
      func_lvl = 2,
      os_lvl     = 3;

HW_BIN_DEF
{
   all_pass                        =   1;
   fail_os_test                   = !3;
   fail_func_test                 = !4;
}

SW_BIN_DEF
{
   all_pass                        =   1;
   fail_open_short_test        = !3;
   fail_func_test                 = !4;
}

START_UP()
{
   SET_LEVELN(rst_lvl, IO_ALLPINS, 0.0V,0.0V,0.0V,0.0V,0.0mA,0.0mA,0.0V, -2V, 7V);
   SET_LEVELN( os_lvl, IO_ALLPINS, 0.0V,0.0V,0.0V,0.0V,0.0mA,0.0mA,0.0V, -2V, 7V);
   SET_LEVELN(func_lvl, IO_ALLPINS, 0V, 2.5V, 1V, 2.0V, 0uA, 0uA, 2V, , );
   LEVELS(rst_lvl, 3mS);
   RELAY_OFF( IO_ALLPINS, ALL, 1mS);
   FORCE_VOLT( MLDPS_PINS, MLDPS,_FORCE_V, 0V, _V_RNG, @6V, _I_RNG,
@500mA,
          _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY, OFF,_DELAY, 3mS);
   test_cnt = 1;
   LOAD_ADDA_WAVEFORM (MAWI2, "../Ramp_example_LV.awi2");
   LOAD_PAT("./PAT/example.ppo");
}

test_start()
{
```

```
   LEVELS(rst_lvl, 3mS) ;
   SET_CBIT(POWER_UR,ON,3mS);

   FORCE_VOLT( MLDPS_PINS, MLDPS,_FORCE_V, 0V, _V_RNG, @6V, _I_RNG,
@500mA,
          _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY, OFF,_DELAY, 3mS);
      RELAY_OFF( IO_ALLPINS, ALL, 3mS);
}

test_end()
{
   LEVELS(rst_lvl, 3mS) ;
   FORCE_VOLT( MLDPS_PINS, MLDPS,_FORCE_V, 0V, _V_RNG, @6V, _I_RNG,
@500mA,
          _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY, OFF,_DELAY, 3mS);
   RELAY_OFF( IO_ALLPINS, ALL, 3mS);
   SET_CBIT(UR_ALLPINS,OFF,3mS);

   test_cnt = test_cnt+1;
   }

power_down()
{
   LEVELS(rst_lvl, 3mS) ;

   FORCE_VOLT( MLDPS_PINS, MLDPS,_FORCE_V, 0V, _V_RNG, @6V, _I_RNG,
@500mA,
          _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY, ON,_DELAY, 3mS);

   FORCE_VOLT( MLDPS_PINS, MLDPS,_FORCE_V, 0V, _V_RNG, @6V, _I_RNG,
@500mA,
          _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY, OFF,_DELAY, 3mS);
   FORCE_V_PMU(PMU, 0V, @6V, @100mA, 100mA, OFF, 3mS);
   RELAY_OFF(IO_ALLPINS, ALL, 3mS);
```

```
   SET_CBIT(UR_ALLPINS,OFF,3mS);
}

RELEASE()
{
   LEVELS(rst_lvl, 2mS) ;

   FORCE_VOLT( MLDPS_PINS, MLDPS,_FORCE_V, 0V, _V_RNG, @6V, _I_RNG,
@500mA,
            _MEAS_TYPE, MI, _CLAMP_I, 100mA, _RELAY, OFF,_DELAY, 3mS);
   RELAY_OFF( IO_ALLPINS, ALL, 1mS);
}

TEST_PRO
{
   BEFORE_TEST: test_start;
      example_AWI2_WD_ramp              ? #C(all_pass, all_pass)    : #C(fail_func_test,
fail_func_test) ;
   AFTER_TEST: test_end ;
}

example_AWI2_WD_ramp()
{

   // example ----> MAWI2 WD

   int w,i,j,log_cnt;

   FILE *stream0;
   char msg[1024];
   CUSTOM_3DIM data_val;

     double gain_error[4]={0.0,0.0,0.0,0.0};
     double offset_error[4]={0.0,0.0,0.0,0.0};
```

```
  double inl_max[4]={-999.0,-999.0,-999.0,-999.0};
  double dnl_max[4]={-999.0,-999.0,-999.0,-999.0};
  double inl_min[4]={999.0,999.0,999.0,999.0};
  double dnl_min[4]={999.0,999.0,999.0,999.0};
  //-------------------------------------------------------
  //-------------------------------------------------------
      int      inl_max_addr[4], dnl_max_addr[4], inl_min_addr[4], dnl_min_addr[4];

  DEBUG_BREAK();
  USE_WD_WAVEFORM(MAWI2,   Ramp_example_LV_2048      , 2mS );
  DEBUG_BREAK();

  SET_LEVELN(func_lvl, ALLPINS_G, 0V, 2.5V, 1.0V, 2.0V, 0uA, 0uA, 0V, , );
  LEVELS(func_lvl, 10mS);
  SET_CBIT(General_UR,ON,3mS);
  SET_CBIT(UR_GO+UR_WD_N_G,ON,3mS);
  WD_RELAY_ON (MAWI2, wd0, WD, GND, 2mS);
  RELAY_ON(ALLPINS_G,PDCL,2mS);

  PIN_MODE(ALLPINS_G , NRZ, EDGE, ENABLE, IO_NRZ, OFF);
  PIN_MODE(         SCLK,   RZ, EDGE, ENABLE, IO_NRZ, OFF);

  RATEN(TS1, 100nS); //10Mhz
  MARKERN(TS1, ALLPINS_G,      5nS,     0nS, ,0nS,0nS, 50.0nS, 76nS);
  MARKERN(TS1,   SCLK      ,      0nS,     50nS, ,0nS,0nS, 95.0nS, 76nS);

  RATEN(TS2, 100nS*Rate_X); //10Mhz
  MARKERN(TS2, ALLPINS_G,      5nS,     0nS, ,0nS,0nS, 50.0nS*Rate_X,
76nS*Rate_X);
  MARKERN(TS2,        SCLK,      0nS,     50nS, ,0nS,0nS, 60.0nS*Rate_X,
0.0nS*Rate_X);

  RATEN(TS3, 1600ns); // 625K = 1600nS
  MARKERN(TS3, ALLPINS_G,      5nS,     0nS, ,0nS,0nS, 50nS    , );
```

```
MARKERN(TS3,    SCLK       ,       0nS,      50nS, ,0nS,0nS, 0nS      , );

//===== debug fixed voltage =====//
JUDGE_PAT(fun_ramp_2048_1V,__ fun_ramp_2048_1V);
//===== debug fixed voltage =====//

SOCKET_INC {

        if(AWI2_SW){

            CRAFT_STATEMENT
            {

             READ_WD_DATA_MS (MAWI2, wd0, Ramp_example_LV_2048         ,
data_val    ,WD_Data_cnt );

          for(int dd=0;dd<(WD_Data_cnt);dd++)
          {
             dgt_data[SOCKET][dd]=data_val[SOCKET][0][dd];
          }
          AWI_DATA_DC_ARITH(dgt_data[SOCKET], WD_Data_cnt, -2.5, 2.5,
INL_value[SOCKET], DNL_value[SOCKET], gain_error[SOCKET], offset_error[SOCKET] ) ;

          for(int gg=0;gg<WD_Data_cnt;gg++)
          {
             //-------------------------------------------------
             if(gg==0)
             {
                //-------------------------------------------------
                dnl_max[SOCKET] = DNL_value[SOCKET][gg];
                dnl_min[SOCKET] = DNL_value[SOCKET][gg];
                inl_max[SOCKET] = INL_value[SOCKET][gg];
                inl_min[SOCKET] = INL_value[SOCKET][gg];
                //-------------------------------------------------
```

```
            dnl_max_addr[SOCKET]    = gg ;
            dnl_min_addr[SOCKET]    = gg ;
            inl_max_addr[SOCKET]    = gg ;
            inl_min_addr[SOCKET]    = gg ;
            //-----------------------------------------------
        }
        else
        {
            //-----------------------------------------------
            if(DNL_value[SOCKET][gg]>dnl_max[SOCKET])
            {
                dnl_max        [SOCKET]=    DNL_value[SOCKET][gg];
                dnl_max_addr[SOCKET]=    gg ;
            }
            //-----------------------------------------------
            if(DNL_value[SOCKET][gg]<dnl_min[SOCKET])
            {
                dnl_min        [SOCKET]=    DNL_value[SOCKET][gg];
                dnl_min_addr[SOCKET]=    gg ;
            }
            //-----------------------------------------------
            if(INL_value[SOCKET][gg]>inl_max[SOCKET])
            {
                inl_max        [SOCKET]=    INL_value[SOCKET][gg];
                inl_max_addr[SOCKET]=    gg ;
            }
            //-----------------------------------------------
            if(INL_value[SOCKET][gg]<inl_min[SOCKET])
            {
                inl_min        [SOCKET]=    INL_value[SOCKET][gg];
                inl_min_addr[SOCKET]=    gg ;
            }
            //-----------------------------------------------
        }
```

```
                //-----------------------------------------------------
            }
            //-----------------------------------------------------
            //-----------------------------------------------------
        DATALOG_MSG="ramp_625K_test";

            TEST_NO(10);
            JUDGE_VARIABLE (SOCKET, dnl_max[SOCKET], -10, 10, "DNL_MAX");
            TEST_NO(20);
            JUDGE_VARIABLE (SOCKET, inl_max[SOCKET], -10, 10, "INL_MAX");
            TEST_NO(30);
            JUDGE_VARIABLE (SOCKET, dnl_min[SOCKET], -10, 10, "DNL_MIN");
            TEST_NO(40);
            JUDGE_VARIABLE (SOCKET, inl_min[SOCKET], -10, 10, "INL_MIN");

                //==***************** Print Data to File !!! *****************==//

                        sprintf(msg,"RAW_625K_Site[%d]_%d.dat",SOCKET,test_cnt);
                        stream0 = fopen(msg, "w+");
                    //for(i=0; i<log_cnt; i++)       fprintf(stream0,"%f \n",data_tmp[i]);
                        for(i=0; i<WD_Data_cnt; i++)        fprintf(stream0,"%f
\n",dgt_data[SOCKET][i]);
                    //for(i=15; i<4110; i++)       fprintf(stream0,"%f \n",data_tmp[i]);
                        fclose(stream0);
                //==***************** Print Data to File !!! *****************==//

              }

         }
   }

   JUDGE_PAT(fun_ramp_reset_example,__fun_ramp_reset_example); // reset Vout to 0V

   SET_JUDGE_MODE (NORM);
```

```
   RELAY_OFF(ALLPINS_G,PDCL,2mS);
   WD_RELAY_OFF (MAWI2, wd0, WD, GND, 2mS);
   SET_CBIT(UR_ALLPINS,OFF,3mS);

   power_down();
}
```

4.23.7 READ_WD_DATA_MS

READ_WD_DATA_MS-- CRAFT statement
Read digitizer acquire memory raw data
Release version : **CRAFT_3380_2.1**

4.23.7.1 Format

```
READ_WD_DATA_MS (Brd_type, pin_name, wd_module_name, dgt_data, Log_Cnt);
```

Parameter list:

pin_name : pin_name

**no_entry: illegal

Brd_type :MAWI2
**no_entry; illegal

dgt_data :

type:CUSTOM_3DIM

3-dimension array // array size: [site][pin][count]

**no_entry; illegal

CUSTOM_3DIM : CRAFT auot-allocate array size [site_no][pin_count][ave_cnt];

Log_Cnt : record data of total count

4.23.7.2 Example

```
WD_SINE_func_5M_Without_LPF()
{
   int w,i,j,adc_code,log_cnt;
   double value;
   FILE *stream0;
   char msg[1024];
   double noise,snr,sdr,sndr,mag_fund,dc_offset;
   int Dnl_flag=0,Inl_flag=0,log_cnt_temp=0;
   double max_offset,max_gain,dac_offset,dac_gain;
```

```
   DEBUG_BREAK();
   log_cnt = pow(2.0,test_bit);
   log_cnt_temp = log_cnt+1;

   DEBUG_BREAK();

   SET_CBIT(Audio_UR,ON,3mS);
SET_JUDGE_MODE (DBL);

   PIN_MODE(CTL_A_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   PIN_MODE(          BCK, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   PIN_MODE(          SCK,   RZ, EDGE, ENABLE, IO_NRZ, OFF);

   RATEN(TS1, 20.0nS); //50.00Mhz
   MARKERN(TS1, CTL_A_PINS,     0.000nS,      0.0nS, ,0nS,0nS, 17.449nS, 19.66nS);
   MARKERN(TS1,             BCK,        0.0nS,      0.0nS, ,0nS,0nS, , );
   MARKERN(TS1,             SCK,10.0nS, 20.0nS, ,0nS,0nS, , );

   SET_LEVELN(func_lvl, ALLPINS_A, 0V, 3.3V, 1.0V, 2.3V, 0uA, 0uA, 0V, , );
   LEVELS(func_lvl, 10mS);

   RELAY_ON(ALLPINS_A,PDCL,2mS);

   SOCKET_INC {
            if(AWI2_SW){
                CRAFT_STATEMENT
             {
                USE_WD_WAVEFORM(MAWI2,Without_LPF,5mS);
                WD_RELAY_ON(MAWI2,WD0,WD,GND,5mS);

                JUDGE_PAT(PCM5102_func_20140604,__PCM5102_func_20140604);
            //==***************** Print Data to File !!! *****************==//
                  SOCKET_INC
```

```
                    {
                        if(SOCKET==0)
                        {

                    READ_WD_DATA_MS(MAWI2,WD0,Without_LPF,data_tmp,8192);

                    sprintf(msg,"Sin_f_12K_amp_0_2V_Sa_5M_Without_LPF.dat");
                        stream0 = fopen(msg, "w+");
                     for(i=0; i<8192; i++)
                        fprintf(stream0,"%f \n",data_tmp[SOCKET][0][i]);
                        fclose(stream0);
                        }
                    }
                //==***************** Print Data to File !!! *****************==//
                    WD_RELAY_OFF(MAWI2,WD0,ALL,ALL,5mS);
                }

            }
    }

    SET_JUDGE_MODE (NORM);
    RELAY_OFF(ALLPINS_A,PDCL,2mS);

    SET_CBIT(UR_ALLPINS,OFF,3mS);

    power_down();
}
```

4.23.8 WG_RELAY_ON

WG_RELAY_ON – CRAFT statement
Release version : **CRAFT_3380_2.1**

4.23.8.1 Format

```
WG_RELAY_ON (Brd_type, Pin_name, P_Relay_type, N_Relay_type, Delay_time);
```

Parameter comment:

Brd_type :MAWI2
**no_entry; illegal

pin_name :pin_def, pin_group

**no_entry; illegal

P_Relay_type : PMU | WG

**no_entry; illegal

N_Relay_type : PMU | WG | GND

**no_entry; illegal

[Notice]: PMU Relay on➔ WG relay off.

WG Relay on➔ PMU relay off.

Delay_time : delay time after realy on, variables and constants are allowed.
Type: TIME or double
**no_entry; 1mS

4.23.8.2 Example

```
WD_SINE_func_test()
{
   int w,i,j,adc_code,log_cnt;
   double value;
   FILE *stream0;
```

```
char msg[1024];
double noise,snr,sdr,sndr,mag_fund,dc_offset;
double THD_value,SNR_value,SND_value, SFDR_value, THD_n_value;
int Dnl_flag=0,Inl_flag=0,log_cnt_temp=0;
double max_offset,max_gain,dac_offset,dac_gain;

DEBUG_BREAK();

log_cnt = pow(2.0,test_bit);
log_cnt_temp = log_cnt+1;

DEBUG_BREAK();

FORCE_V_DPS(VDD, 5.0V, @8V, @1A, 100mA, NORM, ON, 3mS);

PIN_MODE( CTL_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE(DATA_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);

RATEN(TS1, 100.0nS);
MARKERN(TS1, CTL_PINS,     0.000nS,      0.0nS, ,0nS,0nS, 25.0nS, 50.0nS);
MARKERN(TS1,DATA_PINS,     0.000nS,      0.0nS, ,0nS,0nS, 25.0nS, 50.0nS);

SET_LEVELN(AWI2_lvl, ALLPINS, 0V, 3.3V, 0.8V, 2.4V, 0uA, 0uA, 0V, , );
LEVELS(AWI2_lvl, 10mS);

RELAY_ON(ALLPINS,PDCL,2mS);

WAIT(10mS);

USE_WG_WAVEFORM (MAWI2,WG_1K,2mS);
WG_RELAY_ON (MAWI2, WG0, WG, GND, 2mS);

DEBUG_BREAK();
```

```
  SOCKET_INC {

        if(AWI2_SW){

            CRAFT_STATEMENT
            {
                USE_WD_WAVEFORM(MAWI2,Without_LPF,5mS);
                WD_RELAY_ON(MAWI2,WD0,WD,GND,5mS);

                DEBUG_BREAK();

                JUDGE_PAT(AD7111_func,__AD7111_func);


              //==***************** Print Data to File !!! *****************==//
                SOCKET_INC
                {

READ_WD_DATA_MS(MAWI2,WD0,Without_LPF,data_tmp,8192);

                    sprintf(msg,"AWI2_AD7111_site%d.dat",SOCKET);
                    stream0 = fopen(msg, "w+");
                    for(i=0; i<3125; i++)        fprintf(stream0,"%f
\n",data_tmp[SOCKET][0][i]);
                    fclose(stream0);
                }

                SOCKET_INC{

                AWI_DATA_AC_ARITH(data_tmp[0][0], 3125, THD_value, SNR_value,
SND_value, SFDR_value, THD_n_value) ;

                JUDGE_VARIABLE (SOCKET, SNR_value, 70,       , "SNR_dB");
                JUDGE_VARIABLE (SOCKET, THD_value,      , -70, "THD_dB");
```

```
                    }
            }

        }
    }
    FORCE_V_DPS(VDD, 0.0V, @8V, @1A, 100mA, NORM, ON, 3mS);
    RELAY_OFF(ALLPINS,PDCL,2mS);
    WG_RELAY_OFF(MAWI2,WG0,WG,GND,5mS);
    power_down();
}
```

4.23.9 WG_RELAY_OFF

WG_RELAY_OFF – CRAFT statement
Release version : **CRAFT_3380_2.1**

4.23.9.1 Format

```
WG_RELAY_OFF (Brd_type, Pin_name, P_Relay_type, N_Relay_type, Delay_time);
```

Parameter comment:

Brd_type :MAWI2
**no_entry; illegal

pin_name :pin_def, pin_group

**no_entry; illegal

P_Relay_type : PMU | WG | ALL

**no_entry; illegal

N_Relay_type : PMU | WG | GND | ALL

**no_entry; illegal

Delay_time : delay time after realy off, variables and constants are allowed.
Type: TIME or double
**no_entry; 1mS

4.23.9.2 Example

```
WD_SINE_func_test()
{
   int w,i,j,adc_code,log_cnt;
   double value;
   FILE *stream0;
   char msg[1024];
   double noise,snr,sdr,sndr,mag_fund,dc_offset;
   double THD_value,SNR_value,SND_value, SFDR_value, THD_n_value;
   int Dnl_flag=0,Inl_flag=0,log_cnt_temp=0;
   double max_offset,max_gain,dac_offset,dac_gain;

   DEBUG_BREAK();

   log_cnt = pow(2.0,test_bit);
   log_cnt_temp = log_cnt+1;
```

```
   DEBUG_BREAK();

   FORCE_V_DPS(VDD, 5.0V, @8V, @1A, 100mA, NORM, ON, 3mS);

   PIN_MODE( CTL_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   PIN_MODE(DATA_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);

   RATEN(TS1, 100.0nS);
   MARKERN(TS1, CTL_PINS,     0.000nS,       0.0nS, ,0nS,0nS, 25.0nS, 50.0nS);
   MARKERN(TS1,DATA_PINS,     0.000nS,       0.0nS, ,0nS,0nS, 25.0nS,
50.0nS);

   SET_LEVELN(AWI2_lvl, ALLPINS, 0V, 3.3V, 0.8V, 2.4V, 0uA, 0uA, 0V, , );
   LEVELS(AWI2_lvl, 10mS);

   RELAY_ON(ALLPINS,PDCL,2mS);

   WAIT(10mS);

   USE_WG_WAVEFORM (MAWI2,WG_1K,2mS);
   WG_RELAY_ON (MAWI2, WG0, WG, GND, 2mS);

   DEBUG_BREAK();

   SOCKET_INC {

           if(AWI2_SW){

               CRAFT_STATEMENT
               {
                       USE_WD_WAVEFORM(MAWI2,Without_LPF,5mS);
                       WD_RELAY_ON(MAWI2,WD0,WD,GND,5mS);

                       DEBUG_BREAK();

                       JUDGE_PAT(AD7111_func,__AD7111_func);


                   //==***************** Print Data to File !!! *****************==//
                       SOCKET_INC
                       {

READ_WD_DATA_MS(MAWI2,WD0,Without_LPF,data_tmp,8192);

                               sprintf(msg,"AWI2_AD7111_site%d.dat",SOCKET);
                               stream0 = fopen(msg, "w+");
                               for(i=0; i<3125; i++)        fprintf(stream0,"%f
\n",data_tmp[SOCKET][0][i]);
                               fclose(stream0);
                       }

                       SOCKET_INC{

                       AWI_DATA_AC_ARITH(data_tmp[0][0], 3125, THD_value,
SNR_value, SND_value, SFDR_value, THD_n_value) ;
```

```
                    JUDGE_VARIABLE (SOCKET, SNR_value, 70,       , "SNR_dB");
                    JUDGE_VARIABLE (SOCKET, THD_value,     , -70, "THD_dB");
                    }
            }

        }
    }

    FORCE_V_DPS(VDD, 0.0V, @8V, @1A, 100mA, NORM, ON, 3mS);
    RELAY_OFF(ALLPINS,PDCL,2mS);
    WG_RELAY_OFF(MAWI2,WG0,WG,GND,5mS);
    power_down();
}
```

4.23.10 WD_RELAY_ON

WD_RELAY_ON – CRAFT statement
Release version : **CRAFT_3380_2.1**

4.23.10.1 Format

```
WD_RELAY_ON (Brd_type, Pin_name, P_Relay_type, N_Relay_type, Delay_time);
```

Parameter comment:

Brd_type :MAWI2
**no_entry; illegal

pin_name :pin_def, pin_group

**no_entry; illegal

P_Relay_type : PMU | WD

**no_entry; illegal

N_Relay_type : PMU | WD | GND

**no_entry; illegal

[Notice]: PMU Relay on➔ WD relay off.

WD Relay on➔ PMU relay off.

Delay_time : delay time after realy on, variables and constants are allowed.
Type: TIME or double
**no_entry; 1mS

4.23.10.2 Example

```
WD_SINE_func_5M_Without_LPF()
{
   int w,i,j,adc_code,log_cnt;
   double value;
   FILE *stream0;
```

```
   char msg[1024];
   double noise,snr,sdr,sndr,mag_fund,dc_offset;
   int Dnl_flag=0,Inl_flag=0,log_cnt_temp=0;
   double max_offset,max_gain,dac_offset,dac_gain;

   DEBUG_BREAK();
   log_cnt = pow(2.0,test_bit);
   log_cnt_temp = log_cnt+1;

   DEBUG_BREAK();

   SET_CBIT(Audio_UR,ON,3mS);
SET_JUDGE_MODE (DBL);

   PIN_MODE(CTL_A_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   PIN_MODE(          BCK, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   PIN_MODE(          SCK,   RZ, EDGE, ENABLE, IO_NRZ, OFF);

   RATEN(TS1, 20.0nS); //50.00Mhz
   MARKERN(TS1, CTL_A_PINS,     0.000nS,      0.0nS, ,0nS,0nS, 17.449nS, 19.66nS);
   MARKERN(TS1,             BCK,        0.0nS,      0.0nS, ,0nS,0nS, , );
   MARKERN(TS1,             SCK,10.0nS, 20.0nS, ,0nS,0nS, , );

   SET_LEVELN(func_lvl, ALLPINS_A, 0V, 3.3V, 1.0V, 2.3V, 0uA, 0uA, 0V, , );
   LEVELS(func_lvl, 10mS);

   RELAY_ON(ALLPINS_A,PDCL,2mS);

   SOCKET_INC {

         if(AWI2_SW){

            CRAFT_STATEMENT
            {
```

```
                USE_WD_WAVEFORM(Without_LPF,5mS);
                WD_RELAY_ON(MAWI2,WD0,WD,GND,5mS);

                JUDGE_PAT(PCM5102_func_20140604,__PCM5102_func_20140604);
            //==***************** Print Data to File !!! *****************==//
                SOCKET_INC
                {
                    if(SOCKET==0)
                    {

                    READ_WD_DATA_MS(MAWI2,WD0,Without_LPF,data_tmp,8192);

                    sprintf(msg,"Sin_f_12K_amp_0_2V_Sa_5M_Without_LPF.dat");
                      stream0 = fopen(msg, "w+");
                   for(i=0; i<8192; i++)
                      fprintf(stream0,"%f \n",data_tmp[SOCKET][0][i]);
                      fclose(stream0);
                    }
                }
                //==***************** Print Data to File !!! *****************==//
                WD_RELAY_OFF(MAWI2,WD0,ALL,ALL,5mS);
            }

        }
    }

    SET_JUDGE_MODE (NORM);
    RELAY_OFF(ALLPINS_A,PDCL,2mS);
    SET_CBIT(UR_ALLPINS,OFF,3mS);

    power_down();
}
```

4.23.11 WD_RELAY_OFF

WD_RELAY_OFF – CRAFT statement
Release version : **CRAFT_3380_2.1**

4.23.11.1 Format

```
WD_RELAY_OFF (Brd_type, Pin_name, P_Relay_type, N_Relay_type, Delay_time);
```

Parameter comment:

Brd_type :MAWI2
**no_entry; illegal

pin_name :pin_def, pin_group

**no_entry; illegal

P_Relay_type : PMU | WD | ALL

**no_entry; illegal

N_Relay_type : PMU | WD | GND | ALL

**no_entry; illegal

Delay_time : delay time after realy off, variables and constants are allowed.
Type: TIME or double
**no_entry; 1mS

4.23.11.2 Example

```
WD_SINE_func_5M_Without_LPF()
{
   int w,i,j,adc_code,log_cnt;
   double value;
   FILE *stream0;
   char msg[1024];
   double noise,snr,sdr,sndr,mag_fund,dc_offset;
   int Dnl_flag=0,Inl_flag=0,log_cnt_temp=0;
   double max_offset,max_gain,dac_offset,dac_gain;

   DEBUG_BREAK();
```

```
   log_cnt = pow(2.0,test_bit);
   log_cnt_temp = log_cnt+1;

   DEBUG_BREAK();

   SET_CBIT(Audio_UR,ON,3mS);
SET_JUDGE_MODE (DBL);

   PIN_MODE(CTL_A_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   PIN_MODE(          BCK, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   PIN_MODE(          SCK,   RZ, EDGE, ENABLE, IO_NRZ, OFF);

   RATEN(TS1, 20.0nS); //50.00Mhz
   MARKERN(TS1, CTL_A_PINS,     0.000nS,      0.0nS, ,0nS,0nS, 17.449nS, 19.66nS);
   MARKERN(TS1,             BCK,        0.0nS,      0.0nS, ,0nS,0nS, , );
   MARKERN(TS1,             SCK,10.0nS, 20.0nS, ,0nS,0nS, , );

   SET_LEVELN(func_lvl, ALLPINS_A, 0V, 3.3V, 1.0V, 2.3V, 0uA, 0uA, 0V, , );
   LEVELS(func_lvl, 10mS);

   RELAY_ON(ALLPINS_A,PDCL,2mS);

   SOCKET_INC {

         if(AWI2_SW){

            CRAFT_STATEMENT
            {
               USE_WD_WAVEFORM(Without_LPF,5mS);
               WD_RELAY_ON(MAWI2,WD0,WD,GND,5mS);

               JUDGE_PAT(PCM5102_func_20140604,__PCM5102_func_20140604);
          //==***************** Print Data to File !!! *****************==//
                 SOCKET_INC
```

```
                    {
                            if(SOCKET==0)
                            {

                        READ_WD_DATA_MS(MAWI2,WD0,Without_LPF,data_tmp,8192);

                        sprintf(msg,"Sin_f_12K_amp_0_2V_Sa_5M_Without_LPF.dat");
                            stream0 = fopen(msg, "w+");
                        for(i=0; i<8192; i++)
                            fprintf(stream0,"%f \n",data_tmp[SOCKET][0][i]);
                            fclose(stream0);
                            }
                    }
                //==***************** Print Data to File !!! *****************==//
                    WD_RELAY_OFF(MAWI2,WD0,ALL,ALL,5mS);
                }

            }
    }

    SET_JUDGE_MODE (NORM);
    RELAY_OFF(ALLPINS_A,PDCL,2mS);
      SET_CBIT(UR_ALLPINS,OFF,3mS);

    power_down();
}
```

4.23.12 WG_LS_TRIG

WG_LS_TRIG – CRAFT statement

LS Command to trig WG

Release version : **CRAFT_3380_2.1**

4.23.12.1 Format

```
WG_LS_TRIG (Brd_type, pin_name, Trig_source, Mode, Delay_time);
```

Parameter comment:

Brd_type :MAWI2

**no_entry; illegal

Trig_source :IMMEDATE

Mode :

ON : Enable WG L.S trig mode and waveform start

OFF : Disable WG L.S trig mode and waveform Stop

**no_entry; illegal

Delay_time : variables and constants are allowed.
Type: TIME or double
**no_entry; 1mS

4.23.12.2 Example

```
START_UP()
{
   SET_LEVELN(rst_lvl, IO_ALLPINS, 0.0V,0.5V,0.0V,0.0V,0.0mA,0.0mA,0.0V, -2V, 7V);
   SET_LEVELN( os_lvl, IO_ALLPINS, 0.0V,0.0V,0.0V,0.0V,0.0mA,0.0mA,0.0V, -2V, 7V);
   SET_LEVELN(func_lvl, IO_ALLPINS, 0V, 5V, 1V, 3.0V, 0uA, 0uA, 2V, , );

   LEVELS(rst_lvl, 3mS);
   RELAY_OFF( IO_ALLPINS, ALL, 1mS);
```

```
   FORCE_VOLT( MLDPS_PINS, MLDPS,_FORCE_V, 0V, _V_RNG, @6V, _I_RNG, @500mA, _MEAS_TYPE, MI,
_CLAMP_I, 100mA, _RELAY, OFF,_DELAY, 3mS);

   LOAD_PAT("./PAT/dac_PCM5102.ppo");
  //1.load awg-pattern
   LOAD_ADDA_WAVEFORM("../WD_CAPTURE.awi2");
}

char r_str_0[32];
char r_str_1[32];

char* v_rng_str(int num)
{
   switch(num) {
      case 0 :   strcpy(r_str_0,"@LV")     ; break;
      case 1 :   strcpy(r_str_0,"@HV")     ; break;
      default:   strcpy(r_str_0,"unknow"); break;
   }
   return r_str_0;
}
char* wg_tc_str(int num)
{
   switch(num) {
      case 0 :   strcpy(r_str_1,"@8K")     ; break;
      case 1 :   strcpy(r_str_1,"@384K") ; break;
      default:   strcpy(r_str_1,"unknow"); break;
   }
   return r_str_1;
}
WG_WD_self_test_sample()
{
   FILE* fp;
   CUSTOM_3DIM dgt_data;
```

```
  char file_name[256];
  double thd,snr,snd,sfdr,thd_n;
  char msg_0[256];
  char msg_1[256];
  char     wd_tc_0_str[4][16]={"48K"    , "96K"      , "192K"    , "384K"};
  int       wd_tc_0_log_cnt[4]={352        , 704          , 1408        , 2816};

  RATEN(TS2,5uS);
  MARKERN(TS2, IO_ALLPINS, 0nS, 10nS, , 0nS, 10nS, 90nS, 100nS);

  DEBUG_BREAK();
  for(int wg_tc=1;wg_tc<2;wg_tc++)//change wg sample rate 8K....384K no use
  {
    for(int vrng=0;vrng<1;vrng++)//_LV_ :0 _HV_:1
    {

      fflush(stdout);
      WG_RELAY_ON (MAWI2, wg0 , WG, GND, 2mS);
      WD_RELAY_ON (MAWI2, wd0 , WD,    WD, 2mS);
      DEBUG_BREAK();
//-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------//
          int wg_count,wd_count;
          char wg_map[64][128],wd_map[64][128];
          DEBUG_BREAK();

          //2.wg use waveform set

          if(w_mode==0)
          {
            switch(vrng)
            {
              case 0:
              USE_WG_WAVEFORM(MAWI2, WG1P5K_LV, 5mS);
              break;
```

```
            case 1:
            USE_WG_WAVEFORM(MAWI2, WG1P5K_HV, 5mS);
            break;
          }
        }
        DEBUG_BREAK();

        for(int wd_tc=0;wd_tc<4;wd_tc++)
        {
          sprintf(file_name,"WD_data_%s_%d_%s_WG_%s.dat",wd_tc_0_str[wd_tc], v_rng_str(vrng),
wg_tc_str(wg_tc));
          if ((fp = fopen(file_name, "w")) == NULL)
          {
            printf("Cannot open input file.\n");
            return 1;
          }
           fflush(stdout);
           DEBUG_BREAK();
          //3.wd work mode
          if(w_mode==0)
          {
            switch(vrng)
            {
              case 0:
                if(wd_tc==0)   USE_WD_WAVEFORM(MAWI2, WD48K_LV,    wd_ws_delay );
                if(wd_tc==1)   USE_WD_WAVEFORM(MAWI2, WD96K_LV,    wd_ws_delay );
                if(wd_tc==2)   USE_WD_WAVEFORM(MAWI2, WD192K_LV, wd_ws_delay );
                if(wd_tc==3)   USE_WD_WAVEFORM(MAWI2, WD384K_LV, wd_ws_delay );
              break;

              case 1:
                if(wd_tc==0)   USE_WD_WAVEFORM(MAWI2, WD48K_HV,    wd_ws_delay );
                if(wd_tc==1)   USE_WD_WAVEFORM(MAWI2, WD96K_HV,    wd_ws_delay );
```

```
                if(wd_tc==2)   USE_WD_WAVEFORM(MAWI2, WD192K_HV, wd_ws_delay );
                if(wd_tc==3)   USE_WD_WAVEFORM(MAWI2, WD384K_HV, wd_ws_delay );
            break;
        }
    }

    DEBUG_BREAK();
    //4. trig

    if(trig_mode==0)
    {
        //WG trig
        WG_LS_TRIG (MAWI2, wg0, IMM, ON, 10mS);

        //WD trig
        WD_LS_TRIG (MAWI2, wd0, IMM, ON, 50mS);
    }
    else
    {
        if(wg_tc==0)   JUDGE_PAT(wg_wd_8K,__wg_wd_8K);
        else           JUDGE_PAT(wg_wd_st,wg_wd_sp);
    }

    //5.read data
    DEBUG_BREAK();

    switch(vrng)
    {
        case 0:
            if(wd_tc==0)   READ_WD_DATA_MS (MAWI2, wd0, WD48K_LV , dgt_data,
wd_tc_0_log_cnt[wd_tc]);
            if(wd_tc==1)   READ_WD_DATA_MS (MAWI2, wd0, WD96K_LV , dgt_data,
wd_tc_0_log_cnt[wd_tc]);
            if(wd_tc==2)   READ_WD_DATA_MS (MAWI2, wd0, WD192K_LV, dgt_data,
```

```
wd_tc_0_log_cnt[wd_tc]);
                    if(wd_tc==3)    READ_WD_DATA_MS (MAWI2, wd0, WD384K_LV, dgt_data,
wd_tc_0_log_cnt[wd_tc]);
                  break;


                  case 1:
                    if(wd_tc==0)    READ_WD_DATA_MS (MAWI2, wd0, WD48K_HV , dgt_data,
wd_tc_0_log_cnt[wd_tc]);
                    if(wd_tc==1)    READ_WD_DATA_MS (MAWI2, wd0, WD96K_HV , dgt_data,
wd_tc_0_log_cnt[wd_tc]);
                    if(wd_tc==2)    READ_WD_DATA_MS (MAWI2, wd0, WD192K_HV, dgt_data,
wd_tc_0_log_cnt[wd_tc]);
                    if(wd_tc==3)    READ_WD_DATA_MS (MAWI2, wd0, WD384K_HV, dgt_data,
wd_tc_0_log_cnt[wd_tc]);
                  break;
                }
                DEBUG_BREAK();
            for(int dd=0;dd<wd_tc_0_log_cnt[wd_tc];dd++){
                  fprintf(fp,"%lf\n",dgt_data[0][0][dd]);
                  fflush(stdout);
                }
                fclose(fp);
                DEBUG_BREAK();


                if(AC_Calculate)    AWI_DATA_AC_ARITH_EX(dgt_data[0][0],wd_tc_0_log_cnt[wd_tc], thd, snr, snd,
sfdr,thd_n,DFT);



sprintf(msg_0,"SNR-%s_WG_%d_%s_WD_%d_%s",v_rng_str(vrng),MT_Group_Pins[0],wg_tc_str(wg_tc),MT_Sing
le_Pin[0],wd_tc_0_str[wd_tc]);


sprintf(msg_1,"THD-%s_WG_%d_%s_WD_%d_%s",v_rng_str(vrng),MT_Group_Pins[0],wg_tc_str(wg_tc),MT_Sing
le_Pin[0],wd_tc_0_str[wd_tc]);
```

```
            JUDGE_VARIABLE (S0, snr    , 70.0, , msg_0);//;low lmit
            JUDGE_VARIABLE (S0, thd, ,-70.0    , msg_1);//;low lmit

            //WG trig
            WG_LS_TRIG (MAWI2, wg0, IMM, OFF, 10mS);

            //WD trig
            WD_LS_TRIG (MAWI2, wd0, IMM, OFF, 10mS);
           }
 //-------------------------------------------------------------------------------------------------------------------------------------------------//

        WG_RELAY_OFF (MAWI2, wg0 , ALL, ALL, 2mS);
        WD_RELAY_OFF (MAWI2, wd0 , ALL, ALL, 2mS);
      }
   }
   DEBUG_BREAK();
}
```

4.23.13 WD_LS_TRIG

WD_LS_TRIG – CRAFT statement

LS Command to trig WD

Release version : **CRAFT_3380_2.1**

4.23.13.1 Format

```
WD_LS_TRIG (Brd_type, pin_name, Trig_source, Mode, Delay_time);
```

Parameter comment:

Byd_type :MAWI2

**no_entry; illegal

Trig_source :IMMEDATE

**no_entry; illegal

Mode :

ON : Enable WD L.S trig mode and Capture start

OFF : Disable WD L.S trig mode and Capture Stop

**no_entry; illegal

Delay_time : variables and constants are allowed.
Type: TIME or double
**no_entry; 1mS

4.23.13.2 Example

```
START_UP()
{
   SET_LEVELN(rst_lvl, IO_ALLPINS, 0.0V,0.5V,0.0V,0.0V,0.0mA,0.0mA,0.0V, -2V, 7V);
   SET_LEVELN( os_lvl, IO_ALLPINS, 0.0V,0.0V,0.0V,0.0V,0.0mA,0.0mA,0.0V, -2V, 7V);
   SET_LEVELN(func_lvl, IO_ALLPINS, 0V, 5V, 1V, 3.0V, 0uA, 0uA, 2V, , );

   LEVELS(rst_lvl, 3mS);
   RELAY_OFF( IO_ALLPINS, ALL, 1mS);

   FORCE_VOLT( MLDPS_PINS, MLDPS,_FORCE_V, 0V, _V_RNG, @6V, _I_RNG, @500mA, _MEAS_TYPE, MI,
_CLAMP_I, 100mA, _RELAY, OFF,_DELAY, 3mS);

   LOAD_PAT("./PAT/dac_PCM5102.ppo");
  //1.load awg-pattern
   LOAD_ADDA_WAVEFORM("../WD_CAPTURE.awi2");
}

char r_str_0[32];
char r_str_1[32];

char* v_rng_str(int num)
{
```

```
   switch(num) {
      case 0 :   strcpy(r_str_0,"@LV")      ; break;
      case 1 :   strcpy(r_str_0,"@HV")      ; break;
      default:   strcpy(r_str_0,"unknow"); break;
   }
   return r_str_0;
}
char* wg_tc_str(int num)
{
   switch(num) {
      case 0 :   strcpy(r_str_1,"@8K")      ; break;
      case 1 :   strcpy(r_str_1,"@384K") ; break;
      default:   strcpy(r_str_1,"unknow"); break;
   }
   return r_str_1;
}
WG_WD_self_test_sample()
{
   FILE* fp;
   CUSTOM_3DIM dgt_data;
   char file_name[256];
   double thd,snr,snd,sfdr,thd_n;
   char msg_0[256];
   char msg_1[256];
   char     wd_tc_0_str[4][16]={"48K"    , "96K"      , "192K"    , "384K"};
   int      wd_tc_0_log_cnt[4]={352      , 704        , 1408      , 2816};

   RATEN(TS2,5uS);
   MARKERN(TS2, IO_ALLPINS, 0nS, 10nS, , 0nS, 10nS, 90nS, 100nS);

   DEBUG_BREAK();
   for(int wg_tc=1;wg_tc<2;wg_tc++)//change wg sample rate 8K....384K no use
   {
      for(int vrng=0;vrng<1;vrng++)//_LV_ :0 _HV_:1
      {

         fflush(stdout);
         WG_RELAY_ON (MAWI2, wg0 , WG, GND, 2mS);
         WD_RELAY_ON (MAWI2, wd0 , WD,    WD, 2mS);
         DEBUG_BREAK();
//-------------------------------------------------------------------------------------------------------------------------------------------------------------------//
            int wg_count,wd_count;
            char wg_map[64][128],wd_map[64][128];
            DEBUG_BREAK();

            //2.wg use waveform set

            if(w_mode==0)
            {
               switch(vrng)
               {
                  case 0:
                  USE_WG_WAVEFORM(MAWI2, WG1P5K_LV, 5mS);
                  break;

                  case 1:
                  USE_WG_WAVEFORM(MAWI2, WG1P5K_HV, 5mS);
                  break;
               }
            }
            DEBUG_BREAK();

            for(int wd_tc=0;wd_tc<4;wd_tc++)
            {
               sprintf(file_name,"WD_data_%s_%d_%s_WG_%s.dat",wd_tc_0_str[wd_tc], v_rng_str(vrng),
wg_tc_str(wg_tc));
               if ((fp = fopen(file_name, "w")) == NULL)
               {
                  printf("Cannot open input file.\n");
                  return 1;
               }
                fflush(stdout);
                DEBUG_BREAK();
               //3.wd work mode
               if(w_mode==0)
```

```
            {
              switch(vrng)
              {
                case 0:
                  if(wd_tc==0)   USE_WD_WAVEFORM(MAWI2, WD48K_LV,    wd_ws_delay );
                  if(wd_tc==1)   USE_WD_WAVEFORM(MAWI2, WD96K_LV,    wd_ws_delay );
                  if(wd_tc==2)   USE_WD_WAVEFORM(MAWI2, WD192K_LV, wd_ws_delay );
                  if(wd_tc==3)   USE_WD_WAVEFORM(MAWI2, WD384K_LV, wd_ws_delay );
                break;

                case 1:
                  if(wd_tc==0)   USE_WD_WAVEFORM(MAWI2, WD48K_HV,    wd_ws_delay );
                  if(wd_tc==1)   USE_WD_WAVEFORM(MAWI2, WD96K_HV,    wd_ws_delay );
                  if(wd_tc==2)   USE_WD_WAVEFORM(MAWI2, WD192K_HV, wd_ws_delay );
                  if(wd_tc==3)   USE_WD_WAVEFORM(MAWI2, WD384K_HV, wd_ws_delay );
                break;
              }
            }

            DEBUG_BREAK();
            //4. trig

            if(trig_mode==0)
            {
              //WG trig
              WG_LS_TRIG (MAWI2, wg0, IMM, ON, 10mS);

              //WD trig
              WD_LS_TRIG (MAWI2, wd0, IMM, ON, 50mS);
            }
            else
            {
              if(wg_tc==0)   JUDGE_PAT(wg_wd_8K,__wg_wd_8K);
              else              JUDGE_PAT(wg_wd_st,wg_wd_sp);
            }

            //5.read data
            DEBUG_BREAK();

            switch(vrng)
            {
              case 0:
                if(wd_tc==0)   READ_WD_DATA_MS (MAWI2, wd0, WD48K_LV , dgt_data,
wd_tc_0_log_cnt[wd_tc]);
                if(wd_tc==1)   READ_WD_DATA_MS (MAWI2, wd0, WD96K_LV , dgt_data,
wd_tc_0_log_cnt[wd_tc]);
                if(wd_tc==2)   READ_WD_DATA_MS (MAWI2, wd0, WD192K_LV, dgt_data,
wd_tc_0_log_cnt[wd_tc]);
                if(wd_tc==3)   READ_WD_DATA_MS (MAWI2, wd0, WD384K_LV, dgt_data,
wd_tc_0_log_cnt[wd_tc]);
              break;

              case 1:
                if(wd_tc==0)   READ_WD_DATA_MS (MAWI2, wd0, WD48K_HV , dgt_data,
wd_tc_0_log_cnt[wd_tc]);
                if(wd_tc==1)   READ_WD_DATA_MS (MAWI2, wd0, WD96K_HV , dgt_data,
wd_tc_0_log_cnt[wd_tc]);
                if(wd_tc==2)   READ_WD_DATA_MS (MAWI2, wd0, WD192K_HV, dgt_data,
wd_tc_0_log_cnt[wd_tc]);
                if(wd_tc==3)   READ_WD_DATA_MS (MAWI2, wd0, WD384K_HV, dgt_data,
wd_tc_0_log_cnt[wd_tc]);
              break;
            }
            DEBUG_BREAK();
        for(int dd=0;dd<wd_tc_0_log_cnt[wd_tc];dd++){
              fprintf(fp,"%lf\n",dgt_data[0][0][dd]);
              fflush(stdout);
            }
            fclose(fp);
            DEBUG_BREAK();

            if(AC_Calculate)   AWI_DATA_AC_ARITH_EX(dgt_data[0][0],wd_tc_0_log_cnt[wd_tc], thd, snr, snd,
sfdr,thd_n,DFT);
```

```
sprintf(msg_0,"SNR-%s_WG_%d_%s_WD_%d_%s",v_rng_str(vrng),MT_Group_Pins[0],wg_tc_str(wg_tc),MT_Single
_Pin[0],wd_tc_0_str[wd_tc]);

sprintf(msg_1,"THD-%s_WG_%d_%s_WD_%d_%s",v_rng_str(vrng),MT_Group_Pins[0],wg_tc_str(wg_tc),MT_Single
_Pin[0],wd_tc_0_str[wd_tc]);

            JUDGE_VARIABLE (S0, snr    , 70.0, , msg_0);//;low lmit
            JUDGE_VARIABLE (S0, thd, ,-70.0    , msg_1);//;low lmit

            //WG trig
            WG_LS_TRIG (MAWI2, wg0, IMM, OFF, 10mS);

            //WD trig
            WD_LS_TRIG (MAWI2, wd0, IMM, OFF, 10mS);
           }
 //-------------------------------------------------------------------------------------------------------------------------------------------------------------------//

        WG_RELAY_OFF (MAWI2, wg0 , ALL, ALL, 2mS);
        WD_RELAY_OFF (MAWI2, wd0 , ALL, ALL, 2mS);
     }
   }
   DEBUG_BREAK();
}
```

4.23.14 INIT_WG_STATE

INIT_WG_STATE – CRAFT statement
Low speed command could set WG state at first point of waveform before pattern starts.

Release version : **CRAFT_3380_2.1**

4.23.14.1 Format

```
INIT_WG_STATE (Brd_type, pin_name);
```

Parameter comment:

Brd_type :MAWI2
**no_entry; illegal
pin_name :pin_def, pin_group
**no_entry; illegal

4.23.14.2 Example

The steps of INIT_WG_STATE:
1)Setting " WG_RELAY_ON " for protecting WG relay.
2)Then "INIT_WG_STATE", ATE will send the voltage of WG setting at first point.
3)Finally, setting "JUDGE_PAT".

```
WD_SINE_func_test()
{
   int w,i,j,adc_code,log_cnt;
   double value;
   FILE *stream0;
   char msg[1024];
   double noise,snr,sdr,sndr,mag_fund,dc_offset;
   double THD_value,SNR_value,SND_value, SFDR_value, THD_n_value;
   int Dnl_flag=0,Inl_flag=0,log_cnt_temp=0;
   double max_offset,max_gain,dac_offset,dac_gain;

   DEBUG_BREAK();

   log_cnt = pow(2.0,test_bit);
   log_cnt_temp = log_cnt+1;

   DEBUG_BREAK();

   FORCE_V_DPS(VDD, 5.0V, @8V, @1A, 100mA, NORM, ON, 3mS);

   PIN_MODE( CTL_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   PIN_MODE(DATA_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);

   RATEN(TS1, 100.0nS);
   MARKERN(TS1, CTL_PINS,     0.000nS,      0.0nS, ,0nS,0nS, 25.0nS, 50.0nS);
   MARKERN(TS1,DATA_PINS,     0.000nS,      0.0nS, ,0nS,0nS, 25.0nS, 50.0nS);
```

```
SET_LEVELN(AWI2_lvl, ALLPINS, 0V, 3.3V, 0.8V, 2.4V, 0uA, 0uA, 0V, , );
LEVELS(AWI2_lvl, 10mS);

RELAY_ON(ALLPINS,PDCL,2mS);

WAIT(10mS);

USE_WG_WAVEFORM (MAWI2,WG_1K,2mS);
WG_RELAY_ON (MAWI2, WG0, WG, GND, 2mS);

DEBUG_BREAK();

SOCKET_INC {

        if(AWI2_SW){

            CRAFT_STATEMENT
            {
                USE_WD_WAVEFORM(MAWI2,Without_LPF,5mS);
                WD_RELAY_ON(MAWI2,WD0,WD,GND,5mS);

                INIT_WG_STATE(MAWI2,wg0);

                DEBUG_BREAK();

                JUDGE_PAT(AD7111_func,__AD7111_func);


              //==**************** Print Data to File !!! ****************==//
                SOCKET_INC
                {

READ_WD_DATA_MS(MAWI2,WD0,Without_LPF,data_tmp,8192);

                        sprintf(msg,"AWI2_AD7111_site%d.dat",SOCKET);
                        stream0 = fopen(msg, "w+");
                        for(i=0; i<3125; i++)        fprintf(stream0,"%f
\n",data_tmp[SOCKET][0][i]);
                        fclose(stream0);
                }

                SOCKET_INC{

                AWI_DATA_AC_ARITH(data_tmp[0][0], 3125, THD_value,
SNR_value, SND_value, SFDR_value, THD_n_value) ;

                JUDGE_VARIABLE (SOCKET, SNR_value, 70,       , "SNR_dB");
                JUDGE_VARIABLE (SOCKET, THD_value,     , -70, "THD_dB");
                }
            }

        }
}
```

```
    FORCE_V_DPS(VDD, 0.0V, @8V, @1A, 100mA, NORM, ON, 3mS);
    RELAY_OFF(ALLPINS,PDCL,2mS);
    WG_RELAY_OFF(MAWI2,WG0,WG,GND,5mS);
    power_down();
}
```

4.23.15 SET_WG_OFFSET

SET_WG_OFFSET – CRAFT statement

This statement will set WG offset voltage.

Release version : **CRAFT_3380_2.50-SP3**

4.23.15.1 Format

```
SET_WG_OFFSET (Brd_type, pin_name, Value);
```

Parameter comment:

Brd_type	:MAWI2 **no_entry; illegal
pin_name	:pin_def, pin_group **no_entry; illegal
Value:	-4.0V to +4.0V (LV range) -8.0V to +8.0V (HV range) variables & constants allowed type: VOLTAGE or double unit: V, pV, nV, uV, mV **no_entry: illegal

4.23.15.2 Example

```
WD_SINE_func_test()
{
   int w,i,j,adc_code,log_cnt;
   double value;
   FILE *stream0;
   char msg[1024];
   double noise,snr,sdr,sndr,mag_fund,dc_offset;
   double THD_value,SNR_value,SND_value, SFDR_value, THD_n_value;
   int Dnl_flag=0,Inl_flag=0,log_cnt_temp=0;
   double max_offset,max_gain,dac_offset,dac_gain;

   DEBUG_BREAK();

   log_cnt = pow(2.0,test_bit);
   log_cnt_temp = log_cnt+1;

   DEBUG_BREAK();

   FORCE_V_DPS(VDD, 5.0V, @8V, @1A, 100mA, NORM, ON, 3mS);

   PIN_MODE( CTL_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   PIN_MODE(DATA_PINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);

   RATEN(TS1, 100.0nS);
```

```
MARKERN(TS1, CTL_PINS,     0.000nS,      0.0nS, ,0nS,0nS, 25.0nS, 50.0nS);
MARKERN(TS1,DATA_PINS,     0.000nS,      0.0nS, ,0nS,0nS, 25.0nS, 50.0nS);

SET_LEVELN(AWI2_lvl, ALLPINS, 0V, 3.3V, 0.8V, 2.4V, 0uA, 0uA, 0V, , );
LEVELS(AWI2_lvl, 10mS);

RELAY_ON(ALLPINS,PDCL,2mS);

WAIT(10mS);

USE_WG_WAVEFORM (MAWI2,WG_1K,2mS);
SET_WG_OFFSET (MAWI2, wg0, 1.0);
WG_RELAY_ON (MAWI2, WG0, WG, GND, 2mS);

DEBUG_BREAK();

SOCKET_INC {

        if(AWI2_SW){

            CRAFT_STATEMENT
            {
                USE_WD_WAVEFORM(MAWI2,Without_LPF,5mS);
                WD_RELAY_ON(MAWI2,WD0,WD,GND,5mS);
                INIT_WG_STATE(MAWI2,wg0);
                DEBUG_BREAK();
                JUDGE_PAT(AD7111_func,__AD7111_func);

              //==***************** Print Data to File !!! *****************==//
                SOCKET_INC
                {

READ_WD_DATA_MS(MAWI2,WD0,Without_LPF,data_tmp,8192);

                        sprintf(msg,"AWI2_AD7111_site%d.dat",SOCKET);
                        stream0 = fopen(msg, "w+");
                        for(i=0; i<3125; i++)        fprintf(stream0,"%f
\n",data_tmp[SOCKET][0][i]);
                        fclose(stream0);
                }

                SOCKET_INC{

                AWI_DATA_AC_ARITH(data_tmp[0][0], 3125, THD_value,
SNR_value, SND_value, SFDR_value, THD_n_value) ;

                JUDGE_VARIABLE (SOCKET, SNR_value, 70,       , "SNR_dB");
                JUDGE_VARIABLE (SOCKET, THD_value,     , -70, "THD_dB");
                }
            }

        }
}

FORCE_V_DPS(VDD, 0.0V, @8V, @1A, 100mA, NORM, ON, 3mS);
```

```
    RELAY_OFF(ALLPINS,PDCL,2mS);
    WG_RELAY_OFF(MAWI2,WG0,WG,GND,5mS);
    power_down();
}
```

4.23.16 ADDA_PARAL_INC

This is a program flow control statement to ADDA board (ex: MAWI2…). It provides a compact test loop based on the reserved word PIN_DEFS as the loop increment. PIN_DEFS are used to represent each pin_group in the loop, one at a time.
Release version : **CRAFT_3380_2.20_SP1**

4.23.16.1 Format

```
ADDA_PARAL_INC (pin_name) {
    statement;
    .
    .
    statement;
}
```

Parameter comment:

pin_name: pin definition. (only pin name can't fill pin group name)
No entry: illegal
statement: Any valid tester or C statement.

4.23.16.2 Explanation

To avoid using same PMU channel at one time.

3380 PMU MAP

3380P/3380D			
	MAWI2 channel	MAWI2(Positive & Negative)	PMU channel
WD	0	WD0+	0
	1	WD0-	1
	2	WD2+	8
	3	WD2-	9
	4	WD4+	0
	5	WD4-	1
	6	WD6+	8
	7	WD6-	9
WG	8	WG8+	0
	9	WG8-	1
	10	WG10+	8

	11	WG10-	9
	12	WG12+	0
	13	WG12-	1
	14	WG14+	8
	15	WG14-	9

Software channel assign:
WD+ : channel num=slot*64 + (0, 2, 4, 6)
WG+ : channel num=slot*64 + (8, 10, 12, 14)

Dec File

```
PIN_LIST    (MAWI2_LPC6 ) {
    // e.g; 2 site, per-site 3 pin
      wg1    =   396 : 392        = 0      = MAWI2_WG;
      wg2    =   394 : 398        = 1      = MAWI2_WG;
      wd1    =   386 : 388        = 2      = MAWI2_WD;
}
```

Pln File

```
ADDA_PARAL_INC(wg1+wg2+wd1)
{
      //1st loop : channel   396(PMU0)/394(PMU8)
      //2st loop : channel   392(PMU0)/386(PMU8)
      //3st loop : channel   388(PMU0)/398(PMU8)
}
```

4.23.16.3 Example

```
ADDA_OS_TEST(){

  DATALOG_MSG = "ADDA_OS_TEST" ;

  FORCE_VOLT(VCC , MXUVI ,
                _FORCE_V, 0V,
                _V_RNG, @6V,
                _I_RNG, @100mA,
                _MEAS_TYPE, MI,
                _CLAMP_I, 100mA,
                _RELAY , ON,
                _DELAY, 2mS
            );

  ADDA_PARAL_INC(wg1+wg2)
  {
    WG_RELAY_ON(MAWI2,PIN_DEFS,PMU,PMU,2mS);
    FORCE_I_PMU(PMU, 100uA, @100uA, @6V, 2V, ON, 2mS);
    JUDGE_V_PMU(PMU, IFVM, 0.2V, 1.5V, 2mS, 32, AVE, 50uS, , );

    FORCE_I_PMU(PMU, 0A, @100uA, @6V, 2V, OFF, 2mS);
    WG_RELAY_OFF(MAWI2,PIN_DEFS, PMU,PMU,2mS);

  }

  FORCE_V_PMU (PMU, 0V, @6V, @1mA, 1mA, OFF, 2mS);
    FORCE_VOLT(VCC , MXUVI ,
```

```
                    _FORCE_V, 0V,
                    _V_RNG, @6V,
                    _I_RNG, @100mA,
                    _MEAS_TYPE, MI,
                    _CLAMP_I, 100mA,
                    _RELAY , OFF,
                    _DELAY, 2mS
          );
}
```

4.23.17 ADDA_PIN_INC

This is a program flow control statement to ADDA board (ex: MAWI2…). It provides a compact test loop based on the reserved word PIN_DEF as the loop increment. PIN_DEF are used to represent each pin_group in the loop, one at a time.
Release version : **CRAFT_3380_2.20_SP1**

4.23.17.1 Format

```
ADDA_PIN_INC (pin_name) {
      statement;
      .
      .
      statement;
}
```

Parameter comment:

pin_name: pin definition. (only pin name can't fill pin group name)
No entry: illegal
statement: Any valid tester or C statement.

4.23.17.2 Explanation

Always keep one pin_name at one time, test the loop-pin.
And loop count will be total pin numbers.

3380 PMU MAP

3380P/3380D			
	MAWI2 channel	MAWI2(Positive & Negative)	PMU channel
WD	0	WD0+	0
	1	WD0-	1
	2	WD2+	8
	3	WD2-	9
	4	WD4+	0
	5	WD4-	1
	6	WD6+	8
	7	WD6-	9
WG	8	WG8+	0
	9	WG8-	1
	10	WG10+	8
	11	WG10-	9

	12	WG12+	0
	13	WG12-	1
	14	WG14+	8
	15	WG14-	9

Channel assign:
WD : channel num=slot*64 + (0, 2, 4, 6)
WG : channel num=slot*64 + (8, 10, 12, 14)

Dec File

```
PIN_LIST    (MAWI2_LPC6 ) {
    // e.g; 2 site, per-site 3 pin
      wg1    =   396 : 392        = 0      = MAWI2_WG;
      wg2    =   394 : 398        = 1      = MAWI2_WG;
      wd1    =   386 : 388        = 2      = MAWI2_WD;
}
```

Pln File

```
ADDA_PIN_INC(wg1+wg2+wd1)
{
      //1st loop : channel   396(PMU0)
      //2st loop : channel   392(PMU0)
      //3st loop : channel   394(PMU8)
      //4st loop : channel   398(PMU8)
      //5st loop : channel   388(PMU0)/386(PMU8)
}
```

4.23.17.3 Example

```
ADDA_OS_TEST(){

  DATALOG_MSG = "ADDA_OS_TEST" ;

  FORCE_VOLT(VCC , MXUVI ,
                _FORCE_V, 0V,
                _V_RNG, @6V,
                _I_RNG, @100mA,
                _MEAS_TYPE, MI,
                _CLAMP_I, 100mA,
                _RELAY , ON,
                _DELAY, 2mS
            );

  ADDA_PIN_INC(wg1+wg2)
  {
    WG_RELAY_ON(MAWI2,PIN_DEF,PMU,PMU,2mS);
    FORCE_I_PMU(PMU, 100uA, @100uA, @6V, 2V, ON, 2mS);
    JUDGE_V_PMU(PMU, IFVM, 0.2V, 1.5V, 2mS, 32, AVE, 50uS, , );

    FORCE_I_PMU(PMU, 0A, @100uA, @6V, 2V, OFF, 2mS);
    WG_RELAY_OFF(MAWI2,PIN_DEF, PMU,PMU,2mS);

  }

  FORCE_V_PMU (PMU, 0V, @6V, @1mA, 1mA, OFF, 2mS);
```

```
   FORCE_VOLT(VCC , MXUVI ,
                    _FORCE_V, 0V,
                    _V_RNG, @6V,
                    _I_RNG, @100mA,
                    _MEAS_TYPE, MI,
                    _CLAMP_I, 100mA,
                    _RELAY , OFF,
                    _DELAY, 2mS
             );

}
```

5. Plan Program of Extension

5.1 Others

5.1.1 R_REG

This statement read a value from a register.

5.1.1.1 Format

Parameter list:

R_REG (address, reg_value);

Parameter comment:

address: Register address to be read.
Type: unsigned integer
No entry: illegal

reg_value: The value be read from register.
Type: unsigned integer (If type is int, the returned value will be wrong.)

5.1.1.2 Explanation

Read a value from a register. Must be very familiar with the hardware registers and functionality.

5.1.1.3 Example

```
unsigned int reg_value;

R_REG (0x55, reg_value);
```

5.1.2 W_REG

This statement writes a value to a register.

5.1.2.1 Format

Parameter list:

```
W_REG (address, value);
```

Parameter comment:

address: Register address to be written.
Type: unsigned integer
No entry: illegal

value: Register address to be written.
Type: unsigned integer
No entry: illegal

5.1.2.2 Explanation

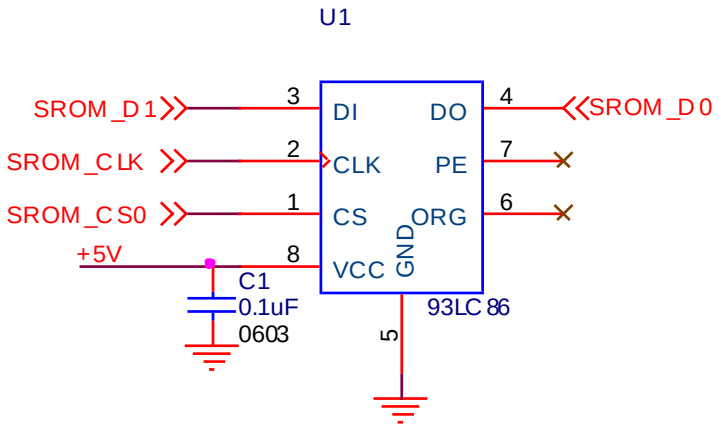
Write a value to a register. Must be very familiar with the hardware registers and functionality.

5.1.2.3 Example

```
unsigned int reg_value;

W_REG (0x55, reg_value);
```

5.1.3 READ_LOAD_BOARD_ID

The statement read back the load board ID. **We can use the statement to read ID of prober card or ID of dut board from EEPROM. User couldn't use this statement for 3380 offical load board.**



5.1.3.1 Format

Parameter list:

status = **READ_LOAD_BOARD_ID** (board_id, serial_no, model, type, date) ;

Parameter comment:

status : type integer

0 : EEPROM not exist

1 : EEPROM exist

board_id: The returned load board serial number.
Type: STRING (Maximum 32 Characters);

serial_no: The returned load board serial number.
Type: STRING (Maximum 32 Characters);

model: The returned load board model.
Type: STRING (Maximum 32 Characters);

type: The returned type.
Type: STRING (Maximum 16 Characters);

date: The returned manufacture date.
Type: STRING (Maximum 16 Characters);

5.1.3.2 Explanation

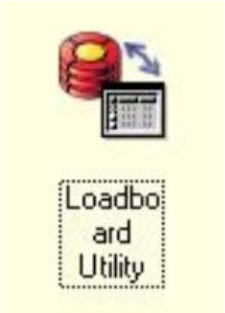
The statement read back the load board ID.

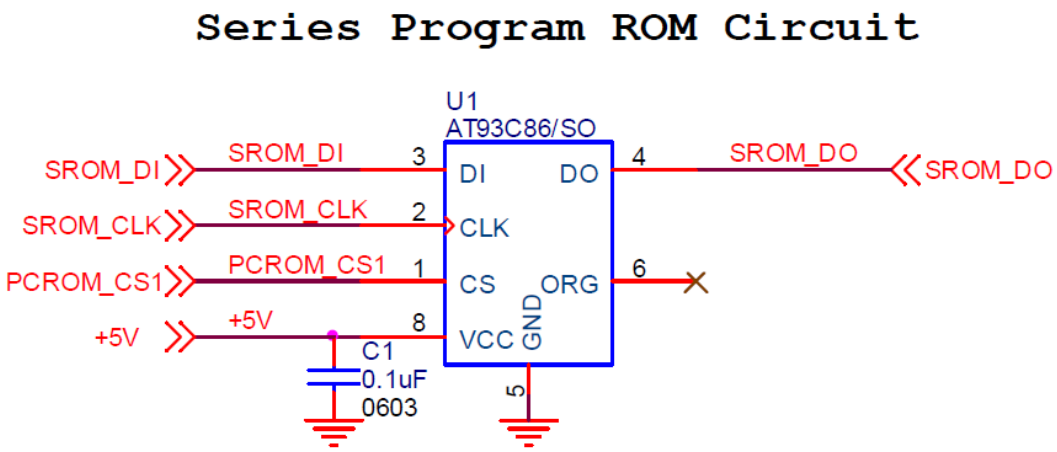
5.1.3.3 Example

```
char    board_id[32], serial_no[32], model[32], type[16], date[16], comment[64];

int status ;

        status = READ_LOAD_BOARD_ID( board_id, serial_no, model, type, date,
        comment);

        if(status == 1)

        {

            if( strcmp("3380_Dut_board", board_id) !=0 )     // ID is not matched.

            {

                printf("board_id = %s \n", board_id );

                printf("the Load Board used for : %s \n", comment);

            ABORT_TEST_PLN(); //exit test program

            }

        }

        else    //EEPROM don't exist.

        {

        }
```

5.1.4 WRITE_LOAD_BOARD_ID

The statement writes the dut board ID to the serial EEPROM on load board. **The statement is for prober card and dut board. Other, user could use CRAFT tool of "Loadboard Utility". The tool has the same ability to write ID in prober card or dut board.**





5.1.4.1 Format

Parameter list:

WRITE_LOAD_BOARD_ID("load_board_id", "serial_no", "model", "type", "date") ;

Parameter comment:

load_board_id: Dut board ID, (Maximum 32 character).
Type: STRING;

serial_no : Serial number, (Maximum 32 character)
Type: STRING;

model: Tester model, (Maximum 32 character)
Type: STRING;

type: Type, (Maximum 16 character)
Type: STRING;

date: Date, (Maximum 16 char)
Type: STRING;

5.1.4.2 Explanation

The statement writes the load board ID to the serial EEPROM on load board.

5.1.4.3 Example

```
WRITE_LOAD_BOARD_ID ("STN3500-AC-Diag-Ver1.0", "123456-032502",
                                        "STN3500", "MULTI-LEVEL", "5/16/2002");
```

```
START_UP()
{

WRITE_LOAD_BOARD_ID ("3380_prober_card_ver1.0", "A",

"3380", "direct mount ", "6/27/2014");

LOAD_PAT (". / PAT / device_CP.ppo" );
}
```

5.1.5 SHOW_TEST_TIME

The statement enables the timer to list all test items' test time and total test time.

5.1.5.1 Format

Parameter list:

```
SHOW_TEST_TIME (flag) ;
```

Parameter comment:
Flag: If flag is true then enable the timer else and disable the timer.

5.1.5.2 Explanation

The statement enable timer to list all test items' test time and total test time.

5.1.5.3 Example

```
GLOBAL   int   list_test_time = 0 ;
SHOW_TEST_TIME (list_test_time);
```

5.1.6 TIMER_ON

The statement enables the timer.

5.1.6.1 Format

Parameter list:

```
TIMER_ON ( ) ;
```

Parameter comment:
None.

5.1.6.2 Explanation

The statement enables the timer and returns the time from "TIMER_OFF" to "TIMER_ON".

5.1.6.3 Example

```
TIMER_ON ( ) ;
…     //statement
time_value = TIMER_OFF( );
printf ("time_value = %f\n", time_value);
```

5.1.7 TIMER_OFF

The statement disables the timer and returns the time from "TIMER_ON" to "TIMER_OFF".

5.1.7.1 Format

Parameter list:

```
time_value = TIMER_OFF ( ) ;
```

Parameter comment:
time_value: Type: double;
Unit: Sec

5.1.7.2 Explanation

The statement disables the timer and returns the time from "TIMER_ON" to "TIMER_OFF".

5.1.7.3 Example

```
TIMER_ON ( ) ;
…     //statement
time_value = TIMER_OFF( );
printf ("time_value = %f\n", time_value);
```

5.1.8 SPEC_DEF

Pre-define spec variable for using in test program.

5.1.8.1 Format

Parameter list:

```
SPEC_DEF   (spec_table_name) {
      Var_type   var_name = min_value : typ_value : max_value ;
      . . . . .
      Var_type   var_name = min_value : typ_value : max_value ;
}
```

Parameter comment:

usage: var_name.min (= min_value)
"var_name" (equal "var_name.typ") (= typ_value)
var_name.max (= max_value)

spec_table_name: The name of load board to be used by the test program.
Max. 32 chars.
Type: string
No entry – illegal

Var_type: TIME, CURRENT, VOLTAGE, FREQUENCY or double.
No entry: illegal

var_name: Variable name with the value (min_value/typ_value/max_value)
No entry: illegal

min_value: Only constant is allowed.
No entry: default 0

typ_value: Only constant is allowed.
No entry: default 0

max_value: Only constant is allowed.
No entry: default 0

5.1.8.2 Explanation

Pre-define spec variable for using in test program.

5.1.8.3 Example

```
SPEC_DEF (FT_spec)   {
            VOLTAGE   vil = 0V : 0V : 0.4V ;
            VOLTAGE   vih = 2.4V : 5V : 5V ;
            CURRENT   iol = 0mA : 0mA : 1mA ;
            CURRENT   ioh = 0mA : 0mA : -1mA ;
            VOLTAGE    vref = 2.5V : 2.5V : 2.5V ;
            VOLTAGE    vdd = 4.5V : 5.0V : 5.5V ;
      }

      SET_LEVELN (func_lvl, Allpins, vil.min, vih.max, 2.5V, 2.5V, iol.typ, ioh.typ,
vref.typ, , );
      SET_LEVELN (func_lvl, in_pins, vdd.typ*0.1, vdd.typ*0.9, vdd.typ*0.3,
vdd.typ*0.7,
                        Iol.typ, ioh.typ, vref.typ, , );
      SET_LEVELN (func_lvl, in_pins, vdd *0.1, vdd *0.9, vdd *0.3, vdd *0.7,
                  Iol, ioh, vref, , );
```

5.1.9 DEBUG_BREAK

The statement is designed specially for setting a break point in Debug-Utility.

5.1.9.1 Format

Parameter list:

```
DEBUG_BREAK ();
```

Parameter comment:
None.

5.1.9.2 Explanation

The statement is designed specially for setting a break point in Debug-Utility when we want to debug a C/C++ program, and it doesn't have any effect in running test program.

5.1.9.3 Example

```
TIMER_ON ( ) ;
…     //statement
time_value = TIMER_OFF( );
DEBUG_BREAK ();
printf ("time_value = %f\n", time_value);
```

5.1.10 DO_WHEN_EOW

This function is executed only when End_Of_Wafer signal detected.

5.1.10.1 Format

Parameter list:

```
DO_WHEN_EOW()
    {
      /* user function */
    }
```

Parameter comment:
None.

5.1.10.2 Explanation

This function is executed only when End_Of_Wafer signal detected.

5.1.10.3 Example

```
DO_WHEN_EOW()
{
    LEVELS(reset_level, 3mS);
    FORCE_V_DPS(dpsAllPower, 0V , @8V , @100mA , 100mA , NORM ,
OFF,2mS);
    FORCE_V_PMU(PMU, 0V ,@6V , @100mA ,100mA ,OFF ,1mS);
    RELAY_OFF(FUNCTIONS, ALL , 1mS);
    SET_CBIT(urAllRelay, OFF ,1mS);
}
```

5.1.11 DATALOG_MSG

A datalog message to be printed on next judge statement.

5.1.11.1 Explanation

The statement can help user to print message to datalog .

```
DATALOG_MSG = "XXX";
```

5.1.11.2 Example

```
FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
RELAY_ON( IO_ALLPINS, PDCL, 3mS);
LEVELS(func_lvl, 3mS);
PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
PIN_MODE(          CLK,   RZ, EDGE,     MASK, IO_NRZ, OFF);
BACKUP_PLAN_RESULT();
DATALOG_MSG = "func_test";
JUDGE_PAT(TEST_1, __TEST_1);

FAIL_REJUDGE( 3 )
{
      JUDGE_PAT(retest_1, __ retest_1);
}
```

5.1.12 PRE-DEFINED PIN_GROUP

IO_ALLPINS – A pin_group name, which include all pins of "IO", "IN", "OUT" pin-type .

DPS_ALLPINS –A pin_group name, which include all pins of "DPS" pin-type.

PREF_ALLPINS –A pin_group name, which include all pins of "PREF" pin-type.

UR_ALLPINS –A pin_group name, which include all pins of "UR" pin-type.

NOTE : these pin_group name can be used in anywhere exclude PIN_GROUP statement .

5.1.13 PRE-DEFINED TYPES

CURRENT	:	variables with A/a (current) unit declaration
FREQUENCY	:	variables with HZ/hz (hertz) unit declaration
TIME	:	variables with S/s (time) unit declaration
VOLTAGE	:	variables with V/v (voltage) unit declaration
RESISTANCE	:	variables with OHM/ohm (ohm) unit declaration

5.1.14 READ_WAFER_ID

The statement read back the wafer ID .

5.1.14.1 Format

Parameter list:

```
READ_WAFER_ID (wafer_id, lot_id) ;
```

Parameter comment:

wafer_id: The returned wafer ID.
Type: STRING (Maximum 32 Characters);

lot_id: The returned lot ID.
Type: STRING (Maximum 32 Characters);

5.1.14.2 Explanation

The statement read back the wafer ID.

5.1.14.3 Example

```
char   wafer_id[32], lot_id[32] ;

READ_WAFER_ID (wafer_id, lot_id);
```

5.1.15 READ_PROBER_XY

The statement read back the xy coordinate of the testing die .

5.1.15.1 Format

Parameter list:

```
READ_PROBER_XY (x_coor, y_coor) ;
```

Parameter comment:

x_coor : The returned x-coordinate of the testing die.
Type: int array, array size 4,
S0 coordinate : x_coor[0]
S1 coordinate : x_coor[1]
S2 coordinate : x_coor[2]
S3 coordinate : x_coor[3]

If not connect GPIB, then default value -1024

y_coor : The returned y-coordinate of the testing die.
Type: int array, array size 16,

S0 coordinate : y_coor[0]
S1 coordinate : y_coor[1]
S2 coordinate : y_coor[2]
S3 coordinate : y_coor[3]

If not connect GPIB, then default value -1024

5.1.15.2 Explanation

The statement read back the xy coordinate of the testing die.

5.1.15.3 Example

```
int   x_coor[4], y_coor[4] ;

READ_PROBER_XY (x_coor, y_coor);
```

5.1.16 SET_DATALOG_MODE

Set datalog memory log mode

5.1.16.1 Format

Parameter list:

SET_DATALOG_MODE(log_mode [, datalog_display_count] [, turn_over_mode]);

Parameter comment:

Log_mode : LOG_CONT : (also turn on FEOP mode)

When fail happened, provide log print on Datalog tool continuous. User can decide log start address by using debug tool "PLAN_DEBUG"

And use datalog_display_count to set display count.

```
                        CSSOOCSSQQQQQQQQOO
                        LEEEELLRABCDEFGHAH
                        RLL12K
                         10
                            m  eeeeeeeeee
SPM          585(73)  1100010XHHHHHHLLHL
SPM          586(74)  1100010XHHHHHLLLHL
SPM          587(75)  1100010XHHHHLLLLHL
SPM          588(76)  1100010XHHH/////HL
SPM          589(77)  1100010XHH//////HL
SPM          590(78)  1100010XH///////HL
SPM          591(79)  1100010X////////LL
SPM          592(80)  1100011X///////HLH
SPM          593(81)  1100010X//////H/LL
SPM          594(82)  1100010X/////H//LL
```

Log data address is continuous

LOG_FAIL_ONLY : (also turn on FEOP mode)Only print fail log on DATALOG TOOL and log fail data

Use datalog_display_count to set display count.

```
                        CSSOOCSSQQQQQQQQOO
                        LEEEELLRABCDEFGHAH
                        RLL12K
                         10
                            m  eeeeeeeeee
SPM          590(78)  1100010XHL//////HL
SPM          591(79)  1100010XLL//////LL
SPM          592(80)  1100011XL//////HLH
SPM          594(82)  1100010XLLL//H/LLL
SPM          599(87)  1100010XHLL/////HL
SPM          604(92)  101001X0LHL/////LL
SPM          605(93)  101001X0L/H///LLLL
SPM          609(97)  101001X0LL////HLLL
SPM         614(102)  1100011XLHL_L_HHLH
SPM         616(104)  1100011XLHL___HHLH
```

Log data address is non-continuous

LOG_SITE_FAIL_ONLY : (also turn off FEOP mode)

Only log first fail content for each socket, if the socket fail(datalog_display_count must equal to 1)

**no_entry : default LOG_SITE_FAIL_ONLY ;

```
                    CSSOOCSSQQQQQQQQOO
                    LEEEELLRABCDEFGHAH
                    RLL12K
                     10
                         m  eeeeeeeeee
SPM          590(78) 1100010XHL//////HL
```

datalog_display_count : Datalog Utility display count value, 1 to **1024**
**no_entry: default = 1 ;

turn_over_mode : ON : if log memory pointer already move to last address, then it will turn over and move to address 0 in next log.
OFF : if log memory pointer already move to last address, then it will keep the same address (last address) in next log.

5.1.16.2 Example

```
//int pel[8],peh[8],pfail[8];
//int vec=0;
//int bb=0;

  SET_DATALOG_MODE(LOG_FAIL_ONLY,1024,ON);
  SET_JUDGE_MODE (NORM );
  JUDGE_PAT(search_turn_on,__search_turn_on);

  vec=0;
  for(bb=0;bb<1024;bb++)
  {
  READ_LOG_MEM(S0, VREG, bb, pel, peh, pfail);
  if(peh[0]>0)vec++;
  if(peh[0]<1) break;
  }

  printf("Vector=%d,TIME=%d nS\n",vec,(vec-1)*100);
```

5.1.16.3 Relative

READ_LOG_MEM

5.1.17 ALARM_ON

If a "Clamp" is detected, it is treated as "alarm fail". For channel which "ALARM_OFF" has been set, alarm detection will not be performed.

Default: ALARM_ON all power which defined in .dec file, excluding PPMU.
(Only show the clamp status in datalog).

After version: **CRAFT_3380_2.20_SP1**, ALARM_ON don't support MLDPS that means MLDPS value will not be alarm in a clamp value.

5.1.17.1 Format

Parameter list:

```
ALARM_ON(pin_name, mode);
```

Parameter comment:

pin_name: pin_def, pin_group
(Follow pin_type of dec file)

Mode:
IGNORE: if clamp happened, then ignore the alarm-fail, but show the clamp status in datalog.

Default: IGNORE

When user program"ALARM_ON", then must program a Alarm-Bin"
SYS_ALARM_HW_BIN"& **"SYS_ALARM_SW_BIN"**
Hardware_bin_name Keyword: **SYS_ALARM_HW_BIN**
Software_bin_name Keyword: **SYS_ALARM_SW_BIN**

5.1.17.2 Example

```
HW_BIN_DEF{
SYS_ALARM_HW_BIN     = !255;
}
SW_BIN_DEF {
SYS_ALARM_SW_BIN = !1023;
}
//============== ======== = DPS ============================//
FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
ALARM_ON(Vddd, FORCE_FAIL);//Set ALARM Fail
JUDGE_PAT(func_pat, __func_pat);//if clamp happen,JUDGE_RESULT will be FAIL.

//============== ======== = PMU ============================//
FORCE_V_DPS ( Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
ALARM_ON((Vcc, FORCE_FAIL);//Set ALARM Fail

PIN_INC(INPUTS)
 {
      FORCE_V_PMU (PMU, 0.4V, @6V, @1mA, 1mA, ON, 1mS);
      RELAY_ON( PIN_DEF, PMU, 1mS);
      JUDGE_I_PMU (PMU, -300uA, 10uA, 2mS, 16, AVE, 50uS, , );
     //if clamp happen,JUDGE_RESULT will be FAIL.

      RELAY_OFF( PIN_DEF, PMU, 3mS);
      FORCE_V_PMU (PMU, 0.4V, @6V, @100uA, 100uA, OFF, 1mS);
 }
```

5.1.18 ALARM_OFF

If a Alarm is detected, it is treated as "alarm fail". For channel which "ALARM_OFF". Has been set, alarm detection will not be performed.

5.1.18.1 Format

Parameter list:

```
ALARM_OFF(pin_name);
```

Parameter comment:

pin_name: pin_def, pin_group

5.1.18.2 Example

```
//============== ======== = DPS ==========================//
FORCE_V_DPS (Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
ALARM_ON(Vddd, FORCE_FAIL );//Set ALARM Fail
JUDGE_PAT (func_pat, __func_pat);//if clamp happen , JUDGE_RESULT will be FAIL.
ALARM_OFF(Vddd);

//============== ======== = PMU ==========================//
FORCE_V_DPS (Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
ALARM_ON(Vee, FORCE_FAIL );//Set ALARM Fail
PIN_INC (INPUTS)
 {
      FORCE_V_PMU (PMU, 0.4V, @6V, @1mA, 1mA, ON, 1mS);
      RELAY_ON( PIN_DEF, PMU, 1mS);
      JUDGE_I_PMU (PMU, -300uA, 10uA, 2mS, 16, AVE, 50uS, , );
     //if clamp happen, JUDGE_RESULT will be FAIL.
      RELAY_OFF (PIN_DEF, PMU, 3mS);
      FORCE_V_PMU (PMU, 0.4V, @6V, @100uA, 100uA, OFF, 1mS);
 }
ALARM_OFF(Vee);
```

5.1.19 GET_ALARM_STATUS

Return the present alarm status.

5.1.19.1 Format

Parameter list:

```
alarm_status = GET_ALARM_STATUS(Socket_no, Type, pin_name );
```

Parameter comment:

Socket_no: S0: socket No. 0
S1: socket No. 1
…
S1023: socket No. 1023
Type: integer
no entry: illegal

Type: DPS (DPS Alarm status)
PREF (PREF Alarm status)
PMU (PMU Alarm status)
UVI (UVI Alarm status)
:

pin_name : pin_def
No entry: illegal (except type = PMU)

alarm _status: return variable name;
Type: integer;
Status value:
"1": clamp happened
"0": clamp not happened

5.1.19.2 Example

```
//============== ======== = DPS ==========================//
FORCE_V_DPS (Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
ALARM_ON (Vddd, FORCE_FAIL);//Set ALARM Fail
JUDGE_PAT (func_pat, __func_pat);//if clamp happen , JUDGE_RESULT will be FAIL.

alarm_status=GET_ALARM_STATUS (S0, DPS, Vddd); //int alarm_status;
if (alarm_status==1)                printf ("clamp happened\n");
else if (alarm_status==0)           printf ("clamp not happened\n");
ALARM_OFF (Vddd);

//============== ======== = PMU ==========================//
FORCE_V_DPS (Vddd, 5V, @8V, @100mA, 100mA, NORM, ON, 1mS) ;
ALARM_ON (INPUTS , FORCE_FAIL);//Set ALARM Fail
PIN_INC (INPUTS)
 {
      FORCE_V_PMU (PMU, 0.4V, @6V, @1mA, 1mA, ON, 1mS);
      RELAY_ON( PIN_DEF, PMU, 1mS);
      JUDGE_I_PMU (PMU, -300uA, 10uA, 2mS, 16, AVE, 50uS, , );
     //if clamp happen, JUDGE_RESULT will be FAIL.
      RELAY_OFF (PIN_DEF, PMU, 3mS);
      FORCE_V_PMU (PMU, 0.4V, @6V, @100uA, 100uA, OFF, 1mS);
 }
alarm_status=GET_ALARM_STATUS (S0, PMU); //int alarm_status;
if (alarm_status==1)                printf ("clamp happened\n");
else if (alarm_status==0)           printf ("clamp not happened\n");

ALARM_OFF (INPUTS);
```

5.1.20 WRITE_PAT_MEM

WRITE_PAT_MEM – CRAFT statement
Write IO pattern vector to specified address, of course replace the original pattern.

5.1.20.1 Format

Parameter list:

```
WRITE_PAT_MEM(socket _no, pin_name, addr, pat_mode, pat_vector) ;
```

Parameter comment:

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S1023 : socket No. 1023
Type : integer
no_entry: illegal

addr: SPM addr is 16M to 32M-1
a SPM address from the pattern file.
Must be represented by a global label [+/- variable or constant]
or module name:local_label [+/- variable or constant] .
type: integer
**no_entry: illegal

pin_name; pin_def, pin_group
**no_entry: illegal

pat_mode: NORM_PAT, DBL_2X_PAT
(the pat_mode will not change the original pattern mode setting,
it just specify the input pin pattern symbol table (1X/2X/..),
**no_entry: NORM_PAT (1X)

pat_vector: type: string

5.1.20.2 Explanation

[Notice]: If device keeps CONT-mode (continue mode), that user can't change IO status while in CONT-mode. It means that high-speed running, user can't change the timing/level/format/pattern.

5.1.20.3 Example

```
WRITE_PAT_MEM(S0, data_bus, sum_addr, NORM_PAT ,"LHLLHLHL");
```

5.1.21 READ_PAT_MEM

READ_PAT_MEM – CRAFT statement
Read IO pattern vector from specified address.

5.1.21.1 Format

```
READ_PAT_MEM(socket _no, pin_name, addr, pat_mode, pat_vector [,
pat_string ]) ;
```

Parameter comment:

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S1023 : socket No. 1023
Type : integer
no_entry: illegal

addr: SPM addr is 16M to 32M-1
a SPM address from the pattern file.
Must be represented by a global label [+/- variable or constant]
or module name:local_label [+/- variable or constant] .
type: integer
**no_entry: illegal

pin_name; pin_def, pin_group
**no_entry: illegal

pat_mode: NORM_PAT, DBL_2X_PAT

(The pat_mode will not change the original pattern mode setting,
It just specifies the input pin pattern symbol table (1X/2X/3X/..),
**no_entry: NORM_PAT (1X)

pat_vector: return data array
type: integer
//== NORM_PAT ==//
// pat_vector[] → pattern symbol
0x0 → X
0x2 → 0
0x3 → 1
0x4 → Z
0x5 → V
0x6 → L
0x7 → H

pat_string: return data string (string length= pin count)
type: char

// pat_ string [] == pattern symbol : X / 0 / 1 / Z / V / L /H

5.1.21.2 Example

```
// data_bus = q0+q1+q2+q3+q4+q5+q6+q7;
        int pat_data[8];
        char pat_str[8];
        READ_PAT_MEM(S0, data_bus, sum_addr, ,pat_data, pat_str);
        //read "q0,q1,q2,q3,q4,q5,q6,q7" to pat_data[] ;
        //q0 pattern →pat_data[0], q1→ pat_data[1], …
```

5.1.22 READ_LOG_MEM

READ_LOG_MEM – CRAFT statement
Read IO or LCD datalog memory; include PEL, PEH and PinFail log memory data.

5.1.22.1 Format

Parameter list:

```
READ_LOG_MEM(socket _no, pin_name, addr, PEL_data, PEH_data, PinFail_data) ;
```

Parameter comment:

```
socket_no :     S0 : socket No. 0
                S1 : socket No. 1
                …
                S31 : socket No. 31
                Type : integer
                **no_entry: illegal

  addr:     datalog log memory address is 0 to 1023
            type: integer
            **no_entry: illegal

  pin_name;     pin_def, pin_group
                **no_entry: illegal

  PEL_data :
  PEH_data :
  PinFail_data :   return Pin Fail log memory,
                     0/1: pass/fail
                     type: int array
```

5.1.22.2 Example

```
// data_bus = q0+q1+q2+q3+q4+q5+q6+q7 ;
        int pel[8],peh[8],pfail[8];
       READ_LOG_MEM(S0, data_bus, 0, pel, peh, pfail);
```

5.1.23 READ_LOG_MEM_BLOCK

READ_LOG_MEM_BLOCK – CRAFT statement
Read IO datalog memory block; include PEL, PEH and PinFail log memory data

5.1.23.1 Format

Parameter list:

```
READ_LOG_MEM_BLOCK(socket _no, pin_name, st_addr, sp_addr, PEL_data,
                   PEH_data, PinFail_data) ;
```

Parameter comment:

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S1023 : socket No. 1023
Type : integer
**no_entry: illegal

st_addr/sp_addr: datalog log memory address is 0 to 1023
type: integer
**no_entry: illegal

pin_name; pin_def, pin_group
**no_entry: illegal

PEL_data : return PEL data
type: int array

PEL	
	FAIL:1(HIGH)
	PASS:0(LOW)

PEH_data : return PEH data
type: int array

PEH	
	PASS:0(HIGH)
	FAIL1:(LOW)

PinFail_data : return Pin Fail log memory,
0/1: pass/fail
type: int array

5.1.23.2 Example

return pel data :

pel[0]	pel [1]	…	pel [3]	pel[4]	pel [5]	…	pel [7]	…	pel[4095]
Addr0: P0	Addr0: P1	…	Addr0: P3	Addr1: P0	Addr1: P1	…	Addr1: P3	…	Addr1023: P3

```
//int pel[1024*4],peh[1024*4],pfail[1024*4];
//int vec=0;
//int bb=0;
//int data_f[1024];
//VREG=P0(MSB)+P1+P2+P3(LSB)

//==================================================================
======//
   SET_DATALOG_MODE(LOG_CONT,1024,ON);
   SET_JUDGE_MODE (NORM );
   JUDGE_PAT(search_turn_on,__search_turn_on);

   READ_LOG_MEM_BLOCK(S0, VREG, 0, 1023, pel, peh, pfail) ;

   j=0;
   for(int bb=0;bb<1024;bb++)
   {
                                        data_f[bb]=0;
                                    for (w=3;w--;w>=0;j++)
                                            {
                                                    data_f[bb]=data_f[bb]   |   (pel[j]<<w);
                                            }
                            Printf("number %d code:=%d", bb ,data_f[bb]);
   }
```

5.1.23.3 Relative

SET_DATALOG_MODE, SET_JUDGE_MODE

5.1.24 SET_UPAT_REG

SET_UPAT_REG – CRAFT statement
Set micro-instruction register

5.1.24.1 Format

Parameter list:

```
SET_UPAT_REG( reg_name , reg_value );
```

Parameter comment:

reg_name : X0 , X1 , X2, … , X7
**no_entry : illegal ;
reg_value : 1 <= reg_value <= 16777215 (24bit register)
**no_entry: illegal

5.1.24.2 Example

```
sw_test()
{
      RATEN(TS1, 1000nS);
      MARKERN(TS1, IO_ALLPINS,0nS,10nS, ,0nS,10nS,900nS,1000nS);
      PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, NORM);
      PIN_MODE(CLK, RZ, EDGE, MASK, IO_NRZ, OFF);
      SET_LEVELN(func_lvl, IO_ALLPINS, 0V, 5V, 0.5V, 2.4V, 150uA, -150uA,
2.5V, , );
      LEVELS(func_lvl, 2mS);
      RELAY_ON(IO_ALLPINS, PDCL, 1mS);
       SET_UPAT_REG( X0, 2 );
       SET_UPAT_REG( X1, 3 );
       SET_UPAT_REG( X2, 4 );
       SET_UPAT_REG( X3, 5 );
       SET_UPAT_REG( X4, 6 );
       SET_UPAT_REG( X5, 7 );
       SET_UPAT_REG( X6, 8 );
       SET_UPAT_REG( X7, 9 );
       JUDGE_PAT(SET_UPAT,__ SET_UPAT)
}
```

```
SPM_PATTERN (SET_UPAT)
{
Start :     * 001000*TS1;
            * 001000*;
            * 001000*;
            * 001000*;
            * 001000*LDXN    X0;
            * 001000*;
```

```
            * 001000*;
            * 001000*;
            * 001000*;
            * 000000*RPTN    X0;
            * 001000*;
            * 001000*;
            * 001000*;
            * 001000*LDXN    X1;
            * 001000*;
            * 001000*;
            * 000000*RPTN    X1;
            * 001000*;
}
```

5.1.24.3 Relative

Micro-Instruction "LDXN", "RPTN"

5.1.25 READ_DMM_VOLT

READ_DMM_VOLT – CRAFT statement
The statement read back the DMM voltage value.

5.1.25.1 Format

Parameter list:

```
status = READ_DMM_VOLT( DMM_type, ave_value [, meas_values, min_value, max_value,
ave_cnt, sampling_interval]) ;
```

Parameter comment:

status :
- 0 : the measurement is ok
- 1 : GPIB error

DMM_type : HP34401
HP34410
HP3458
**no_entry: illegal

ave_value : return average measured values.
Type : VOLTAGE or double
**no_entry: illegal

meas_values : return the all measured values,
array size= ave_cnt
min_value : return the minimum value among "ave_cnt" measurement
max_value : return the maximum value among "ave_cnt" measurement
Type : VOLTAGE or double

ave_cnt: the average count
**no_entry: Default = 1

sampling_interval: the sampling interval between each read
**no_entry: Default = 5mS

5.1.25.2 Example

```
VOLTAGE   dmm_value,min_value,max_value,all_values[128] ;

int status;

status = READ_DMM_VOLT( HP34401, dmm_value );

        status = READ_DMM_VOLT( HP34410, dmm_value, all_values, min_value,
    max_value, 10, 10mS);
```

5.1.26 SET_STEP_LEVEL_MODE

SET_STEP_LEVEL_MODE – CRAFT statement
Set IO pin to drive step-level waveform by high speed trigger

5.1.26.1 Format

Parameter list:

```
SET_STEP_LEVEL_MODE( pin_name , trigger_source , step_level_src,
increment_delta_volt, mode );
```

Parameter comment:

pin_name: pin_def, pin_group (only suit for IO-Pins)
**no_entry; illegal

trigger_source: TS1 ~ TS15
**no_entry; illegal

step_level_src : VIL , VIH , VOL , or VOH
increment_delta_volt : each increase delta-volt

mode: ON/OFF (turn on/off step-level function)`
**no_entry; illegal

[Notice]: It will cost about **23uS** from Trigger signal to step-level output ready .

5.1.26.2 Example

Set start level

```
SET_LEVELN(func_lvl, CTL_PINS+DB_PINS, 0V, 5V, 2.5V, 2.5V, 0uA, 0uA,
0V, , );
SET_LEVELN(func_lvl, WG0, 0V, 5V, 2.5V, 2.5V, 0uA, 0uA, 0V, , );
LEVELS(func_lvl, 8mS);
RELAY_ON(CTL_PINS+DB_PINS+WG0, PDCL, 1mS);

SET_CAPTURE_MEM_MODE(DB_PINS,PEL,ON,64K,1024,LOG_FAIL_ONLY,O
N);

SET_STEP_LEVEL_MODE(WG0,TS2,VIL,0.6103515625mV,ON);
WAIT(5mS);

  CRAFT_STATEMENT
  {
    JUDGE_PAT(squar_st,squar_sp);
    PLAN_RESULT_MS[0]=PASS;
    bit_mask = bit_mask&(0Xfffffffff<<4);
    for(i=0; i<WG_DC_log_cnt; i++)
    {
        READ_CAPTURE_MEM(S0,DB_PINS,i,pel);
        printf("%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d\n", pel[0],
```

Set trigger source ,trigger level and step voltage .
TS2=30uS>23uS for step-level output ready

```
pel[1], pel[2], pel[3], pel[4], pel[5],
            pel[6], pel[7], pel[8], pel[9], pel[10], pel[11], pel[12], pel[13], pel[14],
pel[15]);
            data_f[i]=0;
            for(j=0,w=15; w>=0; w--,j++)
            {
                  data_f[i] = data_f[i] | (pel[j]<<w);
            }
            data_mask[i] = (data_f[i]& bit_mask);
            data_val[i] = ((double)(data_f[i] ^ 0x8000) - 32768.0) / 32768.0 * 10.0;
      }
```

```
PIN_LIST( AD976ar )
            {
            /*name = ATE channel = dut# = type    */
              WG0    = 448        = 1    = IO     ;   //TIP
              DB15 = 160          = 6    = IO     ;   //IO
              DB14 = 128          = 7    = IO     ;   //IO
              DB13 = 152          = 8    = IO     ;   //IO
              DB12 = 184          = 9    = IO     ;   //IO
              DB11 = 168          = 10 = IO     ;   //IO
              DB10 = 208          = 11 = IO     ;   //IO
              DB9    = 224         = 12 = IO     ;   //IO
              DB8    = 240         = 13 = IO     ;   //IO
              DB7    = 248         = 15 = IO     ;   //IO
              DB6    = 232         = 16 = IO     ;   //IO
              DB5    = 216         = 17 = IO     ;   //IO
              DB4    = 192         = 18 = IO     ;   //IO
              DB3    = 200         = 19 = IO     ;   //IO
              DB2    = 176         = 20 = IO     ;   //IO
              DB1    = 144         = 21 = IO     ;   //IO
              DB0    = 136         = 22 = IO     ;   //IO
              R_C    = 134         = 24 = IO     ;   //IO
              Vdig = 576            = 28 = DPS    ;
              Vana = 577            = 27 = DPS    ;
              AGnd =               = 2    = GND    ;   /*GND pin didn't any ate channel */
              DGnd =               = 14 = GND    ;   /*GND pin didn't any ate channel */
            }
```

Assign IO pin for step level used

```
squar_st ::    *XXXX XXXX XXXX XXXX 1          * TS1 ;
                 *XXXX XXXX XXXX XXXX 1          * ;
                 *XXXX XXXX XXXX XXXX 1    * ;
                 *XXXX XXXX XXXX XXXX 1          * RPT 35000;
                 *XXXX XXXX XXXX XXXX 1          * ;
                 *XXXX XXXX XXXX XXXX 1          * ;
                 *XXXX XXXX XXXX XXXX 1          * ;
                 *XXXX XXXX XXXX XXXX 1          * ;
                 *XXXX XXXX XXXX XXXX 1          * ;
                 *ZZZZ ZZZZ ZZZZ ZZZZ 0    * UISYNC,TS2;
                 *ZZZZ ZZZZ ZZZZ ZZZZ 0          * RPT 32766,TS2;
                 *ZZZZ ZZZZ ZZZZ ZZZZ 0          *TS2;
squar_sp ::    *XXXX XXXX XXXX XXXX 1          * TS1
```

Level Trigger by TS2

5.1.26.3 Relative

SET_CAPTURE_MEM_MODE , READ_CAPTURE_MEM

5.1.27 SET_FAIL_BIN

SET_FAIL_BIN -- CRAFT statement

The statement is designed to directly assign the soft-bin and hard-bin in test item and must be declared before the JUDGE statement. the statement will override the bin setting that set up in TEST_PRO function. When CRAFT start in next test item, the "SET_FAIL_BIN" setting will be clear and back to "TEST_PRO" setting.

5.1.27.1 Format

Parameter list:

```
SET_FAIL_BIN ( hw_bin, sw_bin);
```

Parameter comment:

hw_bin : Type : integer
variable allowed, or constant defined in HW_BIN_DEF
**no_entry: illegal

sw_bin : Type : integer
variable allowed, or constant defined in SW_BIN_DEF
**no_entry: illegal

5.1.27.2 Example

1.

```
SOCKET_INC( ) {
    if ( PLAN_RESULT_MS[SOCKET] == PASS)
        SET_FAIL_BIN ( a , a );
    JUDGE_VARIABLE( );
    if ( PLAN_RESULT_MS[SOCKET] == PASS)
        SET_FAIL_BIN ( b , b );
    JUDGE_VARIABLE( );
}
```

2.

```
SET_FAIL_BIN( a , a );
JUDGE_PAT(TEST1,__TEST);
DISABLE_FAIL_SOCKET();
...
…
SET_FAIL_BIN( b , b );
JUDGE_PAT();
DISABLE_FAIL_SOCKET();
```

5.1.27.3 Relative

SOCKET_INC , PLAN_RESULT_MS , JUDGE_VARIABLE , JUDGE_PAT , DISABLE_FAIL_SOCKET

5.1.28 TDO_PRINTF

TDO_PRINTF – CRAFT statement

5.1.28.1 Format

Step1

```
TDO_PRINTF("CSV_FIELD SITE=%d TITLE=%s CONTENT=%s 【TEST_NO=%d】 \n",
SOCKET, "TDO_PRINTF", freq[SOCKET],(int)15);
```

“CSV_FIELD SITE=”, “TITLE=, “CONTENT=”, “TEST_NO=” are key words.

PS. If use “CONTENT=”，the type will be set to “string”.

Step2

```
TDO_PRINTF("CSV_FIELD SITE=%d TITLE=%s VALUE=%f 【TEST_NO=%d】\n", SOCKET,
"TDO_PRINTF", freq[SOCKET],(int)15);
```

“CSV_FIELD SITE=”, “TITLE=, “VALUE=”, “TEST_NO=” are key words.

PS. If use “VALUE=”, the type will be set to “float”.

5.1.28.2 Explanation

The statement can help user to replace C_function "printf" and print message to datalog TDO file (CSV file) and Datalog tool. Normal version is 512 characters.
4096 characters of release version: **CRAFT_3380_2.40**

5.1.28.3 Example

```
SOCKET_INC

{

TDO_PRINTF("CSV_FIELD SITE=%d TITLE=%s VALUE=%f TEST_NO=%d\n",

SOCKET, "PLL", freq[SOCKET],(int)15);

TDO_PRINTF("CSV_FIELD SITE=%d TITLE=%s CONTENT=%s TEST_NO=%d\n",

SOCKET, "TRY", try[SOCKET],(int)15);
}
```

5.1.29 CRAFT_STATEMENT

The statement is designed specially for setting a compound statement in test plan. when we want to debug a compound statement in Debug-Utility , and it will be regarded as a single CRAFT statement.

5.1.29.1 Format

```
CRAFT_STATEMENT
    {
            Statement 1
            Statement 2
            ...................
            ...................
            ...................
            ...................
    }
```

5.1.29.2 Example

```
CRAFT_STATEMENT
{
    JUDGE_PAT(squar_st,squar_sp);
    PLAN_RESULT_MS[0]=PASS;
    READ_CAPTURE_MEM_BLOCK(S0,DB_PINS,0,1023,temp);//temp[1024*16]
    for(i=0; i< 1024 ; i++)
    {
        data_f[i]=0;
        for(j=0,w=15; w>=0; w--,j++)
        {
            data_f[i] = data_f[i] | (pel[j]<<w);
        }
        data_val[i] = ((double)(data_f[i] ^ 0x8000) - 32768.0) / 32768.0 * 10.0;
    }
}
```

All statement can be a single CRAFT_STATEMENT

5.1.30 WRITE_FILE

Write data of the number of elements specified by "data_cnt" to a file from the "Data" array.

5.1.30.1 Format

Parameter list:

```
Status = WRITE_FILE ("file_name", Data, data_cnt );
```

Parameter comment:

Status: 0 (write file valid)
1 (open file error)
2 (other fail)

5.1.30.2 Example

```
 wg_wd_self_test()
{
   int i,j,k , aa,status;
   double data_tmp[1024]={0.0};
   double wd_spec[1024]={0.0};
   double lower[1024]={0.0};
   double upper[1024]={0.0};
   double thd,snr,snd,sfdr,thd_n;
   double dc, signal,harmonic,noise;

   LOAD_AWG_PAT(wg1,"./sin_15v.awi", 16, RPT, 1);
   SET_WG_WORK_MODE(wg1, SINGLE_END, @RNG3, , R50, OFF, ON,
MODE_A);
   SET_WD_WORK_MODE(wd1, @RNG1, 0V, OFF, OFF, OFF, ON);
   SET_WD_WAVEFORM_SET(wd1, 16, 10.0e-6, 10.0e-6, 0x0, 1024);
   JUDGE_PAT(clr_st, clr_sp);
   READ_WD_DATA (S0, wd1, 16, 1024, data_tmp);

//=====================store WD data to
file============================//
   status = WRITE_FILE ("D:\\AWI_TEST.dat", data_tmp, 1024 );
   if(status==0)          TDO_PRINTF( " >>write file valid<< \n") ;
   else if (status==1)   TDO_PRINTF( " >> open file error << \n") ;
   else                   TDO_PRINTF( " >> other fail << \n") ;


   //==================read SPEC from file=========================//
   status = READ_FILE
   ("D:\\AWI_SELF_TEST\\LXAWI_WG_WD_Self_Test_1ch\\spec.dat", wd_spec,
   1024 );

   if(status==0)          TDO_PRINTF (" READ file valid\n");
```

```
  else if (status==1)   TDO_PRINTF (" open file error\n");
  else                  TDO_PRINTF (" other fail\n");

  for(aa=0;aa<1024;aa++)
  {
    lower[aa]= 0.9* wd_spec[aa];
    upper[aa]= 1.1* wd_spec[aa];
  }
  DEBUG_BREAK();
  JUDGE_ARRAY (S0, data_tmp, lower, upper , "WD_DATA_CHECK" );
}
```

5.1.31 READ_FILE

Read data of the number of elements specified by "data_cnt" from a file to the "Data" array.

5.1.31.1 Format

Parameter list:

```
Status = READ_FILE ("file_name", Data, data_cnt );
```

Parameter comment:

Status: 0 (read file valid)
1 (open file error)
2 (other fail)

5.1.31.2 Example:

```
wg_wd_self_test()
{
   int i,j,k , aa,status;
   double data_tmp[1024]={0.0};
   double wd_spec[1024]={0.0};
   double lower[1024]={0.0};
   double upper[1024]={0.0};
   double thd,snr,snd,sfdr,thd_n;
   double dc, signal,harmonic,noise;

   LOAD_AWG_PAT(wg1,"./sin_15v.awi", 16, RPT, 1);
   SET_WG_WORK_MODE(wg1, SINGLE_END, @RNG3, , R50, OFF, ON,
MODE_A);
   SET_WD_WORK_MODE(wd1, @RNG1, 0V, OFF, OFF, OFF, ON);
   SET_WD_WAVEFORM_SET(wd1, 16, 10.0e-6, 10.0e-6, 0x0, 1024);
   JUDGE_PAT(clr_st, clr_sp);
   READ_WD_DATA (S0, wd1, 16, 1024, data_tmp);


//=====================store WD data to
file============================//
   status = WRITE_FILE ("D:\\AWI_TEST.dat", data_tmp, 1024 );
   if(status==0)          TDO_PRINTF( " >>write file valid<< \n") ;
   else if (status==1)   TDO_PRINTF( " >> open file error << \n") ;
   else                   TDO_PRINTF( " >> other fail << \n") ;


   //==================read SPEC from file========================//
   status = READ_FILE
   ("D:\\AWI_SELF_TEST\\LXAWI_WG_WD_Self_Test_1ch\\spec.dat", wd_spec,
   1024 );
```

```
  if(status==0)          TDO_PRINTF (" READ file valid\n");
  else if (status==1)    TDO_PRINTF (" open file error\n");
  else                   TDO_PRINTF (" other fail\n");

  for(aa=0;aa<1024;aa++)
  {
     lower[aa]= 0.9* wd_spec[aa];
     upper[aa]= 1.1* wd_spec[aa];
  }
  DEBUG_BREAK();
  JUDGE_ARRAY (S0, data_tmp, lower, upper , "WD_DATA_CHECK" );
}
```

5.1.32 JUDGE_ARRAY

Make a Pass/Fail decision by comparing the value of a program array against specified array limits.

5.1.32.1 Format

Parameter list:

```
JUDGE_ARRAY (socket_no, var_array, lower_limit_array, upper_limit_array [,
"string" ]);
```

Parameter comment:

socket_no : S0 : socket No. 0
S1 : socket No. 1
…
S1023 : socket No. 1023
Type : integer
**no_entry: illegal

var_array: array variable with the value to be tested
type: TIME, CURRENT, VOLTAGE, FREQUENCY or double
**no_entry: illegal

lower_limit_ array :array variable and array constant allowed
type: TIME, CURRENT, VOLTAGE, FREQUENCY or double
**no_entry: only high limit

upper_limit_ array :array variable and array constant allowed
type: TIME, CURRENT, VOLTAGE, FREQUENCY or double
**no_entry: only low limit

string: Only used to be printed by datalog.
**no_entry: do not print any string .

5.1.32.2 Example

```
wg_wd_self_test()
{
   int i,j,k , aa,status;
   double data_tmp[1024]={0.0};
   double wd_spec[1024]={0.0};
   double lower[1024]={0.0};
   double upper[1024]={0.0};
   double thd,snr,snd,sfdr,thd_n;
   double dc, signal,harmonic,noise;

   LOAD_AWG_PAT(wg1,"./sin_15v.awi", 16, RPT, 1);
   SET_WG_WORK_MODE(wg1, SINGLE_END, @RNG3, , R50, OFF, ON, MODE_A);
   SET_WD_WORK_MODE(wd1, @RNG1, 0V, OFF, OFF, OFF, ON);
   SET_WD_WAVEFORM_SET(wd1, 16, 10.0e-6, 10.0e-6, 0x0, 1024);
   JUDGE_PAT(clr_st, clr_sp);
   READ_WD_DATA (S0, wd1, 16, 1024, data_tmp);


//=====================store WD data to file==========================//
   status = WRITE_FILE ("D:\\AWI_TEST.dat", data_tmp, 1024 );
   if(status==0)          TDO_PRINTF( " >>write file valid<< \n") ;
   else if (status==1)    TDO_PRINTF( " >> open file error << \n") ;
   else                   TDO_PRINTF( " >> other fail << \n") ;


   //==================read SPEC from file=========================//
   status = READ_FILE ("D:\\AWI_SELF_TEST\\LXAWI_WG_WD_Self_Test_1ch\\spec.dat",
   wd_spec, 1024 );

   if(status==0)          TDO_PRINTF (" READ file valid\n");
   else if (status==1)    TDO_PRINTF (" open file error\n");
   else                   TDO_PRINTF (" other fail\n");

   for(aa=0;aa<1024;aa++)
   {
      lower[aa]= 0.9* wd_spec[aa];
      upper[aa]= 1.1* wd_spec[aa];
   }
   DEBUG_BREAK();
   JUDGE_ARRAY (S0, data_tmp, lower, upper , "WD_DATA_CHECK" );
}
```

5.1.33 INPUT_BOX

INPUT_BOX statement provides for receiving external parameter function during executing test program.

5.1.33.1 Format

Parameter list:

```
INPUT_BOX(msg ,scan_type,sacn_variable);
```

Parameter comment:

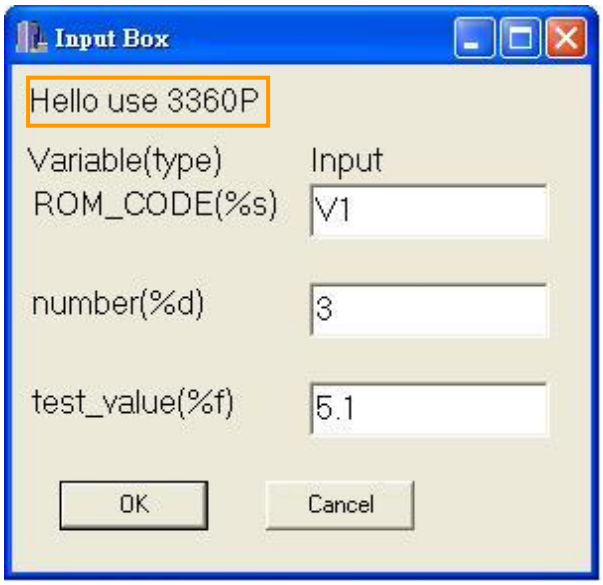
msg : string or variable

scan_type: "%d" , "%c" ,"%f"

scan_variable : variable for storing input value

5.1.33.2 Example

```
START_UP()
{
      SET_LEVELN( rst_lvl, IO_ALLPINS, 0.0V,0.5V,0.0V,0.0V,0.0mA,0.0mA,0.0V, , );
      LEVELS(rst_lvl, 3mS);

       //==============INPUT_BOX================================//
       INPUT_BOX("Hello use 3360P","%s %d %f ", ROM_CODE , number , test_value );
       printf("ROM_CODE:%s ,Number:%d ,test_value:%f\n",ROM_CODE,number,test_value)
;

}
```



5.1.34 ACCEPT_VER_EQ

It limit the test plan can only run in specified Craft Version and specified equipment model, if version is not correct ,system will pop up a warning message window and unload test program.

5.1.34.1 Format

Parameter list:

```
ACCEPT_VER_EQ { Craft Version }
```

Parameter comment:
Craft Version: Include Equipment model, software version and Service Pack information.

Check flow:

1. Check Equipment.
2. Check software version.
   Example: 2.00>1.02>1.01
3. Check Service Pack
   Example: SP3>SP2>SP1

5.1.34.2 Example

```
//========== Check software Version Craft1.01 SP3 in 3360 model ===============//
ACCEPT_VER_EQ { CRAFT_3360_1.01_SP3 } //use the statement in the first of test plan

//========== Check software Version Craft1.01 SP3 in 3360P model ===============//
ACCEPT_VER_EQ { CRAFT_3360P_1.01_SP3 } //use the statement in the first of test
plan
```

5.1.34.3 Relative

ACCEPT_VER_GE

5.1.35 ACCEPT_VER_GE

It limit the test plan can only run in above (include) Craft Version and specified equipment model, if version is not correct, system will pop up a warning message window and unload test program.

5.1.35.1 Format

Parameter list:

```
ACCEPT_VER_EQ { Craft Version }
```

Parameter comment:
Craft Version: Include Equipment model, software version and Service Pack information.

Check flow:

1. Check Equipment.
2. Check software version.
   Example: 2.00>1.02>1.01
3. Check Service Pack
   Example: SP3>SP2>SP1

5.1.35.2 Example

```
//======== Check software Version above(include) Craft1.01 SP3 in 3360 model
============//
ACCEPT_VER_GE { CRAFT_3360_1.01_SP3 } //use the statement in the first of test plan

//======== Check software Version above(include) Craft1.01_SP3 in 3360P model
============//
ACCEPT_VER_GE { CRAFT_3360P_1.01_SP3 } //use the statement in the first of test
plan
```

5.1.35.3 Relative

ACCEPT_VER_EQ

5.1.36 GET_LABEL_ADDR

Get the memory address from the pattern label name

5.1.36.1 Format

Parameter list:

```
GET_LABEL_ADDR( Label_name, Address, Status );
```

Parameter comment:

Label_name:

Pattern label name or string

s: variable

Type: string, variable allowed

**no_entry: illegal

Address: returned label memory address

Type: integer

Status: type : integer ;

0: valid

1: failed

5.1.36.2 Example

```
//int    st_addr,sp_addr,status;
//char label[32];
GET_LABEL_ADDR( rom_st, st_addr, status);

GET_LABEL_ADDR( rom_pat:st, st_addr, status);

strcpy( label, "rom_sp");
GET_LABEL_ADDR( s:label, sp_addr, status);//use variable
```

5.1.37 READ_TESTER_MODEL

The statement read back the tester model.

5.1.37.1 Format

Parameter list:

```
READ_TESTER_MODEL( model_name ) ;
```

Parameter comment:

model_name :
the retuerned system model name
3500
3502
3360
3362
3380
Type: STRING (Maximum 32 Characters);

5.1.37.2 Example

```
test_start()
{
        READ_TESTER_MODEL( model_name );    //char model_name[32];
        printf("model name = %s \n", model_name );
}
```

5.1.38 ABORT_TEST_PLN

The statement will abort test plan and exit.

5.1.38.1 Format

Parameter list:

```
ABORT_TEST_PLN();
```

Parameter comment:
Null

5.1.38.2 Example

```
test_start()
{
    SET_LEVELN(func_lvl, IO_ALLPINS, 0V, 3V, 1V, 2V, 0uA, 0uA, 1.5V, , );
    LEVELS(func_lvl, 3mS);
    JUDGE_PAT(st,sp)
      ....

    if ( PLN_RESULT_MS[0]==FAIL )
   {
            ABORT_TEST_PLN( );    // abort and exit program
   }
   ...

}
```

5.1.39 PAT_SECURITY

The statement with be used to set Patern protect mode .When function is turned on; pattern file can not be read from ATE system.

5.1.39.1 Format

Parameter list:

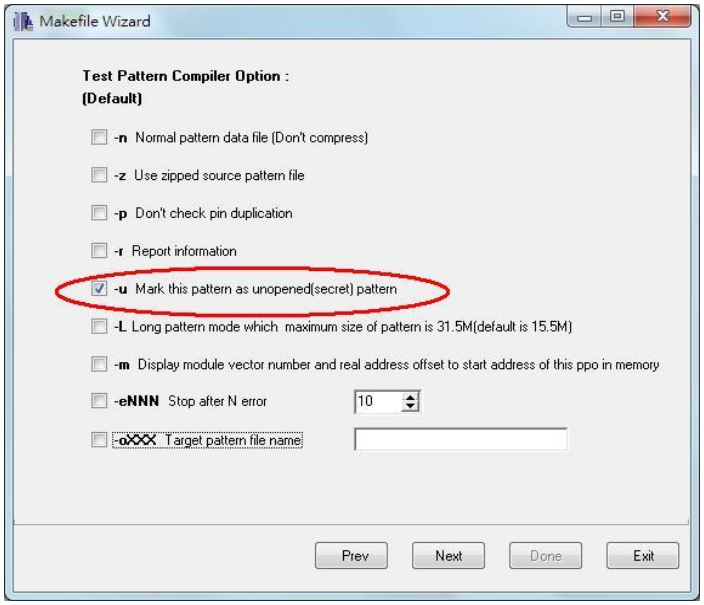
```
PAT_SECURITY ( status) ;
```

Parameter comment:
Status: _ON_ Protect turn on
OFF Protect turn off
NO Entry: _ON_ Default

[Notice:] when Protect mode turn on,READ_PAT_MEM and VECTOR_EXTRACT are invalid.
Program will be unloaded.

5.1.39.2 Example

When user makes program, one of steps is "Mark this pattern as unopened(secret) pattern" in Makefile Wizard.



```
test_example(){
      LOAD_PAT (". / PAT / ls299pat.ppo" ) // load secret pattern
      LEVELS( rst_lvl, 2mS) ;
      RELAY_ON( IO_ALLPINS, PDCL, 1mS);
      PAT_SECURITY(_OFF_);//turn off PAT_SECURITY
      READ_PAT_MEM(     S0, p0, idd_cont_st, ,pat_data, pat_str); // valid
      VECTOR_EXTRACT( S0, freq_cont_st, freq_cont_sp, 0, idd_cont_st,
idd_cont_sp, 5,   "vector_extract.pat",   ve_status);   //vaild
}
```

```
 test_example(){
      LOAD_PAT (". / PAT / ls299pat.ppo" ) // load secret pattern
      LEVELS( rst_lvl, 2mS) ;
      RELAY_ON( IO_ALLPINS, PDCL, 1mS);
      READ_PAT_MEM( S0, p0, idd_cont_st, ,pat_data, pat_str);
      // invalid and show error message.program unload

}
```

5.1.40 TRM_PRINTF

TRM_PRINTF – CRAFT statement

5.1.40.1 Explanation

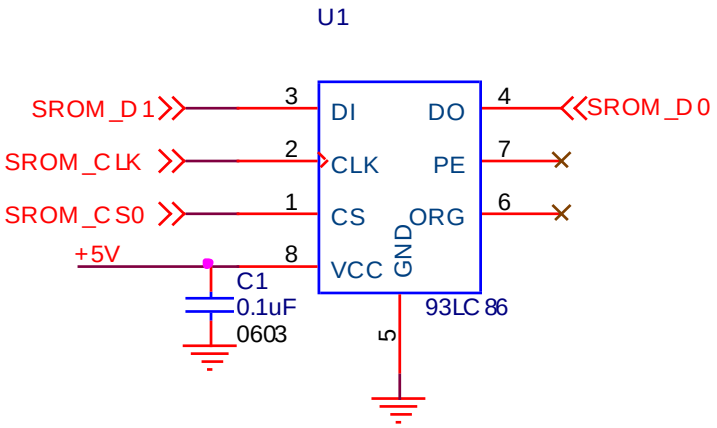
Print user message to TRM datalog file (.trm).
TRM file : for laser trim using; defined in Production-ID Tools.

5.1.40.2 Example

```
int aa=10;

char bb[10]= "ppp";

TRM_ PRINTF ("%s %d",bb,aa) ;
```

5.1.41 READ_SYSTEM_LOAD_BOARD_ID

The statement read back the load board ID. **The Load board means 3380 official version.**
Release version : **CRAFT_3380_1.30**



5.1.41.1 Format

Parameter list:

status = **READ_SYSTEM_LOAD_BOARD_ID**(board_id, serial_no, model, type, date, comment) ;

Parameter comment:

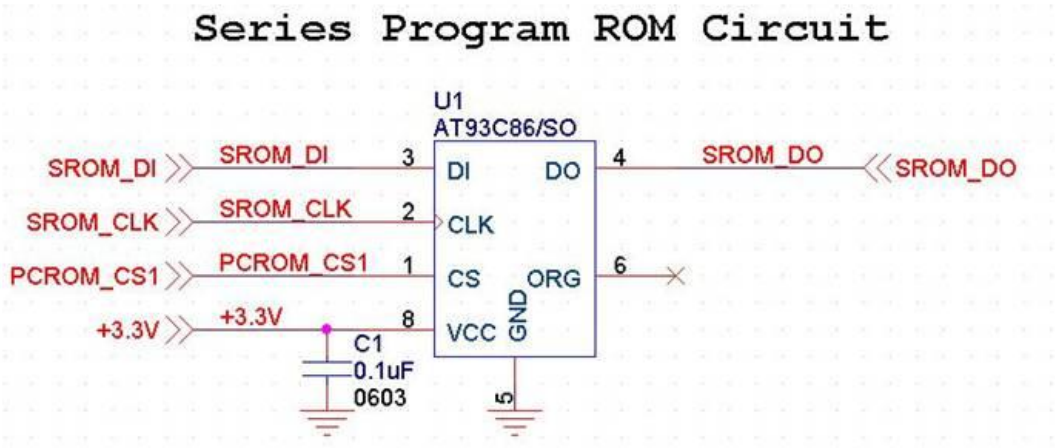
status : type integer
0 : EEPROM not exist
1 : EEPROM exist
board_id : the retuerned load board serial number
Type : STRING (Maximum 32 Charaters);
serial_no : the retuerned load board serial number
Type : STRING (Maximum 32 Charaters);
model : the retuerned load board model
Type : STRING (Maximum 32 Charaters);
type : the retuerned type
Type : STRING (Maximum 16 Charaters);
date : the retuerned manufacture date
Type : STRING (Maximum 16 Charaters);
comment : the retuerned comment
Type : STRING (Maximum 64 Charaters);

5.1.41.2 Example

```
char   board_id[32], serial_no[32], model[32], type[16], date[16], comment[64];
int status;
 status = READ_SYSTEM_LOAD_BOARD_ID (board_id, serial_no, model, type,
date, comment);
            if(status == 1)

            {

                if( strcmp("Chroma-DC-Diag", board_id) !=0 )     // ID is not matched.

                {

                printf("System Load Board ID : board_id: %s \n",board_id);

                printf("System Load Board ID : serial_no: %s \n",serial_no);

                printf("System Load Board ID : model: %s \n",model);

                printf("System Load Board ID : type: %s \n",type);

                printf("System Load Board ID : date: %s \n",date);

                printf("System Load Board ID : comment: %s \n\n",comment);

                ABORT_TEST_PLN();   //exit test program

                }

            }

            else   //EEPROM don't exist.

            {

}
```

5.1.42 READ_LOAD_BOARD_ID_2

The statement read back the remote load board ID. For probe card ID or DUT board. Please follow MXPG3 pogo pin definition.
Release version: **CRAFT_3380_2.20**



5.1.42.1 Format

Parameter list:

status = **READ_LOAD_BOARD_ID_2**(board_id, serial_no, model, type, date, comment) ;

Parameter comment:
status : type integer

0 : EEPROM not exist

1 : EEPROM exist

board_id : returned load board serial number

Type : STRING (Maximum 32 Characters);

serial_no : returned load board serial number

Type : STRING (Maximum 32 Characters);

model : returned load board model

Type : STRING (Maximum 32 Characters);

type : returned type

Type : STRING (Maximum 16 Characters);

date : returned manufacture date

Type : STRING (Maximum 16 Characters);

comment : the returned comment

Type : STRING (Maximum 64 Characters);

5.1.42.2 Example

```
char   board_id[32], serial_no[32], model[32], type[16], date[16], comment[64];

int status ;

status = READ_LOAD_BOARD_ID_2( board_id, serial_no, model, type, date,
comment);

if(status == 1)

{

        if( strcmp("Probe Card ID: 0001", board_id) !=0 )    // ID not match

            {

                printf("board_id = %s \n", board_id );

                printf("the Load Board used for : %s \n", comment);

                exit(0);   //exit test program

            }

}

else   //EEPROM not exist or board_id not "0001"

{

    printf("NO EEPROM !");

}
```

5.1.43 READ_SYSYEM_VENDOR_ID

The statement read back the system vendor ID. The system ID is unique and programmed by Chroma.

5.1.43.1 Format

READ_SYSTEM_VENDOR_ID(model, type, sub_type, ab_revision, serial_no) ;

model : the returned load board model

Type : STRING (Maximum 4 Characters);

type : the returned type

Type : STRING (Maximum 5 Characters);

Sub_type : the returned type

Type : STRING (Maximum 2 Characters);

Ab_revision : the returned load board serial number

Type : STRING (Maximum 4 Characters);

serial_no : the returned load board serial number

Type : STRING (Maximum 6 Characters);

5.1.43.2 Example

```
char model[8] , type[8], sub_type[8], ab_revision[8], serial_no[8];

READ_SYSTEM_VENDOR_ID(model, type, sub_type, ab_revision, serial_no);
printf("model-type-sub_type-ab_revision-serial_no : %s-%s-%s-%s-%s \n",
        model, type, sub_type, ab_revision, serial_no);

// output may : model-type-sub_type-ab_revision-serial_no : 3380-XXX-XX-0101-000001
```

5.1.44 JUDGE_SPI_WRITE

The statement can support user to enter device register quickly by HEX or binary data format. Release version : **CRAFT_3380_2.40-SP5.**

5.1.44.1 Format

```
JUDGE_SPI_WRITE(DataPinName,
                _PAT_ST, St_Addr,
                _PAT_SP, Sp_Addr
                [ ,_DATAINFO, Start_Address, DataString, DataMask, SignBitFirst ] +
                );
```

DataPinName: SPI Data Pin, support pin group.
St_Addr : Burst Pattern Start address
Sp_Addr : Burst Pattern Stop address

- **Data information(_DATAINFO)**

Start_Address: Would like to modify Pattern start address
DataString : Would like to modify data, ex. ”0xA5” or ”10100101”
DataMask : Mask Data
: ”0xA5”, DataMask ”0xC0” Means Data would be “10100101”
SignBitFirst : **MSB_FIRST** or **LSB_FIRST**
DataMask :Mask Data
:”0xA5”, DataMask ”0xC0” Means Data is “10100101” ➔”10100101”➔ “**100101**”
:”0xA5”, DataMask “0x03” Means Data is ”10100101” ➔”**101001**01”➔ “**101001**”
SignBitFirst :statement KeyWord: **MSB_FIRST** or **LSB_FIRST**

single Pin:
: **MSB_FIRST**, Data”0x5”
: P0
: 0 (**Vector 0**)
: 1 (Vector 1)
: 0 (Vector 2)
: 1 (Vector 3)

: **LSB_FIRST**, Data”0x5”
: P0
: 1 (**Vector 0**)
: 0 (Vector 1)
: 1 (Vector 2)
: 0 (Vector 3)

multi Pin: PG=P0+P1+P2+P3;
: **MSB_FIRST**, Data”0xA5”
: P0 P1 P2 P3
: 1 0 1 0 (A) **Vector 0**
: 0 1 0 1 (5) Vector 1

: **LSB_FIRST**, Data”0xA5”
: P0 P1 P2 P3
: 0 1 0 1 (5) Vector 0
: 1 0 1 0 (A) Vector 1

5.1.44.2 Example

```
PIN_MODE(SPI_CK, RZ, EDGE, ENABLE,IO_NRZ, OFF);
PIN_MODE(SPI_D0+SPI_D1+SPI_D2+SPI_EN, NRZ, EDGE, ENABLE,IO_NRZ, OFF);
JUDGE_SPI_WRITE(SPI_D0+SPI_D1+SPI_D2,
        _PAT_ST, B1_CW111_pattern,
        _PAT_SP, __B1_CW111_pattern,
        _DATAINFO, B1_CW111_pattern:
A_72_D_CB882,“0x072CB882”,“0xC0000000”,MSB_FIRST
      );
CUSTOM_3DIM_BOOL ReturnData;
JUDGE_SPI_READ(SPI_D0+SPI_D1+SPI_D2,
      SPI_D0+SPI_D1+SPI_D2,
      X1,
      7,
      ReturnData,
        _PAT_ST, A_6F_D_B7_CW111,
        _PAT_SP, __A_6F_D_B7_CW111,
        _DATAINFO, A_6F_D_B7,“0x5BC”,“0x000”,MSB_FIRST
      );

1)  Pattern Format:
      甲、 User create by test case.
      乙、 SET_DEC_FILE “AAAA.dec”
      丙、 HEADER
      丁、 SPI_CK, SPI_D0, SPI_D1, SPI_D2, SPI_EN, CLK_SEL, DCXO_32KEN, EN_BB,
           TMEAS, TST1,
      戊、 TST2,TST3,TST4,XMODE;
      己、 SPM_PATTERN (B1_CW111_pattern) {
      庚、                    *000001010XXXX0*TS2;
      辛、                    *000001010XXXX0*;
      壬、                    *000001010XXXX0*RPT 32;
      癸、                    *000011010XXXX0*;
      11、 A_72_D_CB882:        *100011010XXXX0*;      //   A7/-/-
      12、                    *111111010XXXX0*;      //   A4/A5/A6
      13、                    *110011010XXXX0*;      //   A1/A2/A3
      14、                    *111011010XXXX0*;      //   D18/D19/A0
      15、                    *110011010XXXX0*;      //   D15/D16/D17
      16、                    *111011010XXXX0*;      //   D12/D13/D14
      17、                    *100111010XXXX0*;      //   D9/D10/D11
      18、                    *101011010XXXX0*;      //   D6/D7/D8
      19、                    *100011010XXXX0*;      //   D3/D4/D5
      20、                    *101011010XXXX0*;      //   D0/D1/D2
      21、                    *000011010XXXX0*;
      22、                    *000001010XXXX0*;
      23、                    }


A.  SET_DEC_FILE “AAAA.dec"
B.
C.  HEADER
D.  SPI_CK,SPI_D0,SPI_D1,SPI_D2,SPI_EN,CLK_SEL,DCXO_32KEN,EN_BB,TMEAS,TS
    T1,
E.  TST2,TST3,TST4,XMODE;
F.  SPM_PATTERN (A_6F_D_B7_CW111) {
G.              *000001010XXXX0*TS2;
H.              *000001010XXXX0*;
I.              *000001010XXXX0*;
J.              *000001010XXXX0*RPT 32;
K.              *000011010XXXX0*; //T = eq_2_ts_1_1 ; EN_High
```

```
L.   A_6F_D_B7:      *101011010XXXX0*; //T = eq_2_ts_1_1 SPI_WR = send     ; A7/-/-
M.                   *101111010XXXX0*; //T = eq_2_ts_1_1 SPI_WR = send     ; A4/A5/A6
N.                   *111111010XXXX0*; //T = eq_2_ts_1_1 SPI_WR = send     ; A1/A2/A3
O.                   *100111010XXXX0*; //T = eq_2_ts_1_1 SPI_WR = send     ; -/-/A0
P.              *100001010XXXX0*; //T = eq_2_ts_1_1 ; TAR
Q.              *000001010XXXX0*; //T = eq_2_ts_1_1 ;
R.              *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store; D17/D18/D19
S.              *000001010XXXX0*; //T = eq_2_ts_1_1 ;
T.              *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store; D14/D15/D16
U.              *000001010XXXX0*; //T = eq_2_ts_1_1 ;
V.              *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store; D11/D13/D13
W.              *000001010XXXX0*; //T = eq_2_ts_1_1 ;
X.              *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store; D8/D9/D10
Y.              *000001010XXXX0*; //T = eq_2_ts_1_1 ;
Z.              *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store; D5/D6/D7
AA.             *000001010XXXX0*; //T = eq_2_ts_1_1 ;
BB.             *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store; D2/D3/D4
CC.             *000001010XXXX0*; //T = eq_2_ts_1_1 ;
DD.             *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store; X/D0/D1
EE.             *000001010XXXX0*; //T = eq_2_ts_1_1 ;
FF.             *000001010XXXX0*; //T = eq_2_ts_1_1 ;
GG.             *000001010XXXX0*; //T = eq_2_ts_1_1 ;
HH.             *000001010XXXX0*; //T = eq_2_ts_1_1 ;
II.             *000001010XXXX0*; //T = eq_2_ts_1_1 ;
JJ.             }
```

5.1.45 JUDGE_SPI_READ

The statement can support user to read device register quickly by HEX or binary data format. Release version : **CRAFT_3380_2.40-SP5.**

5.1.45.1 Format

```
JUDGE_SPI_READ(DataPinName,
               CapturePin,
               CondenseMode,
               CaptureCounts,
               ReturnData,
               _PAT_ST, St_Addr,
               _PAT_SP, Sp_Addr
               [ ,_DATAINFO, Start_Address, DataString, DataMask, SignBitFirst ] +
               );
```

DataPinName : SDI Data Pin, support pin group.
CapturePin : Data Read Back; **Log mode system default set: PEL " K " symbol;**
CondenseMode :
X1 (each 32pin block maximum only 32 capture pin)→ log size maximum 16M-1K
X4 (each 32pin block maximum only 8 capture pin)→ log size maximum 16M*4-1K
X8 (each 32pin block maximum only 4 capture pin)→ log size maximum 16M*8-1K
X16 (each 32pin block maximum only 2 capture pin)→ log size maximum 16M*16-1K
X32 (each 32pin block maximum only 1 capture pin)→ log size maximum 16M*32-1K
CaptureCounts : Capture Data counts
ReturnData : Type **CUSTOM_3DIM_BOOL**; ReturnData[Site][Pin Index][Count]
Pattern_St_Addr : Burst Pattern start address
Pattern_Sp_Addr : Burst Pattern Stop address

- **Data information(_DATAINFO)**

Start_Address: Would like to modify Pattern start address
DataString : Would like to modify data, ex. "0xA5" or "10100101"
DataMask : Mask Data
: "0xA5", DataMask "0xC0" Means Data would be "10100101"
SignBitFirst : **MSB_FIRST** or **LSB_FIRST**

DataMask :Mask Data
:"0xA5", DataMask "0xC0" Means Data is "10100101" ➔"10100101"➔ "**100101**"
:"0xA5", DataMask "0x03" Means Data is "10100101" ➔"**101001**01"➔ "**101001**"

SignBitFirst :statement KeyWord: **MSB_FIRST** or **LSB_FIRST**

single Pin:
: **MSB_FIRST**, Data"0x5"
: P0
: 0 (**Vector 0**)
: 1 (Vector 1)
: 0 (Vector 2)
: 1 (Vector 3)

: **LSB_FIRST**, Data"0x5"
: P0
: 1 (**Vector 0**)
: 0 (Vector 1)
: 1 (Vector 2)
: 0 (Vector 3)

multi Pin: PG=P0+P1+P2+P3;
: **MSB_FIRST**, Data"0xA5"

```
: P0 P1 P2 P3
: 1   0   1   0 (A) Vector 0
: 0   1   0   1 (5) Vector 1

: LSB_FIRST, Data"0xA5"
: P0 P1 P2 P3
: 0   1   0   1 (5) Vector 0
: 1   0   1   0 (A) Vector 1
```

5.1.45.2 Example

```
PIN_MODE(SPI_CK, RZ, EDGE, ENABLE,IO_NRZ, OFF);
PIN_MODE(SPI_D0+SPI_D1+SPI_D2+SPI_EN, NRZ, EDGE, ENABLE,IO_NRZ, OFF);
JUDGE_SPI_WRITE(SPI_D0+SPI_D1+SPI_D2,
        _PAT_ST, B1_CW111_pattern,
        _PAT_SP, __B1_CW111_pattern,
        _DATAINFO, B1_CW111_pattern:
A_72_D_CB882,"0x072CB882","0xC0000000",MSB_FIRST
      );
CUSTOM_3DIM_BOOL ReturnData;
JUDGE_SPI_READ(SPI_D0+SPI_D1+SPI_D2,
      SPI_D0+SPI_D1+SPI_D2,
      X1,
      7,
      ReturnData,
        _PAT_ST, A_6F_D_B7_CW111,
        _PAT_SP, __A_6F_D_B7_CW111,
        _DATAINFO, A_6F_D_B7,"0x5BC","0x000",MSB_FIRST
      );

2)  Pattern Format:
      甲、 User create by test case.
      乙、 SET_DEC_FILE "AAAA.dec"
      丙、 HEADER
      丁、 SPI_CK, SPI_D0, SPI_D1, SPI_D2, SPI_EN, CLK_SEL, DCXO_32KEN, EN_BB,
           TMEAS, TST1,
      戊、 TST2,TST3,TST4,XMODE;
      己、 SPM_PATTERN (B1_CW111_pattern) {
      庚、                    *000001010XXXX0*TS2;
      辛、                    *000001010XXXX0*;
      壬、                    *000001010XXXX0*RPT 32;
      癸、                    *000011010XXXX0*;
      11、 A_72_D_CB882:        *100011010XXXX0*;      //   A7/-/-
      12、                    *111111010XXXX0*;      //   A4/A5/A6
      13、                    *110011010XXXX0*;      //   A1/A2/A3
      14、                    *111011010XXXX0*;      //   D18/D19/A0
      15、                    *110011010XXXX0*;      //   D15/D16/D17
      16、                    *111011010XXXX0*;      //   D12/D13/D14
      17、                    *100111010XXXX0*;      //   D9/D10/D11
      18、                    *101011010XXXX0*;      //   D6/D7/D8
      19、                    *100011010XXXX0*;      //   D3/D4/D5
      20、                    *101011010XXXX0*;      //   D0/D1/D2
      21、                    *000011010XXXX0*;
      22、                    *000001010XXXX0*;
      23、                    }


KK. SET_DEC_FILE "AAAA.dec"
```

```
LL.
MM.  HEADER
NN. SPI_CK,SPI_D0,SPI_D1,SPI_D2,SPI_EN,CLK_SEL,DCXO_32KEN,EN_BB,TMEAS,TS
    T1,
OO.  TST2,TST3,TST4,XMODE;
PP. SPM_PATTERN (A_6F_D_B7_CW111) {
QQ.             *000001010XXXX0*TS2;
RR.             *000001010XXXX0*;
SS.             *000001010XXXX0*;
TT.             *000001010XXXX0*RPT 32;
UU.             *000011010XXXX0*; //T = eq_2_ts_1_1 ; EN_High
VV. A_6F_D_B7:      *101011010XXXX0*; //T = eq_2_ts_1_1 SPI_WR = send     ; A7/-/-
WW.                 *101111010XXXX0*; //T = eq_2_ts_1_1 SPI_WR = send     ; A4/A5/A6
XX.                 *111111010XXXX0*; //T = eq_2_ts_1_1 SPI_WR = send     ; A1/A2/A3
YY.                 *100111010XXXX0*; //T = eq_2_ts_1_1 SPI_WR = send     ; -/-/A0
ZZ.             *100001010XXXX0*; //T = eq_2_ts_1_1 ; TAR
AAA.            *000001010XXXX0*; //T = eq_2_ts_1_1 ;
BBB.            *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store; D17/D18/D19
CCC.                *000001010XXXX0*; //T = eq_2_ts_1_1 ;
DDD.                *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store;
    D14/D15/D16
EEE.            *000001010XXXX0*; //T = eq_2_ts_1_1 ;
FFF.            *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store; D11/D13/D13
GGG.                *000001010XXXX0*; //T = eq_2_ts_1_1 ;
HHH.                *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store; D8/D9/D10
III.            *000001010XXXX0*; //T = eq_2_ts_1_1 ;
JJJ.            *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store; D5/D6/D7
KKK.            *000001010XXXX0*; //T = eq_2_ts_1_1 ;
LLL.            *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store; D2/D3/D4
MMM.                *000001010XXXX0*; //T = eq_2_ts_1_1 ;
NNN.                *1KKK01010XXXX0*; //T = eq_2_ts_1_1 SPI_RD = store; X/D0/D1
OOO.                *000001010XXXX0*; //T = eq_2_ts_1_1 ;
PPP.            *000001010XXXX0*; //T = eq_2_ts_1_1 ;
QQQ.                *000001010XXXX0*; //T = eq_2_ts_1_1 ;
RRR.                *000001010XXXX0*; //T = eq_2_ts_1_1 ;
SSS.            *000001010XXXX0*; //T = eq_2_ts_1_1 ;
TTT.            }
```

5.2 CRAFT Macro Define

5.2.1 SELECTED_PIN_LIST

5.2.1.1 Explanation

CRAFT macro-define, user can read the selected PIN_LIST name by this way

5.2.1.2 Example

```
// char sel_pin_list[64];
Strcpy(sel_pin_list,SELECTED_PIN_LIST);
```

5.2.2 SELECTED_CONDITION

5.2.2.1 Explanation

CRAFT macro-define, user can read the selected CONDITION_SET name by this way

5.2.2.2 Example

```
// char sel_cond[64];
Strcpy(sel_cond,SELECTED_CONDITION);
```

5.2.3 SELECTED_SPEC_DEF

5.2.3.1 Explanation

CRAFT macro-define, user can read the selected SPEC_DEF name by this way.

5.2.3.2 Example

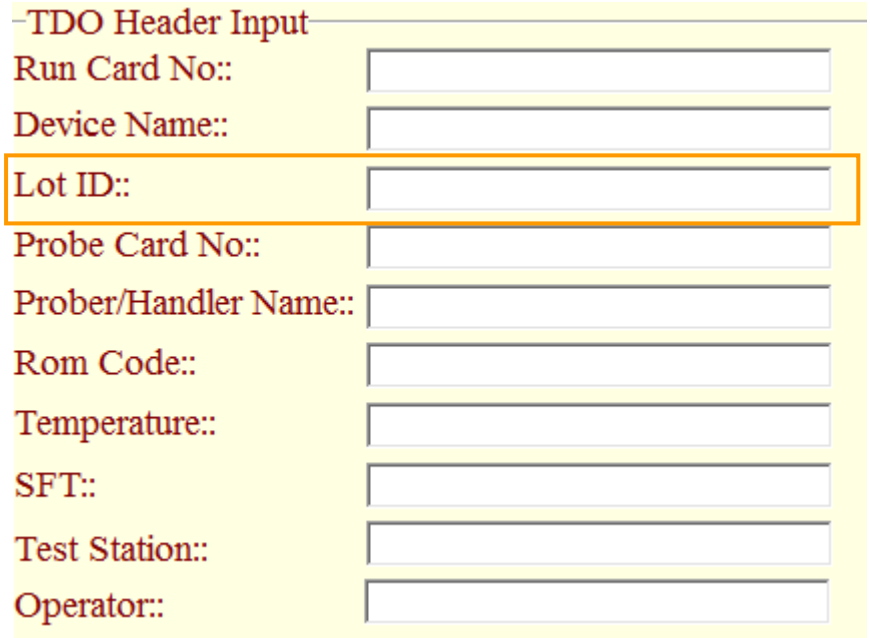
```
// char sel_spec_def [64];
Strcpy(sel_spec_def, SELECTED_SPEC_DEF);
```

5.2.4 TEST_LOT_ID

5.2.4.1 Explanation

Provide user to get LOD_ID from operation mode input information
In CP case, the name will get from Prober, not from operation key in.

5.2.4.2 Example



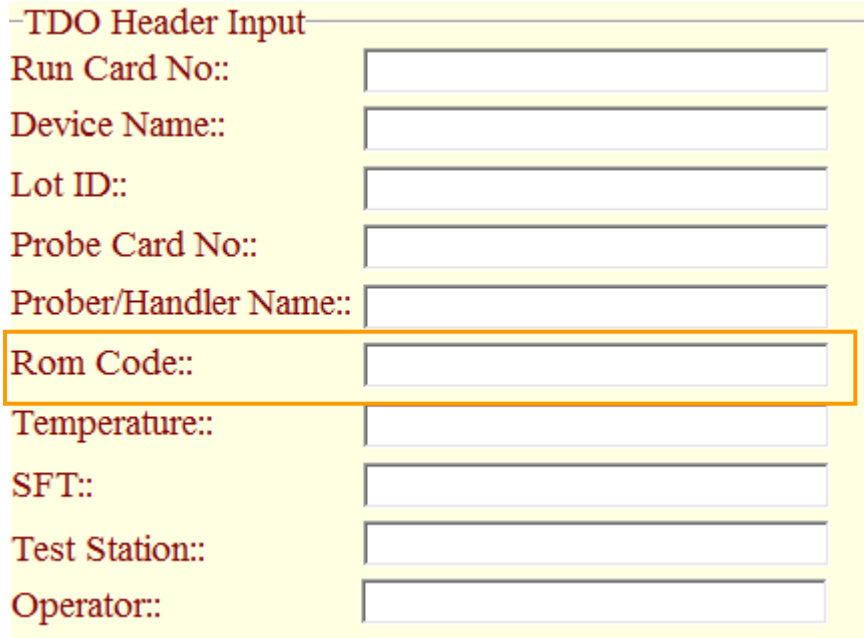
printf("Lot_ID:%s\n", TEST_LOT_ID);

5.2.5 TEST_ROM_CODE

5.2.5.1 Explanation

Provide user to get Rom Code from operation mode input information

5.2.5.2 Example



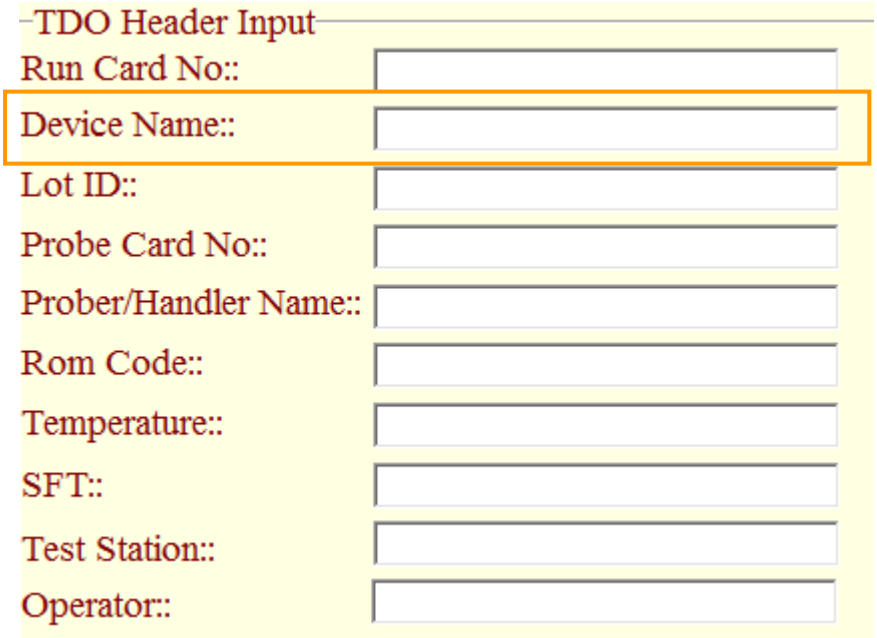

```
printf("Rom code :%s\n", TEST_ROM_CODE);
```

5.2.6 TEST_DEVICE_NAME

5.2.6.1 Explanation

Provide user to get Device name from operation mode input information

5.2.6.2 Example



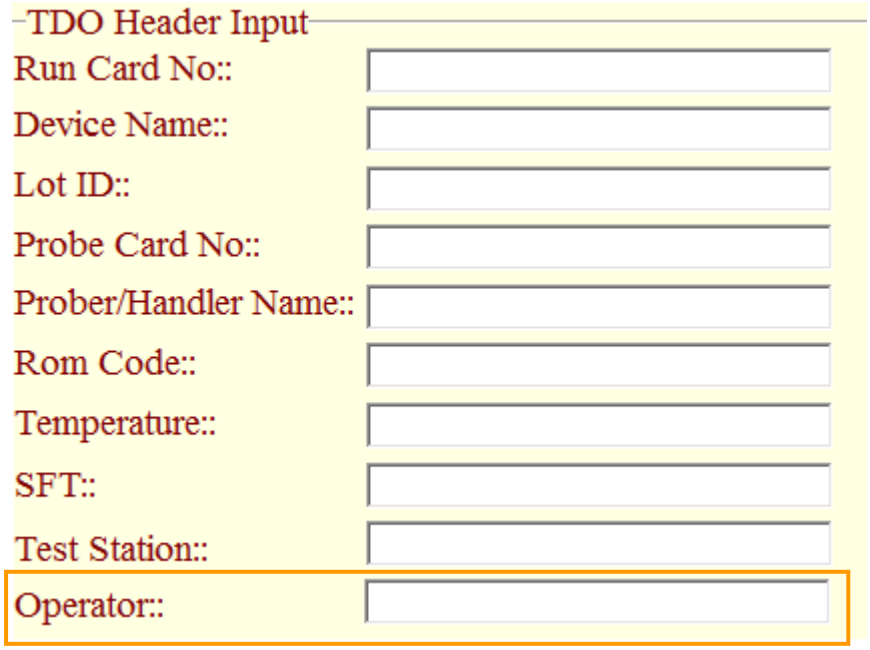
printf("Device name :%s\n", TEST_DEVICE_NAME);

5.2.7 TEST_OPERATOR

5.2.7.1 Explanation

Provide user to get OPERATOR from operation mode input information

5.2.7.2 Example



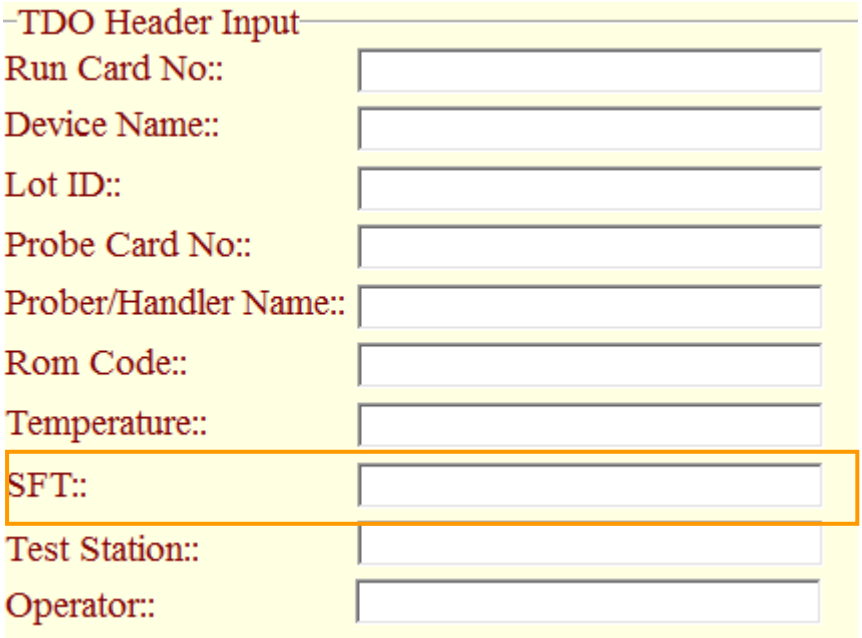

```
printf("OPERATOR :%s\n", TEST_OPERATOR);
```

5.2.8 TEST_SFT

5.2.8.1 Explanation

Provide user to get SFT from operation mode input information

5.2.8.2 Example



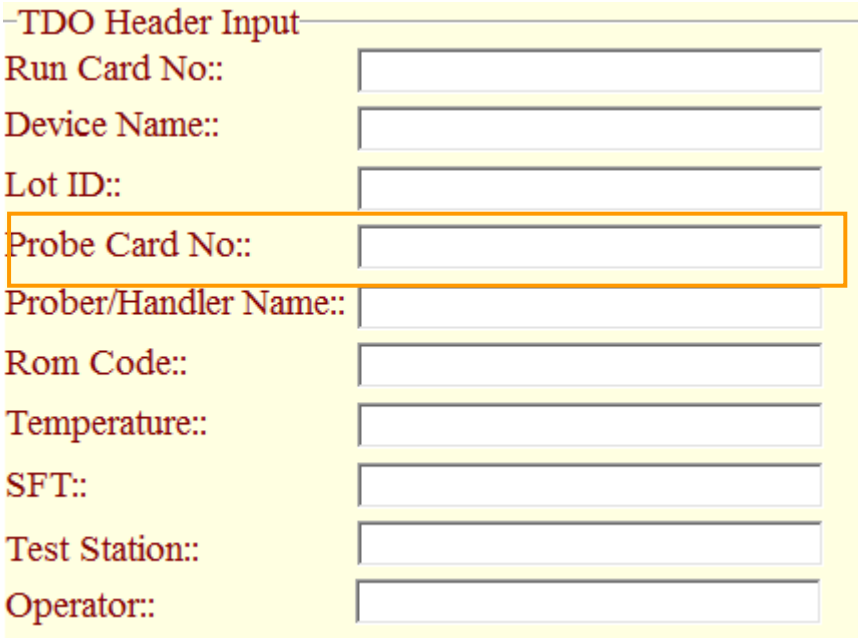

```
printf("SFT :%s\n", TEST_SFT);
```

5.2.9 TEST_PROBE_CARD_NO

5.2.9.1 Explanation

Provide user to get Probe Crad No from operation mode input information

5.2.9.2 Example



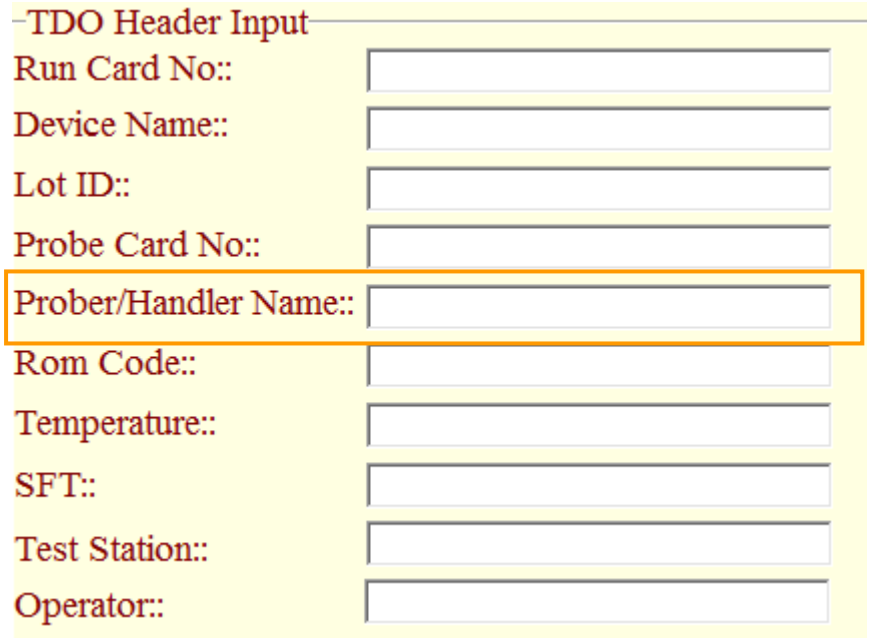

```
printf("Probe Crad No :%s\n", TEST_PROBE_CARD_NO);
```

5.2.10 TEST_PROBER_NAME

5.2.10.1 Explanation

Provide user to get Prober/Handler Name from operation mode input information
In CP case, the name will get from Prober, not from operation key in.

5.2.10.2 Example



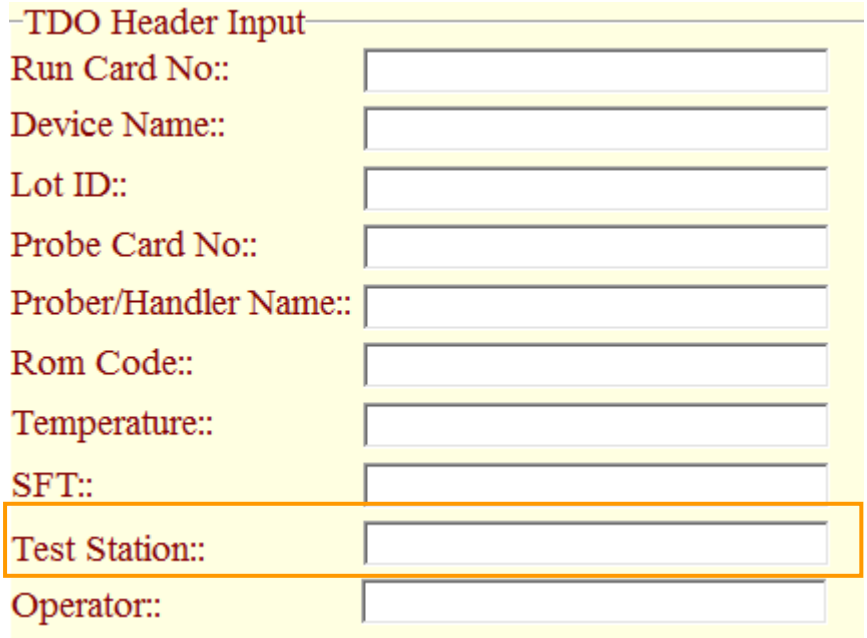
printf("Prober_Handler Name :%s\n", TEST_PROBER_NAME);

5.2.11 TEST_STATION

5.2.11.1 Explanation

Provide user to get Prober/Handler Name from operation mode input information

5.2.11.2 Example



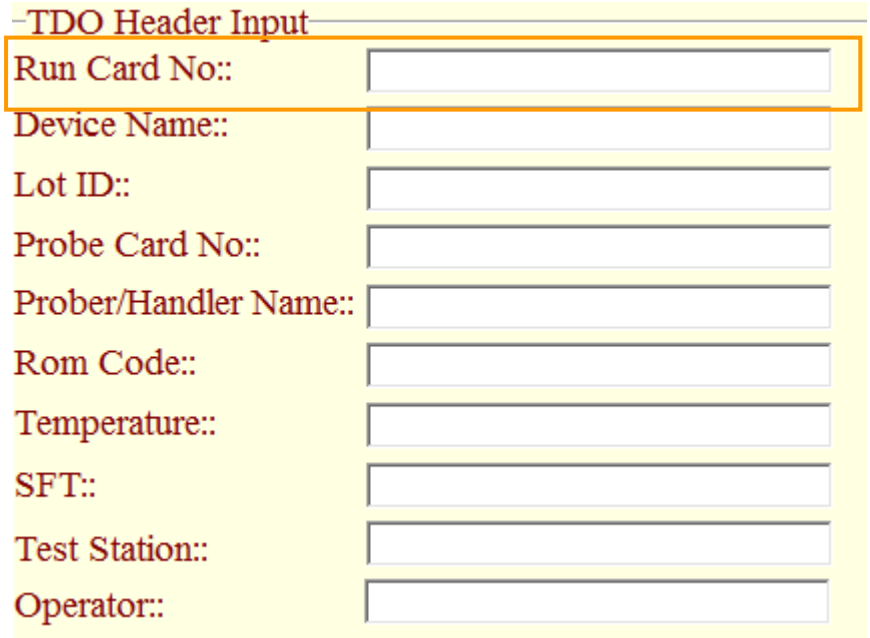

```
printf("Test Station :%s\n", TEST_STATION);
```

5.2.12 TEST_RUN_CARD_NO

5.2.12.1 Explanation

Provide user to get Run Card No Name from operation mode input information

5.2.12.2 Example



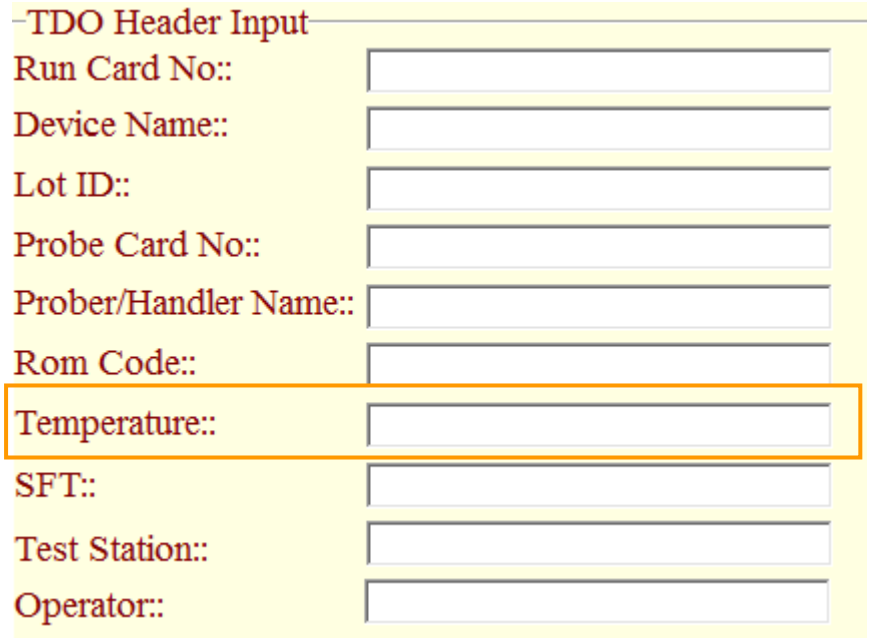
printf("Run Card No :%s\n", TEST_RUN_CARD_NO);

5.2.13 TEST_TEMPERATURE

5.2.13.1 Explanation

Provide user to get TEMPATURE from operation mode input information
In CP case, the name will get from Prober, not from user key in.

5.2.13.2 Example



printf("TEMPERATURE :%s\n", TEST_ TEMPERATURE);

5.2.14 DISABLE_SYSTEM_EOT_RESET

bitwise control each power board if do board reset after end-of-test

5.2.14.1 Format

Parameter list:

DISABLE_SYSTEM_EOT_RESET = **keyword** ;

Parameter comment:

Keyword 0x0 : All power brd will do brd-reset
BRD_MXDPS: the CRAFT will not do MXDPS board-reset after end-of-test
BRD_MXUVI: the CRAFT will not do MXUVI board-reset after end-of-test
BRD_MXREF: the CRAFT will not do MXREFI board-reset after end-of-test
BRD_MLDPS: the CRAFT will not do MLDPSI board-reset after end-of-test
Default: 0x0

NOTE: THIS LIST NEEDS TO CONTAIN ALL OF THE MENU SUPPORT ITEMS,such as Datalog, serial number, etc..

5.2.14.2 Example

```
DISABLE_SYSTEM_EOT_RESET = 0x0;   // the CRAFT will do all power board board-reset after end-of-test (system default)

//DISABLE_ SYSTEM _EOT_RESET = BRD_MXDPS ;   // the CRAFT will not do MXDPS board-reset after end-of-test


//DISABLE_ SYSTEM _EOT_RESET = BRD_MXDPS | BRD_MXUVI | BRD_MXREF;
// the CRAFT will not do MXDPS & MXUVI & MXREF & MLDPS board-reset after end-of-test
```

5.2.15 GET_TEST_COUNT

5.2.15.1 Explanation

Get pattern total test count

```
count_value = GET_TEST_COUNT ( socket_no ) ;
socket_no :
            S0 : socket No. 0
            S1 : socket No. 1
            …
            S31 : socket No. 31
            Type : integer

count_value : return total test count

e.g.;
int count_value;
   count_value = GET_TEST_COUNT ( S0 ) ;
   Printf("Total test count = %d\n", count_value);
```

5.2.15.2 Example

NORM-mode : the Pass/Fail condition returned test count value --> ok !

DBL-mode : A-B cycle
the Pass condition returned test count value --> ok !
the A-cycle Fail condition returned test count value --> ok !
the B-cycle Fail condition returned test count value --> same as A-cycle

```
SPM_PATTERN( func_pat, DBL)
{
    func_st ::
            *01LH* TS1;    // if Fail here    return Fail test count = 1;
            *01LH* ;       // if Fail here    return Fail test count = 1;
            *01HL* ;  //A-cycle: if Fail here    return Fail test count = 3;
            *01HH* ;   //B-cycle: if Fail here    return Fail test count = 3;
            …
}
```

DBL_2X-mode : A-B-C-D cycle
the Pass condition returned test count value --> ok !
the A-cycle Fail condition returned test count value --> ok !
the B-cycle Fail condition returned test count value --> same as A-cycle
the C-cycle Fail condition returned test count value --> same as A-cycle
the D-cycle Fail condition returned test count value --> same as A-cycle

```
SPM_PATTERN( func_pat, DBL_2X)
{
    func_st ::      *01LL* TS1;    // if Fail here    return Fail test count = 1;
            *01LL* ;       // if Fail here    return Fail test count = 1;
            *01LL* ;       // if Fail here    return Fail test count = 1;
            *01LL* ;
            *01HL* ;  // A-cycle: if Fail here    return Fail test count = 5;
            *01HH* ;   // B-cycle: if Fail here    return Fail test count = 5;
            *01HH* ;   // C-cycle: if Fail here    return Fail test count = 5;
            *01HH* ;   // D-cycle: if Fail here    return Fail test count = 5;
            …
}
```

5.2.16 GLOBAL PRODID

5.2.16.1 Explanation

The normal way "GLOBAL" doesn't display in Prodid utility.It changes GLOBAL PRODID, that is a global variable in 3380 production mode.It couldn't be initial value and will compiler fail.

5.2.16.2 Example

Dec file

```
GLOBAL PRODID   int   Log_opens_and_shorts ;
```

5.2.17 PLAN_VERSION

The statement allow users to program their test plan version and display on Datalog.

5.2.17.1 Format

Parameter list:

```
PLAN_VERSION("version_msg");
```

Parameter comment:

version_msg: String or variable
max. 32 chars

5.2.17.2 Example

PLN file

```
PLAN_VERSION("LS299 version 1.01");
```

Datalog output

```
############################################# END OF TEST
#############################################
#File : C:\CRAFT\TestPlanExamples\3380_64Site_AteChk\3380_atechk.pln (LS299 version 1.01)          #
#ATE ID :                                    (CRAFT_3380_1.10                                )          #
#LoadBoard Name: LPC0                                    ID : Not Provide LoadBoard ID                  #
#Condition : No Condition                              Spec. : FT_Test                                #
#LotID :                                                WaferID :                                      #
#DATE : 2012/07/11 10:34:28                             SERIAL NUMBER : 1                              #
#=================================================================================================
=============#
#          | P/F |                          HW Category                |                SW Category          #
#--------+-----+-----------------------------------------------+-----------------------------------#
# Site   0| F     |!13                fail_strb_window_test                |!13 fail_strb_window_test          #
#################################################################################################
#########
```

5.2.18 DISABLE_EOT_RESET

THIS IS A SPECIAL FUNCTION. To control which pin will not do reset action after end-of-test.
Release version: **CRAFT_3380_1.30**

5.2.18.1 Format

Parameter list:

```
DISABLE_EOT_RESET( pin_name );
```

Parameter comment:

pin_name: pin_def, pin_group (IO pin or Power pin)
**no_entry: illegal

5.2.18.2 Explanation

Power pin is setting for channel unit. That other power channel will be turn off on EOT.
IO pins are setting for 16 pins unit. The same unit IO channel will not turn off on EOT.

5.2.19 SET_PE_RELAY_XOR

if true then set PE relay exclusive. Set PE relay exclusive, means always only 1-relay ON.

5.2.19.1 Format

Parameter list:

SET_PE_RELAY_XOR = **keyword** ;

Parameter comment:

Keyword TRUE / 1 : set PE relay exclusive, means always only 1-relay ON
FALSE / 0 : turn-off the function (system default)
Default: FALSE

5.2.19.2 Example

```
SET_PE_RELAY_XOR = TRUE;

//SET_PE_RELAY_XOR = FALSE;

//SET_PE_RELAY_XOR = 1;   //set PE relay exclusive, means always only 1-relay ON
//SET_PE_RELAY_XOR = 0;   //turn-off the function
```

5.2.20 CHECK_PE_RELAY_HOT_SWITCHING

if true then check PE relay if hot switching

5.2.20.1 Format

Parameter list:

```
CHECK_PE_RELAY_HOT_SWITCHING = keyword ;
```

Parameter comment:

Keyword TRUE / 1 : if TRUE & relay hot switching event happened → then CRAFT will pop-up error message and unload the test program
FALSE / 0 : if FALSE & relay hot switching event happened → then CRAFT only print warning message on Datalog window.

Default: TRUE

5.2.20.2 Example

```
SET_PE_RELAY_XOR = FALSE;

CHECK_PE_RELAY_HOT_SWITCHING = TRUE;   //the CRAFT will check if relay hot switching in program (system default)
//CHECK_PE_RELAY_HOT_SWITCHING = FALSE;   //the CRAFT will not check if relay hot switching in program
//CHECK_PE_RELAY_HOT_SWITCHING = 1;   //the CRAFT will check if relay hot switching in program
//CHECK_PE_RELAY_HOT_SWITCHING = 0;   //the CRAFT will not check if relay hot switching in program
```

5.2.21 DISABLE_RELAY_HOT_SWITCHING_MESSAGE

if true then disable PE relay hot switching message

5.2.21.1 Format

Parameter list:

DISABLE_RELAY_HOT_SWITCHING_MESSAGE = **keyword** ;

Parameter comment:

Keyword TRUE : the CRAFT will not print relay hot switching message even really relay hot-switching event happened

FALSE : the CRAFT will print relay hot switching message when relay hot-switching event happened (system default)

Default: FALSE

5.2.21.2 Example

```
SET_PE_RELAY_XOR = FALSE;

CHECK_PE_RELAY_HOT_SWITCHING = TRUE;

DISABLE_RELAY_HOT_SWITCHING_MESSAGE = TRUE;    //the CRAFT will not print relay hot switching message even really relay hot-switching event happened

//DISABLE_RELAY_HOT_SWITCHING_MESSAGE = FALSE;    // the CRAFT will print relay hot switching message when relay hot-switching event happened (system default)
```

5.2.22 ALLOW_MULTI_JUDGE_IN_FAIL_REJUDGE

5.2.22.1 Explanation

Add the declaration **#define ALLOW_MULTI_JUDGE_IN_FAIL_REJUDGE**, when user using FAIL_REJUDGE command, contents of more than two JUDGE, only in each re-load program first show select / warning window.
Release version: **CRAFT_3380_2.40-SP5**

5.2.22.2 Example

PLN file

```
#define ALLOW_MULTI_JUDGE_IN_FAIL_REJUDGE
```

5.2.23 HWBIN

5.2.23.1 Explanation

User can use the marco definition to get back the hardware bin result by site.

5.2.23.2 Example

```
printf("site%d,hardware_bin:%d\n",SOCKET,HWBIN[SOCKET]);
```

5.2.24 SWBIN

5.2.24.1 Explanation

User can use the marco definition to get back the software bin result by site.

5.2.24.2 Example

```
printf("site%d,software_bin:%d\n",SOCKET,SWBIN[SOCKET]);
```

5.2.25 PHIF_X

5.2.25.1 Explanation

User can use the marco definition to get back prober X-axis coordinate by site.
Prober will provide only site0 coordinate. ATE will calculate coordinates of other sites based on site location.

5.2.25.2 Example

```
printf("site%d,XY_coor%d %d\n",SOCKET, PHIF_X[SOCKET],PHIF_Y[SOCKET]);
```

5.2.26 PHIF_Y

5.2.26.1 Explanation

User can use the marco definition to get back prober Y-asix coordinate by site.
Prober will provide only site0 coordinate. ATE will calculate coordinates of other sites based on site location.

5.2.26.2 Example

```
printf("site%d,XY_coor%d %d\n",SOCKET, PHIF_X[SOCKET],PHIF_Y[SOCKET]);
```

5.2.27 TEST_WAFER_ID

5.2.27.1 Explanation

User can use the marco definition to get back WAFER_ID from prober.

5.2.27.2 Example

```
printf("site%d,waferid=%s\n",SOCKET, TEST_WAFER_ID);
```

5.2.28 TEST_WAFERNO

5.2.28.1 Explanation

User can use the marco definition to get back WAFER number within PHIF.utility setting from prober.

5.2.28.2 Example

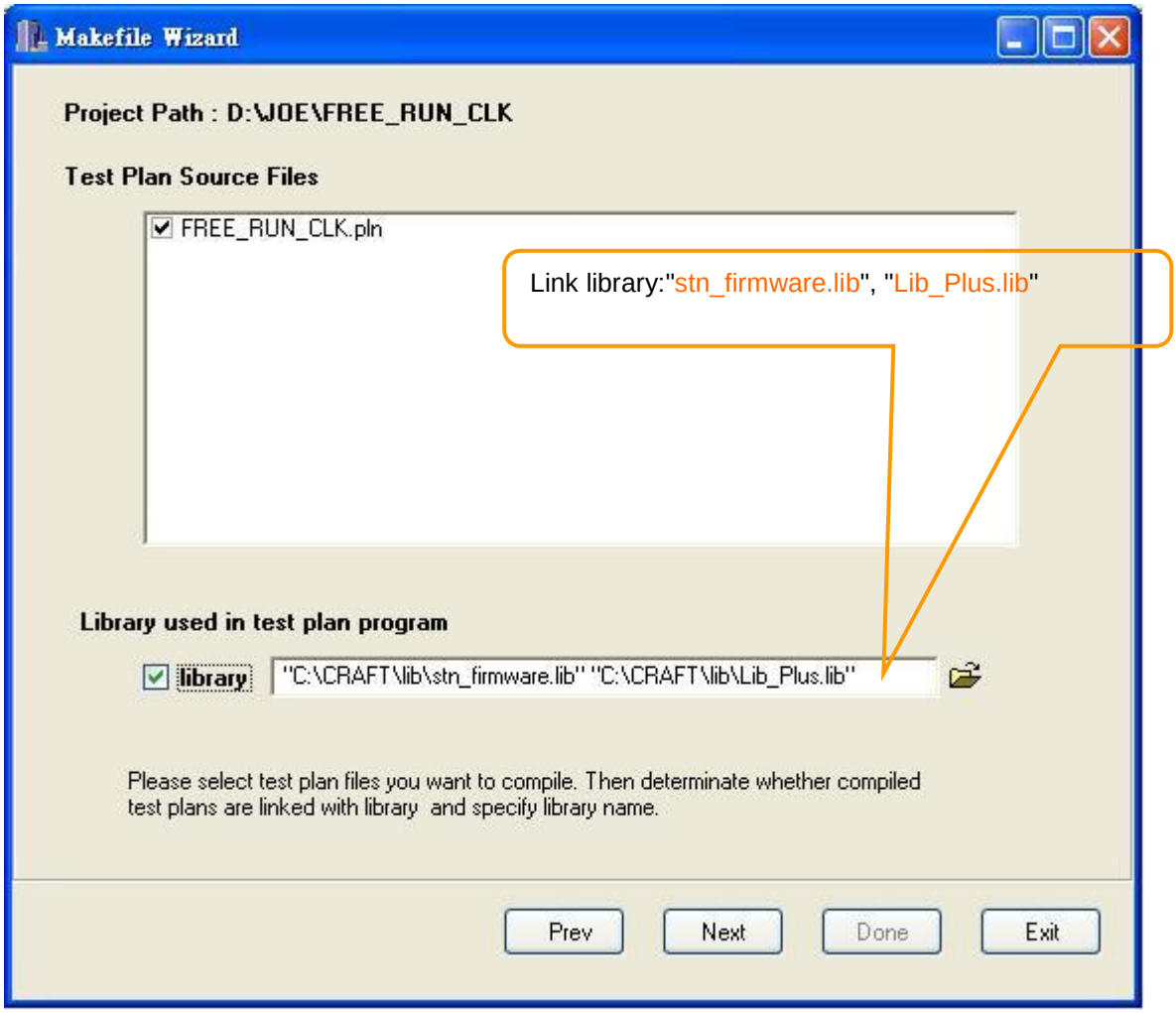
```
printf("site%d,waferno=%d\n",SOCKET, TEST_WAFERNO);
```

5.3 C Language Library

5.3.1 Method

5.3.1.1 Program compile

Need to link library:"stn_firmware.lib", "Lib_Plus.lib"
Use tool "Makefile Wizard" to compile
Makefile Wizard" to compile



5.3.1.2 Check file

Check rtlib.h in "C:\CRAFT\include\Lib_Plus"
Check Lib_Plus.lib in "C:\CRAFT\lib"
Check Lib_Plus.dll in "C:\CRAFT\bin"

Any files were lost, please contact CHROMA service.

5.3.2 GetPGChannel

5.3.2.1 Explanation

Get Pin Group channel for all sites

5.3.2.2 Format

```
GetPGChannel(char * , int *&);
```

5.3.2.3 Example

Example1
dec file

```
PIN_LIST( Amc_8_pins )
          {
            SB      = 0   : 20   = 1   = IO     ;   //IO
            QIN     = 1   : 21   = 2   = IO     ;   //IO   //D0
            WB      = 2   : 22   = 3   = IO     ;   //IO
            DI0     = 3   : 23   = 5   = IO     ;   //IO
            CL      = 4   : 24   = 6   = IO     ;   //IO
            HOLDB = 5   : 25   = 7   = IO     ;   //IO
            VCC =     576   : 577     = 8   = DPS   ;
           }
PIN_GROUP
{
   Data_pins       = QIN+DI0;
}
```

Pln file

```
//GET PIN_GRUP Channel //
int *getpgch;
int pin_cnt;
int i;
printf("GET PIN_GROUP Channel");
pin_cnt = GetPGChannel("Data_pins",getpgch);
printf("Data_pins pin count=%d\n",pin_cnt);

for(i=0;i<pin_cnt;i++)
{
      printf("Pin_number=%d\n",getpgch[i])
}
```

<Result>

```
GET PIN_GROUP Channel
Data_pins pin count=4
Pin_number=1
Pin_number=3
Pin_number=21
Pin_number=23
```

Example2

```
int *getpgch, *getpinch;
int pin_cnt;

//= if pin_no[] come from Pin_Group name =//
pin_cnt = GetPGChannel("Data_pins",getpgch);

CRAFT_c_relay_on_w(getpgch,pin_cnt,_PDCL_,5.0E-3); //relay setting
……..
CRAFT_c_relay_off_w(getpgch,pin_cnt,_PDCL_,5.0E-3); //relay setting


//= if pin_no[] come from single Pin name =//
pin_cnt = GetPinChannel("SB",getpinch);

CRAFT_c_relay_on_w(getpinch,pin_cnt,_PDCL_,5.0E-3); //relay setting
……..
CRAFT_c_relay_off_w(getpgch,pin_cnt,_PDCL_,5.0E-3); //relay setting
```

5.3.3 GetPGSiteChannel

5.3.3.1 Explanation

Get Pin Group channel for selected site

5.3.3.2 Format

```
GetPGSiteChannel(char * , unsigned short ,    int    *&);
```

5.3.3.3 Example

Get Site 0 PIN Group "Data_pins" channel print on datalog
Dec file

```
PIN_LIST( Amc_8_pins )
        {
          SB      = 0   : 20   = 1   = IO     ;   //IO
          QIN     = 1   : 21   = 2   = IO     ;   //IO   //D0
          WB      = 2   : 22   = 3   = IO     ;   //IO
          DI0     = 3   : 23   = 5   = IO     ;   //IO
          CL      = 4   : 24   = 6   = IO     ;   //IO
          HOLDB = 5   : 25   = 7   = IO     ;   //IO
          VCC =     576   : 577     = 8   = DPS   ;
         }
PIN_GROUP
{
  Data_pins       = QIN+DI0;
}
```

Pln file

```
//GET PIN_GRUP Channel for Site0 //
int *getpgsitech;
int pin_cnt;
int i;
printf("GetPGChannel for Site0\n");
pin_cnt = GetPGSiteChannel("Data_pins",S0,getpgsitech);
printf("Data_pins pin count=%d\n",pin_cnt);

for(i=0;i<pin_cnt;i++)
{
      printf("Pin_number=%d\n",getpgsitech[i])
}
```

<Result>

```
 8 GetPGSiteChannel for Site 0
 9 Data_pins pin count=2
10 Pin_number=1
11 Pin_number=3
```

5.3.4 GetPinSiteAllIOChannel

5.3.4.1 Explanation

Get all IO channel for selected Site

5.3.4.2 Format

```
GetPinSiteAllIOChannel(unsigned short    , int *&);
```

5.3.4.3 Example

Get Site 0 all IO channel

Dec file

```
PIN_LIST( Amc_8_pins )
        {
          SB      = 0   : 20   = 1   = IO     ;   //IO
          QIN     = 1   : 21   = 2   = IO     ;   //IO   //D0
          WB      = 2   : 22   = 3   = IO     ;   //IO
          DI0     = 3   : 23   = 5   = IO     ;   //IO
          CL      = 4   : 24   = 6   = IO     ;   //IO
          HOLDB = 5   : 25   = 7   = IO     ;   //IO
          VCC =     576   : 577     = 8   = DPS   ;
         }
PIN_GROUP
{
   Data_pins       = QIN+DI0;
}
```

Pln file

```
//GET All IO Channel for Site0//
int *getalliositech;
int pin_cnt;
int i;
printf("Get All IO Channel for Site 0 \n");
pin_cnt = GetPinSiteAllIOChannel(S0, getalliositech);
printf("Site 0 all io pin count=%d\n",pin_cnt);

for(i=0;i<pin_cnt;i++)
{
      printf("Pin_channel_number=%d\n", getalliositech[i])
}
```

<Result>

```
Get All IO Channel for Site 0
Site 0 all io pin count=6
Pin_channel_number=0
Pin_channel_number=1
Pin_channel_number=2
Pin_channel_number=3
Pin_channel_number=4
Pin_channel_number=5
```

5.3.5 GetPinSiteAllDPSChannel

5.3.5.1 Explanation

Get all DPS channel for selected Site

5.3.5.2 Format

```
GetPinSiteAllDPSChannel(unsigned short    , int *&);
```

5.3.5.3 Example

Dec file

```
PIN_LIST( Amc_8_pins )
        {
          SB      =   0     : 20     = 1 = IO     ;   //IO
          QIN     =   1     : 21     = 2 = IO     ;   //IO   //D0
          WB      =   2     : 22     = 3 = IO     ;   //IO
          DI0     =   3     : 23     = 5 = IO     ;   //IO
          CL      =   4     : 24     = 6 = IO     ;   //IO
          HOLDB =   5     : 25     = 7 = IO     ;   //IO
          VCC     =   576 : 577    = 8 = DPS    ;
          DP2     =   578 : 579    = 9 = DPS ;
          DP3     =   580 : 581    = 10 = DPS;
          PF1     =   448 : 449    = 11 = PREF;
          PF2     =   450 : 451    = 12 = PREF;
          PF3     =   452 : 453    = 13 = PREF;
        }
```

Pln file

```
//GET All DPS Channel for Site0//
int *getalldpssitech;
int pin_cnt;
int i;
printf("Get All DPS Channel for Site 0 \n");
pin_cnt = GetPinSiteAllDPSChannel(S0, getalldpssitech);
printf("Site 0 all DPS pin count =%d\n",pin_cnt);

for(i=0;i<pin_cnt;i++)
{
        printf("DPS_Pin_channel_number=%d\n", getaldpssitech[i])
}
```

<Result>

```
Get All DPS Channel for Site 0
Site 0 all DPS pin count =3
DPS_Pin_channel_number=576
DPS_Pin_channel_number=578
DPS_Pin_channel_number=580
```

5.3.6 GetPinSiteAllPREFChannel

5.3.6.1 Explanation

Get all PREF channel for selected Site

5.3.6.2 Format

```
GetPinSiteAllPREFChannel(unsigned short    , int *&);
```

5.3.6.3 Example

Dec file

```
PIN_LIST( Amc_8_pins )
        {
          SB      =   0     : 20     = 1 = IO     ;   //IO
          QIN     =   1     : 21     = 2 = IO     ;   //IO   //D0
          WB      =   2     : 22     = 3 = IO     ;   //IO
          DI0     =   3     : 23     = 5 = IO     ;   //IO
          CL      =   4     : 24     = 6 = IO     ;   //IO
          HOLDB =   5     : 25     = 7 = IO     ;   //IO
          VCC     =   576 : 577    = 8 = DPS    ;
          DP2     =   578 : 579    = 9 = DPS ;
          DP3     =   580 : 581    = 10 = DPS;
          PF1     =   448 : 449    = 11 = PREF;
          PF2     =   450 : 451    = 12 = PREF;
          PF3     =   452 : 453    = 13 = PREF;
        }
```

Pln file

```
//GET All pref Channel for Site0//
int *getallprefsitech;
int pin_cnt;
int i;
printf("Get All PREF Channel for Site 0 \n");
pin_cnt = GetPinSiteAllPREFChannel(S0, getallprefsitech);
printf("Site 0 all PREF pin count =%d\n",pin_cnt);

for(i=0;i<pin_cnt;i++)
{
        printf("PREF_Pin_channel_number=%d\n", getalprefsitech[i])
}
```

<Result>

```
Get All PREF Channel for Site 0
Site 0 all PREF pin count =3
PREF_Pin_channel_number=448
PREF_Pin_channel_number=450
PREF_Pin_channel_number=452
```

5.3.7 GetPinSiteAllUVIChannel

5.3.7.1 Explanation

Get all UVI channel for selected Site

5.3.7.2 Format

```
GetPinSiteAllUVIChannel(unsigned short    , int *&);
```

5.3.7.3 Example

Dec file

```
PIN_LIST( Amc_8_pins )
        {
          SB      =   0     : 20     = 1 = IO     ;   //IO
          QIN     =   1     : 21     = 2 = IO     ;   //IO   //D0
          WB      =   2     : 22     = 3 = IO     ;   //IO
          DI0     =   3     : 23     = 5 = IO     ;   //IO
          CL      =   4     : 24     = 6 = IO     ;   //IO
          HOLDB =   5     : 25     = 7 = IO     ;   //IO
          VCC     =   576 : 577    = 8 = DPS    ;
          DP2     =   578 : 579    = 9 = DPS ;
          DP3     =   580 : 581    = 10 = DPS;
          PF1     =   448 : 449    = 11 = PREF;
          PF2     =   450 : 451    = 12 = PREF;
          PF3     =   452 : 453    = 13 = PREF;
          UV1     =   224 : 225    = 14 = UVI;
          UV2     =   226 : 227    = 15 = UVI;
          TG1     =   6     : 26     = 16 = TRG;
        }
```

Pln file

```
//GET All uvi Channel for Site0//
int *getalluvisitech;
int pin_cnt;
int i;
pin_cnt = GetPinSiteAllUVIChannel(S0, getalluvisitech);
printf("Site 0 all UVI pin count =%d\n",pin_cnt);

for(i=0;i<pin_cnt;i++)
{
    printf("UVI_Pin_channel_number=%d\n", getaluvisitech[i])
}
```

<Result>

```
Site 0 all UVI pin count =2
PREF_Pin_channel_number=224
PREF_Pin_channel_number=226
```

5.3.8 GetPinSiteAllTRGChannel

5.3.8.1 Explanation

Get all TRG channel for selected Site

5.3.8.2 Format

```
GetPinSiteAllTRGChannel(unsigned short    , int *&);
```

5.3.8.3 Example

Dec file

```
PIN_LIST( Amc_8_pins )
        {
          SB      =   0     : 20     = 1 = IO     ;   //IO
          QIN     =   1     : 21     = 2 = IO     ;   //IO   //D0
          WB      =   2     : 22     = 3 = IO     ;   //IO
          DI0     =   3     : 23     = 5 = IO     ;   //IO
          CL      =   4     : 24     = 6 = IO     ;   //IO
          HOLDB =   5     : 25     = 7 = IO     ;   //IO
          VCC     =   576 : 577    = 8 = DPS    ;
          DP2     =   578 : 579    = 9 = DPS ;
          DP3     =   580 : 581    = 10 = DPS;
          PF1     =   448 : 449    = 11 = PREF;
          PF2     =   450 : 451    = 12 = PREF;
          PF3     =   452 : 453    = 13 = PREF;
          UV1     =   224 : 225    = 14 = UVI;
          UV2     =   226 : 227    = 15 = UVI;
          TG1     =   6     : 26     = 16 = TRG;
        }
```

Pln file

```
//GET All TRG Channel for Site0//
int *getalltrgsitech;
int pin_cnt;
int i;
pin_cnt = GetPinSiteAllTRGChannel(S0, getalltrgsitech);
printf("Site 0 all TRG pin count =%d\n",pin_cnt);

for(i=0;i<pin_cnt;i++)
{
    printf("TRG_Pin_channel_number=%d\n", getaltrgsitech[i])
}
```

<Result>

```
Site 0 all TRG pin count =1
TRG_Pin_channel_number=6
```

5.3.9 DPS2SITE

5.3.9.1 Explanation

Use the function to know which Site use DPS channel

5.3.9.2 Format

```
DPS2SITE(unsigned short);
```

5.3.9.3 Example

Dec file

```
PIN_LIST( Amc_8_pins )
        {
          SB      =   0     : 20     = 1 = IO     ;   //IO
          QIN     =   1     : 21     = 2 = IO     ;   //IO   //D0
          WB      =   2     : 22     = 3 = IO     ;   //IO
          DI0     =   3     : 23     = 5 = IO     ;   //IO
          CL      =   4     : 24     = 6 = IO     ;   //IO
          HOLDB =   5     : 25     = 7 = IO     ;   //IO
          VCC     =   576 : 577    = 8 = DPS    ;
          DP2     =   578 : 579    = 9 = DPS ;
          DP3     =   580 : 581    = 10 = DPS;
          PF1     =   448 : 449    = 11 = PREF;
          PF2     =   450 : 451    = 12 = PREF;
          PF3     =   452 : 453    = 13 = PREF;
          UV1     =   224 : 225    = 14 = UVI;
          UV2     =   226 : 227    = 15 = UVI;
          TG1     =   6     : 26     = 16 = TRG;
        }
```

Pln file

```
// Get SOCKET Number to DPS //
  int getsite2dps

printf("Get site to DPS channel \n");
getsite2dps=DPS2SITE(577);
printf("DPS577 is used by SOCKET %d \n", getsite2dps);
```

<Result>

```
Get site to DPS channel
DPS577 is used by SOCKET 1
```

5.3.10 PREF2SITE

5.3.10.1 Explanation

Use the function to know which Site use PREF channel

5.3.10.2 Format

```
PREF2SITE(unsigned short);
```

5.3.10.3 Example

Dec file

```
PIN_LIST( Amc_8_pins )
        {
          SB      =   0     : 20     = 1 = IO     ;   //IO
          QIN     =   1     : 21     = 2 = IO     ;   //IO   //D0
          WB      =   2     : 22     = 3 = IO     ;   //IO
          DI0     =   3     : 23     = 5 = IO     ;   //IO
          CL      =   4     : 24     = 6 = IO     ;   //IO
          HOLDB =   5     : 25     = 7 = IO     ;   //IO
          VCC     =   576 : 577    = 8 = DPS    ;
          DP2     =   578 : 579    = 9 = DPS ;
          DP3     =   580 : 581    = 10 = DPS;
          PF1     =   448 : 449    = 11 = PREF;
          PF2     =   450 : 451    = 12 = PREF;
          PF3     =   452 : 453    = 13 = PREF;
          UV1     =   224 : 225    = 14 = UVI;
          UV2     =   226 : 227    = 15 = UVI;
          TG1     =   6     : 26     = 16 = TRG;
        }
```

Pln file

```
// Get SOCKET Number to PREF //
  int getsite2pref

printf("Get site to PREF channel \n");
getsite2pref=PREF2SITE(448);
printf("PREF448 is used by SOCKET %d \n", getsite2pref);
```

<Result>

```
Get site to PREF channel
PREF448 is used by SOCKET 0
```

5.3.11 UVI2SITE

5.3.11.1 Explanation

Use the function to know which Site use UVI channel.

5.3.11.2 Format

```
UVI2SITE(unsigned short);
```

5.3.11.3 Example

Dec file

```
PIN_LIST( Amc_8_pins )
        {
          SB      =   0     : 20     = 1 = IO     ;   //IO
          QIN     =   1     : 21     = 2 = IO     ;   //IO   //D0
          WB      =   2     : 22     = 3 = IO     ;   //IO
          DI0     =   3     : 23     = 5 = IO     ;   //IO
          CL      =   4     : 24     = 6 = IO     ;   //IO
          HOLDB =   5     : 25     = 7 = IO     ;   //IO
          VCC     =   576 : 577    = 8 = DPS    ;
          DP2     =   578 : 579    = 9 = DPS ;
          DP3     =   580 : 581    = 10 = DPS;
          PF1     =   448 : 449    = 11 = PREF;
          PF2     =   450 : 451    = 12 = PREF;
          PF3     =   452 : 453    = 13 = PREF;
          UV1     =   224 : 225    = 14 = UVI;
          UV2     =   226 : 227    = 15 = UVI;
          TG1     =   6     : 26     = 16 = TRG;
        }
```

Pln file

```
// Get SOCKET Number to UVI //
  int getsite2uvi

printf("Get site to UVI channel \n");
getsite2uvi=UVI2SITE(224);
printf("UVI224 is used by SOCKET %d \n", getsite2uvi);
```

<Result>

```
Get site to UVI channel
PREF224 is used by SOCKET 0
```

5.3.12 GetPinNameByChannel

5.3.12.1 Explanation

Use the function to know PIN name from channel

5.3.12.2 Format

```
GetPinNameByChannel(unsigned short);
```

5.3.12.3 Example

```
PIN_LIST( Amc_8_pins )
          {
            SB      = 0     : 20   = 1     = IO     ;   //IO
            QIN     = 1     : 21   = 2     = IO     ;   //IO   //D0
            WB      = 2     : 22   = 3     = IO     ;   //IO
            DI0     = 3     : 23   = 5     = IO     ;   //IO
            CL      = 4     : 24   = 6     = IO     ;   //IO
            HOLDB = 5     : 25   = 7     = IO     ;   //IO
            VCC     = 576 : 577 = 8     = DPS    ;
          }
PIN_GROUP
{
  Data_pins      = QIN+DI0;
}
```

```
char *Pname;
PIN_INC(Data_pins)
{
      RELAY_OFF(PIN_DEF, PDCL, 1mS);
      RELAY_ON(PIN_DEF, PMU, 1mS);
      FORCE_I_PMU(PMU, 200uA, @1mA, @6V, 6V, ON, 2mS);
      JUDGE_V_PMU(PMU, IFVM, 200mA, 1.2V, 2mS, 16, AVE, 50uS, , );
      RELAY_OFF(PIN_DEF, PMU, 1mS);
      RELAY_ON(PIN_DEF, PDCL, 1mS);
      SOCKET_INC{
            If(JUDGE_RESULT_MS[SOCKET]==FAIL)
            Pname=GetPinNameByChannel(PIN_INC_ATE[0]);
            printf("SOCKET %d , channel %2d , PIN_NAME %s is fail \n",SOCKET,
PIN_INC_ATE[SOCKET],Pname);
      }
}
```

<Result>

```
51    -1      1  *F      +0.000  V   +200.000 mV     +1.200  V     0  open_all (QIN(1))
52    -1     21  *F      +0.000  V   +200.000 mV     +1.200  V     1  open_all (QIN(21))
53
54 SOCKET 0 ,channel  1 ,PIN_NAME QIN  is fail
55 SOCKET 1 ,channel 21 ,PIN_NAME QIN  is fail
56    -1      3  *F      +0.000  V   +200.000 mV     +1.200  V     0  open_all (DIO(3))
57    -1     23  *F      +0.000  V   +200.000 mV     +1.200  V     1  open_all (DIO(23))
58
59 SOCKET 0 ,channel  3 ,PIN_NAME DIO  is fail
60 SOCKET 1 ,channel 23 ,PIN_NAME DIO  is fail
```

5.3.13 GetDutPin

5.3.13.1 Explanation

To get Dut number of selected pin.

5.3.13.2 Format

```
Char *=GetDutPin(char *);
```

5.3.13.3 Example

```
PIN_LIST( Amc_8_pins )
          {
            SB      = 0     : 20   = 1     = IO     ;   //IO
            QIN     = 1     : 21   = 2     = IO     ;   //IO   //D0
            WB      = 2     : 22   = 3     = IO     ;   //IO
            DI0     = 3     : 23   = 5     = IO     ;   //IO
            CL      = 4     : 24   = 6     = IO     ;   //IO
            HOLDB = 5     : 25   = 7     = IO     ;   //IO
            VCC     = 576 : 577 = 8     = DPS    ;
           }
PIN_GROUP
{
   Data_pins       = QIN+DI0;
}
```

```
char *Pname;
char *dutpin;
PIN_INC(Data_pins)
{
      RELAY_OFF(PIN_DEF, PDCL, 1mS);
      RELAY_ON(PIN_DEF, PMU, 1mS);
      FORCE_I_PMU(PMU, 200uA, @1mA, @6V, 6V, ON, 2mS);
      JUDGE_V_PMU(PMU, IFVM, 200mA, 1.2V, 2mS, 16, AVE, 50uS, , );
      RELAY_OFF(PIN_DEF, PMU, 1mS);
      RELAY_ON(PIN_DEF, PDCL, 1mS);
      SOCKET_INC{
            If(JUDGE_RESULT_MS[SOCKET]==FAIL)
             Pname=GetPinNameByChannel(PIN_INC_ATE[0]);
             printf("SOCKET %d , channel %2d , PIN_NAME %s is fail \n",SOCKET,
PIN_INC_ATE[SOCKET],Pname);
            dutpin=GetDutPin(Pname);
            printf("%s dut number is %s \n",Pname,GetDutPin(Pname));
      }
}
```

<Result>

```
61    -1      1  *F     +0.000  V   +200.000 mV     +1.200  V     0  --NULL-- (QIN(1))
62    -1     21  *F     +0.000  V   +200.000 mV     +1.200  V     1  --NULL-- (QIN(21))
63
64 SOCKET 0 ,channel  1 ,PIN_NAME QIN  is fail
65 QIN dut number is 2
66 SOCKET 1 ,channel 21 ,PIN_NAME QIN  is fail
67 QIN dut number is 2
68    -1      3  *F     +0.000  V   +200.000 mV     +1.200  V     0  --NULL-- (DIO(3))
69    -1     23  *F     +0.000  V   +200.000 mV     +1.200  V     1  --NULL-- (DIO(23))
70
71 SOCKET 0 ,channel  3 ,PIN_NAME DIO  is fail
72 DIO dut number is 5
73 SOCKET 1 ,channel 23 ,PIN_NAME DIO  is fail
74 DIO dut number is 5
```

5.3.14 GetAllPinName

5.3.14.1 Explanation

To get all declared pin name.

5.3.14.2 Format

```
GetAllPinName(char    **    &);
```

5.3.14.3 Example

```
PIN_LIST( Amc_8_pins )
        {
          SB      = 0     : 20   = 1     = IO     ;   //IO
          QIN     = 1     : 21   = 2     = IO     ;   //IO   //D0
          WB      = 2     : 22   = 3     = IO     ;   //IO
          DI0     = 3     : 23   = 5     = IO     ;   //IO
          CL      = 4     : 24   = 6     = IO     ;   //IO
          HOLDB = 5     : 25   = 7     = IO     ;   //IO
          VCC     = 0     : 1    = 8     = DPS    ;
          DP2     = 2     : 3    = 9     = DPS;
          DP3     = 4     : 5    = 10    = DPS;
          PF1     = 0     : 1    = 11    = PREF;
          PF2     = 2     : 3    = 12    = PREF;
          PF3     = 4     : 5    = 13    = PREF;
          UV1     = 257 : 258 = 14    = UVI;
          UV2     = 259 : 260 = 15    = UVI;
          TG1     = 6     : 26   = 16    = TRG;

         }
```

```
//get all pin name//
char *allpinname;
int pin_cnt;

printf("All PIN name list");
pin_cnt=GetAllPinName(allpinname);
for(i=0;i<pin_cnt;i++)
{
      printf("number %d of all pin name is %s \n",i, allpinname[i]);
}
```

<Result>

```
All PIN name list
Number 0 of all pin name is SB
Number 1 of all pin name is QIN
Number 2 of all pin name is WB
Number 3 of all pin name is DI0
Number 4 of all pin name is CL
Number 5 of all pin name is HOLDB
Number 6 of all pin name is VCC
Number 7 of all pin name is DP2
Number 8 of all pin name is DP3
Number 9 of all pin name is PF1
Number 10 of all pin name is PF2
Number 11 of all pin name is PF3
Number 12 of all pin name is UV1
Number 13 of all pin name is UV2
Number 14 of all pin name is TG1
```

5.3.15 GetPinChannel

5.3.15.1 Explanation

Get Pin channel for all sites

5.3.15.2 Format

```
GetPinChannel(char * , int *&);
```

5.3.15.3 Example

<Get SB channel>

```
PIN_LIST( Amc_8_pins )
        {
          SB      = 0     : 20   = 1     = IO     ;   //IO
          QIN     = 1     : 21   = 2     = IO     ;   //IO   //D0
          WB      = 2     : 22   = 3     = IO     ;   //IO
          DI0     = 3     : 23   = 5     = IO     ;   //IO
          CL      = 4     : 24   = 6     = IO     ;   //IO
          HOLDB = 5     : 25   = 7     = IO     ;   //IO
          VCC     = 0     : 1    = 8     = DPS    ;
          DP2     = 2     : 3    = 9     = DPS;
          DP3     = 4     : 5    = 10    = DPS;
          PF1     = 0     : 1    = 11    = PREF;
          PF2     = 2     : 3    = 12    = PREF;
          PF3     = 4     : 5    = 13    = PREF;
          UV1     = 257 : 258 = 14    = UVI;
          UV2     = 259 : 260 = 15    = UVI;
          TG1     = 6     : 26   = 16    = TRG;

         }
```

```
// GET PIN Channel //
int *getpinch;
int cnt;
printf("Get PIN Channel\n");
cnt=GetPinChannel("SB",getpinch);
for(int i=0;i<cnt;i++)
{
        printf("SITE %d SB_channel_number=%d\n",i,getpinch[i]);
}
```

<Result>

```
Get PIN Channel
SITE 0 SB_channel_number = 0
SITE 1 SB_channel_number = 20
```

5.3.16 CRAFT_c_get_brd_type

5.3.16.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.16.2 Format

```
LoadRTLibrary();// Load Library
int CRAFT_c_get_brd_type (int &io_brd_type);
UnloadRTLibrary();//Unload Library
```

io_brd_type:
_3500_NULL_SLOT : Mean IO Board Type does NULL (_3500_NULL_SLOT=0).
_3500_LPC_SLOT : Mean IO Board Type does STLPC (_3500_LPC_SLOT=1).
_3500_LPC32_SLOT: Mean IO Board Type does LPC32 or SNPEC (_3500_LPC32_SLOT=8).

Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.
>> LoadRTLibrary():.suggest to Execute it in "START_UP or test_start" item
>> UnloadRTLibrary():.suggest to Execute it in " RELEASE or test_end" item
Return Value

On success return 0,

5.3.16.3 Example

```
open_short_test()
{
   int io_brd_type;
   int sqpg_config, dps_config, lpc_config[19], ii;
   if(CRAFT_c_get_brd_type(io_brd_type)= =0) printf("CRAFT_c_get_brd_type
SUCCESS\n");
   else   printf("CRAFT_c_get_brd_type FAIL \n");
   if(io_brd_type= = _3500_LPC_SLOT)   //STLPC
   {
         printf("io_brd_type is STLPC\n");
         DATALOG_MSG="O/S negative-PMU";
         FORCE_V_DPS( Vdd,0V,@8V, @100mA, 100mA, NORM, ON, 1mS) ;
         SET_LEVELN(os_lvl, IO_ALLPINS,0.0V, 0.5V, -1.5V, -0.2V, 150uA, -150uA, -2V, , );
         LEVELS(os_lvl, 5mS );
         RELAY_ON(IO_ALLPINS,PDCL, 1mS);
         RATEN(TS15, 10uS);
         MARKERN( TS15, IO_ALLPINS, 0nS, 0nS, , 0nS, 0nS, 9uS, 9.1uS);
         PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, NORM);
         TEST_NO(17);
         JUDGE_PAT(os_st, os_sp );
         LEVELS(rst_lvl, 2mS );
         RELAY_OFF(IO_ALLPINS,PDCL, 1mS);
   }
   else   //NULL
   {printf("io_brd_type=%d, io slot is null !! \n",io_brd_type);}
```

5.3.17 CRAFT_c_get_slot_config

5.3.17.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.17.2 Format

```
LoadRTLibrary();// Load Library
            int CRAFT_c_get_slot_config(
            int &sqpg_config,
            int &dps_config,
            int *lpc_config);
UnloadRTLibrary();//Unload Library
```

sqpg_config:
_3500_NULL_SLOT: Mean SQPG slot configuration does NULL (_3500_NULL_SLOT=0).
_3360_LXPG2_SLOT: Mean SQPG slot configuration does LXPG2 (_3360_LXPG2_SLOT=20).

dps_config:
_3500_NULL_SLOT: Mean DPS slot configuration does NULL (_3500_NULL_SLOT=0).
_3500_DPS_SLOT: Mean DPS slot configuration does STDPS (_3500_DPS_SLOT=4).
_3360_HCDPS_SLOT: Mean DPS slot configuration does HCDPS (_3360_HCDPS_SLOT=30).
_3362_LXPDG_SLOT: Mean DPS slot configuration does LXPDG (_3362_LXPDG_SLOT=24).

lpc_config:
the array variable "lpc_config" array size must be 19; //3360 have 19 IO-Slot
_3500_NULL_SLOT: Mean LPC slot configuration does NULL (_3500_NULL_SLOT=0).
_3500_LPC_SLOT: Mean LPC slot configuration does STLPC (_3500_LPC_SLOT=1).
_3500_LPC32_SLOT: Mean LPC slot configuration does LPC32 (_3500_LPC32_SLOT=8).
_3500_DLPC_SLOT: Mean LPC slot configuration does SDLPC (_3500_DLPC_SLOT=7).
_3360_LXUVI_SLOT: Mean LPC slot configuration does LXUVI (_3360_LXUVI_SLOT=13).
.
_3360_LXAWI_SLOT: Mean LPC slot configuration does LXAWI (_3360_LXAWI_SLOT=16).
_3360_PTRLY_SLOT: Mean LPC slot configuration does PTRLY (_3360_PTRLY_SLOT=17).

_3360_LXREF_SLOT: Mean LPC slot configuration does LXREF (_3360_LXREF_SLOT=18).
_3360_SNPEC_SLOT: Mean LPC slot configuration does SNPEC (_3360_SNPEC_SLOT=23).
_3360_LXPXI_SLOT: Mean LPC slot configuration does LXPXI (_3360_LXPXI_SLOT=26).
_3360_MVUVI_SLOT: Mean LPC slot configuration does MVUVI (_3360_MVUVI_SLOT=31).
_3360_HVREF_SLOT: Mean LPC slot configuration does HVREF (_3360_HVREF_SLOT=33).

Return Value

On success return 0

Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.

>> LoadRTLibrary():.suggest to Execute it in "START_UP or test_start" item

>> UnloadRTLibrary():.suggest to Execute it in " Release or test_end" item

5.3.17.3 Example

```
Get slot config( )
{
    int sqpg_config, dps_config, lpc_config[19], ii;
    if(CRAFT_c_get_slot_config(sqpg_config, dps_config, lpc_config)= =0)
    {
            printf("CRAFT_c_get_slot_config SUCCESS\n");
    }
    else    printf("CRAFT_c_get_slot_config FAIL \n");

    printf("sqpg_config=%d \n",sqpg_config);
    printf("dps_config=%d \n",dps_config);
    for (ii=0; ii<19; ii++)
    {
    printf("lpc_config[%d]=%d \n",ii,lpc_config[ii]);
    }
}
```

5.3.18 CRAFT_c_pin_mode_w

This function is the similar "PIN_MODE" statement only using for setting the IO pins driver format, IO format, compare mode, and load mode

5.3.18.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

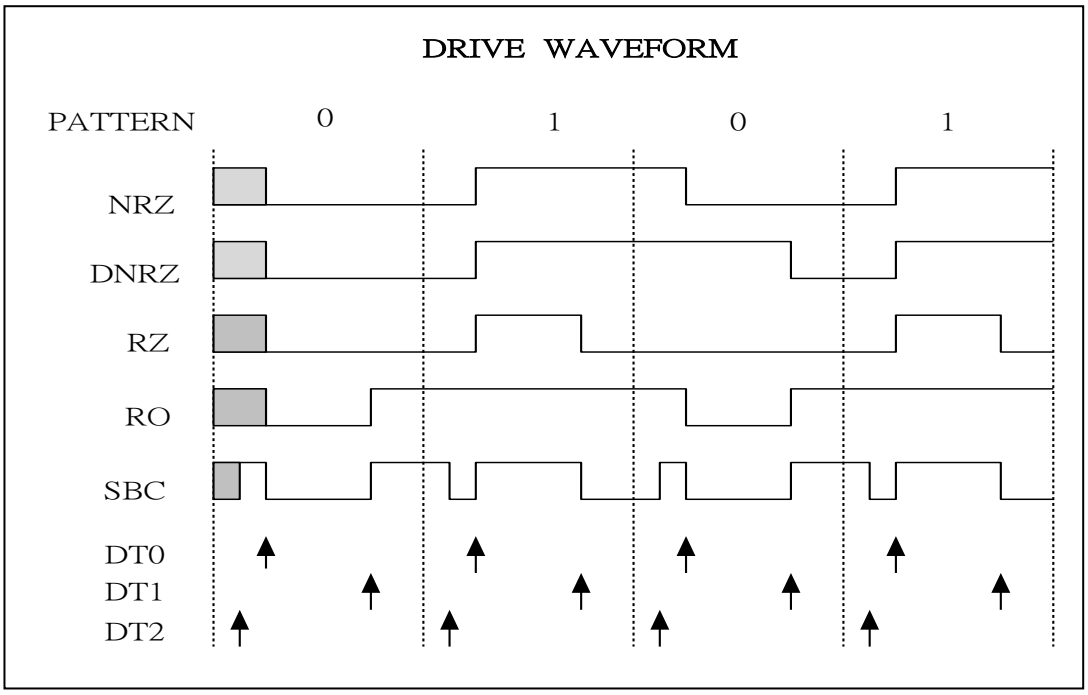
5.3.18.2 Format

```
LoadRTLibrary();// Load Library
int CRAFT_c_force_v_stdps_w (
        int *dps_no,
        int dps_cnt,
        double f_volt,
        ATECalibTable *calibtab,
          );
UnloadRTLibrary();//Unload Library
```

Pin_no : "int array"
Example: use channel: "0", "10"
➔int Pin_no l[2]={ 0 , 10 };

Pin_cnt : pin count ; Type: integral
Example: use channel: "0", "10"
➔ Pin_cnt =2;

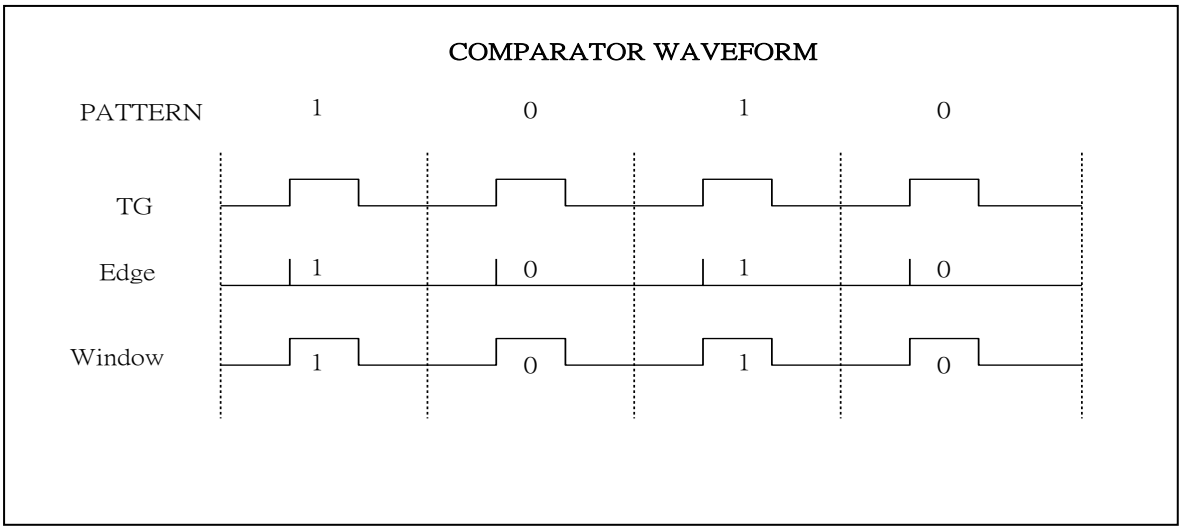
d_format :_NRZ_, _DNRZ_, _RZ_, _RO_, _SBC_, _FCLK_, _F0_and _F1_
(_F0_: Force Low, _F1_: Force High)
FCLK: free-run-clock format
type: integer



c_mode: One of two possible comparator strobe modes. These modes are set prior to the
pattern execution starts, and it will not change during the pattern execution.

WINDOW : Strobe occurs only at the first marker and are edge triggered.

EDGE : Strobe occurs for the duration of an internal pulse (a window), generated between T1 and T2 strobe markers.
Type integer



c_mask : _MASK_: master mask on
ENABLE: master mask off
Type integer

io_format : _IO_NRZ_ , _IO_RZ_ , _IO_RO_ , _IO_ON_ , _IO_OFF_ .
Type integer

load_mode : NORM, ON, OFF ;
_LOAD_NORM_: load is on when driver is off, and off when driver is on.
_LOAD_ON_: load is always on whether driver is on or off.
_LOAD_OFF_: load is always off whether driver is on or off.
Type integer

Return Value
On success return 0,

Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.

>> LoadRTLibrary():.suggest to Execute it in "START_UP or test_start" item

>> UnloadRTLibrary():.suggest to Execute it in " Release or test_end" item

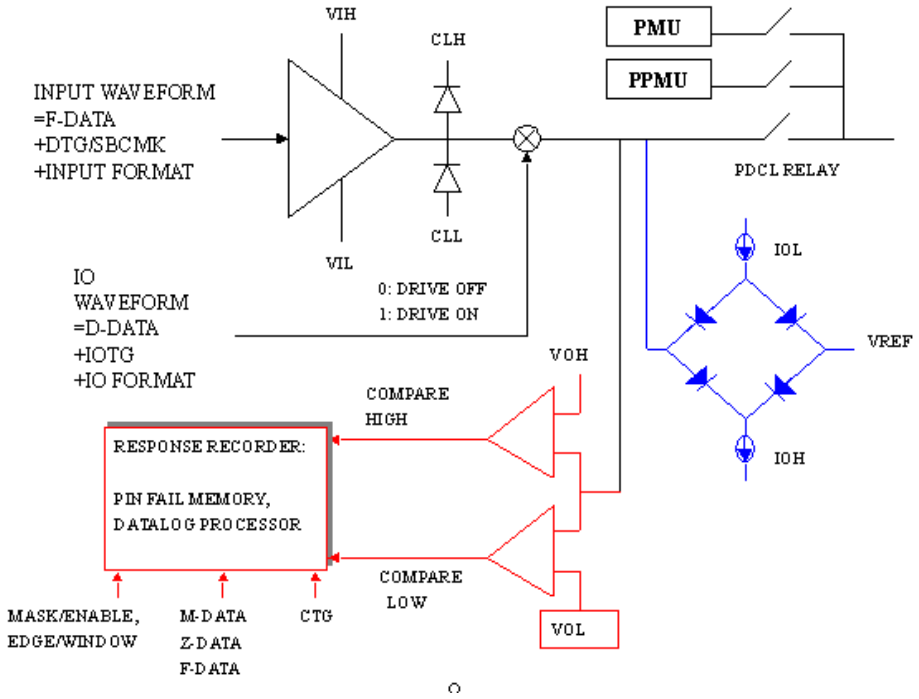
5.3.18.3 Example

```
fclk_setting_by_plus_function ()
{
int pin_no[2]={0,1}; // channel number
int pin_cnt=2;
CRAFT_c_pin_mode_w(pin_no,pin_cnt,_FCLK_,_EDGE_,_MASK_,_IO_NRZ_,_L
OAD_OFF_);

CRAFT_c_frun_clk_w(pin_no,pin_cnt,100.0E-9,25.0E-9,75.0E-9,
                    (&(Plugin.ShmPtr->CalibTab)) );
```

```
CRAFT_c_set_leveln_w (15, pin_no, pin_cnt, 0.0, 3.0, 1.0, 2.5,
                        0.0, 0.0, 1.5, -1.0, 5.0, (&(Plugin.ShmPtr->CalibTab)) );

CRAFT_c_relay_on_w(pin_no,pin_cnt,_PDCL_,5.0E-3); //relay setting
……..
……..
……..
……..

CRAFT_c_relay_off_w(pin_no,pin_cnt,_PDCL_,5.0E-3); //relay setting
}
```

5.3.19 CRAFT_c_relay_on_w

This function is the similar "RELAY_ON" statement only used to connect the specified relays to the DUT.



5.3.19.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.19.2 Format

```
LoadRTLibrary();// Load Library
CRAFT_c_relay_on_w(
        int *pin_no,
        int pin_cnt,
        int relay_resource,
        double delay
        );
UnloadRTLibrary();//Unload Library
```

Pin_no : "int array"

Example: use channel: "0", "10"

➔int Pin_no l[2]={ 0 , 10 };

Pin_cnt : pin count ; Type: integral

Example: use channel: "0", "10"

➔ Pin_cnt =2;

relay_resource : The instrument to be connected to the pin under test. Valid entries are:
PMU (select the pin PMU relays of the specified pins in pin_name.)
PDCL (the relay of IO Pin Driver, Comparator and active Load)
PPMU (the PPMU relay of IO pin)
Type: int (integer)
No entry: illegal

Delay :Specify the wait time after conditions setup.
Variables and constants are allowed.
Type: double

Return Value
On success return 0,

Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.

>> LoadRTLibrary():.suggest to Execute it in "START_UP or test_start" item

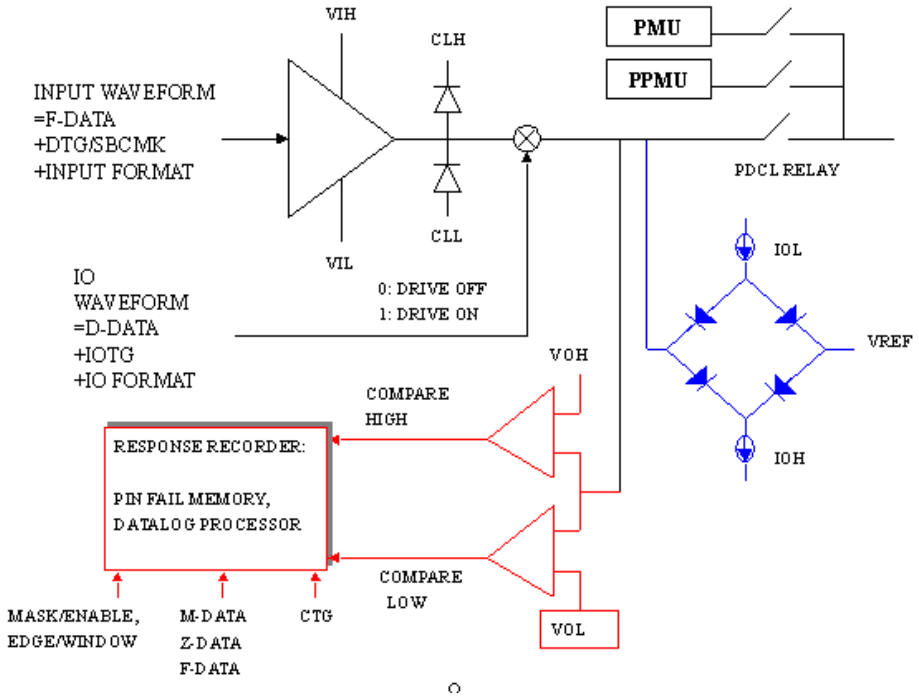
>> UnloadRTLibrary():.suggest to Execute it in " Release or test_end" item

5.3.19.3 Example

```
fclk_setting_by_plus_function ()
{
int pin_no[2]={0,1}; // channel number
int pin_cnt=2;
CRAFT_c_pin_mode_w(pin_no,pin_cnt,_FCLK_,_EDGE_,_MASK_,_IO_NRZ_,_
LOAD_OFF_);

CRAFT_c_frun_clk_w(pin_no,pin_cnt,100.0E-9,25.0E-9,75.0E-9,

      (&(Plugin.ShmPtr->CalibTab)) );

CRAFT_c_set_leveln_w (15, pin_no, pin_cnt, 0.0, 3.0, 1.0, 2.5,
                                    0.0, 0.0, 1.5, -1.0, 5.0,
(&(Plugin.ShmPtr->CalibTab)) );

CRAFT_c_relay_on_w(pin_no,pin_cnt,_PDCL_,5.0E-3); //relay setting
……..
……..
……..
……..

CRAFT_c_relay_off_w(pin_no,pin_cnt,_PDCL_,5.0E-3); //relay setting
}
```

5.3.20 CRAFT_c_relay_off_w

This function is the similar "RELAY_OFF" statement only used to connect the specified relays to the DUT



5.3.20.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.20.2 Format

```
LoadRTLibrary();// Load Library
CRAFT_c_relay_off_w(
        int *pin_no,
        int pin_cnt,
        int relay_resource,
        double delay
        );
UnloadRTLibrary();//Unload Library
```

Pin_no : "int array"

Example: use channel: "0", "10"

➔int Pin_no I[2]={ 0 , 10 };

Pin_cnt : pin count ; Type: integral

Example: use channel: "0", "10"

➔ Pin_cnt =2;

relay_resource : The instrument to be connected to the pin under test. Valid entries are:
PMU (select the pin PMU relays of the specified pins in pin_name.)
PDCL (the relay of IO Pin Driver, Comparator and active Load)
PPMU (the PPMU relay of IO pin)
ALL
Type: int (integer)
No entry: illegal

Delay : Specify the wait time after conditions setup.
Variables and constants are allowed.
Type: double

Return Value
On success return 0,

Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.

>> LoadRTLibrary():.suggest to Execute it in "START_UP or test_start" item

>> UnloadRTLibrary():.suggest to Execute it in " Release or test_end" item

5.3.20.3 Example

```
fclk_setting_by_plus_function ()
{
int pin_no[2]={0,1}; // channel number
int pin_cnt=2;
CRAFT_c_pin_mode_w(pin_no,pin_cnt,_FCLK_,_EDGE_,_MASK_,_IO_NRZ_,_LO
AD_OFF_);

CRAFT_c_frun_clk_w(pin_no,pin_cnt,100.0E-9,25.0E-9,75.0E-9,

        (&(Plugin.ShmPtr->CalibTab)) );

CRAFT_c_set_leveln_w (15, pin_no, pin_cnt, 0.0, 3.0, 1.0, 2.5,
                                    0.0, 0.0, 1.5, -1.0, 5.0,
(&(Plugin.ShmPtr->CalibTab)) );

CRAFT_c_relay_on_w(pin_no,pin_cnt,_PDCL_,5.0E-3); //relay setting
……..
……..
……..
……..

CRAFT_c_relay_off_w(pin_no,pin_cnt,_PDCL_,5.0E-3); //relay setting
}
```

5.3.21 CRAFT_c_frun_clk_w

This function is the similar "FRUN_CLK" statement only used to set Free run clock timing

5.3.21.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.21.2 Format

```
LoadRTLibrary();// Load Library
CRAFT_c_frun_clk_w(
        int *pin_no,
        int pin_cnt,
        double period_value,
        double rising_edge,
        double falling_edge,
        ATECalibTable *calibtab
        )
UnloadRTLibrary();//Unload Library
```

Pin_no : "int array"
Example: use channel: "0", "10"
➔int Pin_no I[2]={ 0 , 10 };

Pin_cnt : pin count : Type: integral
Example: use channel: "0", "10"
➔ Pin_cnt =2;

period_value :period value 5ns to 5.0mS
type: double
**no_entry: illegal

rising_edge :Setting range: 0nS~(1cycle –0.625nS);
**no_entry: illegal

falling_edge :Setting range: 0nS~(1cycle –0.625nS);
**no_entry: illegal

calibtab : (&(Plugin.ShmPtr->CalibTab)))

Return Value
On success return 0,

Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.

>> LoadRTLibrary():.suggest to Execute it in "START_UP or test_start" item

>> UnloadRTLibrary():.suggest to Execute it in " Release or test_end" item

5.3.21.3 Example

```
fclk_setting_by_plus_function ()
{
int pin_no[2]={0,1}; // channel number
int pin_cnt=2;
CRAFT_c_pin_mode_w(pin_no,pin_cnt,_FCLK_,_EDGE_,_MASK_,_IO_NRZ_,_LO
AD_OFF_);

CRAFT_c_frun_clk_w(pin_no,pin_cnt,100.0E-9,25.0E-9,75.0E-9,

        (&(Plugin.ShmPtr->CalibTab)) );

CRAFT_c_set_leveln_w (15, pin_no, pin_cnt, 0.0, 3.0, 1.0, 2.5,
                                        0.0, 0.0, 1.5, -1.0, 5.0,
(&(Plugin.ShmPtr->CalibTab)) );

CRAFT_c_relay_on_w(pin_no,pin_cnt,_PDCL_,5.0E-3); //relay setting
……..
……..
……..
……..

CRAFT_c_relay_off_w(pin_no,pin_cnt,_PDCL_,5.0E-3); //relay setting
}
```

5.3.22 CRAFT_c_set_leveln_w

This function is the similar "SET_LEVELN" statement only used to set I/O PIN level

5.3.22.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.22.2 Format

```
LoadRTLibrary();// Load Library
CRAFT_c_set_leveln_w (
        int lvl_no,
        int *pin_no,
        int pin_cnt,
        double vil,
        double vih,
        double vol,
        double voh,
        double iol,
        double ioh,
        double vref,
        double cll,
        double clh,
        ATECalibTable *calibtab
        )
UnloadRTLibrary();//Unload Library
```

lvl_no :level set number from 1 to 240
Type integer
No entry: illegal

Pin_no : "int array"
Example: use channel: "0", "10"
➔int Pin_no l[2]={ 0 , 10 };

Pin_cnt : pin count : Type: integral
Example: use channel: "0", "10"
➔ Pin_cnt =2;

Vil :-1.9V to 6.0V
type double

vih :-2.0V to 6.0V
type double

vol : -2.0V to 6.0V
type double

voh : -2.0V to 6.0V
type double

iol: Define current value is from Tester to DUT. (Sink current)
Scope of setting is from 0mA to +12mA.

Variables and constants are allowed.
type double

ioh: Define current value from DUT to Tester. (Source current)
Scope of setting is from -12mA to 0mA.
Variables and constants are allowed.
type double

vref: -1.75V to 5.75V
type double

cll: -2V to 5V
type double

clh: -1.5V to 6V
For practical reasons, as well as hardware constraints, clh > cll +300mV.
(2 pins share the same value)
type double

Return Value
On success return 0,

Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.
>> LoadRTLibrary():.suggest to Execute it in "START_UP or test_start" item

>> UnloadRTLibrary():.suggest to Execute it in " Release or test_end" item

5.3.22.3 Example

```
fclk_setting_by_plus_function ()
{
int pin_no[2]={0,1}; // channel number
int pin_cnt=2;
CRAFT_c_pin_mode_w(pin_no,pin_cnt,_FCLK_,_EDGE_,_MASK_,_IO_NRZ_,_LO
AD_OFF_);

CRAFT_c_frun_clk_w(pin_no,pin_cnt,100.0E-9,25.0E-9,75.0E-9,

        (&(Plugin.ShmPtr->CalibTab)) );

CRAFT_c_set_leveln_w (15, pin_no, pin_cnt, 0.0, 3.0, 1.0, 2.5,
                                        0.0, 0.0, 1.5, -1.0, 5.0,
(&(Plugin.ShmPtr->CalibTab)) );

CRAFT_c_relay_on_w(pin_no,pin_cnt,_PDCL_,5.0E-3); //relay setting
……..
……..
……..
……..

CRAFT_c_relay_off_w(pin_no,pin_cnt,_PDCL_,5.0E-3); //relay setting
}
```

5.3.23 CRAFT_c_meter_GPIB_Init

This function is used for GPIB init.
Support Meter: HP34401A / HP34410A / HP3458A

5.3.23.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.23.2 Format

```
LoadRTLibrary();// Load Library

int CRAFT_c_meter_GPIB_Init ( int addr );

UnloadRTLibrary();//Unload Library
```

addr: meter address

Return Value
On success return handle number
On FAIL return -1

Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.

>> LoadRTLibrary():.suggest to Execute it in "START_UP or test_start" item

>> UnloadRTLibrary():.suggest to Execute it in " Release or test_end" item

5.3.23.3 Example

```
gpib_meas()
{
        int address, address1;
        int Handle;
        int Handle1;
        double ave_value, meas_value[1024], ave_vmin, ave_vmax, sam_interval;
        int meas_cnt;
        double ave_value1, meas_value1[1024], ave_vmin1, ave_vmax1;


        LoadRTLibrary();

        address=22;
        Handle = CRAFT_c_meter_GPIB_Init(address);
        address1=23;
        Handle1 = CRAFT_c_meter_GPIB_Init(address1);
```

```
        meas_cnt = 10;
        sam_interval = 10.0e-3;

        CRAFT_c_meter_GPIB_Meas_Volt(
            Handle,          //int handle,                //input used meter handle.
            ave_value,     //double &ave_value,     //return average measuring value.
            meas_value,    //double *meas_value,    //return every meas_value, array
size equals to meas_cnt.
            ave_vmin,       //double &ave_vmin,       //minimun value among "meas_cnt"
counts.
            ave_vmax,       //double &ave_vmax,       //maximun value among
"meas_cnt" counts.
            meas_cnt,       //int meas_cnt,
            sam_interval //double sam_interval
            );
```

```
CRAFT_c_meter_GPIB_Meas_Volt(
            Handle1,          //int handle,                //input used meter handle.
            ave_value1,     //double &ave_value,     //return average measuring value.
            meas_value1,    //double *meas_value,    //return every meas_value, array
size equals to meas_cnt.
            ave_vmin1,       //double &ave_vmin,       //minimun value among "meas_cnt"
counts.
            ave_vmax1,       //double &ave_vmax,       //maximun value among
"meas_cnt" counts.
            meas_cnt,       //int meas_cnt,
            sam_interval //double sam_interval
            );

        printf("ave_value=%f, ave_vmin=%f, ave_vmax=%f \n", ave_value, ave_vmin,
ave_vmax);
        for (int i=0; i<meas_cnt; i++)
       printf("meas_cnt=%d, meas_value=%f \n", i, meas_value[i]);

        printf("ave_value1=%f, ave_vmin1=%f, ave_vmax1=%f \n", ave_value1,
ave_vmin1, ave_vmax1);
        for (int i=0; i<meas_cnt; i++)
       printf("meas_cnt=%d, meas_value1=%f \n", i, meas_value1[i]);



        CRAFT_c_meter_GPIB_Close(Handle);
        CRAFT_c_meter_GPIB_Close(Handle1);

        UnloadRTLibrary();

}
```

5.3.24 CRAFT_c_meter_GPIB_Close

This function is used to close GPIB port.
Support Meter: HP34401A / HP34410A / HP3458A

5.3.24.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.24.2 Format

```
LoadRTLibrary();// Load Library

void   CRAFT_c_meter_GPIB_Close ( int handle );

UnloadRTLibrary();//Unload Library
```

handle: handle number (from CRAFT_c_meter_GPIB_Init() return value)

Return Value
On success return 0

Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.

>> LoadRTLibrary():.suggest to Execute it in "START_UP or test_start" item

>> UnloadRTLibrary():.suggest to Execute it in " Release or test_end" item

5.3.24.3 Example

```
gpib_meas()
{
        int address, address1;
        int Handle;
        int Handle1;
        double ave_value, meas_value[1024], ave_vmin, ave_vmax, sam_interval;
        int meas_cnt;
        double ave_value1, meas_value1[1024], ave_vmin1, ave_vmax1;


        LoadRTLibrary();

        address=22;
        Handle = CRAFT_c_meter_GPIB_Init(address);
        address1=23;
        Handle1 = CRAFT_c_meter_GPIB_Init(address1);


        meas_cnt = 10;
        sam_interval = 10.0e-3;
```

```
CRAFT_c_meter_GPIB_Meas_Volt(
    Handle,          //int handle,                //input used meter handle.
    ave_value,       //double &ave_value,         //return average measuring value.
    meas_value,      //double *meas_value,        //return every meas_value, array
size equals to meas_cnt.
    ave_vmin,        //double &ave_vmin,          //minimun value among "meas_cnt"
counts.
    ave_vmax,        //double &ave_vmax,          //maximun value among
"meas_cnt" counts.
    meas_cnt,        //int meas_cnt,
    sam_interval //double sam_interval
    );
```

```
CRAFT_c_meter_GPIB_Meas_Volt(
    Handle1,         //int handle,                //input used meter handle.
    ave_value1,      //double &ave_value,         //return average measuring value.
    meas_value1,     //double *meas_value,        //return every meas_value, array
size equals to meas_cnt.
    ave_vmin1,       //double &ave_vmin,          //minimun value among "meas_cnt"
counts.
    ave_vmax1,       //double &ave_vmax,          //maximun value among
"meas_cnt" counts.
    meas_cnt,        //int meas_cnt,
    sam_interval //double sam_interval
    );


printf("ave_value=%f, ave_vmin=%f, ave_vmax=%f \n", ave_value, ave_vmin,
ave_vmax);
for (int i=0; i<meas_cnt; i++)
printf("meas_cnt=%d, meas_value=%f \n", i, meas_value[i]);

printf("ave_value1=%f, ave_vmin1=%f, ave_vmax1=%f \n", ave_value1,
ave_vmin1, ave_vmax1);
for (int i=0; i<meas_cnt; i++)
printf("meas_cnt=%d, meas_value1=%f \n", i, meas_value1[i]);



CRAFT_c_meter_GPIB_Close(Handle);
CRAFT_c_meter_GPIB_Close(Handle1);

UnloadRTLibrary();

}
```

5.3.25 CRAFT_c_meter_GPIB_Meas_volt

This function is used to get measure value from Meter.
Support Meter: HP34401A / HP34410A / HP3458A

5.3.25.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.25.2 Format

```
LoadRTLibrary();// Load Library

void   CRAFT_c_meter_GPIB_Close (
          int handle,                    //input used meter handle.
          double &ave_value,             //return average measuring value.
          double *meas_value,            //return every meas_value, array size equals
to meas_cnt.
          double &ave_vmin,              //minimun value among "meas_cnt" counts.
          double &ave_vmax,              //maximun value among "meas_cnt" counts.
          int meas_cnt,                  //average counts.
          double sam_interval            //sampling interval between each
measurement(default              5mS if none keyin
      );

UnloadRTLibrary();//Unload Library
```

handle : handle number (from CRAFT_c_meter_GPIB_Init() return value)

ave_value : return average measuring value.

meas_value : return every meas_value, array size equals to meas_cnt

ave_vmin : minimun value among "meas_cnt" counts

ave_vmax : maximun value among "meas_cnt" counts

meas_cnt : average counts

sam_interval : sampling interval between each measurement
No entry:default 5mS

Return Value
On success return 0

Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.

>> LoadRTLibrary():.suggest to Execute it in "START_UP or test_start" item
>> UnloadRTLibrary():.suggest to Execute it in " Release or test_end" item

5.3.25.3 Example

```
gpib_meas()
{
        int address, address1;
        int Handle;
        int Handle1;
        double ave_value, meas_value[1024], ave_vmin, ave_vmax, sam_interval;
        int meas_cnt;
        double ave_value1, meas_value1[1024], ave_vmin1, ave_vmax1;


        LoadRTLibrary();

        address=22;
        Handle = CRAFT_c_meter_GPIB_Init(address);
        address1=23;
        Handle1 = CRAFT_c_meter_GPIB_Init(address1);

meas_cnt = 10;
        sam_interval = 10.0e-3;

        CRAFT_c_meter_GPIB_Meas_Volt(
            Handle,          //int handle,                //input used meter handle.
            ave_value,     //double &ave_value,     //return average measuring value.
            meas_value,    //double *meas_value,    //return every meas_value, array
size equals to meas_cnt.
            ave_vmin,       //double &ave_vmin,       //minimun value among "meas_cnt"
counts.
            ave_vmax,       //double &ave_vmax,       //maximun value among
"meas_cnt" counts.
            meas_cnt,       //int meas_cnt,
            sam_interval //double sam_interval
            );

        CRAFT_c_meter_GPIB_Meas_Volt(
            Handle1,          //int handle,                //input used meter handle.
            ave_value1,     //double &ave_value,     //return average measuring value.
            meas_value1,    //double *meas_value,    //return every meas_value, array
size equals to meas_cnt.
            ave_vmin1,       //double &ave_vmin,       //minimun value among "meas_cnt"
counts.
            ave_vmax1,       //double &ave_vmax,       //maximun value among
"meas_cnt" counts.
            meas_cnt,       //int meas_cnt,
            sam_interval //double sam_interval
            );


        printf("ave_value=%f, ave_vmin=%f, ave_vmax=%f \n", ave_value, ave_vmin,
```

```
ave_vmax);
        for (int i=0; i<meas_cnt; i++)
        printf("meas_cnt=%d, meas_value=%f \n", i, meas_value[i]);
        printf("ave_value1=%f, ave_vmin1=%f, ave_vmax1=%f \n", ave_value1,
ave_vmin1, ave_vmax1);
        for (int i=0; i<meas_cnt; i++)
        printf("meas_cnt=%d, meas_value1=%f \n", i, meas_value1[i]);

        CRAFT_c_meter_GPIB_Close(Handle);
        CRAFT_c_meter_GPIB_Close(Handle1);

        UnloadRTLibrary();

}
```

5.3.26 CRAFT_c_pma_address_r

Read Back Pattern Memory address.

5.3.26.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.26.2 Format

```
LoadRTLibrary (); // Load Library

int CRAFT_c_pma_address_r(int log_address);

UnloadRTLibrary ();//Unload Library
```

log_address :0~1023

Return Value
On success, return Patten Memory Address,
If return -1 means invalid.
Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.

>> LoadRTLibrary():.suggest to Execute it in "START_UP or test_start" item

>> UnloadRTLibrary():.suggest to Execute it in " Release or test_end" item

5.3.26.3 Example

```
drv_nrz_test()
{
   int PEL_data[1024];
   int PEH_data[1024];
   int PinFail_data[1024];
   ……..
   .....……..
   SET_DATALOG_MODE(LOG_FAIL_ONLY,1024,ON);//set log count and log mode
   JUDGE_PAT(drv_nrz_ut_st, drv_nrz_ut_sp);
   for(int ii=0;ii<1024;ii++)
   {
      READ_LOG_MEM(0, p5, ii, PEL_data, PEH_data, PinFail_data) ;
      if(PinFail_data[0]==1)
      printf("p5 Pattern Fail Address :%8d \n",CRAFT_c_pma_address_r(ii));
      }
}
```

Data Log Message

```
p5 Pattern Fail Address : 4195364
p5 Pattern Fail Address : 4195366
p5 Pattern Fail Address : 4195368
p5 Pattern Fail Address : 4195370
p5 Pattern Fail Address : 4195372
p5 Pattern Fail Address : 4195374
p5 Pattern Fail Address : 4195376
p5 Pattern Fail Address : 4195378
p5 Pattern Fail Address : 4195380
p5 Pattern Fail Address : 4195382
```

5.3.27 GET_TRM_PATH

User could get the path of data location from OPER or APP mode.
Release version : **CRAFT_3380_2.00**

5.3.27.1 Format

```
char* path = GET_TRM_PATH();
```

5.3.27.2 Example

Operator mode (OPER):The path is from Prod ID setting.

```
//In Operator Mode path return Prod ID setup value
char* path = GET_TRM_PATH();
printf("TRM Path = %s",path);
```

Application mode (APP):The path is from local program file.

```
//In Application Mode path return test plan path
char* path = GET_TRM_PATH();
printf("TRM Path = %s",path);
```

5.3.28 CRAFT_c_get_slot_temperature

Read Board calibration and current temperature.

5.3.28.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.28.2 Format

```
LoadRTLibrary (); // Load Library
int CRAFT_c_get_slot_temperature(int *slot_no, int slot_cnt, double *cal_value,
double *brd_value);
UnloadRTLibrary ();//Unload Library
```

slot_no : “int array”
Example: read board: “0”, “10”
int **slot_no** l[2]={ 0 , 10 };

slot_cnt : Type: integral
Example: read board: “0”, “10”
slot_cnt=2;

cal_value : “double array”
Example: read board: “0”, “10” calibration temperature
double **cal_value[2];**
cal_value[0]=40; //board 0 temperature 40 degree
cal_value[1]=45; //board 10 temperature 40 degree

brd_value : “double array”
Example: read board: “0”, "10” current temperature
double **brd_value[2];**
brd_value[0]=44; //board 0 temperature 44 degree
brd_value[1]=48; //board 10 temperature 48 degree

Note: Before using the function, must execute “LoadRTLibrary()” function to load library first.
>> LoadRTLibrary():.suggest to Execute it in “START_UP or test_start” item
>> UnloadRTLibrary():.suggest to Execute it in “ Release or test_end“ item

5.3.28.3 Example

```
drv_nrz_test()
{
int slot_no[3]
int slot_cnt=3
double cal_value[3]
double brd_value[3]

slot_no[0]=1;
slot_no[1]=8;
slot_no[2]=13;

CRAFT_c_get_slot_temperature(slot_no, slot_cnt, cal_value, brd_value );

cal_value[0]=40; //board 1 calibration temperature
cal_value[1]=44; //board 8 calibration temperature
cal_value[2]=38; //board 13 calibration temperature

brd_value [0]=51; //board 1 current temperature
brd_value [1]=49; //board 8 current temperature
brd_value [2]=48; //board 13 current temperature

}
```

5.3.29 CRAFT_c_shuttle_relay_w

Relay of MXPDG is definition in plus c function.

5.3.29.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.29.2 Format

```
LoadRTLibrary (); // Load Library
int CRAFT_c_shuttle_relay_w (int shuttle, double delay );
UnloadRTLibrary ();//Unload Library
```

shuttle :Bitwise, 0:turn off relay
1:turn on relay
bit0(0x01)→ ATE shuttle 0
bit1(0x02)→ ATE shuttle 1
bit2(0x04)→ ATE shuttle 2
bit3(0x08)→ ATE shuttle 3
bit4(0x10)→ Capture Card shuttle 0
bit5(0x20)→ Capture Card shuttle 1
bit6(0x40)→ Capture Card shuttle 2
bit7(0x80)→ Capture Card shuttle 3

Delay : Specify the wait time after conditions setup.
Variables and constants are allowed.
Type: double

Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.

>> LoadRTLibrary():.suggest to Execute it in "START_UP or test_start" item

>> UnloadRTLibrary():.suggest to Execute it in " Release or test_end" item

5.3.29.3 Explanation

There are ATE shuttle control bit0(0x01)~ bit3(0x08), that can't be work together.

5.3.29.4 Example

```
#include <Lib_Plus\rtlib.h>
#include <mxpdg\mxpdg_firmware.h>
START_UP()
{
      LoadRTLibrary();
}
RELEASE()
{
      UnloadRTLibrary();
}
MXPDG_control()
{
    //// control MXPDG +15V ON or OFF
    //// relay_resource = _ON_ ;
    //// _3362_shuttle_power_relay_w( int relay_resource, double delay );
    _3362_shuttle_power_relay_w( _ON_, 3.0e-3 );
    ////shuttle realy 1 ON
    CRAFT_c_shuttle_relay_w(0x1,3.0e-3);

    DEBUG_BREAK();

    ////shuttle realy all OFF
    CRAFT_c_shuttle_relay_w(0x0,3.0e-3);
    //// control MXPDG +15V OFF
    //// relay_resource = _OFF_ ;
    _3362_shuttle_power_relay_w( _OFF_, 3.0e-3 );
}
```

5.3.30 CRAFT_c_get_load_board_id

The statement read back the load board ID and is the same with READ_LOAD_BOARD_ID.

5.3.30.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.30.2 Format

```
LoadRTLibrary();// Load Library
int CRAFT_c_get_load_board_id (
        int cs, //0: load board 1:remote board
        int    &rom_size,      //rom_id(ROM)-->4 char, rom_size(1024)-->8 char
        char *board_id,      //maximum 32 char
        char *serial_no,    //maximum 32 char
        char *model,          //maximum 32 char; tester model, e.g.
        char *type,          //maximum 16 char; load board type
        char *date,          //maximum 16 char;
        char *comment,       //maximum 64 char;
        int    &status        //return status: PASS/FAIL
         );
UnloadRTLibrary();//Unload Library
```

cs : chip select
0➔ load board
1➔ remote board
rom_size : return ROM SIZE
board_id : return load board ID
serial_no : return load board ID
model : return model
type : return type
date : return date
comment : return comment
status : return status (PASS/FAIL)

Return Value
On success return 3,

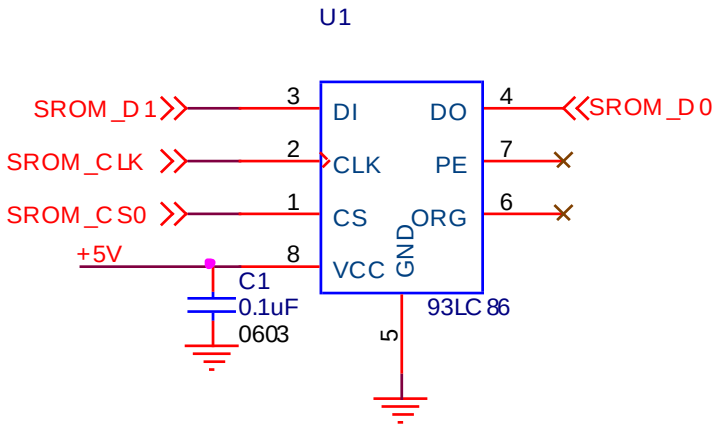
Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.

>> LoadRTLibrary():.suggest to Execute it in "**START_UP** or **test_start**" item

>> UnloadRTLibrary():.suggest to Execute it in " **Release** or **test_end**" item

5.3.30.3 Example

<Circuit>
IC: 93LC86



MXPG3 PoGo Pin Define

Pogo Pin No	define	define	define	define	define	define	define	define	Pogo Pin No
	a	b	c	d	e	f	g	h	
1	**NC**	**NC**	**AG**	**AG**	**AG**	**AG**	**NC**	**AG**	1
2	**+15V**	NC	**IMH_MEET**	**AG**	**POGO_8**	NC	**NC**	NC	2
3	NC	**NC**	NC	**POGO_0**	**AG**	**POGO_16**	NC	**-15V**	3
4	**+15V**	NC	**SYSFAIL**	**AG**	**POGO_9**	NC	**NC**	NC	4
5	NC	**NC**	NC	**POGO_1**	**AG**	**POGO_17**	NC	**-15V**	5
6	**DGND**	NC	**RATEICLK**	**AG**	**POGO_10**	NC	**NC**	NC	6
7	NC	**NC**	NC	**POGO_2**	**AG**	**POGO_18**	NC	**+3.3V**	7
8	**DGND**	NC	**RATEOCLK**	**AG**	**POGO_11**	NC	**NC**	NC	8
9	NC	**NC**	NC	**POGO_3**	**AG**	**POGO_19**	NC	**+3.3V**	9
10	**SYSCLK**	NC	**RPEN**	**AG**	**POGO_12**	NC	**NC**	NC	10
11	NC	**PATEND**	NC	**POGO_4**	**AG**	**POGO_20**	NC	**FROM_DI**	11
12	**IMH_RATE**	NC	**PRINT**	**AG**	**POGO_13**	NC	**POGO_24**	NC	12
13	NC	**N_SPM_CE**	NC	**POGO_5**	**AG**	**POGO_21**	NC	**FROM_DO**	13
14	**BUSY**	NC	**FMEM1_DQ31**	**AG**	**POGO_14**	NC	**POGO_25**	NC	14
15	NC	**LOCKED**	NC	**POGO_6**	**AG**	**POGO_22**	NC	**FROM_CLK**	15
16	**SYNC_FT0**	NC	**POGO_26**	**AG**	**POGO_15**	NC	**FROM_CS1**	NC	16
17	NC	**SQPG**	NC	**POGO_7**	**AG**	**POGO_23**	NC	**FROM_CS0**	17

```
#include<.\Lib_Plus\rtlib.h>
   bid_read( )
   {
            int cs; //0: load board 1:remote board
            int    rom_size;
            char board_id[32];
```

```
char serial_no[32];
char model[32];
char type[16];
char date[16];
char comment[64];
int    status;

 LoadRTLibrary();// Load Library
    CRAFT_c_get_load_board_id (
1, //0: load board 1: remote board
rom_size,
board_id,      //maximum 32 char
serial_no,    //maximum 32 char
model,          //maximum 32 char; tester model, e.g.
type,           //maximum 16 char; load board type
date,           //maximum 16 char;
comment,       //maximum 64 char;
status          //return status: PASS/FAIL
 );

 Printf("BID:%s\n", board_id);
UnloadRTLibrary();//Unload Library

}
```

5.3.31 CRAFT_c_set_frun_clk_mode

User can use the **C-function** to control free run clock mode.

5.3.31.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

[Notice]:
1. Check "rtlib.h" in C:\CRAFT\include\Lib_Plus
2. Check "Lib_Plus.lib" in C:\CRAFT\lib
3. Check "Lib_Plus.dll" in C:\CRAFT\bin

If file lost, please contact CHROMA service.

5.3.31.2 Format

```
LoadRTLibrary();// Load Library
CRAFT_c_set_frun_clk_mode(Pin_Channel , PIN_Count , turn_on_mode ,
frz_mode , ON/OFF );
UnloadRTLibrary();//Unload Library
```

Pin_Channel: **"int array"**

Example: use free run clk channel: "0", "10"

➔int pin_channel[2]={ 0 , 10 };

PIN_Count: pin count ; Type: integral

Example: use free run clk channel: "0", "10"

➔ PIN_Count=2;

turn_on_mode:

0: free run clk output controlled **by pattern burst (BUSY) (system default)**

1: free run clk output controlled **by statement setting and always running**.

It will be stopped by **C-function** setting:

CRAFT_c_set_frun_clk_mode set "OFF"

2: free run clk output controlled **by pattern burst (BUSY) and continue run**.

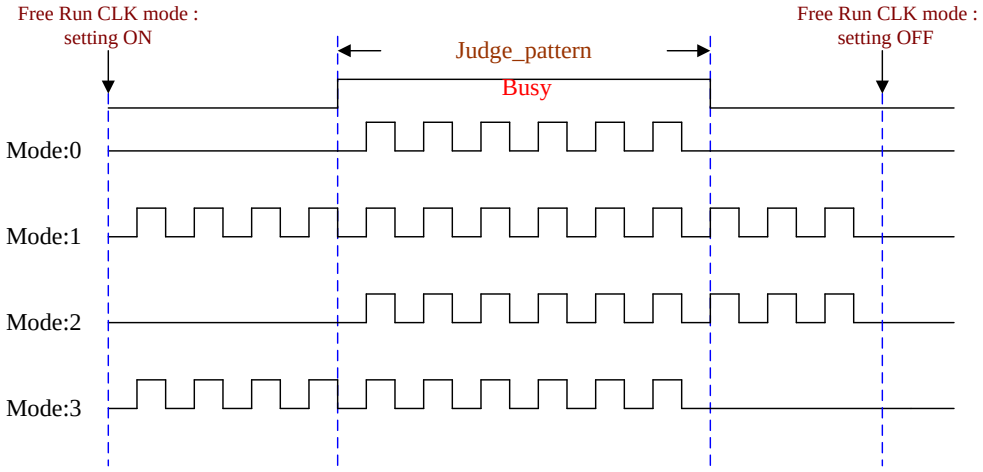
It will be stopped by **C-function** setting:

CRAFT_c_set_frun_clk_mode set "OFF"

3: free run clk output controlled **by C-function setting and turn off while busy done**

type: int

**no_entry: illegal



frz_mode:

0: fclk still running while freeze command find in cycle

1: fclk will be hold if freeze command find in cycle **(system default)**

type: int

**no_entry: illegal

ON/OFF:

0: turn off ----> restore the system default value

1 : trun-on "turn_on_mode" & "frz_mode" setting

**no_entry: illegal

Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.

>> LoadRTLibrary(): Load c_set_frun_clk_mode library .suggest to Execute it in "**START_UP** or **test_start**" item

>> UnloadRTLibrary():Unload c_set_frun_clk_mode library .suggest to Execute it in " **Release** or **test_end**" item

5.3.31.3 Example

```
1. Mode 0
//=====================================//
int *CLK_PIN;
int pin_count=0;
PIN_MODE(CLK, FCLK, EDGE, MASK, IO_ON, OFF); // CLK:pin name
FRUN_CLK(CLK, 5uS, 0S, 2.5uS);
pin_count=GetPinChannel("CLK" , CLK_PIN);
DATALOG_MSG="FREE_RUN_CLK_CHECK_MODE_0";
CRAFT_c_set_frun_clk_mode(CLK_PIN , pin_count , 0 , 0 , 1 );//set mode 0 and
turn on
DEBUG_BREAK();
JUDGE_PAT(func_test,__ func_test);//free run clk output control by
pattern(start and stop)
CRAFT_c_set_frun_clk_mode(CLK_PIN , pin_count , 0 , 0 , 0 );// turn OFF

2. Mode 1
//=====================================//
int *CLK_PIN;
int pin_count=0;
PIN_MODE(CLK, FCLK, EDGE, MASK, IO_ON, OFF); // CLK:pin name
FRUN_CLK(CLK, 5uS, 0S, 2.5uS);
pin_count=GetPinChannel("CLK" , CLK_PIN);
DATALOG_MSG="FREE_RUN_CLK_CHECK_MODE_1";
CRAFT_c_set_frun_clk_mode(CLK_PIN , pin_count , 1 , 0 , 1 );//set mode 1 and
                                                            turn on
                                                          //"CLK output start"
DEBUG_BREAK();
JUDGE_PAT(func_test,__ func_test);
CRAFT_c_set_frun_clk_mode(CLK_PIN , pin_count , 1 , 0 , 0 );// turn OFF and
                                                            "CLK stop"

3. Mode 2
//=====================================//
int *CLK_PIN;
int pin_count=0;
PIN_MODE(CLK, FCLK, EDGE, MASK, IO_ON, OFF); // CLK:pin name
FRUN_CLK(CLK, 5uS, 0S, 2.5uS);
pin_count=GetPinChannel("CLK" , CLK_PIN);
DATALOG_MSG="FREE_RUN_CLK_CHECK_MODE_2";
CRAFT_c_set_frun_clk_mode(CLK_PIN , pin_count , 2 , 0 , 1 );//set mode 2 and
                                                            turn on
DEBUG_BREAK();
JUDGE_PAT(func_test,__ func_test);//free run CLK output start when pattern
start
CRAFT_c_set_frun_clk_mode(CLK_PIN , pin_count , 2 , 0 , 0 );// turn OFF and
                                                            "CLK stop"

4. Mode 3
//=====================================//
int *CLK_PIN;
int pin_count=0;
PIN_MODE(CLK, FCLK, EDGE, MASK, IO_ON, OFF); // CLK:pin name
FRUN_CLK(CLK, 5uS, 0S, 2.5uS);
pin_count=GetPinChannel("CLK" , CLK_PIN);
DATALOG_MSG="FREE_RUN_CLK_CHECK_MODE_3";
CRAFT_c_set_frun_clk_mode(CLK_PIN , pin_count , 3 , 0 , 1 );//set mode 3 and
                                                            turn on
                                                          // "CLK output
```

```
                                                                    start"
    DEBUG_BREAK();
    JUDGE_PAT(func_test,__ func_test);//free run CLK output STOP when pattern
STOP
CRAFT_c_set_frun_clk_mode(CLK_PIN , pin_count , 2 , 0 , 0 );// turn OFF
```

5.3.32 CRAFT_c_get_system_id

User could use the C-function get system id information.

5.3.32.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.32.2 Format

```
LoadRTLibrary (); // Load Library

int CRAFT_c_get_system_id(
        char* model,
        char* type,
        char* sub_type,
        char* ab_revision,
        char* serial
       );

UnloadRTLibrary();//Unload Library
```

model : type char array, size 8 ;ATE model definition.
type : type char array, size 8 ;ATE back-plane type definition.
sub_type : type char array, size 8 ;ATE back-plane sub_type definition.
ab_revision : type char array, size 8 ;ATE back-plane revision definition.
serial : type char array, size 8 ;ATE serial number.

Return Value
On success, return _3500_SYS_PASS,

Note: Before using the function, must execute "LoadRTLibrary()" function to load library first.

>> LoadRTLibrary():.suggest to Execute it in "**START_UP** or **test_start**" item

>> UnloadRTLibrary():.suggest to Execute it in " **Release** or **test_end**" item

5.3.32.3 Example

```
START_UP(){
    LoadRTLibrary();
}
drv_nrz_test()
{
    char ate_model[8];
    char ate_type[8];
    char ate_sub_type[8];
    char ate_ab_revision[8];
    char ate_serial[8];
  ……..
  .....……..
    CRAFT_c_get_system_id(
                ate_model,
                ate_type,
                ate_sub_type,
                ate_ab_revision,
                ate_serial
                );

 printf("Model %s , type %s , sub_type %s , ab %s , serial
%s\n",ate_model,ate_type,ate_sub_type,ate_ab_revision,ate_serial);
}

Release(){
UnloadRTLibrary();
}
```

Data Log Message

```
Model 3380 , type F , sub_type XX , ab 010A , serial 000159
```

5.3.33 CRAFT_c_get_capture_sector_mode

User can use the C-function to get capture memory sector mode function parameter.

5.3.33.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.33.2 Format

```
LoadRTLibrary();// Load Library
CRAFT_c_get_capture_sector_mode(int *pin_no , int pin_cnt , int *sector_no , int
*enable);
UnloadRTLibrary();//Unload Library
```

pin_no : "int array"

Example: use free run clk channel: "0", "10"

➔ int pin channel [2]={ 0 , 10 };

pin_cnt : pin count; Type: integral

Example: use free run clk channel: "0", "10"

➔ pin Count=2;

sector_no :return data, type: "int array"

array size by pin_cnt

enable :return data, type: "int array"

array size by pin_cnt

1:ON

0:OFF

5.3.33.3 Example

```
capture()
{
        int ii=0;
        int data[1]={0};
        int pin_no[2]={32,16};
        int st_no[2]={1,0};
        int sector_no[2]={99,99};
        RATEN(TS1,10uS);
       PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
       MARKERN(TS1, IO_ALLPINS, 0nS, 10nS, , 0nS, 10nS, 5uS, 5uS);
        RELAY_ON(IO_ALLPINS,PDCL,2mS);
        LEVELS(func_lvl, 8mS);

       SET_CAPTURE_MEM_MODE(p32+p16, PEL, ON, 64K);

        DEBUG_BREAK();

        CRAFT_c_set_capture_sector_mode(
                        pin_no,
                        2,
                        st_no,//by pin count (data=0-127)
                        1      //_ON_
                      );

        DEBUG_BREAK();
        JUDGE_PAT(sector_st3,sector_sp3);
        DEBUG_BREAK();
        //read pin channel 16 & 32 capture sector no
        CRAFT_c_get_capture_sector_mode(pin_no , 2 , sector_no , enable);

        printf("channel[%3d], sector mode[%3d],sector no:%d    \n",
                        pin_no[0] , enable [0],sector_no[0]);

        printf("channel[%3d], sector mode[%3d],sector no:%d    \n",
                        pin_no[1] , enable [1],sector_no[1]);

        DEBUG_BREAK();
        LEVELS(rst_lvl, 8mS);
        RELAY_OFF(IO_ALLPINS,PDCL,2mS);
}
```

5.3.34 CRAFT_c_set_capture_sector_mode

The C-function to enhance capture memory function more than 64K.

5.3.34.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.34.2 Format

```
LoadRTLibrary();// Load Library
CRAFT_c_set_capture_sector_mode(int *pin_no , int pin_cnt , int *sector_no , int
enable);
UnloadRTLibrary();//Unload Library
```

pin_no : "int array"

Example: use free run clk channel: "32", "16"

➔ int pin channel [2]={ 32 , 16 };

Channel 32 is 32%8="0"

Channel 16 is 16%8="0"

The remainder must be same that could connect to one pin.

pin_cnt : pin count; Type: integral

Example: use free run clk channel: "32", "16"

➔ pin Count=2;

sector_no :From 0~ 127, array size by pin_cnt

➔ int sector_no[2]={1,0};

[Notice]: sector_no[2]={0,1} means channel 32 is first, channel 16 is second.

enable :type: int

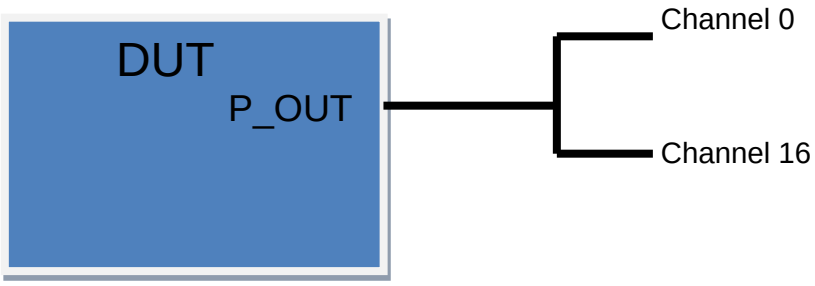
_ON_ :ON

OFF :OFF

**no_entry: illegal

5.3.34.3 Example

<CIRCUIT>



<DEC>

```
P_OUT  =   16  =   1   =   IO;
P_OUT2 =   32  =   2   =   IO;
```

<PLN>

```
capture()
{
        int ii=0;
        int data[1]={0};
        int pin_no[2]={32,16};
        int st_no[2]={1,0};// channel 16 is first , channel 32 is second pin
        RATEN(TS1,10uS);
        PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
        MARKERN(TS1, IO_ALLPINS, 0nS, 10nS, , 0nS, 10nS, 5uS, 5uS);
        RELAY_ON(IO_ALLPINS,PDCL,2mS);
        LEVELS(func_lvl, 8mS);

        SET_CAPTURE_MEM_MODE(p32+p16, PEL, ON, 64K);

        DEBUG_BREAK();

        CRAFT_c_set_capture_sector_mode(
                        pin_no,
                        2,
                        st_no,//by pin count (data=0-127)
                        _ON_      //_ON_
                      );

        DEBUG_BREAK();
        JUDGE_PAT(sector_st3,sector_sp3);
        DEBUG_BREAK();
        //read pin channel 16 capture data
        printf("P16 \n");
        for(ii=0;ii<65536;ii++){
                READ_CAPTURE_MEM(S0, p16, ii, data);
                printf("address[%8d]=%2d \n",ii,data[0]);
        }

        //read pin channel 32 capture data
        printf("P32 \n");
        for(ii=0;ii<65536;ii++){
                READ_CAPTURE_MEM(S0, p32, ii, data);
```

```
            printf("address[%8d]=%2d \n",ii+65536,data[0]);
        }


        DEBUG_BREAK();
        LEVELS(rst_lvl, 8mS);
        RELAY_OFF(IO_ALLPINS,PDCL,2mS);
}
```

<PAT>

```
SET_DEC_FILE "capture_mem_sector_check.dec"

HEADER        p0,p32,p16,p24;
SPM_PATTERN (sector3) {
    sector_st3::            *0XXX* TS1;
                            *1XXX* ;
                            *1XXX* SETX0 13107;
                            *1XXX* ;
                            *1ZZZ* ;    //1
                            *0ZZZ* ;
                            *1ZZZ* ;
                            *1ZZZ* ;
                            *1ZZZ* ;
                            *0ZZZ* JNZ0 .-4;
                            *0XXX* ;
                            *1XXX* ;
                            *1XXX* SETX0 13107;
                            *1XXX* ;
                            *0ZZZ* ;     //2
                            *1ZZZ* ;
                            *0ZZZ* ;
                            *0ZZZ* ;
                            *0ZZZ* ;
                            *1ZZZ* JNZ0 .-4;
                            *0XXX* ;
                            *1XXX* ;
                            *1XXX* ;
                            *1XXX* ;
                            *0XXX* ;
                            *0XXX* ;
                            *1XXX* ;
                            *1XXX* ;
                            *1XXX* ;
   sector_sp3::             *0XXX* ;
}
```

5.3.35 CRAFT_c_set_awi_wf_mem_point_mode

Set MXAWI Waveform memory point mode
This function takes WG/WD channel number and set LXAWI Waveform memory point mode.

5.3.35.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.35.2 Format

```
LoadRTLibrary();// Load Library
int CRAFT_c_set_awi_wf_mem_point_mode(int *pin_no, int pin_cnt, int
turn_on_mode)
UnloadRTLibrary();//Unload Library
```

Pin_Channel: **"int array"**

Example: use AWI channel: "196", "198"

➔int pin_channel[2]={ 196 , 198 };

PIN_Count: pin count ; Type: integral

Example: use AWI channel: "196", "198"

➔ PIN_Count=2;

turn_on_mode:

0: Re-trigger waveform memory point won't set to 0
1: Re-trigger waveform memory point always set to 0

Return Value

On success return 0,
Setting out of definition (ex: illegal channel/mode) return 1.

5.3.35.3 Example

```
#include "rtlib.h "

func_AC_test()
{
   int wg_ch[1] = {196};
   int ret;
   LoadRTLibrary();
   ret = CRAFT_c_set_awi_wf_mem_point_mode (wg_ch, 1, 1);
   printf("Status=%d\n", ret);
   LOAD_AWG_PAT(wg1,"./sin_15v.awi", 16, RPT, 1);
   JUDGE_PAT(clr_st, clr_sp);
   UnloadRTLibrary();
}
```

5.3.36 CRAFT_c_get_awi_wf_mem_point_mode

Read back MXAWI Waveform memory point mode
This function takes WG/WD channel number and return LXAWI Waveform memory point mode (return by ate channel).

5.3.36.1 Header File

```
include<.\Lib_Plus\rtlib.h>
```

5.3.36.2 Format

```
LoadRTLibrary();// Load Library
int CRAFT_c_get_awi_wf_mem_point_mode(int *pin_no, int pin_cnt, int
*turn_on_mode)
UnloadRTLibrary();//Unload Library
```

Pin_Channel: **"int array"**

Example: use AWI channel: "196", "198"

➔int pin_channel[2]={ 196 , 198 };

PIN_Count: pin count ; Type: integral

Example: use AWI channel: "196", "198"

➔ PIN_Count=2;

turn_on_mode: (return by ate channel)

0: Re-trigger waveform memory point won't set to 0
1: Re-trigger waveform memory point always set to 0

Return Value

On success return 0,
Setting out of definition (ex: illegal channel/mode) return 1.

5.3.36.3 Example

```
#include "rtlib.h "

func_AC_test()
{
   int wg_ch[1] = {196};
   int ret, wave_pnt[608];

   LoadRTLibrary();
   CRAFT_c_set_awi_wf_mem_point_mode (wg_ch, 2, 1);
   CRAFT_c_get_awi_wf_mem_point_mode (wg_ch, 2, 1);
   printf("ch[%d] trun_on_mode=%d \n", wg_ch[0], wave_pnt [wg_ch[0]]);// return by
ate channel
   DEBUG_BREAK();
   LOAD_AWG_PAT(wg1,"./sin_15v.awi", 16, RPT, 1);
   JUDGE_PAT(clr_st, clr_sp);
   UnloadRTLibrary();
}
```

5.4 Gang-mode Function

5.4.1 POWER_GANG_PIN_GROUP

Provide the ability to refer to a set of gang-mode power pin by a single name.

5.4.1.1 Format

Parameter list:

```
POWER_GANG_PIN_GROUP {

        ps_pin_group_name = ps_pin_name operand ps_pin_name operand …;

        ps_pin_group_name = ps_pin_name operand ps_pin_name operand…;

}
```

Parameter comment:

ps_pin_group_name : a name given to a pin group. Can define 1024 max.
max. 64 chars
type: string
**no entry: illegal

ps_pin_name : as defined in PIN_LIST statement which the pin type of the pin_name must be DPS or PREF or UVI.

Operand : + > include pin/group in pin group being defined
– > exclude pin/group from pin group being defined

5.4.1.2 Example

```
POWER_GANG_PIN_GROUP   {
        VDD_Gang = Vdd+Vdd_G1+Vdd_G2;
    }
```

5.5 PE HV Function

5.5.1 PIN_MODE_HV

The statement setup the HV pins driver format .The statement only support MXLPC board high-voltage pins.

5.5.1.1 Format

Parameter list:

```
PIN_MODE_HV(pin_name, d_format , io_format );
```

Parameter comment:

```
pin_name:      pin_def, pin_group (only suit for MXLPC HV-Pins)
               **no_entry: illegal

d_format:      NRZ,          //io_format must be IO_NRZ
    F0, F1, and F2           //io_format must be IO_ON

               (F0: Force Low, F1: Force Vih, F2: Force Vihh)
               type: integer
               **no_entry: keep current format

io_format :    IO_NRZ, IO_ON
               **no_entry: illegal
```

5.5.1.2 Explanation

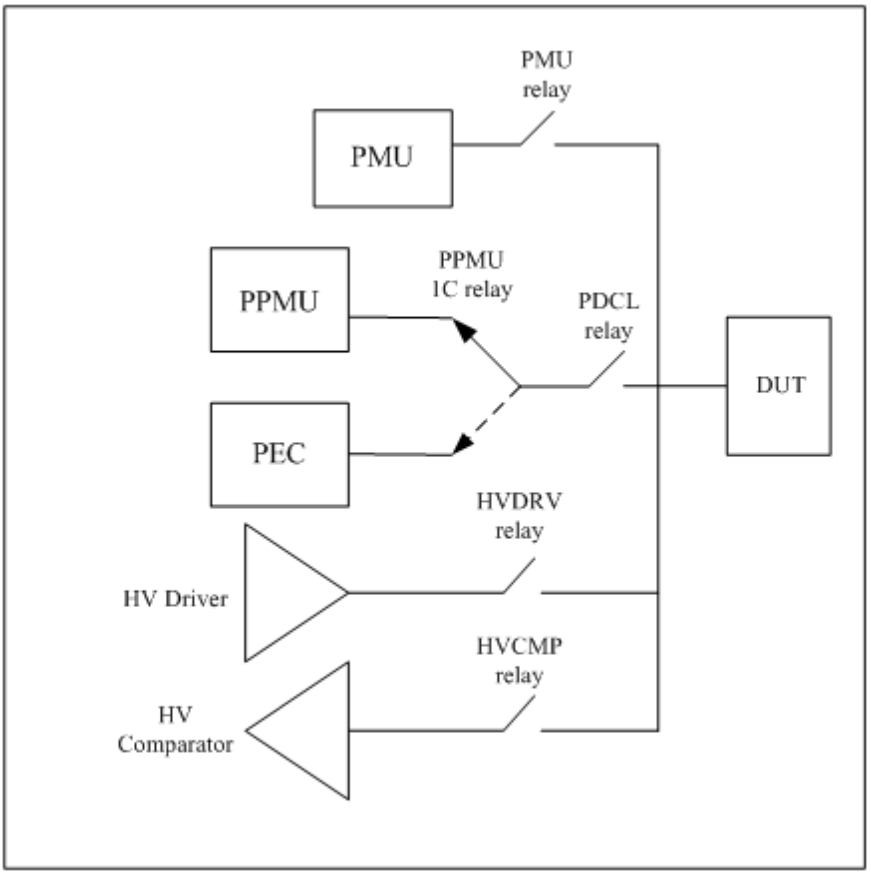
High-voltage pins: 4 channel/broad (ATE channel could use 0, 16, 32, 48)
HV pin does not have active load, so the load mode is useless.

5.5.1.3 Example

```
HV_check()
{
   SET_LEVELN (tmp_lvl, IO_ALLPINS, 0V, 2V, 0.5V, 1.5V, 0uA, 0uA, 1V, , );
   LEVELS (tmp_lvl, 5mS);
   RATEN(TS3,800nS);
   MARKERN(TS3, IO_ALLPINS, 400nS, 0nS, 0nS, 0nS, 0nS, 0nS, 0nS);
   PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   RELAY_ON(TRIG+G1, PDCL, 1mS);
   JUDGE_PAT(initial,__initial);

   RELAY_OFF(TRIG+G1, ALL, 1mS);

   SET_LEVELN_HV (HV_lvl, G1, 0.0V, 2.0V, 10V );
   SET_LEVELN (HV_lvl, TRIG, 0V, 2V, 0.5V, 1.5V, 0uA, 0uA, 1V, , );
   LEVELS(HV_lvl,2mS);
```

```
PIN_MODE_HV (G1, NRZ, EDGE, ENABLE, IO_NRZ );

RATEN(TS3,800nS);
MARKERN(TS3, G1, 400nS, 0nS, 0nS, 0nS, 0nS, 0nS, 0nS);
MARKERN_HV(TS3, G1, 400nS);

RELAY_ON(TRIG, PDCL, 1mS);
RELAY_ON(G1, HVDRV_CMP, 1mS);
JUDGE_PAT(HV_test,__HV_test);

LEVELS (tmp_lvl, 5mS);
PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
RELAY_OFF(IO_ALLPINS, ALL, 1mS);
}
```

```
PIN_MODE_HV (oe, NRZ, IO_NRZ );

PIN_MODE_HV (oe, F2, IO_ON );     /* force Vihh static high */
```



MXLPC PEC

5.5.1.4 Relative

RELAY_ON

5.5.2 SET_LEVELN_HV

Set the HV pin levels for the specified pins and level set number.The statement only support M**XLPC** board high-voltage pins.

5.5.2.1 Format

Parameter list:

SET_LEVELN_HV (lvl_no, pin_name, vil, vih, vihh);

Parameter comment:

pin_name; pin_def, pin_group (4 channel/broad,only suit for HV pins, channel % 16==0 ;e,g:channel : 0,16,32,48)
**no_entry: illegal

lvl_no: a level set number 1 to 255
variabales & constants allowed
type: integer
**no_entry: illegal

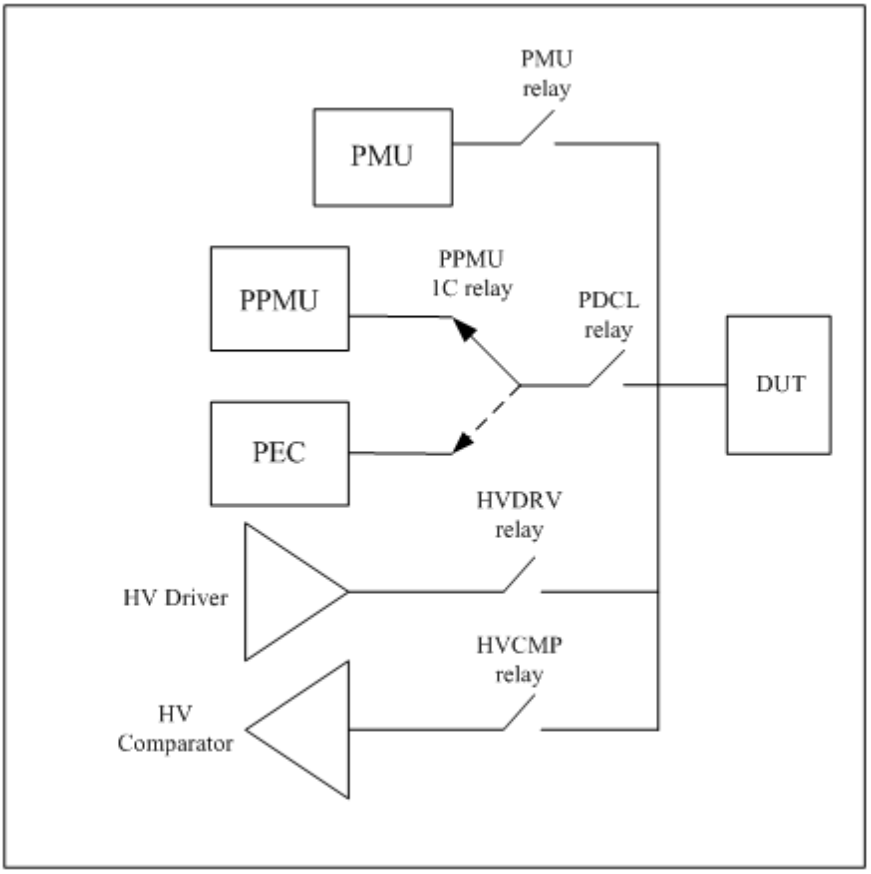
vil: -0.1V to +6.0V
variables & constants allowed
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: keep current value

vih: -0.1V to +6.0V
variables & constants allowed
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: keep current value

vihh: 5.9V to +13.5V
variables & constants allowed
type: VOLTAGE or double
unit: V, pV, nV, uV, mV
**no_entry: keep current value

5.5.2.2 Explanation

High-voltage pins: 4 channel/broad (ATE channel could use 0, 16, 32, 48)
HV pin does not have active load, so the load mode is useless.


PMU
relay
PMU
PPMU
PPMU
1C relay
PDCL
relay
DUT
PEC
HVDRV
relay
HV Driver
HVCMP
relay
HV
Comparator

MXLPC PEC

5.5.2.3 Example

```
HV_check()
{
   SET_LEVELN (tmp_lvl, IO_ALLPINS, 0V, 2V, 0.5V, 1.5V, 0uA, 0uA, 1V, , );
   LEVELS (tmp_lvl, 5mS);
   RATEN(TS3,800nS);
   MARKERN(TS3, IO_ALLPINS, 400nS, 0nS, 0nS, 0nS, 0nS, 0nS, 0nS);
   PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   RELAY_ON(TRIG+G1, PDCL, 1mS);
   JUDGE_PAT(initial,__initial);

   RELAY_OFF(TRIG+G1, ALL, 1mS);

   SET_LEVELN_HV (HV_lvl, G1, 0.0V, 2.0V, 10V );
   SET_LEVELN (HV_lvl, TRIG, 0V, 2V, 0.5V, 1.5V, 0uA, 0uA, 1V, , );
   LEVELS(HV_lvl,2mS);

   PIN_MODE_HV (G1, NRZ, EDGE, ENABLE, IO_NRZ );

   RATEN(TS3,800nS);
   MARKERN(TS3, G1, 400nS, 0nS, 0nS, 0nS, 0nS, 0nS, 0nS);
   MARKERN_HV(TS3, G1, 400nS);

   RELAY_ON(TRIG, PDCL, 1mS);
   RELAY_ON(G1, HVDRV_CMP, 1mS);
   JUDGE_PAT(HV_test,__HV_test);

   LEVELS (tmp_lvl, 5mS);
   PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   RELAY_OFF(IO_ALLPINS, ALL, 1mS);
   }
```

```
SET_LEVELN_HV (func_lvl, vee, 0.8V, 2.0V, 10V );
```

5.5.2.4 Relative

RELAY_ON

5.5.3 MARKERN_HV

Load HV pins' marker value to memory for specified pins and time set number. The statement only supports MXLPC board high-voltage pins.

5.5.3.1 Format

Parameter list:

MARKERN_HV (ts_no, pin_name, DT0) ;

Parameter comment:

pin_name; pin_def, pin_group (4 channel/broad,only suit for HV pins, channel % 16==0 ;e,g:channel : 0,16,32,48)
**no_entry: illegal

ts_no: 1 to 15, or name as defined in TIME_NAME_DEF
variables & constants allowed
type: integer
**no_entry: illegal

DT0 : drive marker ;
Setting range: 0nS~(2cycle - 20nS);
Setting resolution: 1.25nS
**no_entry: Keep current value

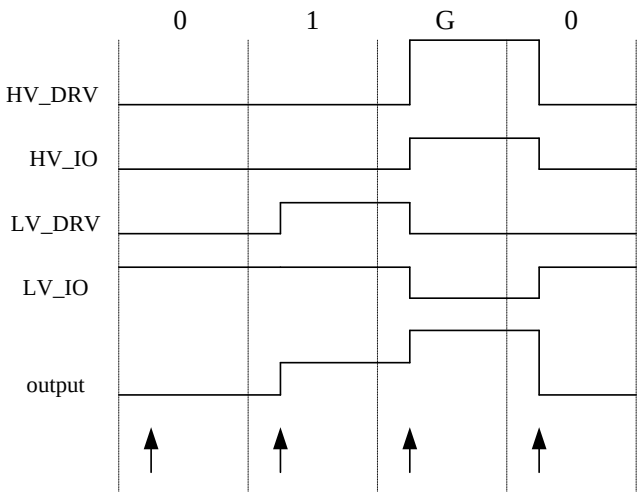
5.5.3.2 Explanation

high-voltage pins : 4 channel/broad (ATE channel could use 0,16,32,48)
HV pin does not have active load, so the load mode is useless.

5.5.3.3 Example

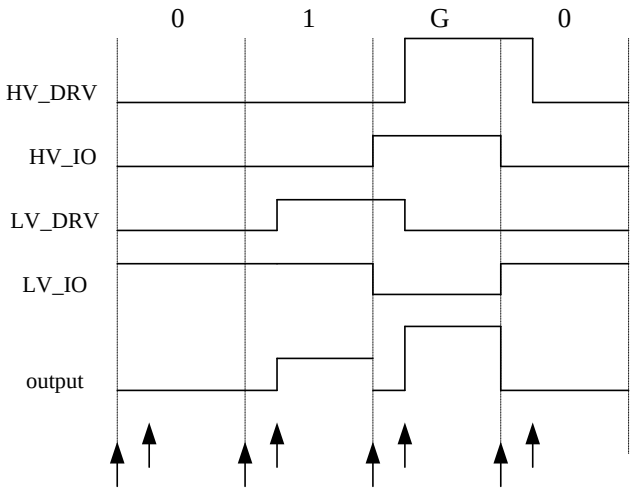
```
HV_check()
{
   SET_LEVELN (tmp_lvl, IO_ALLPINS, 0V, 2V, 0.5V, 1.5V, 0uA, 0uA, 1V, , );
   LEVELS (tmp_lvl, 5mS);
   RATEN(TS3,800nS);
   MARKERN(TS3, IO_ALLPINS, 400nS, 0nS, 0nS, 0nS, 0nS, 0nS, 0nS);
   PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   RELAY_ON(TRIG+G1, PDCL, 1mS);
   JUDGE_PAT(initial,__initial);

   RELAY_OFF(TRIG+G1, ALL, 1mS);

   SET_LEVELN_HV (HV_lvl, G1, 0.0V, 2.0V, 10V );
   SET_LEVELN (HV_lvl, TRIG, 0V, 2V, 0.5V, 1.5V, 0uA, 0uA, 1V, , );
   LEVELS(HV_lvl,2mS);

   PIN_MODE_HV (G1, NRZ, EDGE, ENABLE, IO_NRZ );

   RATEN(TS3,800nS);
   MARKERN(TS3, G1, 400nS, 0nS, 0nS, 0nS, 0nS, 0nS, 0nS);
   MARKERN_HV(TS3, G1, 400nS);

   RELAY_ON(TRIG, PDCL, 1mS);
   RELAY_ON(G1, HVDRV_CMP, 1mS);
   JUDGE_PAT(HV_test,__HV_test);

   LEVELS (tmp_lvl, 5mS);
   PIN_MODE(IO_ALLPINS, NRZ, EDGE, ENABLE, IO_NRZ, OFF);
   RELAY_OFF(IO_ALLPINS, ALL, 1mS);
   }
```

```
      SET_LEVELN_HV (func_lvl, hv_pins, 0.8V, 2.0V, 10V );

      LEVELS(func_lvl,2mS);

      RELAY_ON(hv_pins, HV, 1mS);

      PIN_MODE_HV (hv_pins, NRZ, ,MASK, IO_NRZ );

     MARKERN(TS1, IO_ALLPINS, 250nS, 0nS, 0nS, 250nS, 0nS, 0nS, 0nS);

      MARKERN_HV(TS1, hv_pins, 250nS);
```



HV/LV drv_format: NRZ, 250nS
HV/LV io_format: IO_NRZ, 250nS

```
SET_LEVELN_HV (func_lvl, hv_pins, 0.8V, 2.0V, 10V );

LEVELS(func_lvl,2mS);

RELAY_ON(hv_pins, HVDRV, 1mS);

PIN_MODE_HV (hv_pins, NRZ, ,MASK, IO_NRZ );

MARKERN(TS1, IO_ALLPINS, 250nS, 0nS, 0nS, 0nS, 0nS, 0nS, 0nS);

MARKERN_HV(TS1, hv_pins, 250nS);
```



HV drv_format: NRZ, 250nS
LV io_format: IO_NRZ, 0nS

5.5.3.4 **Relative**

RELAY_ON

5.6 RF Subsystem (Option)

The following statement provides user the easy way to control RF subsystem to measure RF parameters without consider the RF subsystem's control sequence.
Release version: **CRAFT_3380_2.40**

5.6.1 RF Pins Declaration

RF pins are declared in .dec file.

5.6.1.1 Example

```
PIN_LIST(RF)
{
   //pin name =   channel =   dut_pin_no =   pin_type
    VDD          =     23     =     1              =IO;
    CBG          =     24     =     2              =IO;
    CDVDD        =     25     =     3              =IO;
    CE           =     26     =     4              =IO;
    CSN          =     27     =     5              =IO;
    SCK          =     28     =     6              =IO;
    MOSI     =     29     =     7            =IO;
    MISO  =     30     =     8            =IO;
    IRQ          =     31     =     9              =IO;
    RF_1         =     0      =     10             = RF;
    RF_2         =     1      =     11             = RF;
}
```

Main Function

```
START_UP()
{
      RF_Initialize(KEYSIGHT,_IP,"192.168.1.2",_IniFilePath,"CableLoss.ini");
      RF_Load_Modulation_File(KEYSIGHT,_Pin,RF1,_ModulationFile,"D:\\NVARB\\FSK_C
      hroma2.wfm");
      LOAD_PAT("./PAT/PHR182.ppo");
}
POWEROFF()
{
      LEVELS( rst_lvl, 1mS) ;
      RELAY_OFF( ALL_PINS, ALL, 0mS);
      FORCE_V_MLDPS(VDDUSB+VPP, 0.0V, @6V, @500mA, 300mA, NORM, ON, 5mS);
      FORCE_V_MLDPS(VDDUSB+VPP, 0.0V, @6V, @500mA, 300mA, NORM, OFF, 3mS);
      FORCE_V_MLDPS(VDD_DIG+VDD18PA, 0.0V, @6V, @500mA, 300mA, NORM, ON,
5mS);
      FORCE_V_MLDPS(VDD_DIG+VDD18PA, 0.0V, @6V, @500mA, 300mA, NORM, OFF,
3mS);
}

Tx_Power()
{
      DATALOG_MSG="Tx_Power";
      TEST_NO(800);
      FREQUENCY CenterFreq = 2439.8MHz;
      double measureLevel = 5;
      CUSTOM_2DIM pow_V;
```

```
    TIME acquire_time = 1mS;

    RELAY_ON(SPI_PINS,PDCL,3mS);

    LEVELS( Level1, 1mS) ;

PIN_MODE(RESET+P00+P01+P02+P03+P04+P05+PROG,NRZ,EDGE,MASK,IO_NRZ,OF
F);
    PIN_MODE(P02,RO,EDGE,MASK,IO_NRZ,OFF);
    PIN_MODE(P03,NRZ,EDGE,ENABLE,IO_NRZ,OFF);
    RATEN(TS1, 1000nS);
    MARKERN(TS1,ALL_PINS    , 0.0nS, -1,-1 ,0nS ,-1 ,800nS ,-1 );
    MARKERN(TS1,P02          , 250nS,750nS,-1 ,0nS ,-1 ,800nS ,-1 );
    MARKERN(TS1,P03          , 0nS,-1,-1 ,0nS ,-1 ,800nS ,-1 );

    JUDGE_PAT(two_wire_SPI_trapping,__two_wire_SPI_trapping);

    FORCE_V_MLDPS(VDDUSB, 5.0V, @6V, @500mA, 300mA, NORM, ON, 3mS);
    FORCE_V_MLDPS(VPP, 0.0V, @6V, @500mA, 300mA, NORM, ON, 3mS);
    FORCE_V_MLDPS(VDD_DIG+VDD18PA, 0.0V, @6V, @500mA, 300mA, NORM, ON,
3mS);

    SET_JUDGE_MODE( NORM , FEOP_ON    );
    JUDGE_PAT(Rx_BER_all_register,__Rx_BER_all_register);
    SET_JUDGE_MODE( NORM , FEOP_OFF    );
    JUDGE_PAT(Tx_Center_Frequency_close_loop,__Tx_Center_Frequency_close_loop);
    JUDGE_PAT(Tx,__Tx);
    SOCKET_INC(FRZ_OFF)
    {
      PLAN_RESULT_MS[SOCKET]=PASS;
    }

    ///////////////Chroma statement : Measure Power//////////////////////////////////

    RF_Judge(KEYSIGHT,RF1
      ,_Mode, Power
      ,_CenterFreq, CenterFreq
      ,_MeasureLevel, measureLevel
      ,_AcquireTime, acquire_time
      ,_Judge, -6, 2
      ,_Wait, 100mS
    );
    RF_Measure(KEYSIGHT,RF1
              ,_Mode, Power
              ,_CenterFreq, CenterFreq
              ,_MeasureLevel, measureLevel
              ,_AcquireTime, acquire_time
              ,_ReturnPower, pow_V
              ,_Wait, 100mS
    );
    SOCKET_INC(FRZ_OFF)
    {
      JUDGE_VARIABLE (SOCKET, pow_V[SOCKET][0], -6, 2,
"RF_Measure_Tx_Power");
    }


    POWEROFF();
}
```

```
Rx_Gain()
{
      DATALOG_MSG="Rx_Gain";
      TEST_NO(810);
      int data_P03[65000], cnt=7;

      //Set RF Power in = -50dBm
      FREQUENCY CenterFreq = 2440MHz;
      double Source = -50;
      //////////////Chroma statement : Source CW signal//////////////////////////////////

      RELAY_ON(SPI_PINS,PDCL,3mS);

      RF_Modulation_Burst(KEYSIGHT,RF1
         ,_SourceLevel, Source
         ,_CenterFreq, CenterFreq
         ,_Wait, 100mS
      );

      ////////////////////////////////////////////
      SET_LEVELN( Level1          , P03    , 0.0V,3.3V ,1.0V,1.0V,     0.0mA, 0.0mA ,1.65V, , );
      LEVELS( Level1, 1mS) ;

PIN_MODE(RESET+P00+P01+P02+P03+P04+P05+PROG,NRZ,EDGE,MASK,IO_NRZ,OF
F);
      PIN_MODE(P02,RO,EDGE,MASK,IO_NRZ,OFF);
      PIN_MODE(P03,NRZ,EDGE,ENABLE,IO_NRZ,OFF);
      RATEN(TS1, 1000nS);
      MARKERN(TS1,ALL_PINS    , 0.0nS, -1,-1 ,0nS ,-1 ,800nS ,-1 );
      MARKERN(TS1,P02          , 250nS,750nS,-1 ,0nS ,-1 ,800nS ,-1 );
      MARKERN(TS1,P03          , 0nS,-1,-1 ,0nS ,-1 ,800nS ,-1 );

      JUDGE_PAT(two_wire_SPI_trapping,__two_wire_SPI_trapping);
      FORCE_V_MLDPS(VDDUSB, 5.0V, @6V, @500mA, 300mA, NORM, ON, 3mS);
      FORCE_V_MLDPS(VPP, 0.0V, @6V, @500mA, 300mA, NORM, ON, 3mS);
      FORCE_V_MLDPS(VDD_DIG+VDD18PA, 0.0V, @6V, @500mA, 300mA, NORM, ON,
3mS);
      SET_JUDGE_MODE( NORM , FEOP_ON    );
      JUDGE_PAT(Rx_BER_all_register,__Rx_BER_all_register);
      SET_JUDGE_MODE( NORM , FEOP_OFF    );
      JUDGE_PAT(header_register_RF_BankC,__header_register_RF_BankC);
      JUDGE_PAT(calibration_VCO_Tx,__calibration_VCO_Tx);
      JUDGE_PAT(calibration_VCO_Rx,__calibration_VCO_Rx);
      JUDGE_PAT(Rx_enable_Rx_standby,__Rx_enable_Rx_standby);
      JUDGE_PAT(Rx_Gain_write_Bank1,__Rx_Gain_write_Bank1);
      SET_CAPTURE_MEM_MODE(P03,PEL,ON,64K,LOG_FAIL_ONLY,ON,FEOP_ON);
      WAIT(10mS);
      JUDGE_PAT(Rx_Gain_Read_RSSI,__Rx_Gain_Read_RSSI);

SET_CAPTURE_MEM_MODE(P03,PEL,OFF,64K,LOG_FAIL_ONLY,OFF,FEOP_OFF);
      SET_LEVELN( Level1          , P03     , 0.0V,3.3V ,1.65V,1.65V,     0.0mA,
0.0mA ,1.65V, , );

      SOCKET_INC(FRZ_OFF)
      {
         int RSSI=0;
         PLAN_RESULT_MS[SOCKET]=PASS;
         READ_CAPTURE_MEM_BLOCK(SOCKET , P03, 0, 20, data_P03);
         cnt=7;
         for(int i=0;i<8;i++)
```

```
        {
            int bit=data_P03[i];
            RSSI+= bit *pow(2,cnt--);
        }
        if(print){printf("Site :%d RSSI :%d\n",SOCKET,RSSI);fflush(stdout);}

        JUDGE_VARIABLE(SOCKET,RSSI,180,190,"RSSI"); //180~190

    }
    ///////////////Chroma statement : Stop source//////////////////////////////////////

    RF_Stop_Modulation(KEYSIGHT, RF1);

    ////////////////////////////////////////////

    POWEROFF();
}
Tx_Frequency_Deviation()
{
    DATALOG_MSG="Tx_Frequency_Deviation";
    TEST_NO(1010);
    FREQUENCY CenterFreq = 2439.8MHz;
    double measureLevel = 5;
    CUSTOM_2DIM DEV_V;
    int NumSymbolsToAnalyse=400;
    FREQUENCY SymbolRate=1MHz;
    FREQUENCY measurementSpan_v=5MHz;
    TIME acquire_time = 2mS;
    TIME offset_time=0.5mS;

    RELAY_ON(SPI_PINS,PDCL,3mS);

    LEVELS( Level1, 1mS) ;

PIN_MODE(RESET+P00+P01+P02+P03+P04+P05+PROG,NRZ,EDGE,MASK,IO_NRZ,OF
F);
    PIN_MODE(P02,RO,EDGE,MASK,IO_NRZ,OFF);
    PIN_MODE(P03,NRZ,EDGE,ENABLE,IO_NRZ,OFF);
    RATEN(TS1, 1000nS);
    MARKERN(TS1,ALL_PINS    , 0.0nS, -1,-1 ,0nS ,-1 ,800nS ,-1 );
    MARKERN(TS1,P02          , 250nS,750nS,-1 ,0nS ,-1 ,800nS ,-1 );
    MARKERN(TS1,P03          , 0nS,-1,-1 ,0nS ,-1 ,800nS ,-1 );
    JUDGE_PAT(two_wire_SPI_trapping,__two_wire_SPI_trapping);
    FORCE_V_MLDPS(VDDUSB, 5.0V, @6V, @500mA, 300mA, NORM, ON, 3mS);
    FORCE_V_MLDPS(VPP, 0.0V, @6V, @500mA, 300mA, NORM, ON, 3mS);
    FORCE_V_MLDPS(VDD_DIG+VDD18PA, 0.0V, @6V, @500mA, 300mA, NORM, ON,
3mS);

    SET_JUDGE_MODE( NORM , FEOP_ON    );
    JUDGE_PAT(Rx_BER_all_register,__Rx_BER_all_register);
    SET_JUDGE_MODE( NORM , FEOP_OFF    );
    JUDGE_PAT(calibration_VCO_Tx,__calibration_VCO_Tx);
    JUDGE_PAT(calibration_VCO_Rx,__calibration_VCO_Rx);
    JUDGE_PAT(Tx_Power_open_loop,__Tx_Power_open_loop);
    JUDGE_PAT(whitening_remove,__whitening_remove);
    RATEN(TS3, 2000nS);
    MARKERN(TS3,ALL_PINS    , 0.0nS, -1,-1 ,0nS ,-1 ,800nS ,-1 );
    MARKERN(TS3,P02          , 250nS,750nS,-1 ,0nS ,-1 ,800nS ,-1 );
    MARKERN(TS3,P03          , 0nS,-1,-1 ,0nS ,-1 ,800nS ,-1 );
    JUDGE_PAT(continuous_Tx_Freq_Dev,__continuous_Tx_Freq_Dev);
```

```
      SOCKET_INC(FRZ_OFF)
      {
         PLAN_RESULT_MS[SOCKET]=PASS;
      }

      ///////////////Chroma statement : Measure Frequency Deviation//////////


      RF_Judge(
      KEYSIGHT,RF1
      ,_Mode, FreqDeviation
      ,_CenterFreq, CenterFreq
      ,_MeasureLevel, measureLevel
      ,_NumSymbolsToAnalyse,NumSymbolsToAnalyse
      ,_ModulationTypes, FSK2
      ,_SymbolRate,SymbolRate
      ,_MeasurementSpan, measurementSpan_v
      ,_AcquireTime, acquire_time
      ,_OffsetTime, offset_time//default =0
      ,_Judge, 205000, 332000
      ,_Wait, 10mS
      );
      RF_Measure(
      KEYSIGHT,RF1
      ,_Mode, FreqDeviation
      ,_CenterFreq, CenterFreq
      ,_MeasureLevel, measureLevel
      ,_NumSymbolsToAnalyse,NumSymbolsToAnalyse
      ,_ModulationTypes, FSK2
      ,_SymbolRate,SymbolRate
      ,_MeasurementSpan, measurementSpan_v
      ,_AcquireTime, acquire_time
      ,_OffsetTime, offset_time//default =0
      ,_ReturnFreqDev, DEV_V
      ,_Wait, 10mS
      );
      SOCKET_INC(FRZ_OFF)
      {
         JUDGE_VARIABLE (SOCKET, DEV_V[SOCKET][0], 205K, 332K,
"RF_Measure_Tx_Frequency_Deviation");
      }


      ////////////////////////////////////////////////////////////////////
   POWEROFF();
}
```

5.6.2 RF_Initialize

The statement, RF_Initialize will initial the RF Subsystem. It is recommended put in "START_UP" sub-function.

5.6.2.1 Format

```
CRAFT_RF_Initialize(Vendor
[,_IP,IP]
[,_IniFilePath,IniFilePath]
);
```

Vendor : Input the RF vendor brand.
KEYSIGHT (initialize Keysight RF subsystem)
**no_entry: illegal

IP : IP address of RF equipment
Type: string

IniFilePath : Load calibration file path.
Type: string

5.6.2.2 Example

```
START_UP()
{
    RATEN(TS1,800nS);
    MARKERN(TS1,test_pins,0nS,55nS,-1,0nS,-1,400nS,-1);
    SET_LEVELN(rst_LV,test_pins,0V,0V,1.2V,1.3V,0uA,0uA,2V,,);
    SET_LEVELN(OS_LV,test_pins,0V,0.0V,2V,4V,0uA,0uA,0V);
    SET_LEVELN(funhv_LV,test_pins    ,0V,3.3V,1.55V,1.75V,0uA,0uA,2V,,);

    RF_Initialize(KEYSIGHT,_IP,"192.168.1.2",_IniFilePath,"CableLoss.ini");
}
```

5.6.3 RF_System_Alignment

This statement provides the alignment action of RF equipment.

5.6.3.1 Format

```
RF_System_Alignment (Vendor , Pins,);
```

Vendor : Input the RF vendor brand.
KEYSIGHT initialize KEYSIGHT RF subsystem
**no_entry: illegal
Pins : The pins are defined in .dec file. The pin type should be “RF”.

5.6.3.2 Example

```
RF_System_Alignment(KEYSIGHT,RF1);
```

5.6.4 RF_Load_Modulation_File

Load modulation file to the associate channel memory. It is strongly recommended that execute in sub-function “START_UP”.

5.6.4.1 Format

```
RF_Load_Modulation_File(Vendor ,
[_Pin, Pins, _ModulationFile, full_file_path]//Aeroflex
);
```

Vendor : Input the RF vendor brand.
KEYSIGHT initialize KEYSIGHT RF subsystem
**no_entry: illegal
Pins : The pins are defined in .dec file. The pin type should be “RF”.
full_file_path : The modulation file path.//this file is on RF PC.
Type: string

5.6.4.2 Example

```
START_UP()
{
	RATEN(TS1,800nS);
	MARKERN(TS1,test_pins,0nS,55nS,-1,0nS,-1,400nS,-1);
	SET_LEVELN(rst_LV,test_pins,0V,0V,1.2V,1.3V,0uA,0uA,2V,,);
	RF_Initialize(KEYSIGHT,_IP,"192.168.1.2",_IniFilePath,"CableLoss.ini");
	RF_Load_Modulation_File(KEYSIGHT,_Pin,RF1,_ModulationFile,"D:\\NVARB\\
FSK_Chroma2.wfm");
}
```

5.6.5 RF_System_Temperature

This statement provides user the method to return the temperature of RF equipment.

5.6.5.1 Format

```
RF_Measure (
Vendor,Pins
[,_ReturnTemp,temp]
);
```

Vendor : Input the RF vendor brand.
KEYSIGHT (initialize KEYSIGHT RF subsystem)
**no_entry: illegal

Pins : The pins are defined in .dec file. The pin type should be “RF”.

temp : return the measured temperature value
Type: CUSTOM_2DIM
**no_entry: illegal

5.6.5.2 Example

```
CUSTOM_2DIM temp;
RF_System_Temperature(KEYSIGHT,RF1,_ReturnTemp,temp);
```

5.6.6 RF_Measure

RF Measure statement provided user to measure RF signal power, maximum power's frequency and RF signal deviation value.

5.6.6.1 Format

```
RF_Measure (
Vendor,Pins
[,_Mode, Mode]
[,_CenterFreq, freq]
[,_MeasureLevel, level]
[,__AcquireTime, acquire_time]
[,_NumSymbolsToAnalyse,NumSymbolsToAnalyse]
[,_ModulationTypes, FSK2]
[,_SymbolRate,SymbolRate]
[,_MeasurementSpan, measurementSpan_v]
[,_OffsetTime, offset_time]
[,_ReturnPower, Power_value]
[,_ReturnMaxPowerFreq, MaxPowerFreq_value]
[,_ReturnFreqDev, DEV_V]
[,_Wait, wait_time]
);
```

Vendor : Input the RF vendor brand.
KEYSIGHT (initialize KEYSIGHT RF subsystem)
**no_entry: illegal

Pins : The pins are defined in .dec file. The pin type should be "RF".

Mode : Select the measure mode.
Power (measure RF signal power)
Max_Pow_Freq (measure max power's frequency)
FreqDeviation(measure RF signal deviation value)
**no_entry: illegal

freq : measurement center frequency in Hz
Type: double, FREQUENCY
380MHz to 6GHz
**no_entry: illegal

level : measurement level in dBm
Type: double
**no_entry: illegal

acquire_time : acquired time to do analysis
Type: double,TIME
**no_entry: illegal

NumSymbolsToAnalyse : define the number of symbol to do deviation analysis
Type: int
****Only for MODE : FreqDeviation**

FSK2 : acquired time to do analysis

****only two type : OQPSK;FSK2**
****Only for MODE : FreqDeviation**

SymbolRate : define the symbol rate to do deviation analysis
Type: double, FREQUENCY
****Only for MODE : FreqDeviation**

measurementSpan_v: define the span to do deviation analysis
Type: double, FREQUENCY
****Only for MODE : FreqDeviation**

offset_time : define the offset time from acquired time to do analysis
Type: double,TIME
***Only for MODE : FreqDeviation**

wait_time : delay time
Type: double,TIME

Power_value : return the measured power value
Type: CUSTOM_2DIM
**no_entry: illegal

MaxPowerFreq_value : return the measured maximum power frequency value
Type: CUSTOM_2DIM
**no_entry: illegal

DEV_V : return the measured deviation value
Type: CUSTOM_2DIM
**no_entry: illegal

5.6.6.2 Example

Measure Power

```
FREQUENCY CenterFreq = 2439.8MHz;
double measureLevel = 5;
CUSTOM_2DIM pow_V;
TIME acquire_time = 1mS;

RF_Measure(KEYSIGHT,RF1
                ,_Mode, Power
                ,_CenterFreq, CenterFreq
                ,_MeasureLevel, measureLevel
                ,_AcquireTime, acquire_time
                ,_ReturnPower, pow_V
                ,_Wait, 100mS
        );
```

Measure Maximum Power Frequency

```
FREQUENCY CenterFreq = 2439.8MHz;
double measureLevel = 5;
CUSTOM_2DIM MaxPowerFreq_v;
TIME acquire_time = 1mS;
RF_Measure(
                KEYSIGHT,RF1
                ,_Mode, Max_Pow_Freq
                ,_CenterFreq, CenterFreq
                ,_MeasureLevel, measureLevel
                ,_AcquireTime, acquire_time
                ,_ReturnMaxPowerFreq, MaxPowerFreq_v
                ,_Wait, 5mS
        );
```

Measure Devuation

```
FREQUENCY CenterFreq = 2439.8MHz;
double measureLevel = 5;
CUSTOM_2DIM DEV_V;
int NumSymbolsToAnalyse=400;
FREQUENCY SymbolRate=1MHz;
FREQUENCY measurementSpan_v=5MHz;
TIME acquire_time = 2mS;
TIME offset_time=0.5mS;
RF_Measure(
        KEYSIGHT,RF1
        ,_Mode, FreqDeviation
        ,_CenterFreq, CenterFreq
        ,_MeasureLevel, measureLevel
        ,_NumSymbolsToAnalyse,NumSymbolsToAnalyse
        ,_ModulationTypes, FSK2
        ,_SymbolRate,SymbolRate
        ,_MeasurementSpan, measurementSpan_v
        ,_AcquireTime, acquire_time
        ,_OffsetTime, offset_time
        ,_ReturnFreqDev, DEV_V
        ,_Wait, 10mS
        );
```

5.6.7 RF_JUDGE

RF_Judge statement provided user to judge RF signal power, max power's frequency and RF signal deviation value.

5.6.7.1 Format

```
RF_Judge (
Vendor,Pins
[,_Mode, Mode]
[,_CenterFreq, freq]
[,_MeasureLevel, level]
[,__AcquireTime, acquire_time]
[,_NumSymbolsToAnalyse,NumSymbolsToAnalyse]
[,_ModulationTypes, FSK2]
[,_SymbolRate,SymbolRate]
[,_MeasurementSpan, measurementSpan_v]
[,_OffsetTime, offset_time]
[,_Judge, low_limit, high_limit]
[,_Wait, wait_time]
);
```

Vendor : Input the RF vendor brand.
KEYSIGHT (initialize KEYSIGHT RF subsystem)
**no_entry: illegal

Pins : The pins are defined in .dec file. The pin type should be "RF".

Mode : Select the measure mode.
Power (measure RF signal power)
Max_Pow_Freq (measure max power's frequency)
FreqDeviation(measure RF signal deviation value)
**no_entry: illegal

freq : measurement center frequency in Hz
Type: double, FREQUENCY
380MHz to 6GHz
**no_entry: illegal

level : measurement level in dBm
Type: double
**no_entry: illegal

acquire_time : acquired time to do analysis
Type: double,TIME
**no_entry: illegal

NumSymbolsToAnalyse : define the number of symbol to do deviation analysis
Type: int
****Only for MODE : FreqDeviation**

FSK2 : acquired time to do analysis
****only two type : OQPSK;FSK2**
****Only for MODE : FreqDeviation**

SymbolRate : define the symbol rate to do deviation analysis

Type: double, FREQUENCY
****Only for MODE : FreqDeviation**

measurementSpan_v: define the span to do deviation analysis
Type: double, FREQUENCY
****Only for MODE : FreqDeviation**

offset_time : define the offset time from acquired time to do analysis
Type: double,TIME
***Only for MODE : FreqDeviation**

low_limit : The judge result's lower limit. ; default unit = Hz
high_limit : The judge result's higher limit. ; default unit = Hz
wait_time : delay time
Type: double,TIME

5.6.7.2 Example

Judge Power

```
FREQUENCY CenterFreq = 2439.8MHz;
double measureLevel = 5;
CUSTOM_2DIM MaxPowerFreq_v;
TIME acquire_time = 1mS;

RF_Judge(KEYSIGHT,RF1
        ,_Mode, Power
        ,_CenterFreq, CenterFreq
        ,_MeasureLevel, measureLevel
        ,_AcquireTime, acquire_time
        ,_Judge, -6, 2
        ,_Wait, 100mS
    );
```

Judge Maximum Power Frequency

```
FREQUENCY CenterFreq = 2439.8MHz;
double measureLevel = 5;
CUSTOM_2DIM MaxPowerFreq_v;
TIME acquire_time = 1mS;

RF_Judge(
    KEYSIGHT,RF1
    ,_Mode, Max_Pow_Freq
    ,_CenterFreq, CenterFreq
    ,_MeasureLevel, measureLevel
    ,_AcquireTime, acquire_time
    ,_Judge, 0, 20000
    ,_Wait, 5mS
    );
```

Judge Deviation

```
FREQUENCY CenterFreq = 2439.8MHz;
double measureLevel = 5;
CUSTOM_2DIM DEV_V;
int NumSymbolsToAnalyse=400;
FREQUENCY SymbolRate=1MHz;
FREQUENCY measurementSpan_v=5MHz;
TIME acquire_time = 2mS;
TIME offset_time=0.5mS;
RF_Judge(
        KEYSIGHT,RF1
        ,_Mode, FreqDeviation
        ,_CenterFreq, CenterFreq
        ,_MeasureLevel, measureLevel
        ,_NumSymbolsToAnalyse,NumSymbolsToAnalyse
        ,_ModulationTypes, FSK2
        ,_SymbolRate,SymbolRate
        ,_MeasurementSpan, measurementSpan_v
        ,_AcquireTime, acquire_time
        ,_OffsetTime, offset_time//default =0
        ,_Judge, 205000, 332000
        ,_Wait, 10mS
        );
```

5.6.8 RF_Modulation_Burst

Star to force the modulation RF signal or CW signal.

5.6.8.1 Format

```
RF_ Modulation_Burst(
Vendor,Pins
[,_SourceModulationFile, source_file]
[,_SourceLevel, level ]
[,_CenterFreq, freq]
[,_Wait, wait_time]
);
```

Vendor : Input the RF vendor brand.
KEYSIGHT (initialize KEYSIGHT RF subsystem)

Pins : The pins are defined in .dec file. The pin type should be “RF”.

source_file : This is the source modulation file path. Indicate which loaded modulation file will be used for modulation output. If it doesn’t be assigned, CW will output from the RF port.
Type: string

Level : Output RF signal power level.
Type: double.

Freq : Output RF signal center frequency.
380MHz to 6GHz
Type: double, FREQUENCY

wait_time : delay time
Type: double,TIME

5.6.8.2 Example

```
CW signal :

FREQUENCY CenterFreq = 2406MHz;
double power_db=-60;
RF_Modulation_Burst(KEYSIGHT,RF1
        ,_SourceLevel, Source
        ,_CenterFreq, CenterFreq
        ,_Wait, 100mS
    );
Modulation signal :

FREQUENCY CenterFreq = 2406MHz;
double power_db=-60;
RF_Modulation_Burst(KEYSIGHT,RF1
        ,_SourceModulationFile, "FSK_Chroma2.wfm"
        ,_SourceLevel, power_db
        ,_CenterFreq, CenterFreq
        ,_Wait, 100mS
    );
```

5.6.9 RF_Stop_Modulation

Stop forcing modulation signal. It is strongly recommended that user should stop running modulation signal at the end of the test item while the test item had been ever initialed RF_ Modulation_Burst statement.

5.6.9.1 Format

```
RF_Stop_Modulation (Vendor,PINS);
```

Vendor : Input the RF vendor brand.
KEYSIGHT (initialize KEYSIGHT RF subsystem)
**no_entry: illegal

Pins : The pins are defined in .dec file. The pin type should be “RF”.

5.6.9.2 Example

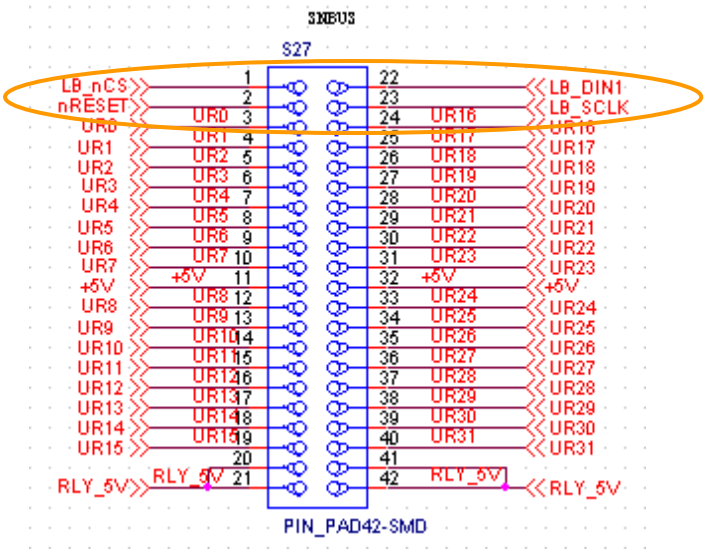
```
RF_Stop_Modulation(KEYSIGHT, RF1);
```

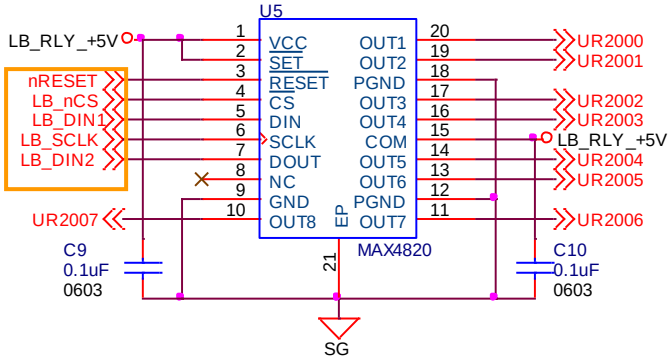
Appendix A How to Use 4820 to Add Relay Control Signal

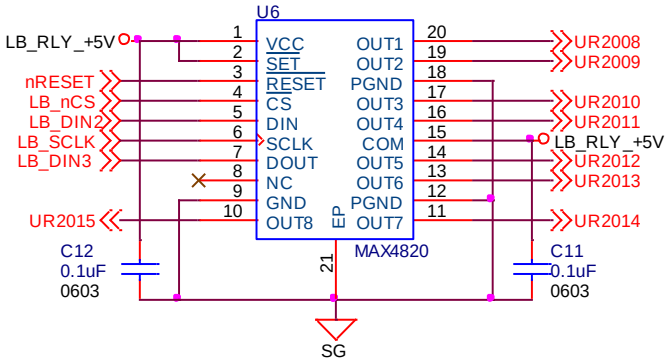
Descriptions:
4820 is a relay control IC. It provides eight relay control signal, and support serial mode.
SNPUS/SPBUS provide four signals to control 4820 on ATE 3380 series
By the way, user can get 2048 relay control bit.

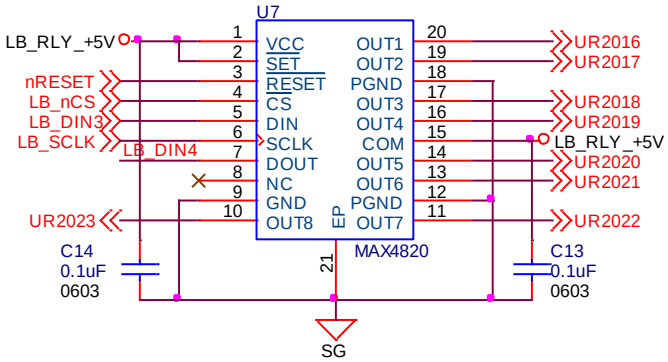
Circuit

4820	SNBUS/SPBUS
RESET	nRESET
CS	LB_nCS
DIN	LB_DIN1
SCLK	LB_SCLK




U5
LB_RLY_+5V
nRESET
LB_nCS
LB_DIN1
LB_SCLK
LB_DIN2
UR2007
C9
0.1uF
0603
VCC
SET
RESET
CS
DIN
SCLK
DOUT
NC
GND
OUT8
OUT1
OUT2
PGND
OUT3
OUT4
COM
OUT5
OUT6
PGND
OUT7
EP
21
MAX4820
UR2000
UR2001
UR2002
UR2003
LB_RLY_+5V
UR2004
UR2005
UR2006
C10
0.1uF
0603
SG


U6
LB_RLY_+5V
nRESET
LB_nCS
LB_DIN2
LB_SCLK
LB_DIN3
UR2015
C12
0.1uF
0603
VCC
SET
RESET
CS
DIN
SCLK
DOUT
NC
GND
OUT8
OUT1
OUT2
PGND
OUT3
OUT4
COM
OUT5
OUT6
PGND
OUT7
EP
21
MAX4820
UR2008
UR2009
UR2010
UR2011
LB_RLY_+5V
UR2012
UR2013
UR2014
C11
0.1uF
0603
SG


U7
LB_RLY_+5V
nRESET
LB_nCS
LB_DIN3
LB_SCLK
LB_DIN4
UR2023
C14
0.1uF
0603
VCC
SET
RESET
CS
DIN
SCLK
DOUT
NC
GND
OUT8
OUT1
OUT2
PGND
OUT3
OUT4
COM
OUT5
OUT6
PGND
OUT7
EP
21
MAX4820
UR2016
UR2017
UR2018
UR2019
LB_RLY_+5V
UR2020
UR2021
UR2022
C13
0.1uF
0603
SG

program

➔Declaration file

Channel assign: 2000 ~ 4048

```
PIN_LIST(1_site_pins)
/*pin_name=L/B channel = dut_pin = pin_type*/
{
P00   =  0      =   0   = IO;
RL0   =  2000   =       = UR;
RL1   =  2001   =       = UR;
RL2   =  2002   =       = UR;
RL3   =  2003   =       = UR;
RL4   =  2004   =       = UR;
RL5   =  2005   =       = UR;
RL6   =  2006   =       = UR;
RL7   =  2007   =       = UR;
}

UR_PIN_GROUP
{
Ur_all = RL0+RL1+RL2+RL3+RL4+RL5+RL6+RL7;
Ur_aa= RL1+RL3+RL5+RL7;
Ur_55= RL0+RL2+RL4+RL6;
}
```

Channel assign: 2000~4048

➔Pln file

Use statement "SET_CBIT ()"

Serial relay control signal don't support hex (0x0) control mode. Only use"PIN name" or "PIN Group".

```
Test()
{
SET_CBIT(Ur_aa,ON,2mS);
DEBUG_BREAK();

msg_ret=MessageBox(NULL,"check if LED Green 2 ,4 ,6 ,8
ON ?","warning",MB_YESNO|MBSYSTEMMODAL);

SET_CBIT(Ur_all,OFF,2mS);

if(msg_ret!=6)
   {
    MessageBox(NULL,"Please check TBUS Board if
OK?","warning",MB_OK|MBSYSTEMMODAL);
    PLAN_RESULT_MS[0]=FAIL;
    return2;

   }
}
```

Reminder

1.3380 MXBUS pogopin definition

MXBUS PoGo Pin Define	//2010/12/17 update								
Pogo Pin No	define	define	define	define	define	define	define	define	Pogo Pin No
	a	b	c	d	e	f	g	h	
1	LBLVDSIN_P0	DG	DG	DG	AG	AG	AG	LBLVDSOUT_P0	1
2	LBLVDSIN_N0	LBLVDSIN_G0	DG	DG	AG	AG	LBLVDSOUT_G0	LBLVDSOUT_N0	2
3	LBLVDSIN_P1	LBLVDSIN_G1	DG	DG	AG	AG	LBLVDSOUT_G0	LBLVDSOUT_P1	3
4	LBLVDSIN_N1	DG	DG	DG	AG	AG	AG	LBLVDSOUT_N1	4
5	LB_SCLK	LB_nCS	LB_DIN	nRESET	SPI_SCK	SPI_nCS	SPI_DIN	SPI_DOUT	5
6	UR0	UR8	UR16	UR24	UR32	UR40	UR48	UR56	6
7	UR1	UR9	UR17	UR25	UR33	UR41	UR49	UR57	7
8	UR2	UR10	UR18	UR26	UR34	UR42	UR50	UR58	8
9	UR3	UR11	UR19	UR27	UR35	UR43	UR51	UR59	9
10	UR4	UR12	UR20	UR28	UR36	UR44	UR52	UR60	10
11	UR5	UR13	UR21	UR29	UR37	UR45	UR53	UR61	11
12	UR6	UR14	UR22	UR30	UR38	UR46	UR54	UR62	12
13	UR7	UR15	UR23	UR31	UR39	UR47	UR55	UR63	13
14	LB_DOUT								14
15	LB_D5V	LB_D5V	LB_D5V	LB_D5V		LB_A5V	LB_A5V	LB_A5V	15
16	LB_D5V	LB_D5V	LB_D5V	LB_D5V		LB_A5V	LB_A5V	LB_A5V	16
17	LB_D5V	LB_D5V	LB_D5V	LB_D5V		LB_A5V	LB_A5V	LB_A5V	17

2. Don’t support hex (0x0) control mode. Only use" PIN name" or "PIN Group".
3. Support 2048 relay control bit. Channel assign from 2000 to 4048

Appendix B How to Use ALPG (Including *.DEC, *.PAT, *.PLN)

Descriptions:

- Using ALPG can help you simplify pattern, specially used in testing memories.
- ALPG pattern symbol including A/B/C/D.A/B for address pattern, C/D for data pattern.
- Using ALPG pattern U.I symbol to explain each ALPG pattern symbol.

Example:

***.DEC**

While using ALPG, user must declare ALPG pin group in DEC file. The format was shown as below:

```
//ALPG_ID_0
ALPG_PIN_GROUP{
ALPG_X=A0+A1+A2+A10+A9+A11;//(LSB---->MSB)
ALPG_Y=A8+A14+A13+A12+A7+A6+A5+A4+A3;//(LSB---->MSB)
ALPG_Z=D1+D2+D3+D4+D5+D6+D7+D8;//(LSB---->MSB)
}
//ALPG_ID_1
ALPG_PIN_GROUP{
ALPG_X1=A0+A2+A10+A9+A11;//(LSB---->MSB)
ALPG_Y1=A8+A14+A13+A12+A7+A6+A5+A4+A3;//(LSB---->MSB)
ALPG_Z1=D1+D2+D3+D4+D5+D6+D7+D8;//(LSB---->MSB)
}

//ALPG_ID_2
ALPG_PIN_GROUP{
ALPG_X2=A0+A1+A10+A9+A11;//(LSB---->MSB)
ALPG_Y2=A8+A14+A13+A12+A7+A6+A5+A4+A3;//(LSB---->MSB)
ALPG_Z2=D1+D2+D3+D4+D5+D6+D7+D8;//(LSB---->MSB)
}

//ALPG_ID_3
ALPG_PIN_GROUP{
ALPG_X3=A0+A1+A2+A9+A11;//(LSB---->MSB)
ALPG_Y3=A8+A14+A13+A12+A7+A6+A5+A4+A3;//(LSB---->MSB)
ALPG_Z3=D1+D2+D3+D4+D5+D6+D7+D8;//(LSB---->MSB)
}
```

Fig1. ALPG_PIN_GROUP

ALPG_X explain pattern symbol "A", ALPG_Y explain pattern symbol "B", ALPG_Z explain pattern symbol "C" & "D" respectively.

The pin name follows by the ALPG_X (/ALPG_Y/ALPG_Z) respect bit order,

Ex:

ALPG_X = A0+A1+A2+A10+A9 +A11 means ALPG pin group X got 6 bits,

A11-> Bit5, A9-> Bit 4…, A0-> Bit0 respectively.

ALPG_Y= A8+A14+A13+A12 +A7+A6+A5+A4+ A3 means ALPG pin group Y got 9 bits,

A3-> Bit8, A4-> Bit7, A5-> Bit2…, A8-> Bit 0 respectively.

ALPG_Z= D1+D2+D3+D4+D5+D6+D7+ D8means ALPG pin group Z got 8 bits,

D8-> Bit7, D7-> Bit6, D6-> Bit5…, D1-> Bit 0 respectively.

***.PAT**

ALPG pattern is composed by two parts: ALPG pattern symbol and ALPG micro instruction. The format was shown as Fig2.

```
SET_DEC_FILE "./ALPG_TEST.dec"

HEADER   %CE,WE,OE,%D8,D7,D6,D5,D4,D3,D2,D1,
     %A3,A4,A5,A6,A7,A12,A13,A14,A8,%A11,A9,A10,A2,A1,A0;

            /* CWO DDDDDDDD AAAAAAAAA AAAAAA */
            /* EEE 87654321 345671118 191210 */
            /*                  234  1 0    */
            /*                        */

APM_PATTERN(Check_Board_ALPG_pat)
{
//                                          SQPG'UI              X'UI              Y'UI              Z'UI
  CH_ST:* 100 XXXXXXXX 000000000 000000 * TS1;

  W_CH0:* 000 DDDDDDDD BBBBBBBBB AAAAAA * SETXO 511,UISYNC; [XA:0,X=XA,        YA:0,Y=YA,      ZA:0,ZB:0,Z=ZA==ZB]
        * 000 DDDDDDDD BBBBBBBBB AAAAAA * RPT 63;           [++XA,X=XA,       Y=YA,           ++ZA,      Z=ZA==ZB]
        * 000 DDDDDDDD BBBBBBBBB AAAAAA * JNZO .-1;         [XA:0,X=XA,        ++YA,Y=YA,      ZA:0,++ZB,Z=ZA==ZB]
        * 000 DDDDDDDD BBBBBBBBB AAAAAA * RPT 63;           [++XA,X=XA,       Y=YA,           ++ZA,      Z=ZA==ZB]

  R_CH0:* 010 CCCCCCCC BBBBBBBBB AAAAAA * TS1,SETXO 511;    [XA:0,X=XA,        YA:0,Y=YA,      ZA:0,ZB:0,Z=ZA==ZB]
        * 010 CCCCCCCC BBBBBBBBB AAAAAA * RPT 63;           [++XA,X=XA,       Y=YA,           ++ZA,      Z=ZA==ZB]
        * 010 CCCCCCCC BBBBBBBBB AAAAAA * JNZO .-1;         [XA:0,X=XA,        ++YA,Y=YA,      ZA:0,++ZB,Z=ZA==ZB]
        * 010 CCCCCCCC BBBBBBBBB AAAAAA * RPT 63;           [++XA,X=XA,       Y=YA,           ++ZA,      Z=ZA==ZB]

  W_CH1:* 000 DDDDDDDD BBBBBBBBB AAAAAA * TS1,SETXO 511;    [XA:0,X=XA,        YA:0,Y=YA,      ZA:0,ZB:0,Z=ZA!=ZB]
        * 000 DDDDDDDD BBBBBBBBB AAAAAA * RPT 63;           [++XA,X=XA,       Y=YA,           ++ZA,      Z=ZA!=ZB]
        * 000 DDDDDDDD BBBBBBBBB AAAAAA * JNZO .-1;         [XA:0,X=XA,        ++YA,Y=YA,      ZA:0,++ZB,Z=ZA!=ZB]
        * 000 DDDDDDDD BBBBBBBBB AAAAAA * RPT 63;           [++XA,X=XA,       Y=YA,           ++ZA,      Z=ZA!=ZB]

  r_CH1:* 010 CCCCCCCC BBBBBBBBB AAAAAA * TS1,SETXO 511;    [XA:0,X=XA,        YA:0,Y=YA,      ZA:0,ZB:0,Z=ZA!=ZB]
        * 010 CCCCCCCC BBBBBBBBB AAAAAA * RPT 63;           [++XA,X=XA,       Y=YA,           ++ZA,      Z=ZA!=ZB]
        * 010 CCCCCCCC BBBBBBBBB AAAAAA * JNZO .-1;         [XA:0,X=XA,        ++YA,Y=YA,      ZA:0,++ZB,Z=ZA!=ZB]
        * 010 CCCCCCCC BBBBBBBBB AAAAAA * RPT 63;           [++XA,X=XA,       Y=YA,           ++ZA,      Z=ZA!=ZB]

  CH_SP:* 100 XXXXXXXX 000000000 000000 * TS1;
}
```

A B C

Fig2. ALPG PATTERN

User must notice follow points:

a. ALPG Pattern must contain in APM_PATTERN().

b. ALPG U.I. must write after normal micro instruction, and include by "[ ]".

And Each UI separate by a ",".

c. User can mix normal pattern(0/1/L/H) and ALPG pattern(A/B/C/D) at the same time, but user must declare ALPG_PIN_GROUP in DEC file.

-ALPG pattern memory got 1k deep, and it's mapped into FPM.

-APM symbol to select X-Driver, Y-Driver, Z-Driver and Z-compare, or original FPM pattern

- Presently ALPG support maximum 16bits for each address and data pin group.

***.PLN**

In plan file, user must declare SET_ALPG_WORK_MODE before JUDGE_PAT to set ALPG pin group. If user wants to define ALPG mask pin also declare SET_ALPG_WORK_MODE.

```
//===== CHECK BOARD =====
TEST_NO(2);DATALOG_MSG="ALPG_Check_Board";
//SET_ALPG_WORK_MODE( mask_mode [,XAimask,XBimask,YAimask,YBimask,ZAimask,ZBimask]);
SET_ALPG_WORK_MODE(ON,0x0,0x0,0x0,0x0,0xFFFE,0xFFFE,0);
JUDGE_PAT(Check_Board_ALPG_pat:CH_ST ,Check_Board_ALPG_pat:CH_SP);
```

Fig3. ALPG PLAN

Appendix C The Instruction of Scan Function

Descriptions:

Scan pattern Format:

SET_DEC_FILE "Device definition file"	Block	①
HEADER	Block	②
SCANIN_PINS	Block	③
SCANOUT_PINS	Block	④
PATTERN	Block	⑤

① SET_DEC_FILE: The block defines the declaration file name and pin type.
② HEADER: The block defines declaration that is used by the order of entry and display of the actual binary information.
③ SCANIN_PINS: The block defines the SCAN input pin. A name of a single pin defined in the PINS_LIST declaration. Only constant names from the PIN_LIST are allowed. 〝C〞 type：STRING，64 characters maximum.
④ SCANOUT_PINS: The block defines the SCAN output pin. A name of a single pin defined in the PIN_LIST declaration. Only constant names from the PIN_LIST are allowed. 〝C〞 type: STRING, 64 characters maximum.
⑤ Pattern: This block defines the pattern function and format.

```
SET_DEC_FILE "1s299_16sites_pin.dec"

HEADER   CLR,%SEL0,SEL1,%G1,G2,%CLK,%SL,SR,%QA,QB,QC,
         QD,QE,QF,QG,QH;

SCANIN_PINS  SR ;
SCANOUT_PINS QA ;
           /*C SS GG C SS ABCDEFGH */
           /*L EE 12 L LR QQQQQQQQ */
           /*R LL    K    ABCDEFGH */
           /*  01                  */
SPM_PATTERN (scan_func_pat) {
            *0 00 00 0 00 LLLLLLLL *TS2;
            *0 11 00 0 00 XXXXXXXX *;

            *1 11 11 1 00 00000000 *;//-----load
            *1 00 00 0 00 LLLLLLLL *;//-----hold
            *1 11 11 1 00 11111111 *;
            *1 00 00 0 00 HHHHHHHH *;
            *1 11 11 1 00 10101010 *;
            *1 00 00 0 00 HLHLHLHL *;
            *1 11 11 1 00 01010101 *;
            *1 00 00 0 00 LHLHLHLH *;

            *0 00 00 0 00 LLLLLLLL *; //sfr_st::
            *1 10 00 1 01 HXXXXXXX *; //Scan Setup, scan hold pin setup
  <SCAN_ST>
   <SR>   *11111111 00000000 11111111 00000000 11111111 00000000
           10101010 10101010 10101010 10101010 10101010 10101010
           11111111 00000000 11111111 00000000 11111111 00000000 *;
   <QA>   *HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL
           HLHLHLHL HLHLHLHL HLHLHLHL HLHLHLHL HLHLHLHL HLHLHLHL
           HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL *;
  <SCAN_SP>
            *1 10 00 1 00 XXXXXXXX *; //sfr_sp::
}
```

① ET_DEC_FILE

② EADER

③SCANIN_PINS
④SCANOUT_PINS

⑤Pattern

Scan function specification：

SCAN	Chain mode	32 chain/ board	16 chain/ board	8 chain/ board	4 chain/ board
	Max scan pin#/slot	1 scan-pin/slot	2 scan-pin/slot	4 scan-pin/slot	8 scan-pin/slot
	NO. of scan pin	16	32	64	128
	Pattern size SPM =16M	1G (= 32*2*16M)	512M (= 16*2*16M)	256M (= 8*2*16M)	128M (= 4*2*16M)
	Pattern size SPM =32M	2G (= 32*2*32M)	1G (= 16*2*32M)	512M (= 8*2*32M)	256M (= 4*2*32M)

☆ System will base on the SCAN pin defined to select how many chain/board mode.
☆ Pattern Size = 32M. MXPG3 & MXLPC hardware configulation need to 32M memory size.

SET_SCAN_WORK_MODE

SET_SCAN_WORK_MODE (scan_in_pin_name, scan_out_pin_name, mode);
Set SCAN pin work mode. Only SPM pattern memory can use scan function.
Only the channel which the ate-channel%4 ==0 can be assigned as scan pin

Example:

1. SCAN PIN：User defines the SCAN pins in SCAN pattern file, including SCAN IN and SCAN OUT. The pin name of the SCAN signal pins and ATE channel should be defined in the PINS_LIST declaration

```
SET_DEC_FILE "ls299_16sites_pin.dec"

 HEADER    CLR,%SEL0,SEL1,%G1,G2,%CLK,%SL,SR,%QA,QB,QC,
           QD,QE,QF,QG,QH;

  SCANIN_PINS   SR ;
  SCANOUT_PINS  QA ;

            /*C SS GG C SS ABCDEFGH */
            /*L EE 12 L LR QQQQQQQQ */
            /*R LL    K    ABCDEFGH */
            /*  01                  */
 SPM_PATTERN (scan_func_pat) {
```

Define declaration file

Define declaration file

2. SCAN pattern file：

```
SPM_PATTERN (scan_func_pat) {

            *0 00 00 0 00 LLLLLLLL *TS2;
            *1 00 00 0 00 LHLHLHLH *;

            *0 00 00 0 00 LLLLLLLL *; //sfr_st::
            *1 10 00 1 01 HXXXXXXX * ;  //Scan Setup, scan hold pin setup
  <SCAN_ST>
   <SR>    *11111111 00000000 11111111 00000000 11111111 00000000
            10101010 10101010 10101010 10101010 10101010 10101010
            11111111 00000000 11111111 00000000 11111111 00000000 *;
   <QA>    *HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL
            HLHLHLHL HLHLHLHL HLHLHLHL HLHLHLHL HLHLHLHL HLHLHLHL
            HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL *;
  <SCAN_SP>

            *1 10 00 1 00 XXXXXXXX *; //sfr_sp::
}
```

The Normal pattern last vector

The SCAN pattern first vector

```
SPM_PATTERN (scan_func_pat) {

            *0 00 00 0 00 LLLLLLLL *TS2;
            *1 00 00 0 00 LHLHLHLH *;

            *0 00 00 0 00 LLLLLLLL *; //sfr_st::
            *1 10 00 1 00 XXXXXXXX * ;  //Scan Setup, scan hold pin setup
 <SCAN_ST>
  <SR>    *11111111 00000000 11111111 00000000 11111111 00000000
           10101010 10101010 10101010 10101010 10101010 10101010
           11111111 00000000 11111111 00000000 11111111 00000000 *;
  <QA>    *HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL
           HLHLHLHL HLHLHLHL HLHLHLHL HLHLHLHL HLHLHLHL HLHLHLHL
           HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL HHHHHHHH LLLLLLLL *;
 <SCAN_SP>

            *1 10 00 1 00 XXXXXXXX *; //sfr_sp::
}
```

The SCAN pin name and pattern data

3. SET_SCAN_WORK_MODE

SET_SCAN_WORK_MODE(scan_in_pin_name,scan_out_pin_name,mode);

scan_in_pin_name：pin_def,pin_group (only suit for IO-Pins).

scan_out_pin_name：pin_def,pin_group(only suit for IO-Pins).

mode：ON/OFF (turn on/off scan function)

```
SET_SCAN_WORK_MODE(SR, QA, ON);

JUDGE_PAT(scan_func_pat, __scan_func_pat);

SET_SCAN_WORK_MODE(SR, QA, OFF);
```

Turn ON scan mode

Judge SCAN pattern

Turn OFF scan mode

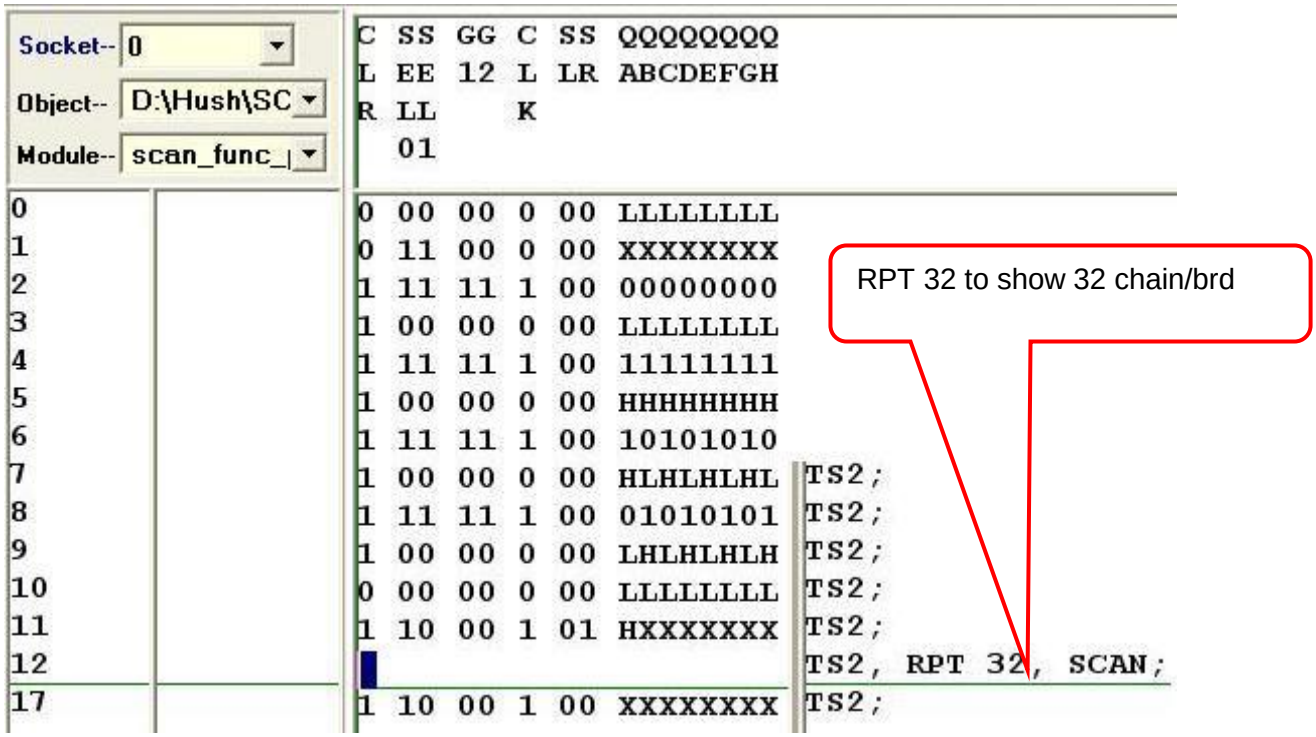
Appendix D How to Debug Scan Function

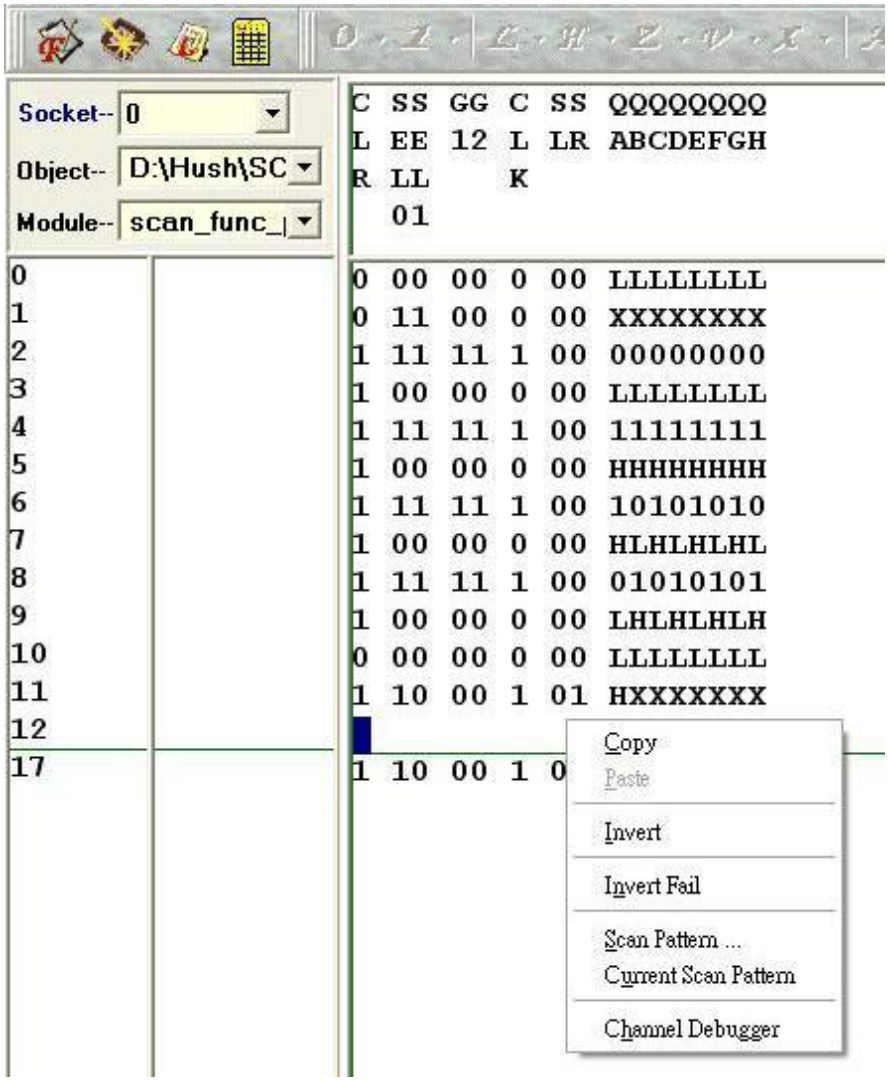
Descriptions:

1.Datalog: How to read datalog message when SCAN function fail?
2.Pattern editor: How to use Pattern edotor when debugging SCAN function?

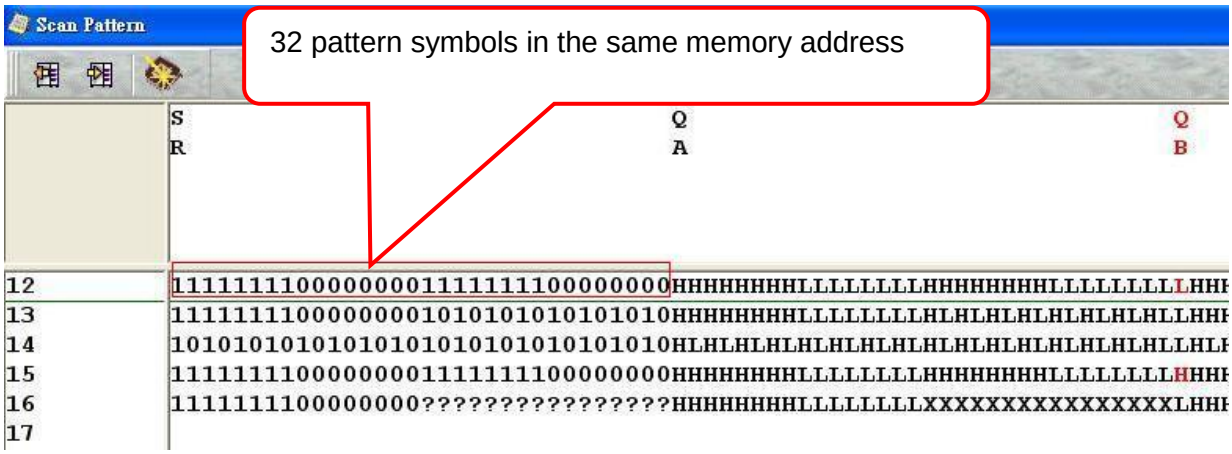
Example:

1. Pattern editor utility:





Push the right button of mouse can open menu.Select scan pattern item can open SCAN Pattern window



System will select the chain/brd mode in accord with declaration file defined by user. This example is 32 chain/brd modes. For the reason, scan pattern memory have 32 pattern symbols in the same pattern address. We can know the SCAN RPT 32 times.

2. Datalog message :

```
FAIL    13            30       1     FUNC(ls299_func_scan_F.pat)      0     SPM 12(scan_func_pat_f+12)
                     ***
             CSSGGCSSQQQQQQQQ
             LEE12LLRABCDEFGH
             RLL  K
              01
                  m  eeeeeeee
SPM       12 11000101H XXXXXX  < SCAN_RPT/SCAN_CYCLE: 1/1 >
SPM       12 11000100L XXXXXX  < SCAN_RPT/SCAN_CYCLE: 9/9 >
SPM       13 11000101H/XXXXXX  < SCAN_RPT/SCAN_CYCLE: 17/49 >
SPM       13 11000101H/XXXXXX  < SCAN_RPT/SCAN_CYCLE: 25/57 >
SPM       15 11000101H/XXXXXX  < SCAN_RPT/SCAN_CYCLE: 1/97 >
SPM       15 11000100L XXXXXX  < SCAN_RPT/SCAN_CYCLE: 9/105 >
```

SCAN_RPT: The fail vector in the current address.

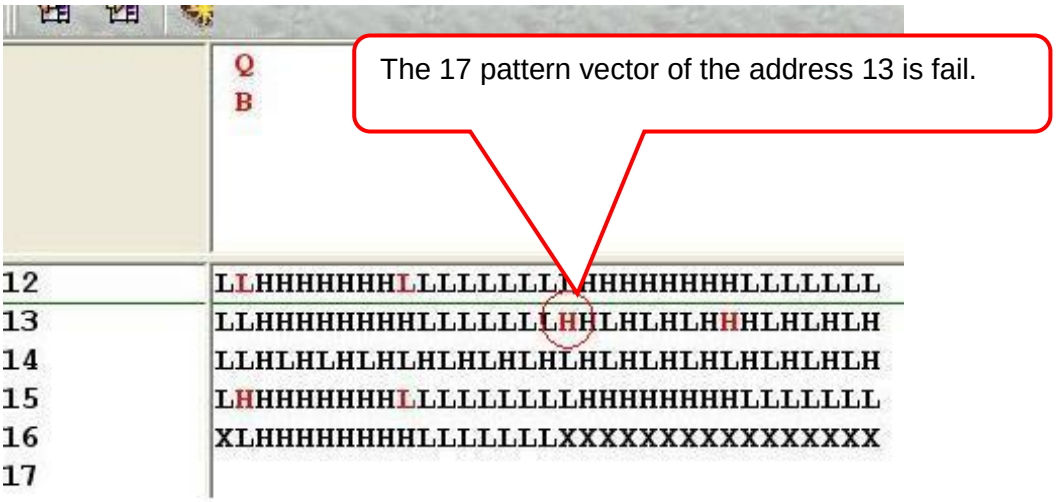
SCAN_CYCLE: The fail vector in the scan pattern.

For the example:

SPM 13 11000101H/XXXXXX <SCAN_RPT/SCAN_CYCLE: 17/49>

SCAN_RPT 17 : The 17 pattern vector of the address 13 is fail.

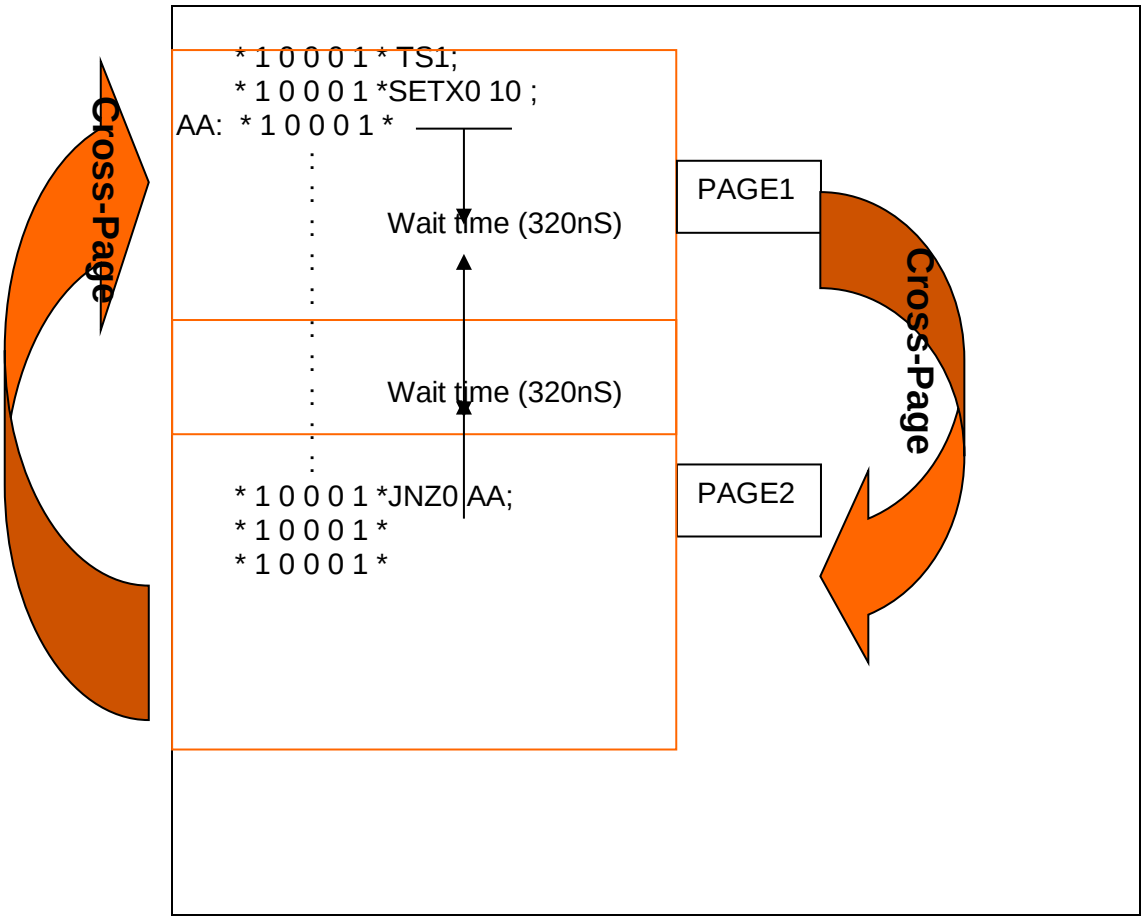
SCAN_CYCLE 49 : In Address 12 have 32 pattern vectors + the 17 pattern vector of the address 13 = 49 (32+17).



Appendix E Micro Instruction "JNZ, JMP, JSR Limitation

Descriptions:

Each page in SPM memory has 256 vectors. Before cross-page happen, SPM memory needs 320nS refresh time to prepare.
When using micro instruction "JNZ, JMP and JSR", the limitation will need be take carefor avoiding error happened.
In other words, no cross-page will be no limitation.



Example:

<Wrong Example>

TEST rate:50nS

```
     * 1 0 0 0 1 * TS1;← period 50nS
     * 1 0 0 0 1 *SETX0 400 ;
     * 1 0 0 0 1 *;                    PAGE1
AA:  * 1 0 0 0 1 *;
     * 1 1 0 0 1 *JNZ0 AA;
     * 1 0 0 0 1 *;
     * 1 0 0 0 1 *;                    PAGE2
     * 1 0 0 0 1 *;
```

Correction 1: add loop range to increase delay cycle (time)

```
     * 1 0 0 0 1 * TS1;← period 50nS
     * 1 0 0 0 1 *SETX0 50 ;
     * 1 0 0 0 1 *;
AA:  * 1 0 0 0 1 *;
     * 1 1 0 0 1 *;
     * 1 0 0 0 1 *;
     * 1 1 0 0 1 *;                    PAGE1
     * 1 0 0 0 1 *;                    Delay time>320nS
     * 1 1 0 0 1 *;
     * 1 0 0 0 1 *;
     * 1 1 0 0 1 *;
     * 1 0 0 0 1 *;
     * 1 1 0 0 1 *;                    Delay time>320nS
     * 1 0 0 0 1 *;                    PAGE2
     * 1 1 0 0 1 *;
     * 1 0 0 0 1 *;
     * 1 1 0 0 1 *;
     * 1 0 0 0 1 *;
     * 1 1 0 0 1 *JNZ0 AA;
     * 1 0 0 0 1 *;
     * 1 0 0 0 1 *;
     * 1 0 0 0 1 *;
```

Correction 2: fix JNZ loop range in the same PAGE.

```
     * 1 0 0 0 1 * TS1;← period 50nS
     * 1 0 0 0 1 *SETX0 100 ;
     * 1 0 0 0 1 *;                    PAGE1
AA:  * 1 0 0 0 1 *;
     * 1 1 0 0 1 *JNZ0 AA;
     * 1 0 0 0 1 *;
     * 1 0 0 0 1 *;                    PAGE2
     * 1 0 0 0 1 *;
```

Example:

Open short test

```
open_short_test()
{
	FORCE_V_DPS ( Vdd, 0V, @8V, @100mA, 100mA, NORM, ON, 1mS);
	SET_LEVELN( os_lvl, IO_OS_PINS,0.0V, 0.5V, 0V, 0V, 0uA, 0uA, 0V);
	LEVELS(os_lvl,5mS);

	RELAY_ON(IO_OS_PINS,PDCL,1mS);
	PIN_MODE(IO_OS_PINS,F0,EDGE,MASK,IO_ON,OFF);
	TEST_NO(10);
	PIN_INC(IO_OS_PINS)
	{
		RELAY_OFF(PIN_DEF,PDCL,1mS);
		RELAY_ON(PIN_DEF,PMU,1mS);
		FORCE_I_PMU(PMU,-200uA,@1mA,@6V,2V,ON,1mS);
		JUDGE_V_PMU(PMU,IFVM,-1.2V,-0.2V,2mS);
		FORCE_I_PMU(PMU,0uA,@100uA,@6V,2V,OFF,1mS);
		RELAY_OFF(PIN_DEF,PMU,1mS);
		RELAY_ON(PIN_DEF,PDCL,1mS);
	}
}
```

Appendix F MAWI2 of Tool Instruction

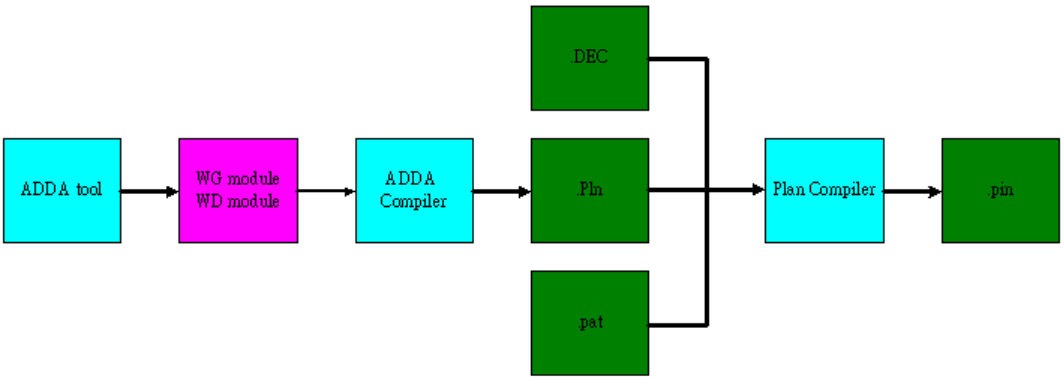
Descriptions:

2-1. AWI2 programming will highly depend on ADDA tool. In this chapter we will explain how to manipulate ADDA tool in detail.

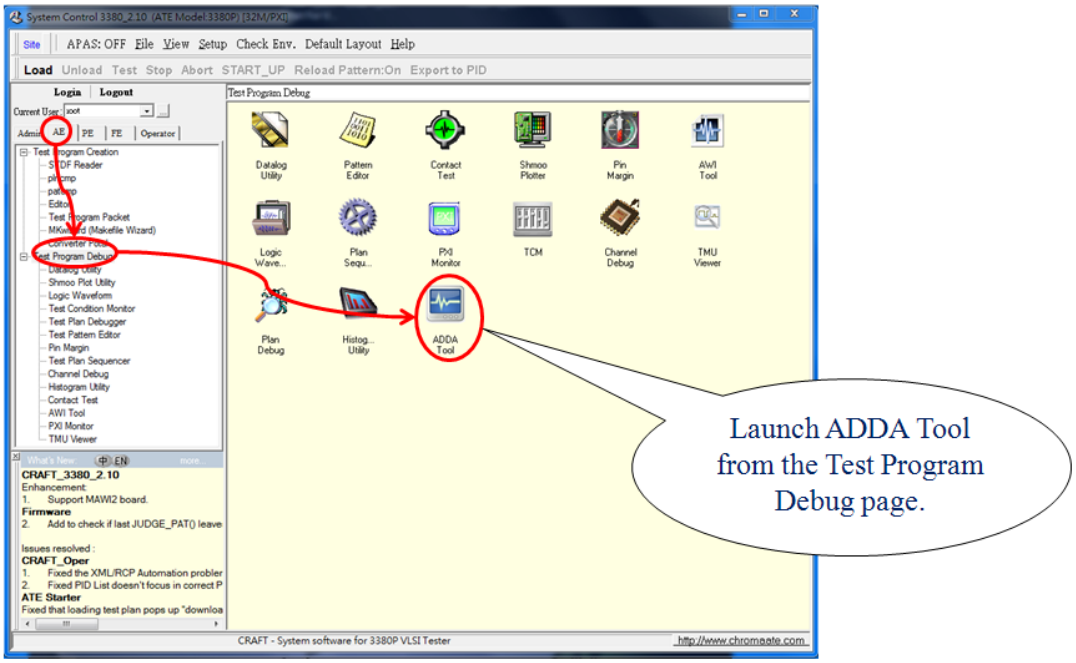
2-2. How to program AWI2 also include in this chapter.

2-3. We will use an AWI2 self test program as an example to demo coding program.

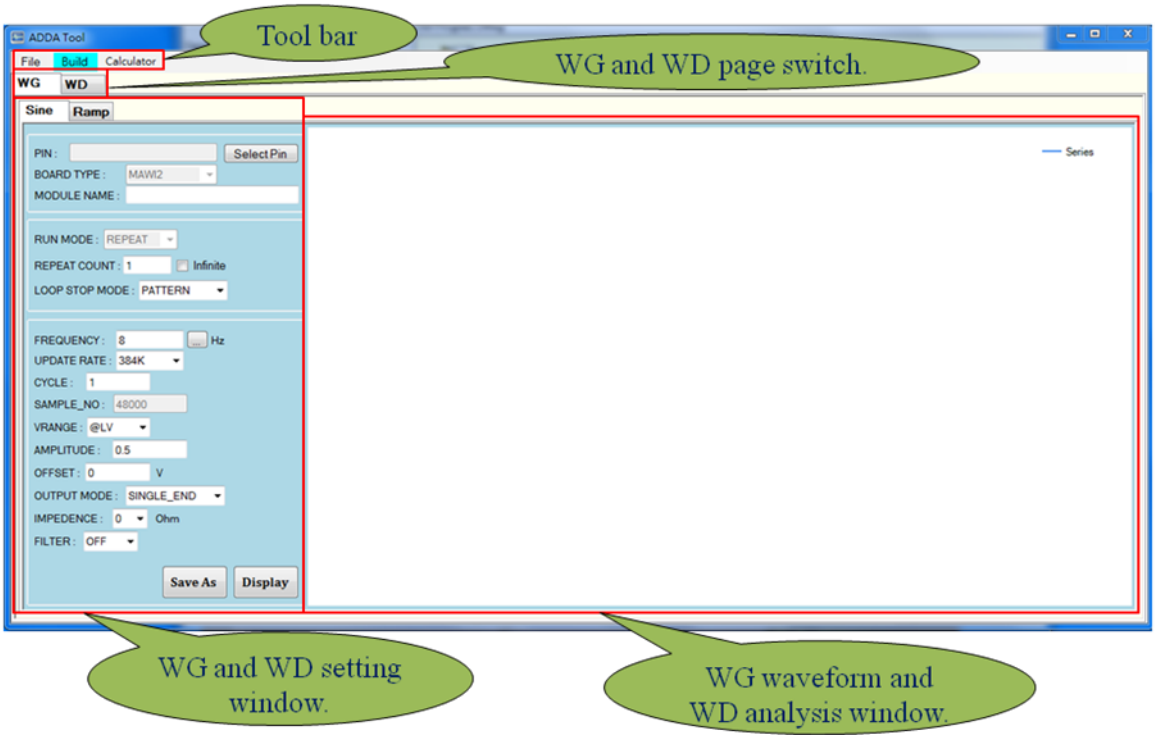
Example:



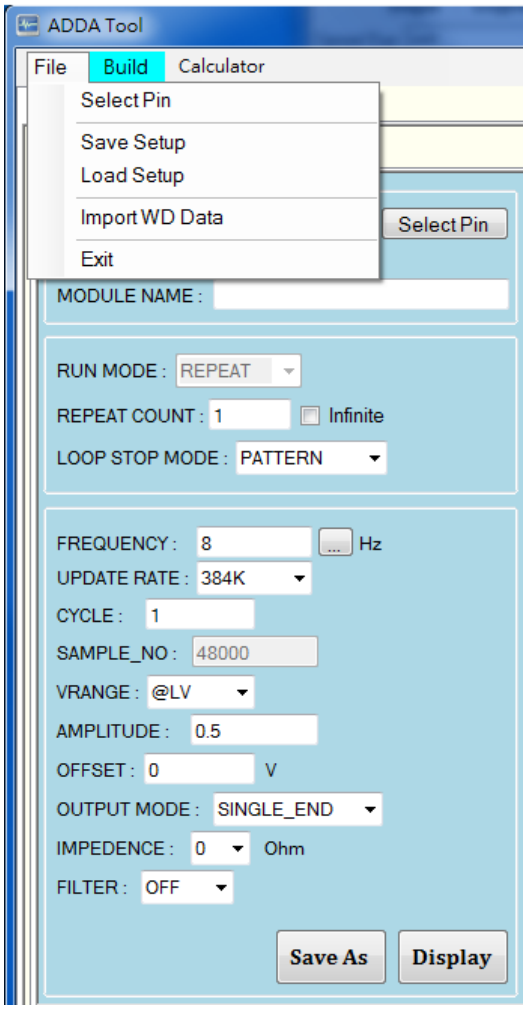
1. Launch ADDA tool



2. ADDA tool



3. File tool bar



Select Pin: Load .DEC file to select WG or WD channel.

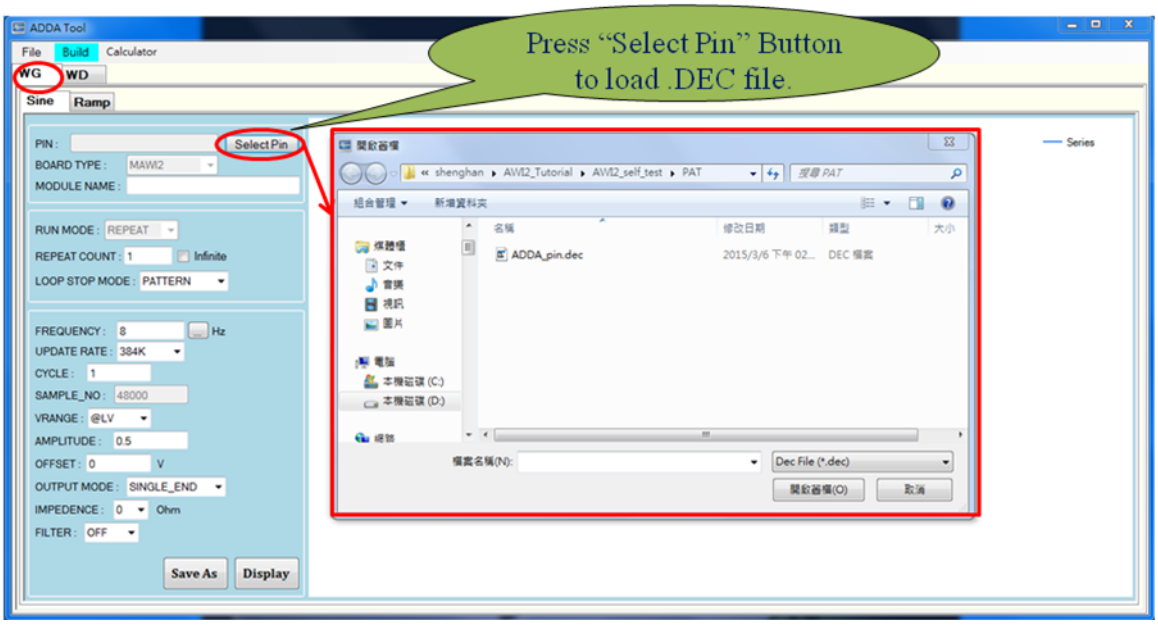
Save Setup: Saving the current WG or WD setting to a file.

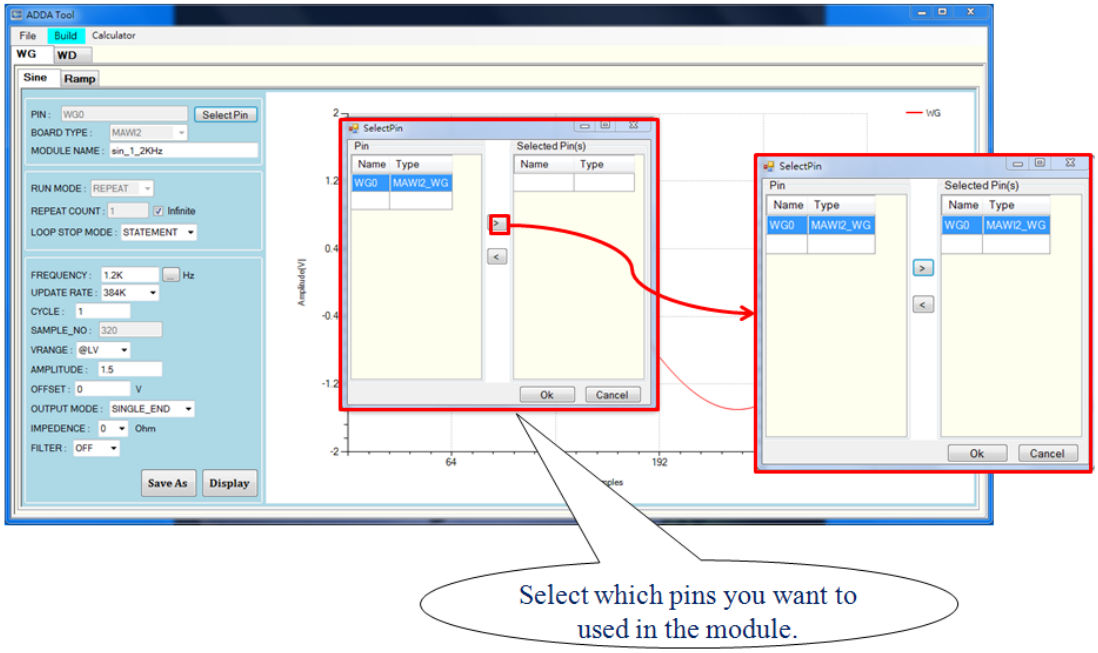
Load Setup: Loading the saved WG or WD setting file.

Import WD Data: Loading the WD capture result data for calculates SNR, THD, and DNL … etc.

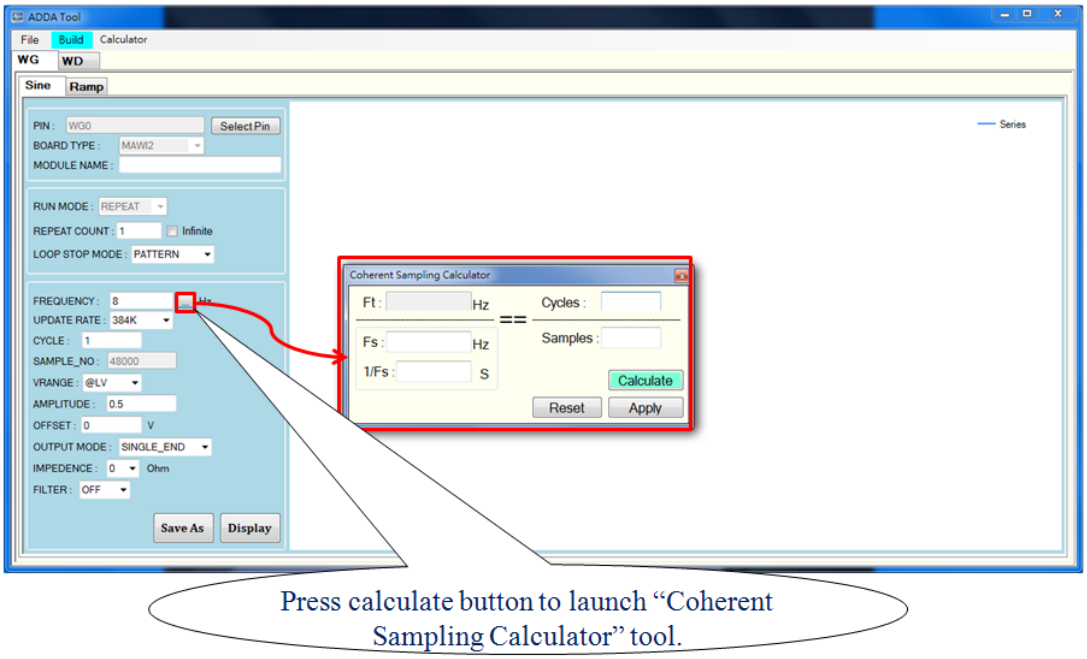
4. ADDA tool WG program example

- Step 1 : Select DEC file

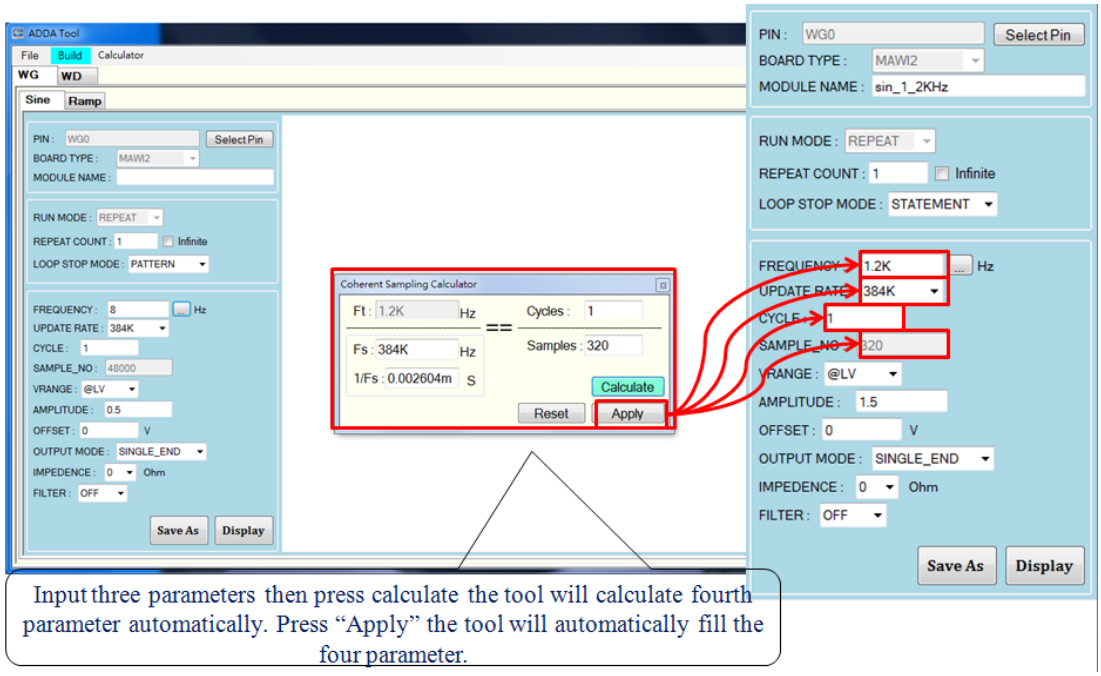




- Step 2 : Setting WG waveform



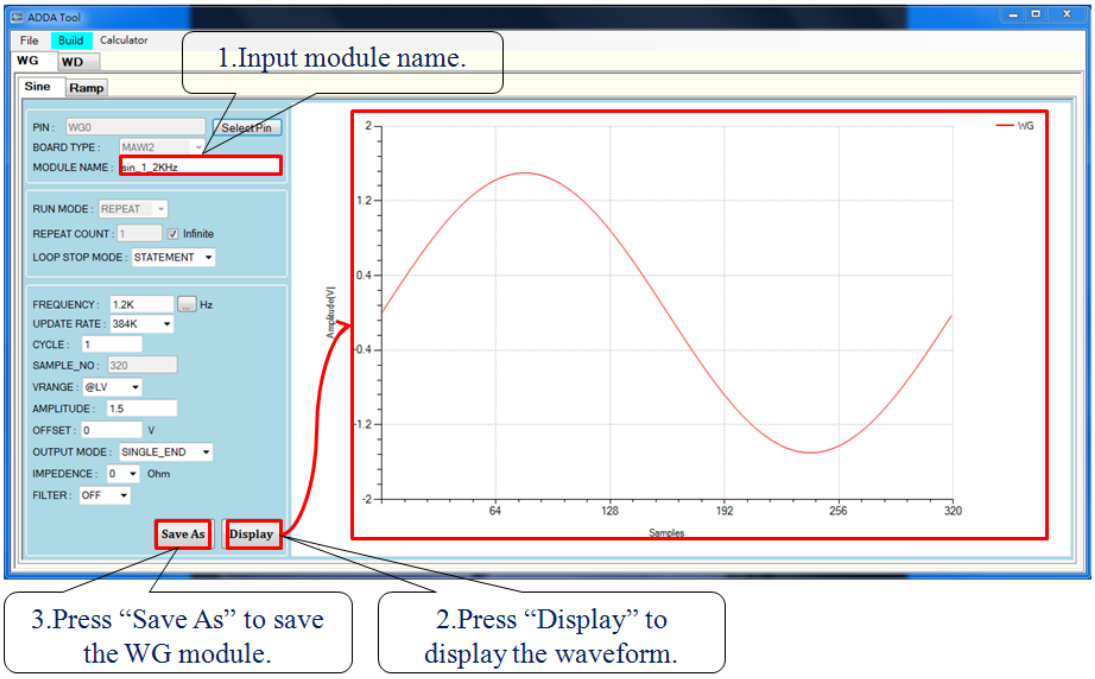
Press calculate button to launch "Coherent Sampling Calculator" tool.



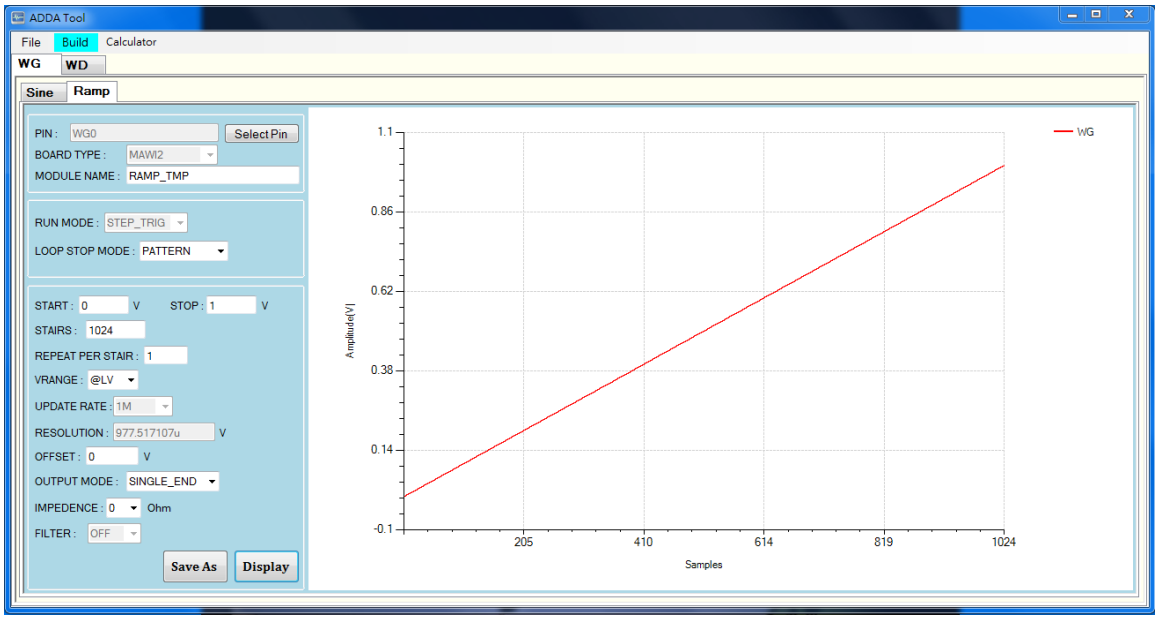
Input three parameters then press calculate the tool will calculate fourth parameter automatically. Press "Apply" the tool will automatically fill the four parameter.

- Step 3 : Save WG module

Sine wave

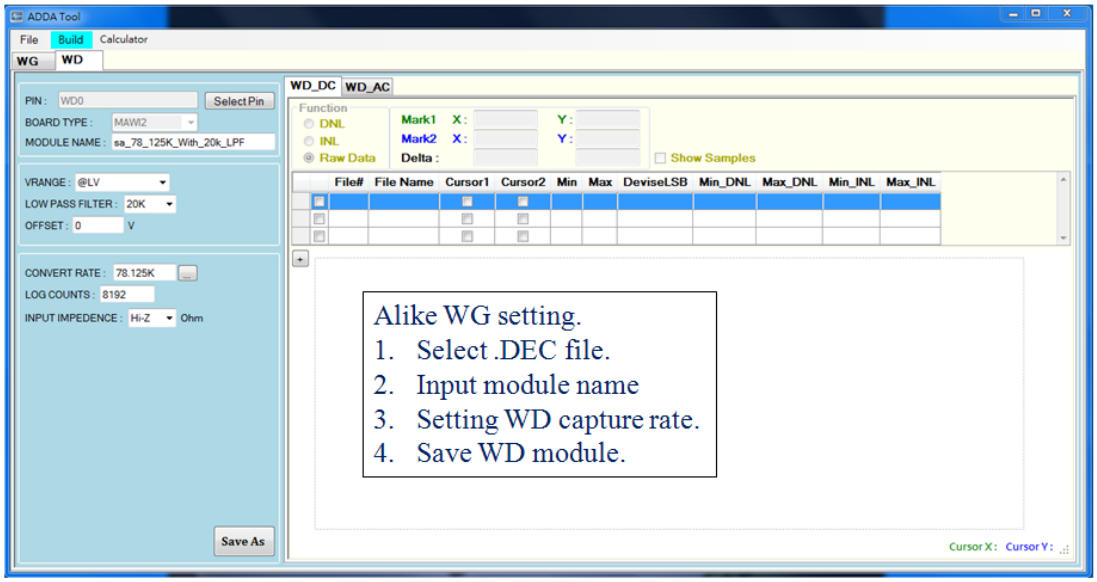


Ramp wave



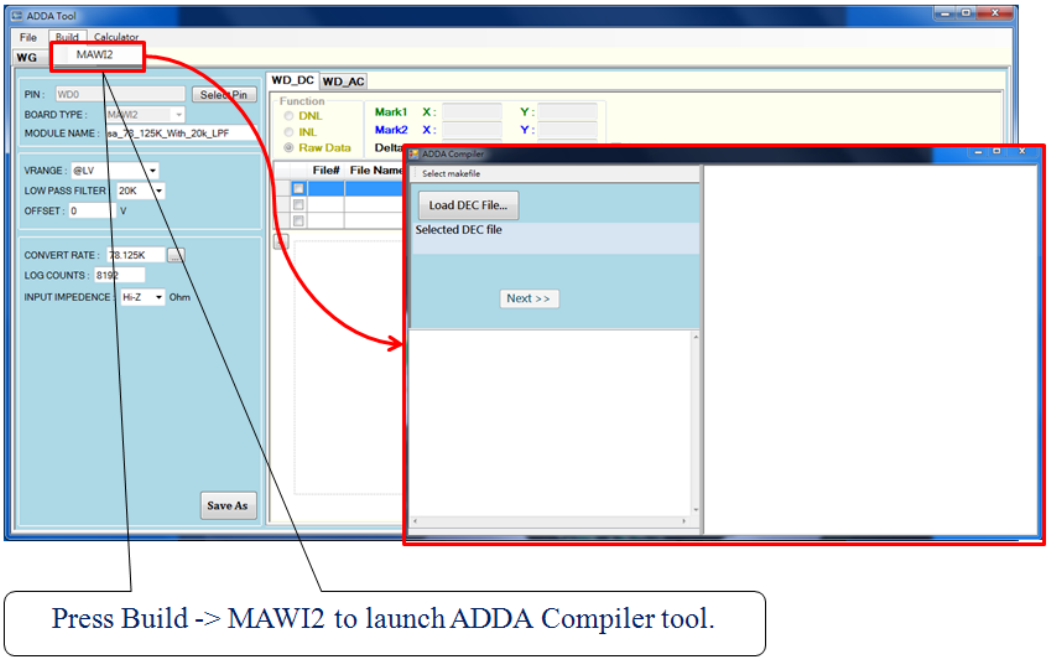
5. ADDA tool WD program example

- Setting WD waveform

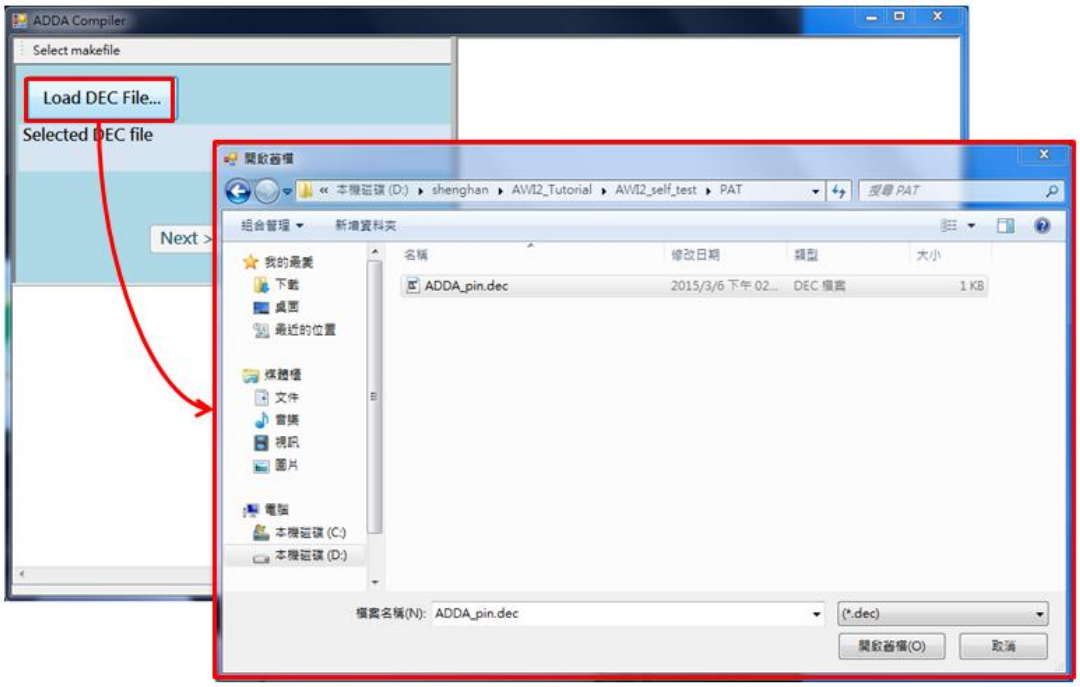


6. ADDA complier

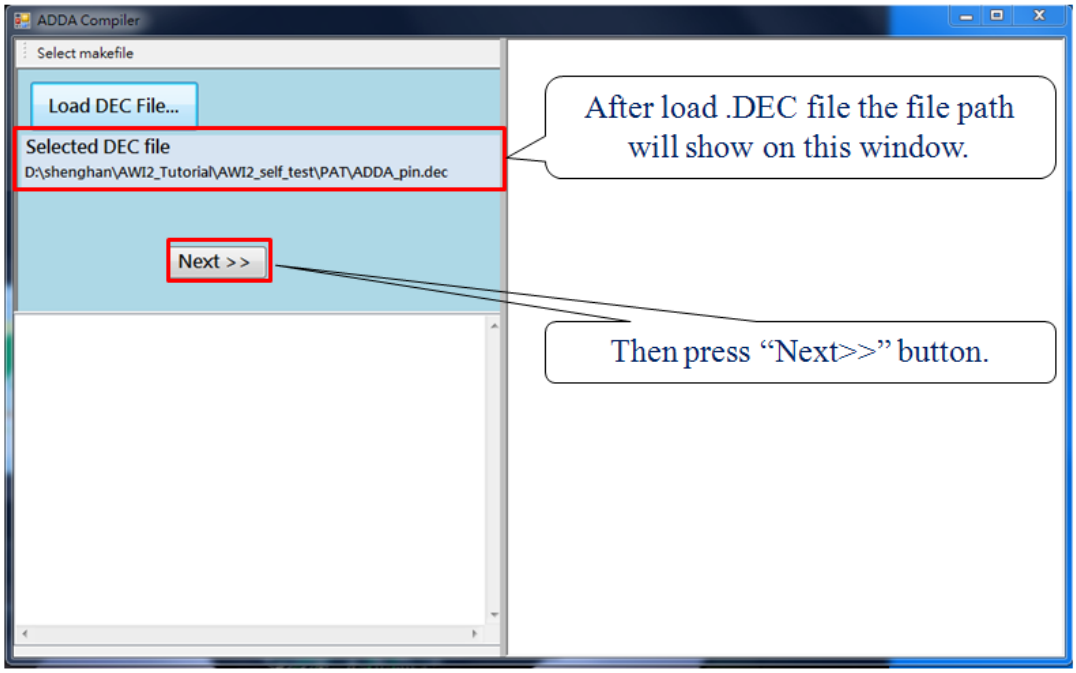
- Launch ADDA complier



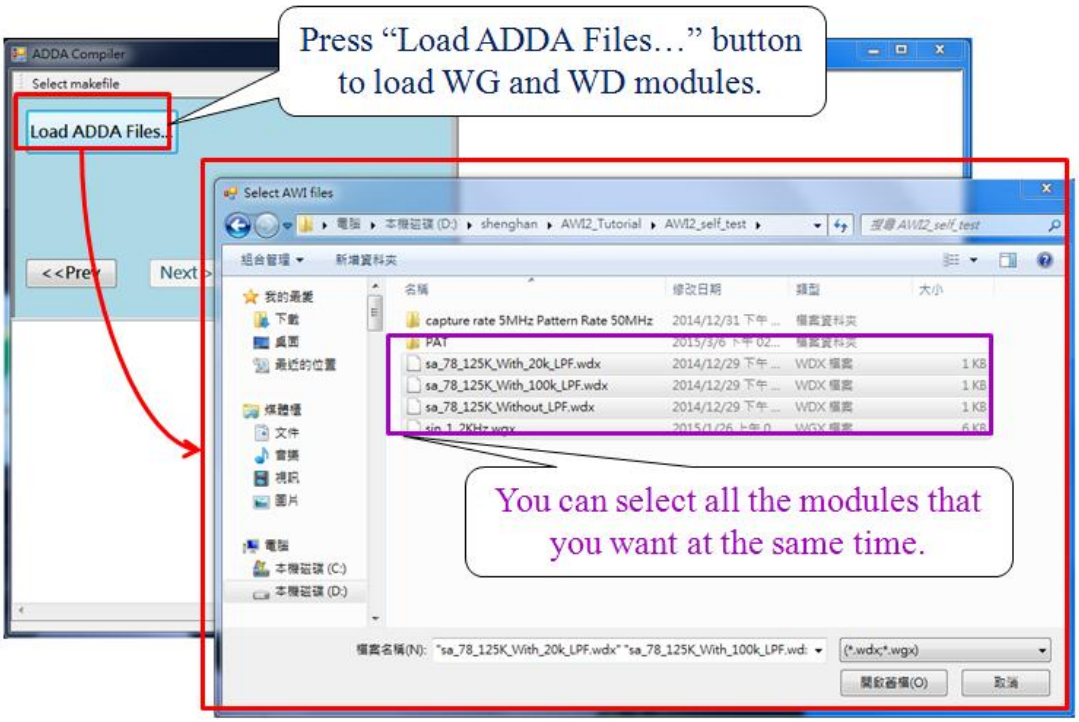
- Load DEC file

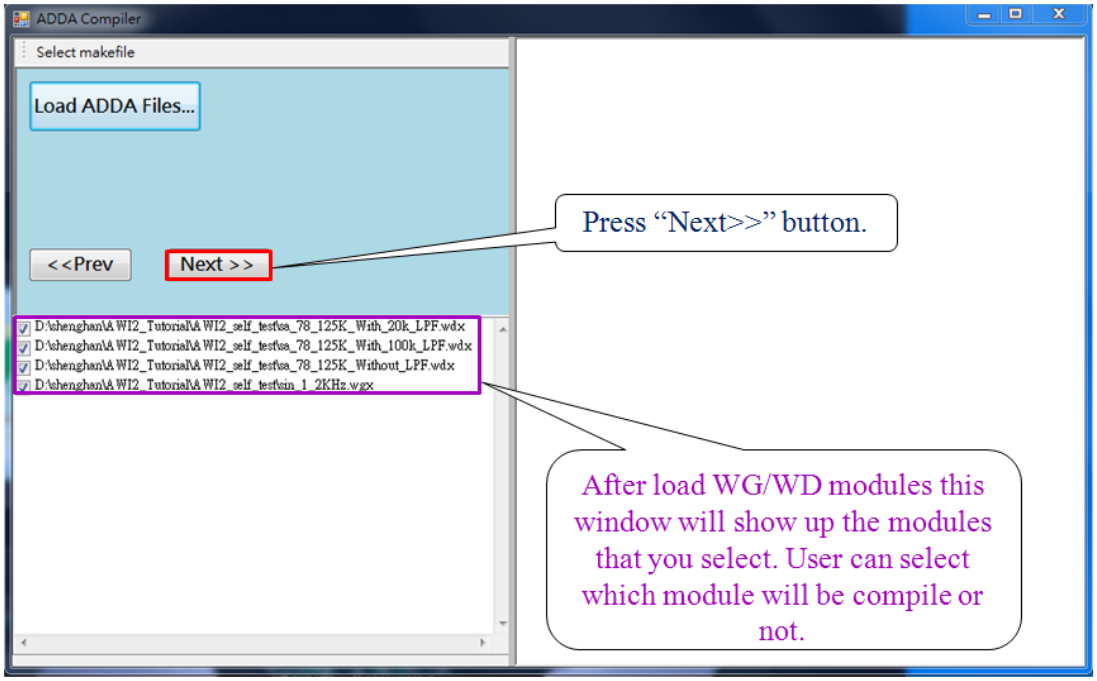


Press "Load DEC File…" button to load .DEC file.

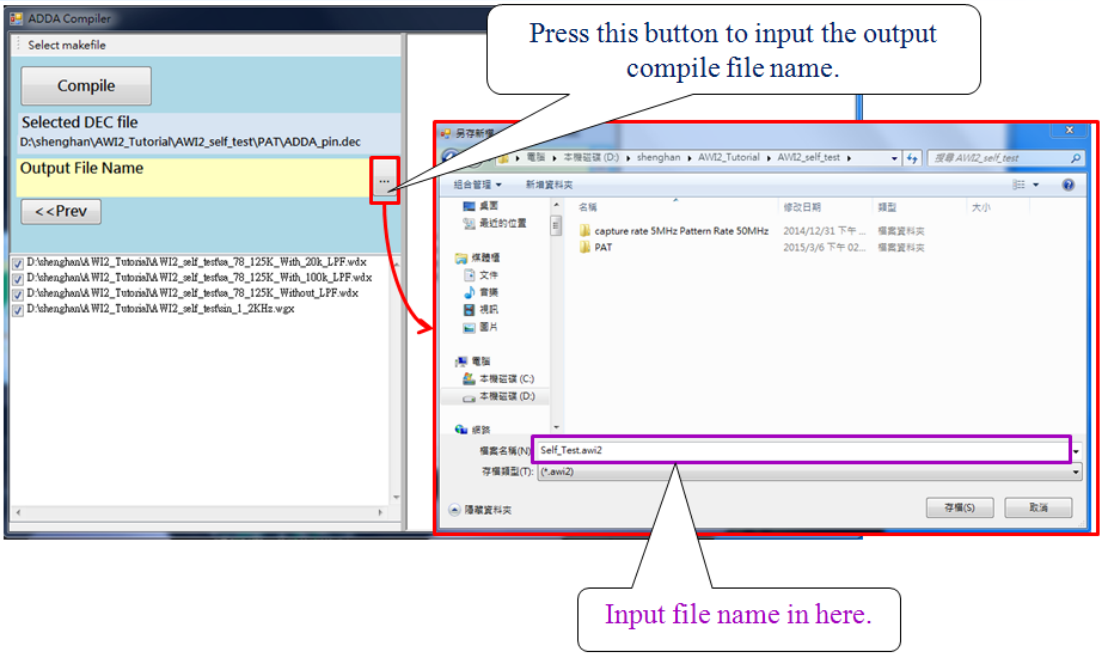


- Load WG/WD modules

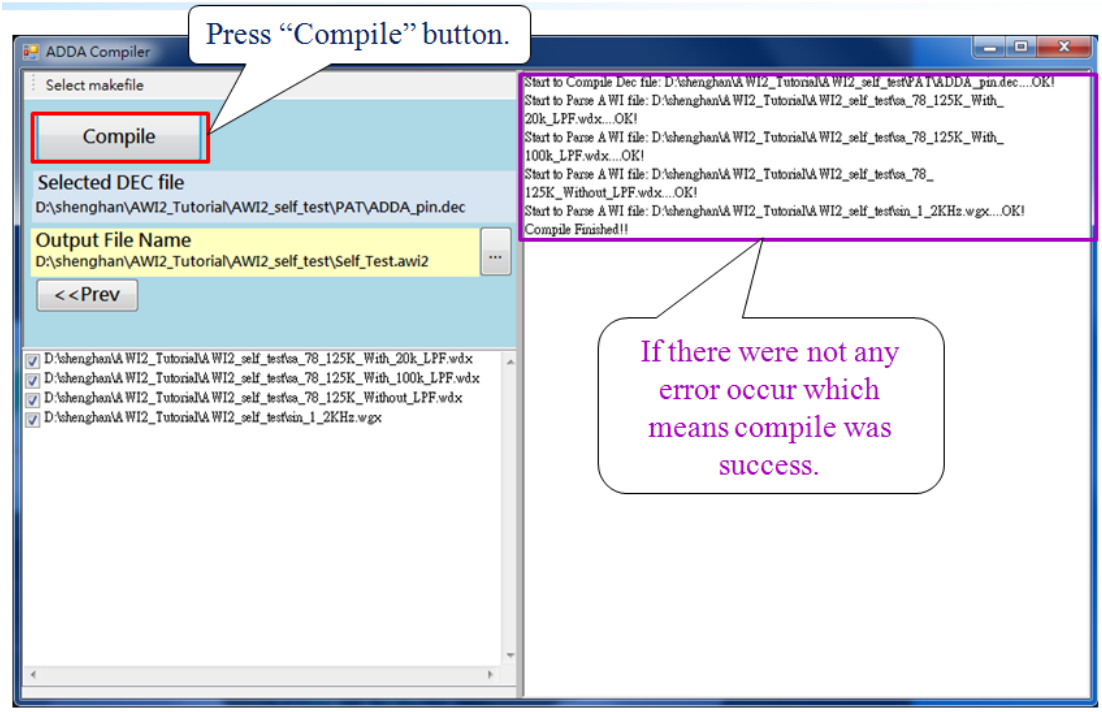




- Input complier output file name



- Complier



7. DEC file coding

- AWI2 WG/WD pin type

```
PIN_LIST(AWI2_4site )
{
    WG0          = 392 : 394 : 396 : 398   =   1   = MAWI2_WG;
    WD0          = 384 : 386 : 388 : 390   =   2   = MAWI2_WD;

    Test_pin   = 0    : 16   : 32   : 48    = 3    = IO;
}
```

8. PLN file coding

- START_UP

  Load the waveform generator pattern file into the hardware pattern memories.It is alike “LOAD_PAT” for pattern test.

```
START_UP()
{
  …
  LOAD_ADDA_WAVEFORM(MAWI2,"../Self_Test.awi2");
  LOAD_PAT("./PAT/ADDA_pat.ppo");
}
```

- LOAD_ADDA_WAVEFORM

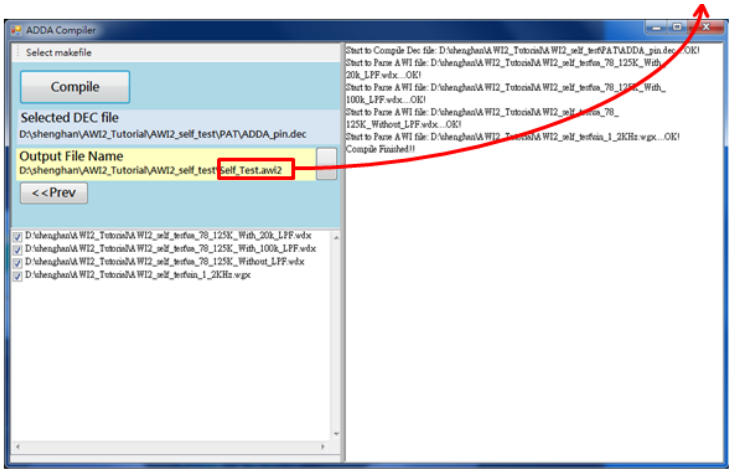
LOAD_ADDA_WAVEFORM(Board type, File_Name) ;

Board type: MAWI2

File Name: AWG pattern file name. It was the compile result file from the ADDA compile tool.

Example:

```
LOAD_ADDA_WAVEFORM(MAWI2,"../Self_Test.awi2");
```



9. WG/WD self test by low speed trigger

```
WD_SINE_1_2K_sa_78_125K_With_100k_LPF_LS_Trig()
{
    USE_WG_WAVEFORM(MAWI2,sin_1_2KHz,2mS);
    WG_RELAY_ON(MAWI2,WG0,WG,GND,5mS);
    USE_WD_WAVEFORM(MAWI2,sa_78_125K_With_100k_LPF,5mS);
    WD_RELAY_ON(MAWI2,WD0,WD,GND,5mS);
    WG_LS_TRIG(MAWI2,WG0,IMM,ON,100mS);
    WD_LS_TRIG(MAWI2,WD0,IMM,ON,10mS);

    READ_WD_DATA_MS(MAWI2,WD0,sa_78_125K_With_100k_LPF,data_tmp,8
    192);
    SOCKET_INC
    {
            ....
            AWI_DATA_AC_ARITH_EX(SNR_cal, 3125, THD_V, SNR_V, SND_V,
      SFDR_V,THD_n_V,DFT);
            JUDGE_VARIABLE (SOCKET, SNR_V, 90, 100 , "SNR" );
    }
    WD_LS_TRIG(MAWI2,WD0,IMM,OFF,5mS);
    WG_LS_TRIG(MAWI2,WG0,IMM,OFF,5mS);
    WD_RELAY_OFF(MAWI2,WD0,ALL,ALL,5mS);
    WG_RELAY_OFF(MAWI2,WG0,ALL,ALL,5mS);
}
```

- USE_WG_WAVEFORM

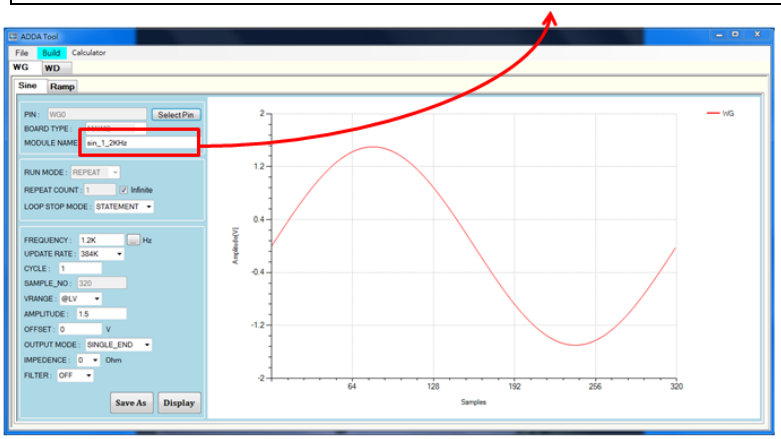
USE_WG_WAVEFORM(Board type, WG module name, Delay_time) ;

Board type: MAWI2

WG module name: WG pattern file name. It was WG module name setting in the ADDA tool.

Example:

USE_WG_WAVEFORM(MAWI2,sin_1_2KHz,2mS);



- WG_RELAY_ON

WG_RELAY_ON(Board type, Pin_name, P_Relay_type, N_Relay_type, Delay_time) ;

Board type: MAWI2

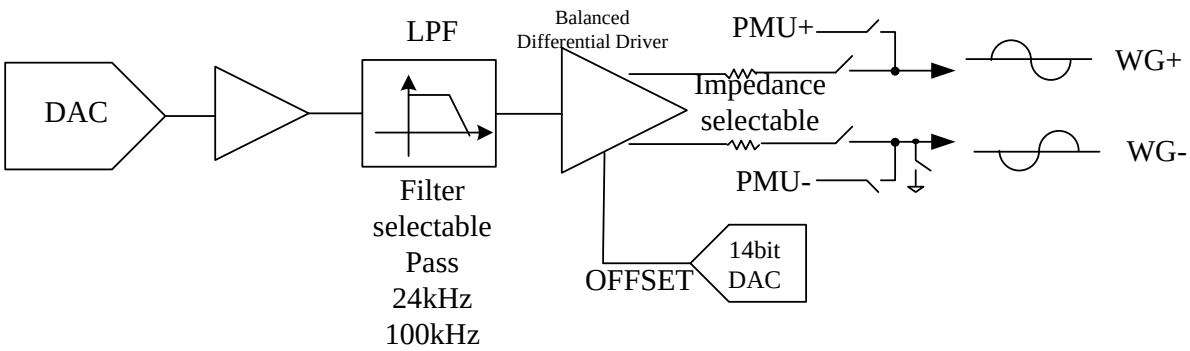
Pin_name: The pin name edit in the .DEC file.

P_Relay_type: PMU | WG

N_Relay_type: PMU | WG | GND

Example:

WG_RELAY_ON(MAWI2,WG0,WG,GND,5mS);



- USE_WD_WAVEFORM

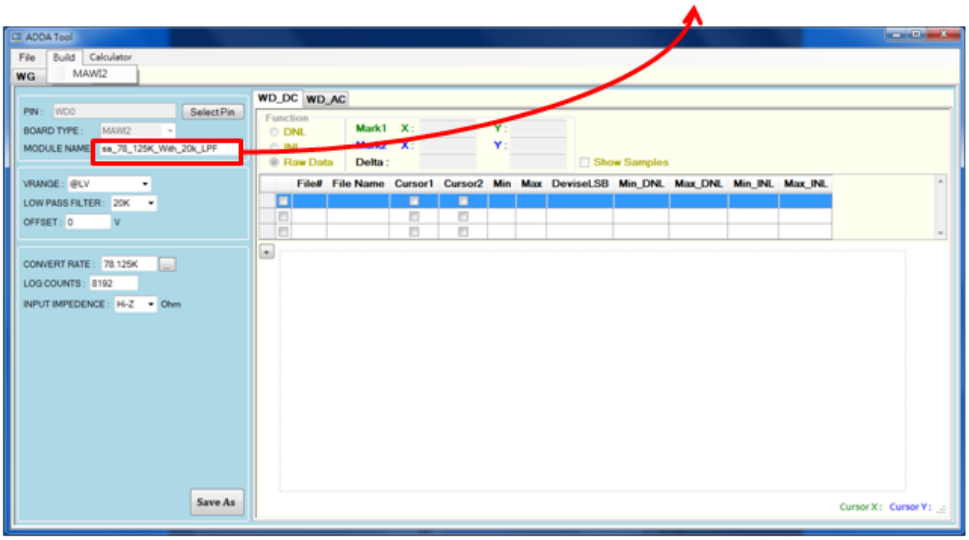
USE_WD_WAVEFORM(Board type, WD module name, Delay_time) ;

Board type: MAWI2

WD module name: WD pattern file name. It was WD module name setting in the ADDA tool.

Example:

USE_WD_WAVEFORM(MAWI2,sa_78_125K_With_100k_LPF,5mS);



- WD_RELAY_ON

WD_RELAY_ON(Board type, Pin_name, P_Relay_type, N_Relay_type, Delay_time) ;

Board type: MAWI2

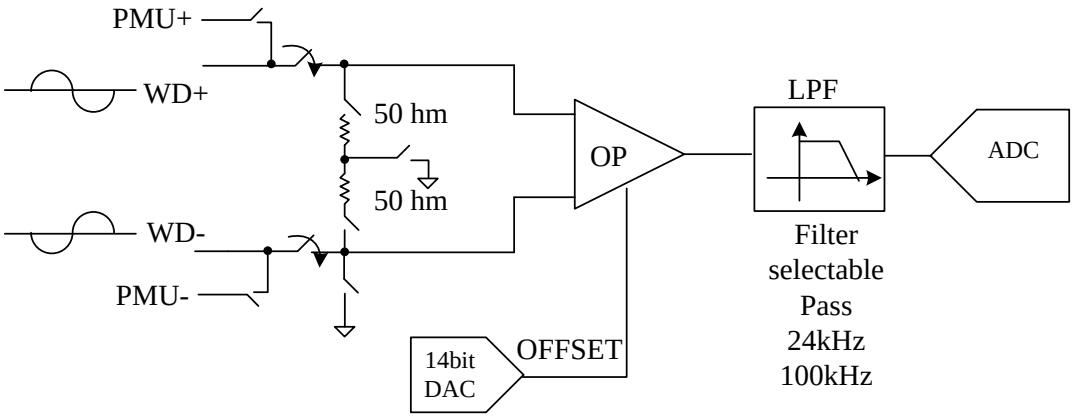
Pin_name: The pin name edit in the .DEC file.

P_Relay_type: PMU | WD

N_Relay_type: PMU | WD | GND

Example:

WD_RELAY_ON(MAWI2,WD0,WD,GND,5mS);



- WG_LS_TRIG

  This commend will trigger WG start to output waveform immediately without pattern trigger.

WG_LS_TRIG(Board type, Pin_name, Trig_source, Mode, Delay_time) ;

Board type: MAWI2

Pin_name: The pin name edit in the .DEC file.

Trig_source: IMM | EXT

Mode: ON | OFF

Example:

```
WG_LS_TRIG(MAWI2,WG0,IMM,ON,100mS);
```

- WD_LS_TRIG

  This commend will trigger WD start to output waveform immediately without pattern trigger.

WD_LS_TRIG(Board type, Pin_name, Trig_source, Mode, Delay_time) ;

Board type: MAWI2

Pin_name: The pin name edit in the .DEC file.

Trig_source: IMM | EXT

Mode: ON | OFF

Example:

```
WD_LS_TRIG(MAWI2,WD0,IMM,ON,10mS);
```

- READ_WD_DATA_MS

READ_WD_DATA_MS(Board type, Pin_name, wd_module_name, dgt_data, Log_Cnt) ;

Board type: MAWI2

Pin_name: The pin name edit in the .DEC file.

Wd_module_name: WD pattern file name. It was WD module name setting in the ADDA tool.

Dgt_data:

Type: CUSTOM_3DIM (3-dimension array)

CUSTOM_3DIM CRAFT auto-allocate array size [site_no][pin_count][ave_cnt]

Log_Cnt: The data of count will be readed.

Example:

```
CUSTOM_3DIM data_tmp;
READ_WD_DATA_MS(MAWI2,WD0,sa_78_125K_With_100k_LPF,data_tmp,8192);
```

- AWI_DATA_AC_ARITH_EX

  This function is designed to calculate WD dynamic characteric data specially for None 2^**n**.

AWI_DATA_AC_ARITH_EX(T_data, point_count, THD_value, SNR_value, SND_value, SFDR_value,THD_n_value, transfer_mode) ;

T_data: Time domain data

Point_count: effective point number, 2**^n use FFT**

, None 2^**n must use DFT mode**

THD_value, SNR_value, SND_value, SFDR_value, THD_n_value: calculate return value

Transfer_mode: FFT

, DFT (None 2^**n point count must assign this mode)**

Example:

```
AWI_ DATA_AC_ARITH_EX(SNR_cal, 3125, THD_V, SNR_V, SND_V,
SFDR_V,THD_n_V,DFT);
```

10. WG/WD self test by high speed trigger

```
WD_SINE_1_2K_sa_78_125K_With_100k_LPF_LS_Trig()
{
    USE_WG_WAVEFORM(MAWI2,sin_1_2KHz,2mS);
    WG_RELAY_ON(MAWI2,WG0,WG,GND,5mS);
    USE_WD_WAVEFORM(MAWI2,sa_78_125K_With_100k_LPF,5mS);
    WD_RELAY_ON(MAWI2,WD0,WD,GND,5mS);
    JUDGE_PAT(ADDA_test,__ADDA_test);

    READ_WD_DATA_MS(MAWI2,WD0,sa_78_125K_With_100k_LPF,data_tmp,8
    192);
    SOCKET_INC
    {
        ….
        AWI_DATA_AC_ARITH_EX(SNR_cal, 3125, THD_V, SNR_V, SND_V,
```

```
        SFDR_V,THD_n_V,DFT);
            JUDGE_VARIABLE (SOCKET, SNR_V, 90, 100 , "SNR" );
    }
    WD_LS_TRIG(MAWI2,WD0,IMM,OFF,5mS);
    WG_LS_TRIG(MAWI2,WG0,IMM,OFF,5mS);
    WD_RELAY_OFF(MAWI2,WD0,ALL,ALL,5mS);
    WG_RELAY_OFF(MAWI2,WG0,ALL,ALL,5mS);
}
```

11. ADDA test pattern

Use WG0 (pin name define in .DEC file) as u-instruction to trigger WG start output waveform

Use WD0 (pin name define in .DEC file) as u-instruction to trigger WD start capture waveform

```
SET_DEC_FILE "./ADDA_pin.dec"
HEADER     Test_pin;
SPM_PATTERN(ADDA_test)
{
  * X* TS1 ;
  * X* RPT 100 ;
  * X* WG0 ;
  * X* RPT 1000 ;
  * X* WD0 ;
  * X* RPT 10000000 ;
  * X* ;
  }
```

CHROMA ATE INC.

致茂電子股份有限公司

66 Huaya 1st Road, Guishan,

Taoyuan 33383, Taiwan

台灣桃園市 33383 龜山區

華亞一路 66 號

T +886-3-327-9999

F +886-3-327-8898

Mail: info@chromaate.com

http://www.chromaate.com

Copyright by CHROMA ATE INC. All Rights Reserved.
All other trade names referenced are the properties of their respective companies.