435/Installation ManualInstallation Manual

UNI92K-4DMUNI92K-4DMW

|  |
| --- |
| ***UNI N201*** |
| ***Software Architecture*** |
| Rev. 0.02 |



Copyright ©UNITEST Corporation

**Table of Contents**

[1. Introduction 4](#_Toc23181759)

[1.1. Purpose 4](#_Toc23181760)

[1.2. Reference 4](#_Toc23181763)

[1.3. Overview 4](#_Toc23181764)

[2. Software Architecture 5](#_Toc23181765)

[2.1. UniSDK Block Diagram 5](#_Toc23181766)

[2.2. TEST CORE ENGINE Block Diagram 5](#_Toc23181766)

[2.3. GUI Block Diagram](#_Toc23181766) 6

[2.4. USO Block Diagram 7](#_Toc23181766)

[2.5. Compiler Block Diagram 7](#_Toc23181766)

[2.6. SSO Block Diagram 7](#_Toc23181766)

[2.7. PCAL Block Diagram 8](#_Toc23181766)

[2.7.1. Overall Flow 8](#_Toc23181767)

[2.7.2. PCAL GAIL / OFFSET Calculate 8](#_Toc23181767)

[2.7.3. PCAL Result 8](#_Toc23181767)

[2.8. DIAG Block Diagram 9](#_Toc23181766)

[2.8.1. Overall Flow](#_Toc23181767) 9

[2.8.2. DIAG ITEM Check](#_Toc23181767) 9

[2.8.2. DIAG RESULT](#_Toc23181767) 9

[2.9. POWER Sequence 10](#_Toc23181766)

[2.9.1. USER Set VS](#_Toc23181767) 10

[2.9.2. POWER ON or OFF](#_Toc23181767) 10

[2.10. POWER MONITORING 10](#_Toc23181766)

[2.10.1. Overall Flow 10](#_Toc23181767)

[2.10.2. POWER MONITORING Action 11](#_Toc23181767)

[2.11. FLASH MEMORY 12](#_Toc23181766)

[2.12. FAIL MEMORY 15](#_Toc23181766)

[2.13. CHAMBER CONTROL 16](#_Toc23181766)

[2.14. CHAMBER EMULATOR 21](#_Toc23181766)

[2.15. SOCKET COMPILER 24](#_Toc23181766)

[3. Supporting Information 27](#_Toc23181777)

[3.1. Software Require Architecture 27](#_Toc23181778)

[3.1.1. Revision History 27](#_Toc23181779)

[3.1.2. 참고 27](#_Toc23181780)

# Introduction

본 문서는 UNIN201의 구조를 설명하는 SRA(Software Requirements Architecture)이다.

## Purpose

UNIN201의 UniSDK에 대한 내용을 Black Diagram으로 표현한다.

문서의 사용자는 UNITEST UNIN201 개발자(SW, FW, HW, TE)를 대상으로 작성한다.

SW 개발팀은 본 SRA에 따라 기능 설계 및 구현을 한다.

UNIN201은 설계서에 맞도록 개발 한다.

## Reference

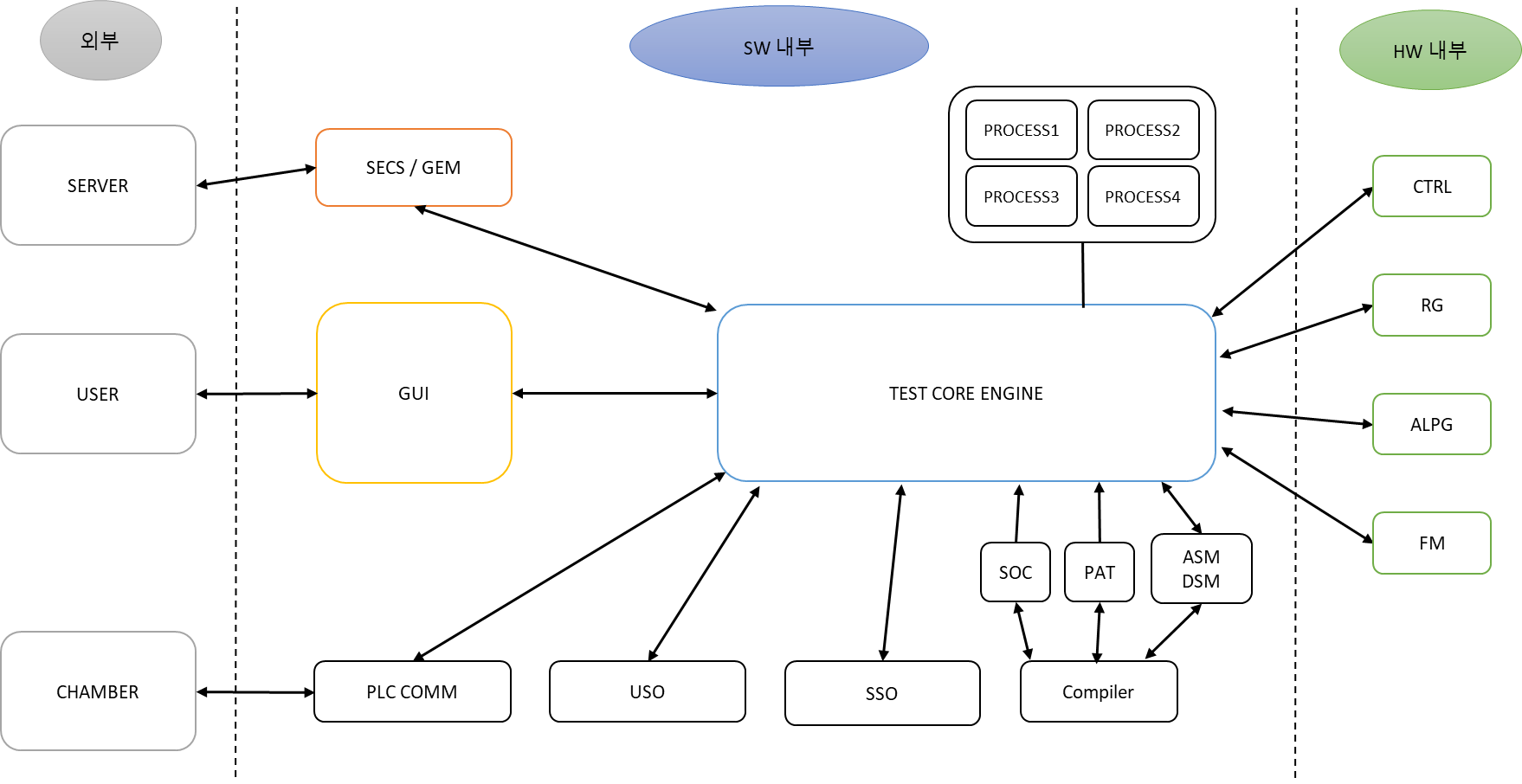
## Overview

본 문서는 다음과 같이 구성되었다.

2장에서는 UNIN201의 UNISDK의 전체적인 Architecture와, 그것이 세부적으로 어떻게 구성 되어 있는지 살펴 본다.

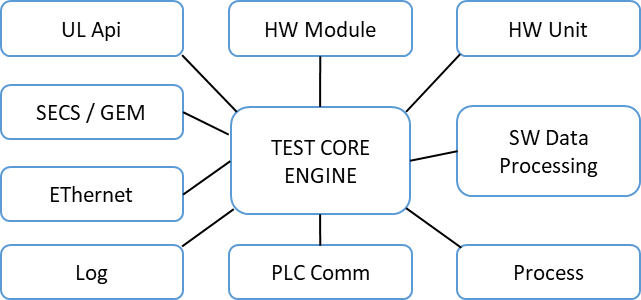
# Software Architecture

## UniSDK Block Diagram



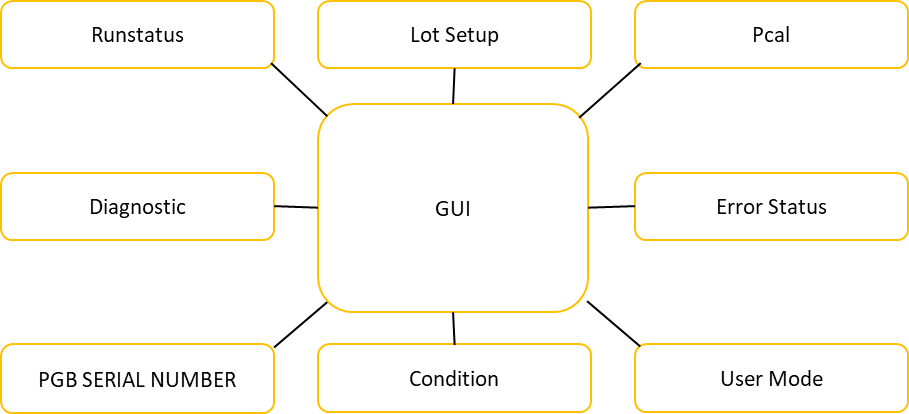
* UniSDK Block Diagram은 크게 3가지로 분류한다.
  + 외부에서부터 UniSDK로의 입력 요인을 외부, UniSDK가 제공하고 있는 부분을 SW 내부, UniSDK에서 명령을 전달하는 HW 내부로 나누어진다.
* 외부로는 사용자가 사용하는 SERVER, 직접 장비를 운영하는 사용자, 온도 및 기구 제어를 하는 CHAMBER로 나뉘어 진다.
* SW 내부는 TEST CORE ENGINE이 중심이며, MASTER 1개, SLAVE 4개의 Process를 가진다.
* TEST CORE ENGINE은 GUI, USO, SSO에게 제어 및 data 조회 기능을 제공한다.
* TEST CORE ENGINE은 통신과 관련하여 SECS / GEM, PLC COMM, HW 내부와의 통신 기능을 지원한다.
* TEST CORE ENGINE은 Compiler에서 만들어진 Socket, Pattern file 등을 처리한다.

## TEST CORE ENGINE Block Diagram



* TEST CORE ENGINE의 구조는 위의 그림과 같다.
* Ethernet은 RG, ALPG, FM, CTRL과 통신하는 기능이다.
* HW Unit은 Ethernet의 상위 단계로 Chip의 종류를 선택하고, Register 영역에 Data를 Read / Write 하는 기능들의 집합이다.
* HW Module는 HW Unit의 상위 단계로 HW Unit와 data를 주고 받거나, 받은 data를 가공는 기능들의 집합이다
* SW Data Processing은 HW 동작과는 별개로 Software 적으로 data를 가공하는 기능들의 집합이다.
* UL Api는 UniSDK에서 사용자에게 제공하는 api들이며, TEST CORE ENGINE의 내부 Driver와 연결 되어 있다..
* Log는 User에게 제공하기 위한 정보나 동작과 관련된 내용을 기록하는 기능들의 집합이다.
* PLC Comm은 Test Core Engine이 PLC의 Protocol에 Read / Write 하는 기능들의 집합이다.
* Test Core Engine의 Process는 MASTER 1개, SLAVE 4개로 구성된다.
* SECS / GEM을 지원하며, 고객 환경에 맞춰 구현한다.

## GUI Block Diagram



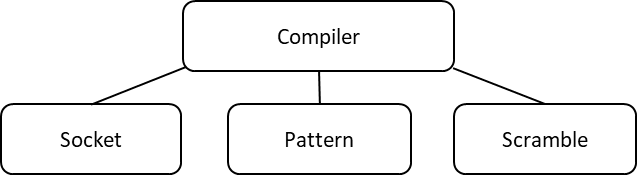
* Runstatus는 UniSDK의 메인 GUI창으로 Test와 관련된 동작과 Test Result와 관련된 내용들을 사용자에게 보여준다.
* Lot Setup은 사용자가 Lot Name, Operator ID, Test Pgm path, Test 진행할 PGB 선택 등을 지원한다.
* Pcal은 Pcal의 Item condition을 설정할 수 있고, PGB, Item별 Result를 보여준다.
* Diagnostic은 Diagnostic Item Condition을 설정할 수 있고, PGB, Item별 Result를 보여준다.
* Error Status는 Error가 발생했거나, 오늘 발생했던 Error, Error List 등을 보여준다.
* PGB SERIAL NUMBER는 PGB 내 FLASH 공간에 Serial number를 READ / WRITE 할 수 있는 창이다.
* Condition은 PGB MASK, DUT MASK, LOG Level, Monitoring 정보 등을 설정 할 수 있는 창이다.
* User Mode는 Engineer Mode, Operator Mode를 설정할 수 있는 창이다.

## USO Block Diagram



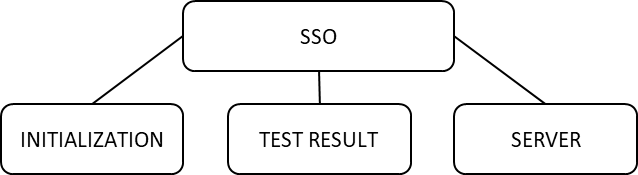
* USO는 Utility 기능이 share library file 형태로 만들어진 것을 말한다.
  + File은 .uso file로 만들어진다.
* PCAL은 PCAL 동작이 코드로 구현 되어 있으며, share library 파일 형태이다.
* DIAG는 DIAG 동작이 코드로 구현 되어 있으며, share library 파일 형태이다.

## Compiler Block Diagram



* Socket은 DUT, POWER, IO, PIN, GROUP 정보 등이 담겨 있는 socket file을 compile 하여 .soc file을 만든다.
  + Test Program에서 사용자가 원하는 socket file을 load하여 Unisdk에게 socket file 내용을 전달한다.
* Pattern은 ALPG에게 명령어를 전달하는 pattern file을 compile 하여 .pat file을 만든다.
  + Test Program에서 사용자가 원하는 pattern file을 load하면 Unisdk에서 pat file 내용을 alpg에게 전달한다.

## SSO Block Diagram



* SSO는 여러 종류의 파일을 share library file 형태로 만들어진 것을 말한다.
  + 크게 INITIALIZATION, TEST RESULT, SERVER로 나뉠 수 있다.
* INITIALIZATION은 UNISDK 실행하면서 HW 초기화를 진행하는 내용과 HW 초기화를 진행하지 않는 sso file들을 말한다..
* TEST RESULT는 Pretest Start, End 시점, Test Start, End 시점에서 만들어지는 result file을 만드는 sso file들이다.
* SERVER는 UniSDK 실행 중 사용자가 사용하는 Server에서 필요로 하는 정보 들에 대해 코드로 구현하여 sso file로 만들어진 것들이다.

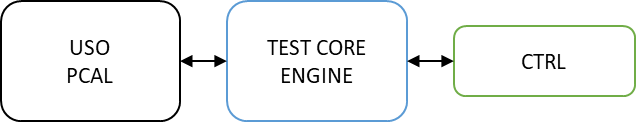
## PCAL Block Diagram

### Overall Flow



* User는 GUI PCAL에서 Condition을 설정 하고 Start를 클릭한다.
* GUI PCAL은 Start 신호를 받고 USO PCAL을 실행한다.
* USO PCAL은 내부 동작을 진행하며, Item에 맞도록 TEST CORE ENGINE을 제어한다.
* TEST CORE ENGINE은 CTRL Register에 Data를 전달하거나, 읽어온다.
* CTRL은 TEST CORE ENGINE으로부터 READ 일 경우 Data를 넘겨주고, WRITE 일 경우 해당 Data에 맞게 동작한다.

### PCAL GAIN / OFFSET Calculate



### USO PCAL Voltage Write.

### TCE는 CTRL에게 전달.

### CTRL Voltage 동작.

### USO PCAL Voltage Read.

### TCE는 CTRL에게 전달.

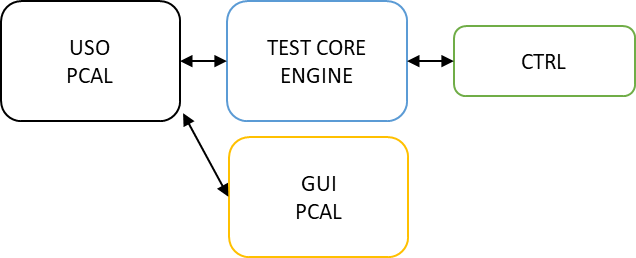
### CTRL Voltage 값 Register에 Write.

### TCE는 CTRL Register Read.

### USO PCAL는 TCE로부터 Voltage 값 Read.

### Read 된 Voltage 값으로부터 GAIN / OFFSET 값 계산.

### PCAL RESULT



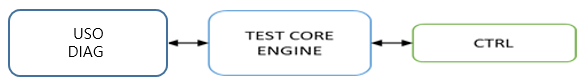
* Voltage는 2.7.2와 같은 방식으로 동작한다.
  + 단, Voltage는 GAIN과 OFFSET 값이 적용된 값으로 전달한다.
* 사용자가 설정한 Tolerance가 벗어나면 FAIL, 아니라면 PASS처리한다.
* PASS / FAIL을 TEST CORE ENGINE의 LOG와, GUI PCAL에 전달한다.

## DIAG Block Diagram

### Overall Flow

* User는 GUI DIAG에서 Condition을 설정하고 Start를 클릭한다.
* GUI DIAG은 Start 신호를 받고 USO DIAG을 실행한다.
* USO DIAG은 내부 동작을 진행하며, Item에 맞도록 TEST CORE ENGINE을 제어한다.
* TEST CORE ENGINE은 CTRL Register에 Data를 전달하거나, 읽어온다.
* CTRL은 TEST CORE ENGINE으로부터 READ 일 경우 Data를 넘겨주고, WRITE 일 경우 해당 Data에 맞게 동작한다.

### DIAG ITEM Check



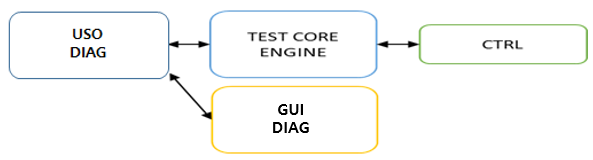
### USO DIAG Condition에 설정된 item에 따라 진행.

### TCE는 Condition의 해당된 DIAG Item에 대한 REGISTER영역에 접근.

### CTRL는 TCE가 접근하고자한 REGISTER영역에 대한 READ VALUE를 TCE에 전달.

### TCE는 받은 READ VALUE를 통하여 각 DIAG ITEM의 검증 사항에 맞혀서 진행함.

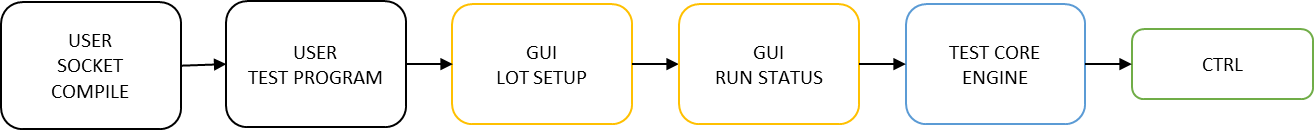
### DIAG RESULT



* 사용자가 설정한 DIAG ITEM에 대해 검증을 진행한다.
* 각 DIAG ITEM의 PASS / FAIL을 TEST CORE ENGINE의 LOG와, GUI DIAG에 전달한다.

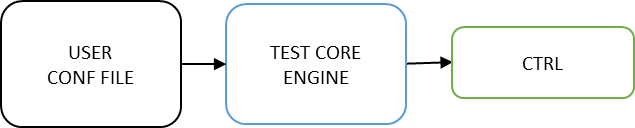
## Power Sequence

### USER Set VS



* User는 Socket File을 통해 Power Channel들을 정의한다.
* Test Program에서 Socket File에 정의된 Power Channel들 중 원하는 Power에 Voltage를 인가한다.
* GUI의 Lot Setup 창에서 Test Program을 Load한다.
* GUI의 Run Status 창에서 Test Program을 Start한다.
* Test Program은 내부 동작을 진행하며, Test Core Engine은 CTRL에 READ / WRTIE 한다.
* CTRL은 TEST CORE ENGINE으로부터 READ 일 경우 Data를 넘겨주고, WRITE 일 경우 해당 Data에 맞게 동작한다.

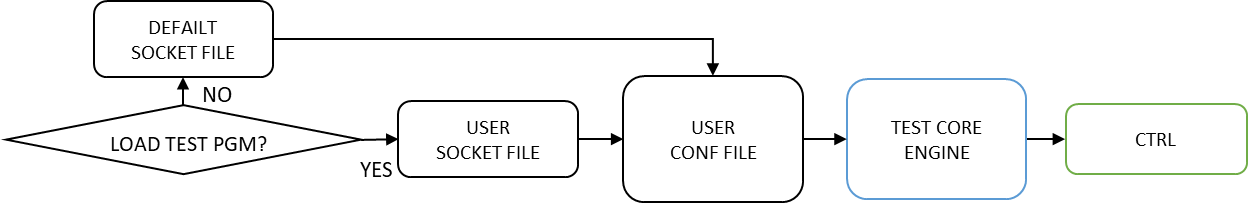
### POWER ON or OFF



* User는 Conf File에 TEST 시점마다 설정하고 싶은 Power channel을 기입한다.
* TEST CORE ENGINE은 User가 기입한 Conf File을 load하여, User가 기입한 순서에 맞게 동작한다.
* TEST CORE ENGINE은 Power channel에 관련되어 0V 는 Power off를 의미하며, 그 외 Voltage는 Power On을 의미한다.
* Test CORE ENGINE은 Power channel과 Voltage를 CTRL에게 전달한다.

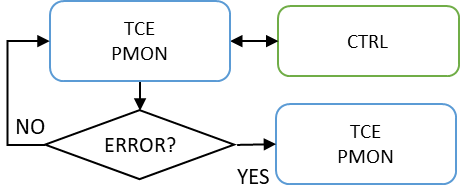
## POWER MONITORING

### Overall Flow



* Test Program이 Load 되어 있다면 User가 원하는 Socket File을 기준으로 동작하고, Load 되어 있지 않다면 Unisdk가 기본으로 가지고 있는 Socket File을 기준으로 동작한다.
  + 동작의 의미는 Socket File 내에 선언되는 Power Channel 들의 Name을 기준으로 동작하겠다는 뜻한다.
* User는 conf file을 통해 Power monitoring의 condition을 설정한다.
* TEST CORE ENGINE에서는 User가 설정한 conf file을 읽어 Power Monitoring 동작을 확인하고, Power Channel들에 대한 Tolerance를 CTRL에 Write한다.
* TEST CORE ENGINE은 주기적으로 CTRL을 읽어 FAIL이 발생한 Power Channel이 없는지 확인한다.

### POWER MONITORING Action



* TEST CORE ENGINE 내부의 Power monitoring을 실행한다.
* TCE PMON에서는 CTRL로부터 COMPARE 신호를 얻어 온다.
* TCE PMON에서는 COMPARE 신호에 ERROR가 있는지 판단한다.
  + ERROR가 있으면 해당 Channel과 관계된 Channel을 모두 MASK 한다.
  + ERROR가 없으면 다시 TCE PMON을 동작한다.

## FLASH MEMORY

* ALPG, RG, FM bit file 을 flash memory 에 write 한다.
* Flash memory size 증가로 인해서 사이즈에 맞게 수정 해야 한다.
* Register Map Page Select

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **ADDRESS[31:0]** | **DATA[7:0]** | **COMMENT** |
| ALPG FLASH PAGE | 0000\_0B80 | [7:0] ALPG PAGE | PAGE 0 ~ 6 default : 0 |
| RG FLASH PAGE | 0000\_0B81 | [7:0] RG PAGE | PAGE 0 ~ 6 default : 0 |
| FM FLASH PAGE | 0000\_0B82 | [7:0] FM PAGE | PAGE 0 ~ 2 default : 0 |
| RECONFIG SELECT | 0000\_0B83 | [3] : ALPG3 | CONFIG 하려는 ALPG 선택 ALL 은 0x0f |
| [2] : ALPG2 | 1 : config enable, 0 : config disable default : 0 |
| [1] : ALPG1 | 총 4개의 GROUP |
| [0] : ALPG0 |  |
| RECONFIG SELECT | 0000\_0B84 | [3] : RG3 | CONFIG 햐려는 RG 선택 ALL 은 0x0f |
| [2] : RG2 | 1 : config enable, 0 : config disable default : 0 |
| [1] : RG1 | 총 4개의 GROUP |
| [0] : RG0 |  |
| RECONFIG SELECT | 0000\_0B85 | [3] : FM3 | CONFIG 하려는 FM 선택 ALL 은 0x0f |
| [2] : FM2 | 1 : config enable, 0 : config disable default : 0 |
| [1] : FM1 | 총 4개의 GROUP |
| [0] : FM0 |  |
| Reconfig\_RUN | 0000\_0BFF | [2] : FM RUN | \* ALPG, RG, FM 은 동시에 RUN 실행 할 수 없다. |
| [1] : RG RUN |  |
| [0] : ALPG RUN |  |

[표] Register Map – Flash Address 변경

* **ul\_loadpg**
  + ALPG 는 7 Page (page : 0 ~ 6)
    - Address change
  + RG 는 7 Page (page : 0 ~ 5)
    - Address change
  + FM 은 4 Page (page : 0 ~ 3)
    - Address change
  + Reconfig RUN
    - Address change
* 사용자 프로그램 진행시 test time 단축을 위해 Flash load check 기능 추가해야 한다.
  + UNIGUI 실행시 loadpg 을 할것인 기능 추가
    - UNIGUI RUN -> CONF Data load -> loadpg (load flag true) -> conf file flash page load check -> loadpg (page load) -> reset -> version read -> conf file save (page and version)
    - UNIGUI RUN -> CONF Data load -> loadpg (load flag false) -> conf file data delete (page and version)

CONF DATA LOAD

(Flag, Page and Version)

UNIGUI START

ALPG/RG/FM LOAD

LOAD PG FLAG

VERSION READ

FLASH LOAD END

PAGE and Version SAVE

**TRUE**

**FALSE**

PAGE and Version Delete

* + 사용자 프로그램 실행시 loadpg 검사 기능
    - MainPgm RUN -> Product page & conf file page check : OK -> version read -> compare Version -> Function Test
    - MainPgm RUN -> Product page & conf file page check : NG -> loadpg -> reset -> version read -> conf file save (page and version) -> Function Test

SCRIPT START

ALPG/RG/FM LOAD

VERSION READ

FLASH LOAD END

PAGE and Version SAVE

Page Check

**FALSE**

Version Check

**TRUE**

**FALSE**

**TRUE**

* 사용자 프로그램 진행시 test time 단축을 위해 Flash Done Signal check 기능 추가해야 한다.
  + Loadpg 함수 안에서 Flash Done Signal 을 완료 될 때 까지 check 한다.
  + Done Signal Check 시간은 MAX 10초로 한다.
  + Polling 주기는 1초로 10초 동안 진행을 하며, 중간에 check complete 되면 break 로 빠져 나온다.
  + 설계시 개별 PGB 와 Global PGB 을 구분해서 설계 한다.

## FAIL MEMORY

* Device Test 후 read fail data 을 PGB의 “Fail Memory” 영역에서 read 한다.
* Fail Memory 은 “UNIN201\_Instruction\_ver1.2(191119).xlsx” 참조 한다.
  + 참조 자료



* Fail memory data 을 이용해서 RA 할 수 있도록 Text and Binary 파일을 생성 해야 한다. (Text은 사용자 확인 용도로 사용 하며, Binary은 RA 할 때 사용한다.)
* Fail memory data format 은 다음 표와 같다.

|  |  |  |
| --- | --- | --- |
| **NAME** | **MEMORY ASSIGN**  **[BIT]** | **SIZE BYTE [BIT]** |
| RESERVED | 0:15 | 2 [0:15] |
| X\_ADDR | 0:23 | 3 [0:23] |
| Y\_ADDR | 0:17 | 3 [0:23] |
| WRITE\_DATA\_H | 0:23 | 3 [0:23] |
| WRITE\_DATA\_L | 0:23 | 3 [0:23] |
| READ\_DATA\_H | 0:96 | 12 [0:95] |
| READ\_DATA\_L | 0:96 | 12 [0:95] |
| CS | 0:15 | 10 [0:79] |
| Z\_ADDR | 0:7 | 1 [0:7] |
| PATTERN\_ADDR | 0:11 | 2 [0:15] |
| TOTAL |  | 51 |

[표] Fail Memory Data Format

* Text Format 은 사용자 요구 조건에 맞춰서 저장한다.

Binary Format 은 [표] Fail Memory Data Format 의 structure data 을 save 및 load 할수 있도록 설계 한다.

## CHAMBER CONTROL

* 개별 챔버 제어 방법
  + PC 두대로 개별 제어
  + PC 한대로 제어시 프로세서 분리 및 안정화 검증기간이 길어짐
    - 챔버, SCRIPT 및 기타 기능 들을 프로세서 분리 해야 제어가능 제어
* 챔버 Address Map 은 “938 통신 ADDRESS MAP.xlsx” 참조 한다.
  + 참조 자료



* 챔버 기능 흐름도

UNISDK START

PLC Library Memory Delete

PLC Library Memory Assign

API Function

UNISDK Close

SW Library

PLC Library Function

GUI and Utility

PLC Library Memory Delete

**egnbackend.cpp**bp\_initCond()

**Chamber Function Control**

* OS 챔버 트리 구조
  + 아래 트리 구조는 PLC Library 을 사용하고 있는 구조를 나타낸다.
  + 아래 트리 구조을 최적화 해서 구현 한다. (불필요한 소스 파일 제거 및 안정화 목적)
  + 불필요한 코드 unidebug.cpp, unicmd.cpp 제거 한다.

**egnmain.cpp**

**egnplc.cpp**

**egnbackend.cpp**

s\_engine

s\_includelib

source

**api.cpp**

**unicmd.cpp**

**unidebug.cpp**

**unihwmdl.cpp**

**uniplcfunc.cpp**

**uniswshm.cpp**

s\_UniSDKFrame

**define.cpp**

**fmainwindow.cpp**

f\_basic

f\_main

**frunstatus.cpp**

**ftmseventtest.cpp**

f\_setup

**flotenddisplay.cpp**

f\_utility

**fchamctrl.cpp**

**ferrorstatus.cpp**

**futility.cpp**

**ftemperature.cpp**

* 온도 제어는 API 와 GUI Utility 화면에서 기본 적인 제어 할 수 있도록 설계 한다.
* Condition file 또는 GUI Setup 에서 기본 설정 할 수 있도록 한다.
* Condition file 경로 1 : /home/unin2/UNIN2\_maint/conf/syscondition.conf
  + 챔버 사용 유무
  + 챔버 온도 제어 MAX/MIN
  + 챔버 통신 주기
  + 챔버 온도 모니터링 / 주기
  + 챔버 모니터링
  + 챔버 온도 차트 / 주기

|  |  |
| --- | --- |
| **[CHAMBER]** | |
| CHAM1\_USE | TRUE |
| CHAM2\_USE | TRUE |
| CHAM\_TEMP\_MAX | 130 |
| CHAM\_TEMP\_MIN | -10 |
| CHAM\_COMM\_PERIOD | 5 |
| **[MONITORING]** | |
| TEMP\_MONITOR | FALSE |
| CHAM\_MONITOR | FALSE |
| TEMP\_CHART | FALSE |
| TEMP\_TIMER | 1 |
| TEMP\_INTERVAL\_FOR\_TEMP\_CHART | 10 |

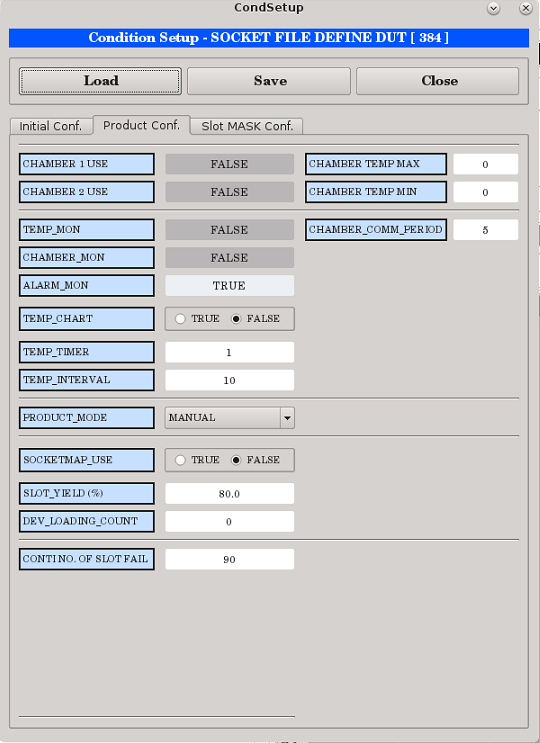
[표] syscondition.conf 설정

* Condition file 경로 2 : /home/unin2/UNIN2\_maint/.conf/unisdk-l938d.conf
  + 챔버 통신 IP / PORT

|  |  |
| --- | --- |
| **[PLC\_COMM]** | |
| IP | 192.168.0.2 |
| PORT | 12291 |

[표] unisdk-l938d.conf 설정

* GUI Condition 화면 :
  + 경로 : MENU > Set up > Condition - Product Conf TAB



[그림] 챔버 설정 화면

* GUI CHAMBER CONTROL 화면
  + (1) READ & WRITE
    - Target TEMP read/write
    - Target Soak Time read/write
    - Current Temp read
    - Lamp Control
  + (2) Mode
    - Local & Remote
    - Soak Time On & Off
    - Execute Run & Stop
    - Door Cover 1,2,3 Lock & UnLock
    - Door Sensor Status Open & Close
  + (3) Control Log Display



(2)

(3)

(1)

[그림] 챔버 제어 화면

## CHAMBER EMULATOR

* + Chamber Emulator Action Sequence Flow

**TEST CORE**

**ENGINE**

**Chamber**

**Emulator**

**Read**

**Write**

**UNIGUI**

* UNISDK와 PLC CHAMBER EMULATOR간에는 TEST CORE ENGINE과의 SW 흐름은 위와 같다.
  + UNISDK와 CHAMBER EMULATOR 통신 구조

**READ/WRITE**

**CLOSE**

**CHAMBER EMUALTOR**

**SOCKET**

**BIND**

**LISTEN**

**ACCEPT**

**UNISDK**

**SOCKET**

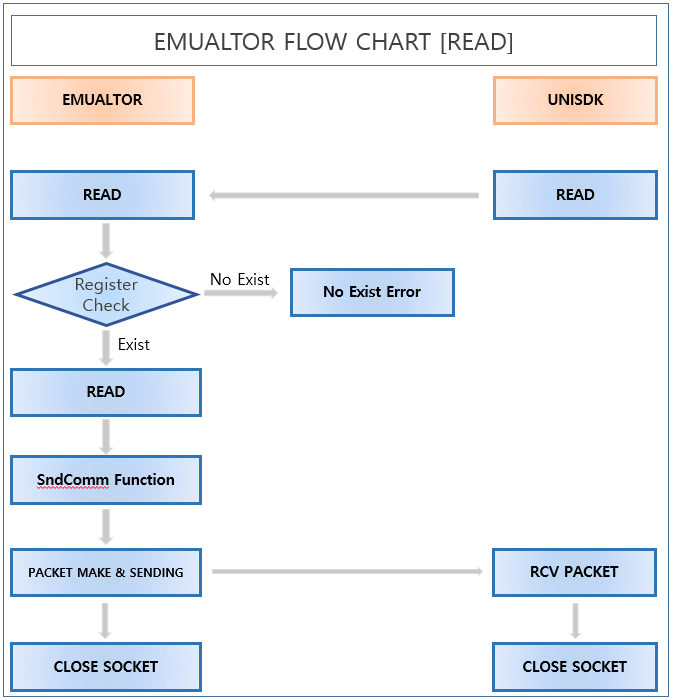
**CONNECT**

**READ/WRITE**

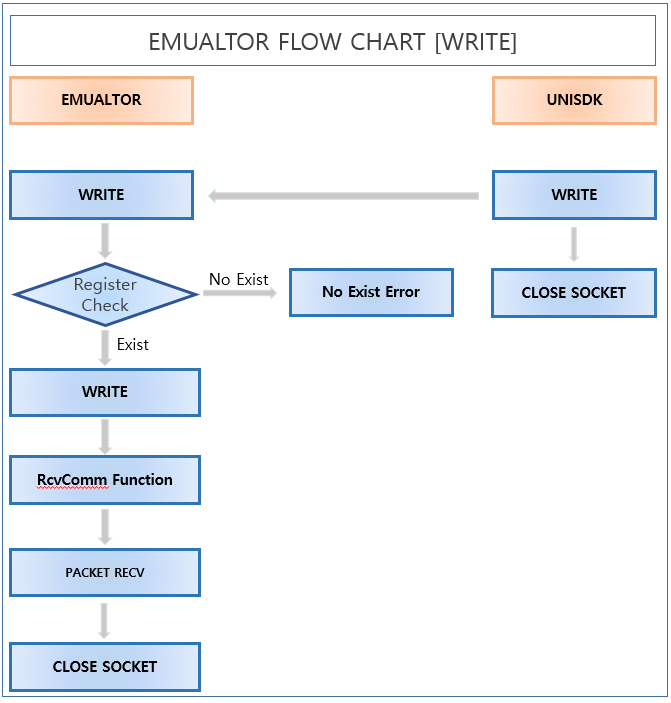
COMMUNICATION STRUCTURE

* UNISDK는 PLC CHAMBER EMULATOR와 TCP/IP 통신을 한다. 위의 통신 구조의 흐름이다.

* + Emulator Flow Chart [Read]



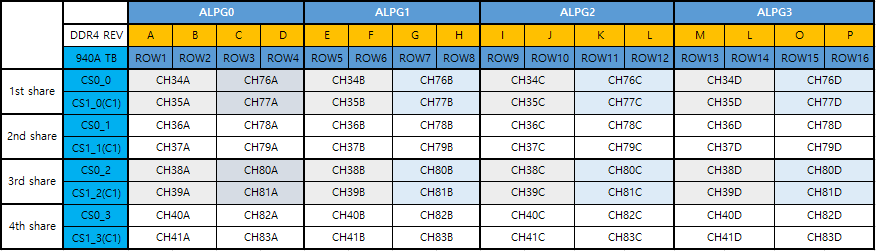
* 위 그림은 앞서 설명한 “통신 구조” 흐름에서 빨간색으로 표시한 부분의 내부 흐름을 나타낸 그림이며, CHAMBER EMULATOR는 UNISDK에서 요청한 REGISTER에 대한 READ를 요청 시에 CHAMBER EMULATOR는 PACKET을 만들어 SEND하며, UNISDK는 받은 PACKET을 통하여 DATA세팅하여 ALARM 및 TEMP, TIME 등을 CHECK하는 용도로 사용한다. 위의 그림은 EMULATOR 흐름도를 나타낸 그림이다.
  + Emulator Flow Chart [Write]



* 위 그림은 앞서 설명한 “통신 구조” 흐름에서 빨간색으로 표시한 부분의 내부 흐름을 나타낸 그림이며, CHAMBER EMULATOR는 UNISDK에서 요청한 REGISTER에 대한 WRITE를 요청 시에 CHAMBER EMULATOR는 RECV한 PACKET을 토대로 UNISDK의 DATA를 READ 요청 시에 전달해주거나, EMULATOR 내부적으로 값에 따른 Simulation하는 용도로 사용한다. 위의 그림은 EMULATOR 흐름도를 나타낸 그림이다.

## SOCKET COMPILER

* 소켓 파일에 기술되어 있는 것을 OS(unigui) 또는 사용자 프로그램에서 사용 할 수 있도록 컴파일 한다.
* 현재 컴파일러의 CS 정보와 Channel Map 의 정보가 불일치 하기 때문에 수정이 필요 하다.
* 소켓 파일의 ROW Line DUT3, DUT4, DUT7, DUT8, … 의 CS 정보 불일 치로 아래와 같이 수정한다.
* CS Channel map info
  + 채널 넘버 0 기준이며, [A: ALPG0], [B: ALPG1], [C: ALPG2], [D: ALPG3]



* CS Socket file info
  + 채널 넘버 1 기준 이며, [A: ALPG0], [B: ALPG1], [C: ALPG2], [D: ALPG3]
  + CSA, CSB, CSC, CSD 는 socket file 의 DUTDEF 에서 사용하는 이름이다.
  + Socket Compiler and Socket File 은 다음과 같이 변경
    - CSA ALPG0 -> CSA, CSB 을 ALPG0
    - CSB ALPG1 -> CSC, CSD 을 ALPG1
    - CSC ALPG2 -> CSE, CSF 을 ALPG2
    - CSD ALPG3 -> CSG, CSH 을 ALPG3
  + Define : ALPG 0, CSA[0-7] CSB[0-7]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PINDEF** | CH | **PINDEF** | CH | **PINDEF** | CH | **PINDEF** | CH |
| **CSA0** | CHA(35) | **CSA4** | CHA(39) | **CSB0** | CHA(77) | **CSB4** | CHA(81) |
| **CSA1** | CHA(36) | **CSA5** | CHA(40) | **CSB1** | CHA(78) | **CSB5** | CHA(82) |
| **CSA2** | CHA(37) | **CSA6** | CHA(41) | **CSB2** | CHA(79) | **CSB6** | CHA(83) |
| **CSA3** | CHA(38) | **CSA7** | CHA(42) | **CSB3** | CHA(80) | **CSB7** | CHA(84) |

* + Define : ALPG 1, CSC[0-7] CSD[0-7]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PINDEF** | CH | **PINDEF** | CH | **PINDEF** | CH | **PINDEF** | CH |
| **CSC0** | CHB(35) | **CSC4** | CHB(39) | **CSD0** | CHB(77) | **CSD4** | CHB(81) |
| **CSC1** | CHB(36) | **CSC5** | CHB(40) | **CSD1** | CHB(78) | **CSD5** | CHB(82) |
| **CSC2** | CHB(37) | **CSC6** | CHB(41) | **CSD2** | CHB(79) | **CSD6** | CHB(83) |
| **CSC3** | CHB(38) | **CSC7** | CHB(42) | **CSD3** | CHB(80) | **CSD7** | CHB(84) |

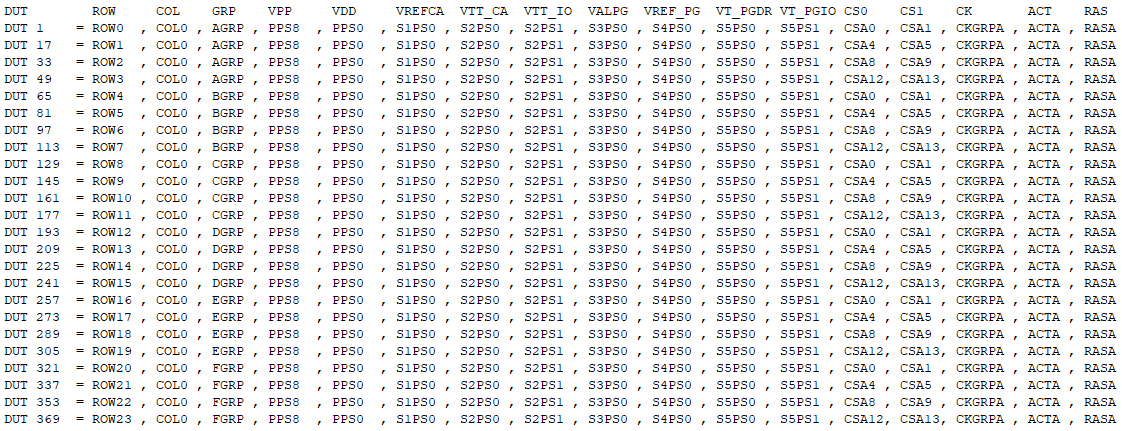
* + Define : ALPG 2, CSE[0-7] CSF[0-7]

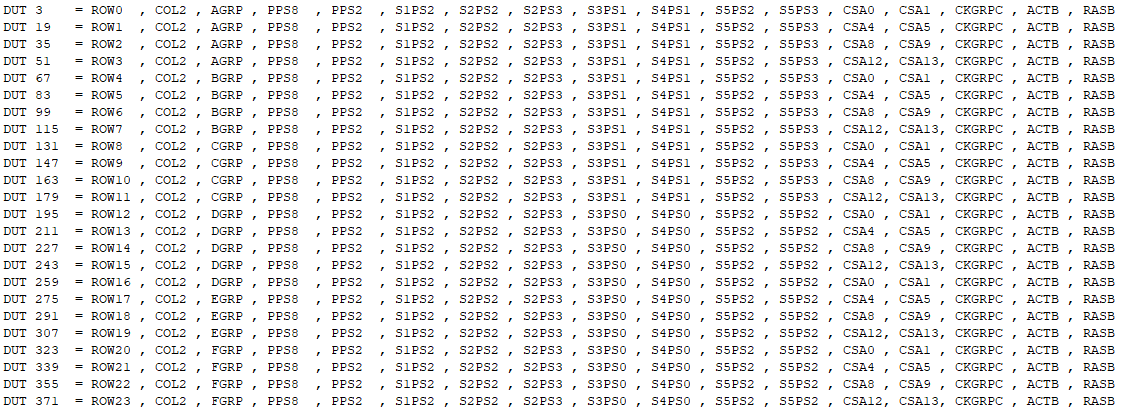
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PINDEF** | CH | **PINDEF** | CH | **PINDEF** | CH | **PINDEF** | CH |
| **CSE0** | CHC(35) | **CSE4** | CHC(39) | **CSF0** | CHC(77) | **CSF4** | CHC(81) |
| **CSE1** | CHC(36) | **CSE5** | CHC(40) | **CSF1** | CHC(78) | **CSF5** | CHC(82) |
| **CSE2** | CHC(37) | **CSE6** | CHC(41) | **CSF2** | CHC(79) | **CSF6** | CHC(83) |
| **CSE3** | CHC(38) | **CSE7** | CHC(42) | **CSF3** | CHC(80) | **CSF7** | CHC(84) |

* + Define : ALPG 3, CSG[0-7] CSH[0-7]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PINDEF** | CH | **PINDEF** | CH | **PINDEF** | CH | **PINDEF** | CH |
| **CSD0** | CHG(35) | **CSD4** | CHG(39) | **CSB0** | CHH(77) | **CSB4** | CHH(81) |
| **CSD1** | CHG(36) | **CSD5** | CHG(40) | **CSB1** | CHH(78) | **CSB5** | CHH(82) |
| **CSD2** | CHG(37) | **CSD6** | CHG(41) | **CSB2** | CHH(79) | **CSB6** | CHH(83) |
| **CSD3** | CHG(38) | **CSD7** | CHG(42) | **CSB3** | CHH(80) | **CSB7** | CHH(84) |

* + 잘못 된 Socket file sample





* + 수정 POINT
    - 아래 테이블과 같이 SOCKET 과 COMPILER 수정

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **소켓 파일 정의가 잘못됨** | | | | **소켓 파일 수정(채널 MAP DEFINE)** | | | |
| **DUT** | PINDEF | **DUT** | PINDEF | **DUT** | PINDEF | **DUT** | PINDEF |
| **DUT1** | CSA0, CSA1 | **DUT3** | **CSA0, CSA1** | **DUT1** | CSA0, CSA1 | **DUT3** | **CSB0, CSB1** |
| **DUT17** | CSA4, CSA5 | **DUT19** | **CSA4, CSA5** | **DUT17** | CSA2, CSA3 | **DUT19** | **CSB2, CSB3** |
| **DUT33** | CSA8, CSA9 | **DUT35** | **CSA8, CSA9** | **DUT33** | CSA4, CSA5 | **DUT35** | **CSB4, CSB5** |
| **DUT49** | CSA12, CSA13 | **DUT51** | **CSA12, CSA13** | **DUT49** | CSA6, CSA7 | **DUT51** | **CSB6, CSB7** |

# Supporting Information

## Software Requirement Architecture

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| 날짜 | 버전 | 설명 | 작성자 |
| 2019-12-06 | 0.01 | UniSDK 전체, PCAL, Power Sequence, Power Monitoring Block Diagram 기입. | 서동호 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 참고



|  |
| --- |
| **UniTest Inc.**  27, Gigok-ro, Giheung-gu, Yongin-si, Gyeonggi-do, Korea  Zip Code: 446-930  Tel : 82-31-547-0300  Fax : 82-31-547-0468 |