FMS RTU Framework 간략 설명서

본 문서는 FMS 개발 서버 구축을 위한 일반 사항에 대해 설명한 문서로써 ㈜ 알림 시스템의 허가 없이 외부에 오픈하지 않는다.

< 기본 규칙 >

* define 문은 대문자로 한다.
* 변수 및 function은 소문자로 한다.
* 전역 변수는 g\_ 접두사 처리 , static 처리한다.

전역 변수 중 숫자 형 데이터는 volatile 처리하고 반드시 초기값을 설정한다.

**static volatile int g\_sts = 0;**

* 로컬 변수는 반드시 초기값을 설정한다.
* 로컬 함수는 static으로 선언한다.
* 포인터 변수는 ptr 접두사를 사용한다.
* print는 util에 있는 print\_dbg 함수로 대체한다.

저장이 필요한 print에 대해서는 debug\_type항목에 DBG\_SAVE 값을 넣는다.

* sleep은 util에 있는 sys\_sleep 함수로 대체 한다.
* struct는 typedef로 정의하고 \_t 접미사를 둔다. 1byte padding 규칙을 둔다.

**typedef struct rtu\_sts\_**

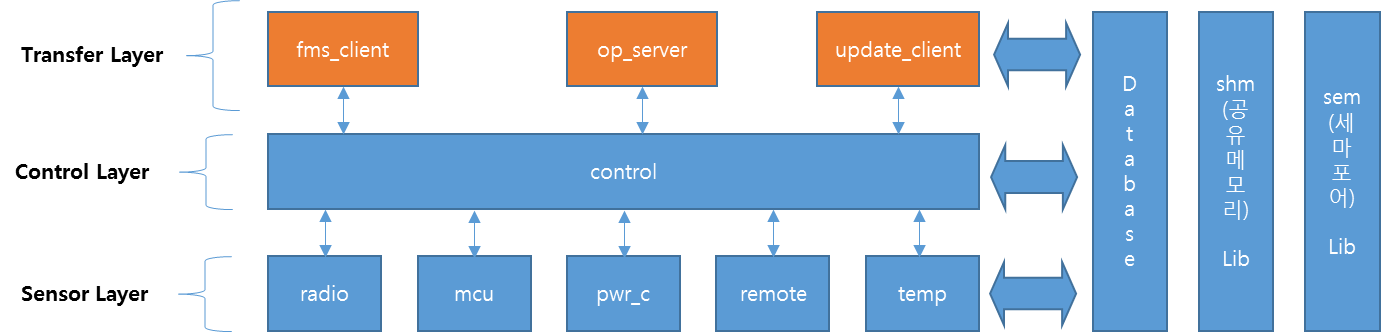
**{**

**int fms\_net\_sts;**

**}\_\_attribute\_\_ ((packed)) rtu\_sts\_t;**

* 컴파일 시 모든 warring은 없도록 한다.
* inline은 사용하지 않는다.
* **fms\_client.c op\_server.c update\_client.c 파일을 보시면 함수 사용법과 변수 작성법에 대해 상세히 설명하였으니 참조 바랍니다.**

< Moduel 구성도 >



* fms\_client : FMS Servr와 연동되는 모듈
* op\_server : OP Client들과 연동되는 모듈
* update\_client : Update Server와 연동되는 모듈
* control : 통신 모듈과 센서 처리 모듈의 데이터를 처리하는 모듈
* radio : 전파 감시 장치 처리를 담당하는 모듈
* mcu : di, do gpio 처리를 담당하는 모듈
* pwr\_c : 전원제어 장치 처리를 담당하는 모듈
* remote : remote control 장치 처리를 담당하는 모듈
* temp : 온도/습도 장치 처리를 담당하는 모듈

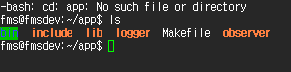
< 개발 주체 >

* fms\_client, op\_server, update\_client : 키로직에서 개발 담당
* control, radio, mcu, pwr\_c, remote, temp, pwr\_m, database, sem : 알림에서 개발 담당
* shm : 알림과 키로직에서 필요시 공용으로 개발

주의 : shm은 알림과 키로직이 필요시 공통으로 수정, 보완할 수 있는 모듈이기 때문에 각사에서 추가 할 경우 반드시 각사에 변경 사항을 알려 파일의 동기화가 이루어져야 한다.

< 디렉토리 및 파일 설명 >

* app\_xxxx.tar.gz 파일을 압축 풀면 아래와 같이 디렉토리가 생성됨.



* bin : make 후 생성되는 파일의 위치
  + fms\_observer : 프로세스 감시 용 프로그램
  + fms\_logger : fms rtu 프로그램
  + 프로그램 실행 : ./fms\_logger
* include : library 파일에 대한 .h 파일 위치
  + com\_def.h : define 선언 파일
  + com\_include.h : 각 모듈에 들어가는 공용 include 문 정의 파일
  + conf.h : 설정 파일 헤드 파일
  + core\_conf.h : core 설정 파일 헤드 파일
  + protocol.h : 프로토콜 선언 파일
  + sem.h : 세마포어 선언 파일
  + shm.h : 공유메모리 선언 파일
  + type.h : 데이터 형신 선언 파일
  + util.h : 유틸리티 선언 파일
  + UT\_List.h : linked list 선언 파일
* Lib : library 정의 파일
  + Conf.c : 설정 정보 입,출력 함수 정의 파일
  + Core\_conf.c : 설정 정보 처리 core 함수 정의 파일
  + Sem.c : 세마 포어 정의 파일
  + Shm.c : 공유 메모리 정의( get, set ) 파일
  + Util.c : 유틸리티 정의 파일
* Logger : RTU 운영 파일
  + fms\_client : FMS Servr와 연동되는 모듈
  + op\_server : OP Client들과 연동되는 모듈
  + update\_client : Update Server와 연동되는 모듈
  + control : 통신 모듈과 센서 처리 모듈의 데이터를 처리하는 모듈
  + radio : 전파 감시 장치 처리를 담당하는 모듈
  + mcu : di, do gpio 처리를 담당하는 모듈
  + pwr\_c : 전원제어 장치 처리를 담당하는 모듈
  + remote : remote control 장치 처리를 담당하는 모듈
  + temp : 온도/습도 장치 처리를 담당하는 모듈
* Make file : 실행 파일 생성 메크로
  + make – s : Make 실행 ( 실행 후 바이러니 파일은 /bin 폴더에 생성됨 )
  + make clean : object 파일 제거

make –s 전 반드시 make clean을 먼저 실행 할것.

* Observer : 프로세스 감시용 프로그램

< 주요 주의 사항 >

* Transfer Layer
  + int recv\_sns\_data\_fms\_client( unsigned short inter\_msg, void \* ptrdata ) : fms\_client
  + int recv\_sns\_data\_op\_server( unsigned short inter\_msg, void \* ptrdata ) : op\_server
  + int recv\_cmd\_data\_update\_client( unsigned short inter\_msg, void \* ptrdata ) :

update\_client

* + - Control Layer에서 데이터를 받는 함수
    - inter\_msg : control에서 전달되는 데이터에 대한 messageID로 프로토콜에 정의 된 \_enumFMS\_CMD\_TAG 데이터를 사용한다.
    - ptrdata : control에서 전달되는 데이터 구조체의 void \* 형으로 inter\_msg에 따라 다른 구조체가 넘어 옴. ( 현재 미 정의 됨 )
  + static int send\_data\_control( UINT16 inter\_msg, void \* ptrdata )

Control로 데이터를 전달하는 함수

* + - Control Layer로 데이터를 전송하는 함수
    - inter\_msg : control에 전달하는 데이터에 대한 messageID로 프로토콜에 정의 된 \_enumFMS\_CMD\_TAG 데이터를 사용한다.
    - ptrdata : control에 전달하는 데이터 구조체의 void \* 형으로 inter\_msg에 따라 다른 구조체를 넘김( 현재 미정의 됨 )
* 세마포어를 사용할 경우 반드시 lock , unlock을 커플로 사용해야한다.

int lock\_sem( int semid, int key )

int unlock\_sem( int semid , int key )

예) lock\_sem( g\_sem\_id, g\_sem\_key )

unlock\_sem( g\_sem\_id , g\_sem\_key );

최대한 짧은 시간동안 사용해야 한다.

Lock과 un\_lock사이의 시간이 길어지면 시스템 전반적으로 느려진다.

* Print 대신 util에 존재하는 print\_dbg함수를 사용한다.

int print\_dbg( unsigned char debug\_type, int print\_consol, char \* fmt, ...)

/\* DEBUG TYPE \*/

#define DBG\_NONE 0x01

#define DBG\_INFO 0x02

#define DBG\_INIT 0x03

#define DBG\_ETC 0x04

#define DBG\_LINE 0x10

#define DBG\_SAVE 0x20

#define DBG\_ERR 0xFF