

## OTA 아키텍처

 PlantUML Web Server

### 1. PC → RP4

- MQTT를 이용하여 RP4에 TCP/IP 통신을 통해 hex파일을 전달함.
- aurix에선 hex파일을 생성해도 empty값 도 포함해서 나와서 용량이 엄청 커짐.  
이걸 필터링하거나, hex파일을 쓸 것.
  - hex파일의 경우 주소+데이터 형태로 나타나있는데 주소 가지고 그대로 인덱싱해서 PFLASH에 쓰거나,
  - Isl파일을 통해 주소가 연속적이게 만들고 데이터를 할당하던가

### 2. RP4 → TC

- ~~CAN을 이용하여 RP4에서 8바이트 단위로 TC375에 데이터를 전달함.~~
- bin파일 크기가 kb라서 CAN으론 너무 자원 소모가 큼. 전용 이더넷 선을 추가해서 해보자.
- 이더넷 통신을 통해 전송할 때, 해당 파일을 온전히 전송할 수 있는가?
  - 파일 데이터 일부분 전송 실패 시 어떻게 처리하나?
  - 파일이 온전히 전송되었는지 어떻게 확인하나?
  - 구분해서 PFLASH업데이트 => 리셋버튼을 눌러서 => 업데이트된 걸 실행하는 구조인데,
    - HW 리셋을 => 현재 펌웨어 버전 같은건 아마 알게 만들 수 있을듯. PFLASH A,B << 전역 변수 설정.
    - SW 리셋을 알면 클러스터에서 업데이트 여부를 결정해서 업데이트를 수행할 수 있을 것

### 3. TC firmware download

- a. ~~8바이트단위로 전역 array에 저장함.~~
- b. kb파일을 메모리에 갖고 있기엔 너무 무겁다. 받을때마다 pflash에 쓰자.

### 4. inactive한 PFLASH영역에 플래시 프로그래밍을 통해 쓰기를 수행함

- a. 기본 가정
  - i. Segment 10을 사용하는 중이라고 가정.  
(non-cached, 플래시프로그래밍 작업에 반드시 사용되어야함)
  - ii. 현재 시스템이 표준 주소 맵 모드로 동작하고 있다고 가정
  - iii. PF0을 사용하고 있다고 가정. PF10이 inactive하다고 가정

b. 수행 방법

i. 프로그래밍 루틴(eraseSectors, enterPageMode, writePage, waitUnbusy 등)을

`copyFunctionsToPSPR()` 를 통해 CPU0의 Program Scratch-Pad SRAM (PSPR)으로 복사

- PFLASH에 쓰기 작업하는 동안 해당 PFLASH의 코드를 실행할 수 없기 때문 (근데 왜 안되는지 모르겠음)

ii. PFLASH1영역에 전역 array의 데이터를 가져와서 쓰기

1. PFLASH1 시작 주소인 A030 0000<sub>H</sub> 부터 시작

2. 데이터를 쓰기 전에 해당 논리 섹터를 지워야 합니다.

`IfxFlash_eraseMultipleSectors()` 함수를 PSPR에서 실행하여 16KB 단위의 논리 섹터를 지울 수 있습니다

3. **페이지 모드 진입:** `IfxFlash_enterPageMode()` 함수를 PSPR에서 실행하여 페이지 모드로 진입합니다.

4. **데이터 로드 (IfxFlash\_loadPage2X32):** 임시 버퍼( `receive_buffer` )에 있는 `bin` 파일 데이터를 PFLASH의 페이지 어셈블리 버퍼로 로드합니다

- a. 전역 array의 하나의 index에 8바이트가 저장되어 있고, 이 함수가 특정 주소에 32bit, 32bit까지 로드하므로 8바이트를 쪼개서 해당 주소에 넣으면 될듯?

5. `IfxScuWdt_clearSafetyEndinitInline()` 함수를 통해 Safety EndInit 보호 비트를 해제

6. **페이지 쓰기 (IfxFlash\_writePage):** 로드된 페이지는 `IfxFlash_writePage()` 함수 (PSPR에서 실행)를 통해 PFLASH에 기록

7. **Safety EndInit 보호 재설정:** 모든 플래시 프로그래밍 작업이 완료된 후,

`IfxScuWdt_setSafetyEndinitInline()` 함수를 호출하여 EndInit 보호 비트를 다시 설정

8. **작업 완료 대기:** 각 단계 후 `IfxFlash_waitUnbusy()` 함수 (PSPR에서 실행)를 사용하여 플래시 메모리가 준비될 때까지 기다려야 합니다

5. 주소맵 전환 후 시스템 리셋을 트리거

a. DMU `Disable Protection` 명령 시퀀스로 암호를 입력하여 보호를 해제

- i. default가 해제라는것 같은데?

b. UCB\_SWAP 의 MARKERLx.SWAP 필드에 000000AA<sub>H</sub> 값을 써서 대체 주소 맵으로 설정

c. `MARKERHx.ADDR` , `CONFIRMATIONHx.ADDR` , `CONFIRMATIONLx.CODE` 와 같은 다른 SWAP 정보도 함께 구성

- i. MARKERL0.SWAP의 시스템 주소를 MARKERH0.ADDR 에 작성

- ii. CONFIRMATIONL0.CODE의 시스템 주소를 CONFIRMATIONH0.ADDR에 작성
- iii. 확인 코드 57B5327F<sub>H</sub>를 CONFIRMATIONL0.CODE에 기록
- d. 이전 ( X-1 ) UCB\_SWAP 항목은 'all-1' 값으로 오버 프로그래밍하여 무효화
- e. **Resume Protection** 명령으로 다시 보호를 활성화
- f. 시스템 리셋 (어케하지?)
- 6. 리셋후, SSW는 UCB\_SWAP\_ORIG/COPY에 저장된 SOTA 구성 평가
  - a. SSW는 **MARKERLx.SWAP** 에 설정된 **000000AA** <sub>H</sub> 값을 기반으로 대체 주소 맵을 시스템에 활성화
    - i. 00000055<sub>H</sub> 값 입력 시 표준 주소 맵화
  - b. 대체 주소 맵(Alternate Address Map)에서 TC37x의 PFLASH 매핑
  - c. **Segment 10 PFLASH:** **A030 0000** <sub>H</sub> ~ **A05F FFFF** <sub>H</sub>에 Program Flash 1 (PF0)이 매핑됩니다.
    - i. 즉, 이전에 표준 주소 맵의 PF1이었던 물리적 뱅크가 대체 주소 맵의 PF0 위치로 주소 매핑이 변경됩니다.
  - d. SSW는 모든 **CPUx\_FLASHCON4.DDIS** 비트를 '1'로 설정하여 PFLASH에 대한 CPU의 직접 접근을 비활성화하고, SRI를 통해 접근하도록 경로를 변경합니다.