Modbus RTU 레지스터 맵(holding/input register 기준) 설계

1. 기본 규칙

- **03 (Read Holding Registers) 04 (Read Input Status) 를 주소 영역으로 논리적으로 구분한다.**
  비트 제어 명령은 사용하지 않는다.
  적용 명령어
  - 0x03 (Read Holding Registers) 슬레이브의 연속된 레지스터(읽기/쓰기 가능한 데이터)를 읽습니다.
  - 0x04 (Read Input Registers) 입력 전용 레지스터(읽기만 가능)를 읽습니다.
  - 0x06 (Write Single Register) 하나의 레지스터에 값을 씁니다.
  - 0x10 (Write Multiple Registers) 여러 연속된 레지스터에 값을 씁니다.
- **모든 주소는 16비트(1워드) 단위** Mdbus RTU uses 16-bit registers.
  모든 레지스터는 16비트 단위로만 접근 가능 (바이트 단위 접근 불가)
- **Modbus 표준은 MSB first(빅엔디안)**
  32-bit data (e.g., position) is represented using 2 consecutive registers.
- **기기 식별 레지스터**:
  UID(96bits), 32bit device id, 펌웨어 버전, slave address(DIP switch),
- **에러 및 상태 레지스터**:
  현재 에러 코드, 상태 플래그 등.
- **PWM heater, NTC 온도센서, 2개의 모터(리미트 츠위치 하나) 제어/상태 레지스터**:
  동작 테스트 및 개별 제어를 위한 영역 (시나리오 모드에서는 제어가 동작하면 안됨)
- **기존 Genhome2 의 시나리오에 따라 동작시 필요한 데이터 영역**:
  비휘발성(필요에 따라서) 파라미터, 설정값 등. - 시나리오 모드 제어 비트 추가(defalut: off(0))
- **Port Configutaion**:
  Baud rate: 256000 bps - 짧은 케이블 거리(<2m)에서 최적화된 고속 통신 => 115200 bps (CRC 에러로 변경)
  Data bits : 8 Parity: Even
  Stop bits: 1
  통신 성능:
  - ~~8개 디바이스 전체 폴링: ~36-44ms (기존 115200 bps 대비 2.2배 향상)~~
  - 8개 디바이스 전체 폴링: ~80-90ms (115200bps 기준)
  - ~~단일 요청-응답: ~4.5-5.5ms~~
  - 단일 요청-응답: ~9-11ms (115200bps 기준)
  - ~~실시간 제어 응답성: 22-28Hz 업데이트 가능~~
  - 실시간 제어 응답성: 10-12Hz 업데이트 가능 (115200bps 기준)
  - CFTerm 테스트 프로그램 호환 (프로그램에서 설정 가능한 최고 속도가 256000)

Modbus RTU Register Map (설계 계획안)

읽기 전용 레지스터 (`0x0000`~)

주소 (Address)	Read/Write	이름 (Name)	설명 (Description)	데이터 타입 (Data Type)	기본값 (Default Value) MSB First
0x0000	R	status	상태 비트 통합 (시스템 상태, 에러 등)	uint16_t	0x0000~0xFFFF
0x0001	R	heater_temp_x10	히터 현재 온도 x10 (예: 1°C=10)	int16_t	
0x0002	R	heater_time_x10	히터 현재 가열 시간 (100msec)	uint16_t	
0x0003	R	pwm_duty_x100	현재 PWM 듀티(%) x100	uint16_t	
0x0004/05	R	motor0_pos	모터 0 현재 위치 step 단위	uint32_t	
0x0006/07	R	motor1_pos	모터 1 현재 위치 step 단위	uint32_t	
0x0008	R	motor0_status	모터 0 상태	uint16_t	세부 설명 참조
0x0009	R	motor1_status	모터 1 상태	uint16_t	세부 설명 참조
0x000A	R	reserved1	사용 안함	uint16_t	
0x000B	R	slave_address	DIP 스위치 기반 슬레이브 주소	uint16_t	0x5F00+slave address(8bit)
0x000C/0D	R	DEVICE_ID	32비트 Device ID (UID XOR)	uint32_t	0x12345678
0x000E~13	R	UID[0~2]	STM32 96bit UID (원본)	uint32_t[3]	MSB first network order 변환
0x0014	R	FW_VERSION	펌웨어 버전 (예: 0x0002=0.2)	uint16_t	0x0002

status (0x0000)

시스템 전체 상태를 나타내는 비트 필드:

- Bit 0: 히터 동작 상태 (0=OFF, 1=ON)
- Bit 1: 모터0 동작 상태 (0=Stop, 1=Running)
- Bit 2: 모터1 동작 상태 (0=Stop, 1=Running)
- Bit 3: 카트리지 탈착 센서 (0 = out, 1 = in)
- Bit 4: 카트리지 버튼 1 (0=OFF, 1=ON)
- Bit 5: 카트리지 버튼 2 (0=OFF, 1=ON)
- Bit 6: 카메라 LED 상태 (0=OFF, 1=ON)
- Bit 7: UV LED 상태 (0=OFF, 1=ON)
- Bit 8-15: senario step (8 bits) **senario cmd 참조**

heater_temp_x10 (0x0001)

히터의 현재 온도를 0.1°C 단위로 표시 (예: 250 = 25.0°C)

heater_time_x10 (0x0002)

현재 히터 가열 시간을 100ms 단위로 표시

pwm_duty_x100 (0x0003)

실제 적용된 PWM 듀티(%) x100 값 (PID/수동/과열 등 모든 제한 반영) 예) 3012 -> 30.12%

motor0_pos (0x0004/05), motor1_pos (0x0006/07)

모터 0-1의 현재 위치 (step 단위, 32비트, MSB first)

motor0_status (0x0008), motor1_status (0x0009)

모터 0-1 동작 상태 및 리미트 스위치 상태 (상태 비트별 상세 설명)

비트	이름	의미/설명
15~12	motor state	motor_hal_state_t
11~10	motor mode	app_motor_mode_t
9~0	Reserved	예약

- 모터 enable/disable은 내부 전력 관리 시스템에서 자동으로 처리됨 (정지 후 1000ms 경과 시 자동 비활성화)

상태 필드 상세 설명:

MotorState (Bit 15~12): 모터 동작 상태

코드 상의 구현 값을 그대로 사용, 변경시 업데트 하겠음.

```
typedef enum {
    MOTOR_STATE_DISABLED = 0,     // 드라이버 비활성화 (EN = LOW)
    MOTOR_STATE_STOPPED,          // 활성화 상태로 정지 (EN = HIGH)
    MOTOR_STATE_ACCEL,           // 가속 구간
    MOTOR_STATE_CONST,           // 정속 이동
    MOTOR_STATE_DECEL,           // 감속 구간
} motor_hal_state_t;
```

MotionMode (Bit 11~10): 모터 동작 방식

코드 상의 구현 값을 그대로 사용, 변경시 업데트 하겠음.

```
typedef enum {
    APP_MOTOR_MODE_NORMAL = 0, // 일반 모드
    APP_MOTOR_MODE_HOMING, // 홈 찾기 모드 상태에서 리미트가 감지되면 ->
HOMING_BACKOFF 상태로 변경됩니다.
    APP_MOTOR_MODE_HOMING_BACKOFF, //HOMING_BACKOFF 상태에서 이동이 완료되면 ->
NORMAL 상태로 변경되고 is_homed 플래그가 설정됩니다.
    APP_MOTOR_MODE_AT_LIMIT // 리미트 스위치 감지// homming 이 아닌 상태에서
} app_motor_mode_t;
```

reserved1 (0x000A)

예약된 레지스터

slave_address (0x000B)

DIP 스위치의 ON/OFF 를 bit 로 환산하여 사용한다. 단, 0 인 경우를 제외시키기 위해 +1 하여 사용함.
특별이 의미는 없지만, 상위 BYTE 를 0x5F 로 채운다. (범위 : 0x5F01 ~0x5F08)

DEVICE_ID (0x000C/0D)

32비트 Device ID는 STM32 내부 UID 워드들의 XOR 연산으로 생성됩니다:

device_id = uid0 ^ uid1 ^ uid2

예시) 만약 STM32 UID가 다음과 같다면:

- UID[0] = 0x12345678
- UID[1] = 0x9ABCDEF0
- UID[2] = 0x13572468
- Device ID = 0x12345678 ^ 0x9ABCDEF0 ^ 0x13572468 = 0x8B8F8A90

Modbus 레지스터에는 big-endian 형식으로 저장: 0x000C=0x8B8F, 0x000D=0x8A90

UID[0~2] (0x000E~13)

STM32 MCU의 96비트 고유 식별자(UID)는 3개의 32비트 워드로 구성됩니다:

High Word (UID[0]):

- 주소: 0x000E/0F
- STM32 내부 주소: 0x1FFF7A10
- 이 값은 lot number와 wafer number를 포함

Middle Word (UID[1]):

- 주소: 0x0010/11
- STM32 내부 주소: 0x1FFF7A14
- wafer의 X/Y 좌표 정보를 포함

Low Word (UID[2]):

- 주소: 0x0012/13
- STM32 내부 주소: 0x1FFF7A18
- 제조 정보를 포함

각 32비트 워드는 MSB first (big-endian) 네트워크 바이트 순서로 저장됩니다: 예) 0x12345678 값은 레지스터에 0x1234, 0x5678로 저장

FW_VERSION (0x0014)

펌웨어 버전 정보. Major(상위 8bit), Minor(하위 8bit) 예) 0x0A0A -> 10.10, 0x0101 -> 1.1, 0x000A -> 0.10

읽기-쓰기 레지스터 (`0x0100`~)

주소란에 (예정) 표시는 아직 구체적인 구현 방향이 확정되지 않은 항목임.
상태가 계속 변하는 레지스터는 읽기 전용으로 제공하고, 읽기-쓰기 레지스터는 기본적으로 write 한 내용을 유지하는 용도로 주로 사용된다.

- **0x0100~0x010F**: 기본 제어 (히터, PID) - 16개 레지스터
- **0x0110~0x011F**: 모터 제어 - 16개 레지스터
- **0x0120~0x012F**: 시나리오 제어 - 16개 레지스터
- 추가 확장 필요시 0x0130~ 이후 사용
- GenHome2 에서 사용된 EEPROM 에 저장하는 기능은 추후 다시 검토하여 필요 여부를 결정하겠음.

기본 제어 영역 (0x0100~0x010F)

주소 (Address)	Read/Write	이름 (Name)	설명 (Description)	데이터 타입 (Data Type)	기본값 (Default Value) MSB First
0x0100	R/W	pwm_control	히터 PWM 듀티 제어 레지스터 (자세한 설명은 하단 참조)	uint16_t	0 (PID 제어 모드)
0x0101	R/W	heater_temp_x10	히터 목표 온도 (0.1도 단위, 자동 모드 전용)	uint16_t	0 (목표온도 없음)
0x0102/03	R/W	pid_kp	PID Kp (2워드)	float (32bit)	3.62 = 0x4067AE14 (상위 워드: 0x4067, 하위 워드: 0xAE14)

주소 (Address)	Read/Write	이름 (Name)	설명 (Description)	데이터 타입 (Data Type)	기본값 (Default Value) MSB First
0x0104/05	R/W	pid_ki	PID Ki (2워드)	float (32bit)	0.10 = 0x3DCCCCCD (상위워드: 0x3DCC, 하위워드: 0xCCCD)
0x0106/07	R/W	pid_kd	PID Kd (2워드)	float (32bit)	0.0 = 0x00000000 (상위워드: 0x0000, 하위워드: 0x0000)
0x0108	R/W	temp_offset_x10	온도 오프셋 x10	int16_t	
0x0109	R/W	temp_period	온도 측정 ADC 주기(ms)	uint16_t	default:100msec (수정 불가)
0x010A	R/W	gpio_control	GPIO 제어 (UV on/off 등)	uint16_t	
0x010B	R/W	uv_off_sec	UV OFF Timer (sec)	uint16_t	20 min .
0x010C~0x010F	-	reserved	예약 영역	-	

- float 타입은 IEEE754 32비트(2워드, MSB first)로 매핑 각 32비트 값은 16비트 워드 단위로만 스왑(MSB first)하여 저장하며, 바이트 스왑은 하지 않는다. 예: STM32 float 3.62 = 0x4067AE14 (리틀엔디안) Modbus 맵: 0x4067(상위워드), 0xAE14(하위워드) Python: (상위워드 << 16) | 하위워드 → struct.unpack('>f', ...)로 해석

PWM 히터 제어 시스템 동작

- 기존 모델 구현을 그대로 사용함.
- 안전을 위한 최대 PWM duty % 를 정수값 1000 으로 환산해서 사용(PID 제어 출력)
- 예: 1000 → 최대 PWM duty %, 500 → (최대 PWM duty %)/2
- **실제 듀티 x 100 으로 변경 고려중(읽기 영역 상태값과 같은 형태) 0 ~ 10000 (0.00% ~ 100.00%)**

pwm_control 레지스터 (0x0100)

- pwm 수동 제어를 위한 조절 값
- 범위 : 0~1000 (0 : 수동 제어 off, 최대값은 pwm_max_value 에 제한됨)
- 과열 보호 동작 (98도 이상에서 PWM=0 강제 설정)

heater_temp_x10 레지스터 (0x0101)

- pid 제어 목표 온도 설정 값 (0.1 도 적용을 위해서 x 10 하여 사용)
- 기존 모델에서 사용하던 최대 출력 예열 단계를 사용함.

1. **예열 단계 (WAITING)**:
   - 온도차 > temp_pid_startx10: 최대 출력으로 예열
   - 온도차 ≤ temp_pid_startx10: PID 제어 단계로 전환
2. **PID 제어 단계 (ACTIVE)**:
   - 100ms 주기로 PID 연산 수행
   - 과열 방지: 온도가 목표보다 과도하게 높으면 PWM=0

heater 동작은 이 두개의 레지스터를 write 하는 경우에만 모드 변경을 함

- 레지스터 write 없이 모드를 변환하는 경우는 제외 시킴

pwm_control > 0 : 수동 모드 **pwm_control = 0** : 정지 (PWM 출력 완전 정지) **heater_temp_x10 > 0** : PID 제어 모드 **heater_temp_x10 = 0** : 정지 (PWM 출력 완전 정지)

실시간 읽기(pwm_duty_x100, 0x0003):

- 0x0003 레지스터는 현재 실제 적용된 PWM 듀티(%) x100 값을 읽기 전용으로 제공
- PID 제어/수동모드/과열 보호 등 모든 제한이 반영된 실제 출력값
- 예: 500 → 5.00%, 1000 → 10.00%

PID 상수 (Kp(0x0102/03), Ki(0x0104/05), Kd(0x0106/07))

- float 타입은 IEEE754 32비트(2워드, MSB first)로 매핑

temp_offset_x10 (0x0108)

- 온도 센서와 실제값 간의 오차를 보정하기 위한 값

temp_period (0x0109)

- 온도 측정의 주기, PID 제어 주기, 100 msec 고정값
- 주기 변경 기능이 필요할 경우를 대비해서 추가됨.

gpio_control (0x010A)

GPIO 제어 레지스터 비트 구성:

```
Bit 15:   System Reset (0=Normal, 1=Reboot) - 쓰기 시 시스템 재부팅
Bit 14-8: 예약 영역
Bit 7-0:  GPIO 제어
          Bit 0: UV LED (led_uvc_en) (0=OFF, 1=ON)
          Bit 1: Camera Light (led_di) (0=OFF, 1=ON)
          Bit 3-7: 확장용
```

uv_off_sec (0x010B)

UV 램프 off 지연시간 (초) ; 20 분 (120 초)

모터 제어 영역 (0x0110~0x011F)

모터 구성 정책 (v0.21.0 이후)

- 두 모터(모터0/모터1)는 기어비 16:1(MOTOR_TYPE_24BYG48)로 통일되었습니다.
- 웜기어(1:30) 기반 각도 변환/설정은 정상적으로 반영되지 않습니다. 관련 항목은 Reserved/Disabled로 취급합니다.
- Modbus 인터페이스는 기본적으로 변환되는 것은 모터 사양에 따라서 변경되도록 설계되었지만, 외부 사향 추가 기어나, 스크류 피치는 아직 반엉하지 않았습니다.
- 향후 다양한 기어비/피치 지원이 필요할 경우, 별도의 설정 레지스터를 추가하는 방안을 검토하겠습니다.
- 기존 각도 기반 명령(MOVE_DEG/MOVE_REV)은 내부적으로 step 단위로 변환되어 처리됩니다.

표준 단위 및 변환식

- 변환 공식은 16:1 모터에 맞춰서 추후 업데이트 될 수 있습니다.

Deprecated/Disabled

- 웜기어(1:30) 전제의 각도 기반 설정/예시는 0.21.0 이후 Disabled입니다.
- 해당 레지스터/항목은 Reserved로 유지하거나, 읽기 시 0/기본값, 쓰기는 무시 정책을 따릅니다.
- 각도 기반 제어가 필요하면 상위에서 steps 또는 mm로 변환하여 사용하세요.

주소 (Address)	Read/Write	이름 (Name)	설명 (Description)	데이터 타입 (Data Type)	기본값 (Default Value) MSB First
0x0110/11	R/W	motor0_target_pos	모터0 제어 명령과 사용되는 변수	uint32_t	
0x0112	R/W	motor0_control	모터0 제어 명령	uint16_t	[7:0]=명령코드
0x0113	-	motor0_reserved	모터0 확장/예약	uint16_t	
0x0114/15	R/W	motor1_target_pos	모터1 제어 명령과 사용되는 변수	uint32_t	
0x0116	R/W	motor1_control	모터1 제어 명령	uint16_t	[7:0]=명령코드
0x0117	-	motor1_reserved	모터1 확장/예약	uint16_t	
0x0118	R/W	motor0_accel	모터0 가속도 (steps/sec^2)	uint16_t	기본값: 10 (범위: 10~100)

주소 (Address)	Read/Write	이름 (Name)	설명 (Description)	데이터 타입 (Data Type)	기본값 (Default Value) MSB First
0x0119	R/W	motor0_speed	모터0 속도 (steps/sec)	uint16_t	기본값: 300
0x011A	R/W	motor1_accel	모터1 가속도 (steps/sec^2)	uint16_t	기본값: 50 (범위: 10~100)
0x011B	R/W	motor1_speed	모터1 속도 (steps/sec)	uint16_t	기본값: 600
0x011C~0x011F	-	reserved	모터 확장 영역	-	

- 모터 제어는 DRV8428 STEP/DIR 방식을 사용하며, 각 모터마다 하나의 limit 스위치가 할당되어 있음
- motor enable/disable 은 1000ms 이상 제어 명령이 없으면 자동으로 disable 됨

motor0_target_pos(0x0110/11) / motor1_target_pos (0x0114/15)

motor0_control(0x0112) 명령에 따라 해석되는 argument 변수입니다. 아래와 같이 명령별로 입력값의 의미와 단위가 다릅니다.

- **STOP(0x00)**: 모터 이동 정지
- **MOVE_TO(0x01)**: 절대 위치 이동. 이 변수에 "목표 위치(step)" 값을 입력하면, 해당 step 위치로 이동합니다.
    - 예: 10000 입력 → 10000 step 위치로 이동
- **MOVE_HOME(0x02)**: HOME 찾기 동작을 시작합니다. target_postion, speed, accel 모두 무시됩니다.
- **MOVE_REL(0x05)**: 상대 위치 이동. 현재 위치에서 입력한 step만큼 상대적으로 이동합니다.
    - 예: 500 입력 → 현재 위치에서 +500 step 이동
- **MOVE_DEG(0x06)**: 각도 이동. 0.01도 단위의 각도값을 입력하면, 내부에서 step 단위로 변환하여 이동합니다.
    - 예: 9000 입력 → 90.00도 회전 (9000 x 0.01 = 90도)
    - 변환 공식: step = (입력값) × (step/degree 환산값)
- **MOVE_REV(0x07)**: 회전수 이동. 0.01회전 단위의 값을 입력하면, 내부에서 step 단위로 변환하여 이동합니다.
    - 예: 100 입력 → 1.00회전 (100 x 0.01 = 1회전)
    - 변환 공식: step = (입력값) × (1회전당 step 수)
- **MOVE_POS(0x08)**: 저장된 기본 위치의 index : 0-2
    - 예: 1 입력 → 저장된 위치 1번(index=1)으로 이동
      motor 0 는 3개의 지정 위치 (OPEN 0, STEP1 1, STEP2 2)

motor 1 는 2개의 지정 우치 (OPEN 0, CLOSE 1)
기구 조립이 완료되며 기본 위치가 미리 지정되며, MOVE_POS 명령 만으로 조작.

- **SET_POS(0x09)**: 현재의 위치를 저장 기본 위치에 지정 : 0-2
    - 예: 2 입력 → 현재 위치를 2번(index=2) 위치로 저장
      미리 설정된 지정 위치를 변경하고자 할때 이동 후 현재 위치로 지정 값을 변경
      ※ 저장된 위치는 전원 리셋 시 초기화되며, 비휘발성 저장 기능은 아직 구현되어 있지 않습니다.

각 명령에서 argument 해석 방식이 다르므로, 반드시 motor1_control에 입력하는 명령과 argument의 단위를 일치시켜야 합니다.

참고:

- step/degree/rev 변환 공식 및 예시는 아래 "step 변환 공식" 섹션을 참고하세요.
- argument 값은 항상 32비트 정수로 입력하며, MSB first(big-endian)로 전송됩니다.

motor0_control (0x0112) / motor1_control (0x0116)

현재 하위 8bit 만 사용

명령어

- STOP 0x00 // 정지 (모터 정지, 이전 명령 취소)
- MOVE_TO 0x01 // 절대 위치로 이동
- HOME 0x02 // 홈 위치로 이동 (리미트 스위치까지)

추가 명령어

- MOVE_REL 0x05 // 상대 위치(relative)로 이동 주어진 양만큼 이동
- MOVE_DEG 0x06 // 각도 이동 (degrees 단위) (0.01 도 단위 입력, x 100 해서 정수로
- MOVE_REV 0x07 // 상대 회전 이동 (revolutions 단위) (0.01 단위 입력, x 100 해서 정수로 전달)전달)
- MOVE_POS 0x08 // 모터에 따라서 3개의 기본 설정된 위치로 이동 (index input : 0-2)
- SET_POS 0x09 // 현재 위치를 설정 위치로 지정 (index input : 0-2). 전원 유지시만 유지(저장 기능 고려 없음)

직접 이동 step 을 계산하려면 아래 수식을 참조하세요.

motor0_pos (0x0004/05), motor1_pos (0x0006/07) 의 현재 위치를 참조해서,
MOVE_TO(0x01) 을 사용하거나, 계산값을 그대로 MOVE_REL(0x05) 을 사용하시면 됩니다.

step 변환 공식

```
1. 모터0 (24BYJ-48) 스테퍼 모터 기본 스펙
   - 스텝 각도: 5.625°/32 = 0.176° per step
   - 1회전 스텝 수: 2048 steps/revolution
   - 동작 전압: 12V DC
   - 최소 속도: 10 steps/sec
   - 최대 속도: 400 steps/sec
   모터 1 (24BYG-48) 스테퍼 모터 기본 스펙
    - 스텝 각도: 5.625°/8 = 0.703125° per step
```

```
     - 1회전 스텝 수: 512 steps/revolution
     - 동작 전압: 12V DC
     - 최소 속도: 10 steps/sec
     - 최대 속도: 650 steps/sec

  2. 각도 변환
    - 모터 0 스텝당 각도: 0.088° (5.625°/32)
    - 1도 회전 = 1° × (2048 steps/360°) = 5.69 steps/degree ≈ 6 steps
    예시:
       90도 회전 = 90° × 5.69 = 512 steps
       180도 회전 = 180° × 5.69 = 1024 steps
       360도 회전 = 360° × 5.69 = 2048 steps (1회전)
    - 모터 1 스텝당 각도: 0.703125° (5.625°/8)
    - 1도 회전 = 1° × (512 steps/360°) ≈ 1.42 steps/degree
    예시:
       90도 회전 = 90° × 1.422 = 128 steps
       180도 회전 = 180° × 1.422 = 256 steps
       360도 회전 = 360° × 1.422 = 512 steps (1회전)

  3. 32비트 position 값 범위
     - 최소값: 0 steps (원점)
     - 최대값: 0x7FFFFFFF steps (약 ±2,147,483,647 steps)
     - 최대 이동 거리: ±1,342,177mm (약 ±1.34km)
```

motor0_accel (0x0118) / motor1_accel (0x011A)

가속도 값이 0 이면 가속도를 사용하지 않습니다. 범위: 10 ~ 100

```
#define MOTOR_ACCEL_MIN     10
#define MOTOR_ACCEL_DEFAULT 25
#define MOTOR_ACCEL_HIGH    50
#define MOTOR_ACCEL_MAX     100
```

- 기구 적용 테스트 해보면서 값이 바뀌면 변경하겠습니다.

motor0_speed (0x0119) / motor1_speed (0x011B)

범위 : 10 - 650 홈으로 이동 시에는 저속으로 변경된 후에 완료 후 기존 속도롤 원복 합니다.

모터 0 (24BYJ-48) : 10 ~ 400
모터 1 (24BYG48) : 10 ~ 650

- 기구 적용 테스트 해보면서 값이 바뀌면 변경하겠습니다.

자동 전력 관리

현재 구현 특징:

스템 모터는 enable 상태를 유지하면 발열이 발생되므로, Stop 이후 일정 시간이 흐르면 disable 상태로 자동 전환합니다. 현제는 1000msec 설정되어 있습니다.
disable 조절 gpio 포트르 두 모터가 공유하고 있으므로, 실제 하드웨어적인 disable 은 두 모터가 모두 정지한 후 1000 msec 입니다.

시나리오 제어 영역 (0x0120~0x012F)

주소 (Address)	Read/Write	이름 (Name)	설명 (Description)	데이터 타입 (Data Type)	기본값 (Default Value) MSB First
0x0120	R/W	scenario_cmd	시나리오 명령 레지스터	uint16_t	0 시나리오 비활성(모든 동작 중지)
0x0121	R/W	step1_sec	1단계 유지 시간 (초)	uint16_t	0(진행안함)
0x0122	R/W	step2_sec	2단계 유지 시간 (초)	uint16_t	0(진행안함)
0x0123	R/W	temp1_x10	step1 목표 온도 (0.1°C 단위)	uint16_t	700
0x0124	R/W	temp2_x10	step2 목표 온도 (0.1°C 단위)	uint16_t	950
0x0125	R/W	temp_ok_sec	온도 OK 유지 시간(초)	uint16_t	30 초 예열 check 시간
0x0126	R/W	temp_pid_start_x10	차이가 이하일 대 PID 제어 시작 온도	uint16_t	100 (10.0°C)
0x0127	R/W	temp_preheatx10	예열 온도 x10	uint16_t	600 (60.0°C)
0x0128	R/W	pwm_max_value	PWM 최대값 (0~1000: % 아님)	uint16_t	300
0x0129~0x012F	-	reserved	시나리오 확장/예약	-	

필드별 상세 설명

- **scenario_cmd (0x0120)**: 시나리오 실행/정지/저장 등 명령을 위한 레지스터.
  쓰기 전용, 현재 UI 상태는 status(0x0000) 비트 8-15에 반영됨
- **step1_sec (0x0121), step2_sec (0x0122)**: 각 단계별 유지 시간(초). 0이면 해당 단계 skip.
- **temp1_x10 (0x0123), temp2_x10 (0x0124)**: 각 단계별 목표 온도(0.1°C 단위, 예: 250=25.0°C).
- **temp_ok_sec (0x0125)**: 목표 온도 도달 후 OK 상태를 유지해야 하는 시간(초). 예열 체크 등에서 사용.

- **temp_pid_start_x10 (0x0126)**: 예열→PID 전환 임계 온도(0.1°C 단위). 예: 100=10.0°C.
- **temp_preheatx10 (0x0127)**: 예열 단계 목표 온도(0.1°C 단위). 예: 600=60.0°C.
- **pwm_max_value (0x0128)**: PWM 최대값(0~1000, % 아님). 시나리오 동작 중 PWM duty 상한 제한.
- **reserved (0x0129~0x012F)**: 확장/예약 영역. 향후 단계 추가, 파라미터 확장 등에 사용.

scenario_cmd (0x0120) 및 상태 설명

enum 값	의미 및 동작 설명
0(0x00) UI_SCENARIO_OFF	시나리오 비활성(모든 시나리오 동작 중지)
1(0x01) UI_STOP	히터 OFF, 대기
2(0x02) UI_READY	기본 예열(60도)
3(0x03) UI_SELECT_SLOT	카트리지 정보에 따라 예열 시작
4(0x04) UI_SLOT_OPEN_READY	2차 예열 완료, 슬롯 오픈 명령 대기
5(0x05) UI_SLOT_OPEN	슬롯 오픈 동작
6(0x06) UI_WAIT_CAT_IN	슬롯 오픈 완료하고 카트리지 삽입 대기(삽입 시 증폭 예열 시작)->삽입되면 다음단계
7(0x07) UI_SLOT_CLOSE_READY	슬롯 클로즈 명력 대기
8(0x08) UI_SLOT_CLOSE	슬롯 클로즈 동작
9(0x09) UI_WAIT_SW1	SW1(버튼1) 이동 시키고 완료 되면 다음 단계
10(0x0A) UI_WAIT_STEP1	1차 증폭 완료 대기
11(0x0B) UI_WAIT_SW2	SW2(버튼2) 이동 시키고 완료 되면 다음 단계
12(0x0C) UI_WAIT_STEP2	2차 증폭 완료 대기
13(0x0D) UI_SLOT_OPEN2_READY	2차 증폭 완료되어 2차 슬롯 오픈 명령 대기
14(0x0E) UI_SLOT_OPEN2	2차 오픈 동작 진행하고 완료되면 다음 단계
15(0x0F) UI_WAIT_CAT_OUT	사용자가 카트리지 분리할때까지 대기
16(0x10) UI_SLOT_CLOSE2_READY	카드리지가 분리되면 2차 클로즈 명령 대기
17(0x11) UI_SLOT_CLOSE2	2차 클로즈 동작 완료되며 다음 단계
18(0x12) UI_UV_ON	UV ON
19(0x13) UI_UV_OFF_WAIT	timeout 지연 후 off
20(0x14) UI_CMD_SAVE_NVM,	NVM 저장 명령 - not for ui change, flash 저장 응답 지연때문에 동작중에 call 자제

enum 값	의미 및 동작 설명
21(0x15) UI_CMD_MAX	상태 개수 한계값
0x40 UI_ERR_MASK	에러 상태 플래그(0x80, OR 연산으로 에러 상태 표현)
0x7F UI_STEP_MAX	상태 최대값(0xFF)

각 커맨드는 상태 전이 시 하드웨어 제어(히터, 모터, 센서 등)와 타이머, 에러 처리, 사용자 입력(버튼/센서) 등을 연동하여 자동화 시나리오를 구현합니다. 에러 상태는 UI_ERR_MASK(0x40)와 OR 연산으로 표현되며, 각 단계별 타임아웃/센서 미동작 시 에러 상태로 전이됩니다.

초기 상태는 UI_SCENARIO_OFF 이므로 UI_READY 명령을 주지 않는 한 시나리오 모드는 동작하지 않는다.

> **정책/구현 주의:**
>
> - 구조체/코드/헤더/문서의 필드 순서, 타입, 기본값, 엔디안 변환 정책이 반드시 100% 일치해야 함.
> - 16bit 값은 그대로, 32bit/float은 MB_HOST_TO_REG_U32/F32 등 변환 매크로 필수.
> - 시나리오 제어 영역은 확장성을 고려해 reserved 필드 확보.
> - 실제 동작/저장/파라미터 변경 시 코드-문서-헤더 동기화 필수.
