app_commands.dart

소스 코드

전체 코드

```dart
import 'dart:typed_data';

enum PORT_COMMANDS {
  close(AppCommands.cmdClose),
  handshake(AppCommands.cmdHandshake),
  open(AppCommands.cmdOpens),
  postper(AppCommands.cmdPostper),
  postData(AppCommands.cmdPostData),
  cmdPhone(AppCommands.cmdPhone),
  recheck(AppCommands.cmdClose),
  sleep(AppCommands.cmdSleep),
  driverTrue(AppCommands.returnDriverTrue),
  driverFalse(AppCommands.returnDriverFalse),
  openB(AppCommands.cmdOpenb),
  ;

  final String command;

  // enum 생성자
  const PORT_COMMANDS(this.command);

  Uint8List toUint8List() {
    return Uint8List.fromList(command.codeUnits);
  }

  static Uint8List toUint8ListByString(String command) {
    return Uint8List.fromList(command.codeUnits);
  }

  static String getValidPhoneCommand(String phone) {
    return "${AppCommands.validPhonePrefix}$phone${AppCommands.validPhoneSuffix}";
  }
}

enum PORT_RESPONSES {
  // 각 enum 멤버는 AppCommands에 정의된 실제 응답 문자열을 값으로 가집니다.
  ok(AppCommands.returnOk),
  fail(AppCommands.returnFail),
  full(AppCommands.returnFull),
  reject(AppCommands.returnReject),
  open(AppCommands.returnOpen),
  notAuth(AppCommands.returnNotAuth),
  driverTrue(AppCommands.returnDriverTrue),
  driverFalse(AppCommands.returnDriverFalse),
  stmSleep(AppCommands.returnStmSleep),
```

```dart
  ;

  // enum이 실제 응답 문자열 값을 저장할 final 변수
  final String response;

  // enum 생성자
  const PORT_RESPONSES(this.response);

  /// 수신된 문자열(value)을 기반으로 일치하는 PORT_RESPONSES enum 멤버를 찾습니다.
  /// 일치하는 멤버가 없으면 null을 반환합니다.
  static PORT_RESPONSES? byValue(String value) {
    for (var resp in values) {
      if (resp.response == value) {
        return resp;
      }
    }
    return null;
  }
}

class AppCommands {
  static const String testMeasure = "[TEST]MEASURE";

  static const String cmdHandshake = "[CMD]HANDSHAKE";

  static const String cmdPhone = "[VALID]--ENDSTR";
  static const String validPhonePrefix = "[VALID]";
  static const String validPhoneSuffix = "ENDSTR"; // 여기도 # 제거

  static const String cmdOpens = "[CMD]OPENS";
  static const String cmdClose = "[CMD]CLOSES";
  static const String cmdPostper = "[CMD]POSTPER";

  static const String cmdSleep = "[CMD]SLEEP";
  static const String cmdOpenb = "[CMD]OPENB";
  static const String cmdOpenv = "[CMD]OPENV";
  static const String cmdClosev = "[CMD]CLOSEV";
  static const String cmdOff = "[CMD]OFF";
  static const String cmdTest = "[TEST]";
  static const String cmdPostData = "[CMD]POSTDATA";
  static const String returnDriverTrue = "${prefixAnswer}DRIVER:TRUE";
  static const String returnDriverFalse = "${prefixAnswer}DRIVER:FALSE";
  static const String returnStmSleep = "${prefixAnswer}STM_SLEEP";

  static const String prefixAnswer = "[ANS]";
  static const String returnOk = "${prefixAnswer}OK";
  static const String returnFail = "${prefixAnswer}FAIL";
  static const String returnReject = "${prefixAnswer}REJECT";
  static const String returnOpen = "${prefixAnswer}OPEN";
  static const String returnNotAuth = "${prefixAnswer}NOTAUTH";
  static const String returnFull = "${prefixAnswer}FULL";

  // 정규식 수정: 끝에 #가 있을 수도 있고 없을 수도 있도록 처리
```

```dart
  static final RegExp regExp = RegExp(r"O(\d+(?:\.\d+)?)W(\d+(?:\.\d+)?).*#?$");

  static (double oil, double water) returnOilWaterFormat(String input) {
    double oil = 0.0;
    double water = 0.0;
    final Match? match = regExp.firstMatch(input);
    if (match != null && match.groupCount == 2) {
      String? oilString = match.group(1);
      String? waterString = match.group(2);

      print("Extracted oil string: $oilString"); // "7.89"
      print("Extracted water string: $waterString"); // "1.23"

      // 추출된 문자열을 double로 변환합니다. 변환 실패 시 null이 될 수 있으므로 tryParse 사용
      if (oilString != null) {
        oil = double.tryParse(oilString) ?? 0.0;
      }
      if (waterString != null) {
        water = double.tryParse(waterString) ?? 0.0;
      }
    }

    return (oil, water);
  }

}
```

1 파일 개요

파일명: `lib/common/appcommands.dart`
역할: Flutter 앱에서 사용되는 **포트 명령어 및 응답 관리** 클래스와 enum 정의
주요 목적:

- 장치와 통신 시 사용되는 **명령어(Command)와 응답(Response)**를 체계적으로 관리
- **문자열 ↔ 바이트(Uint8List)** 변환 제공
- **전화번호 검증 명령어** 생성 기능
- 장치로부터 수신된 **Oil/Water 데이터 파싱**

사용 맥락 예시:

- BLE, UART, Serial 통신 시 명령어 전송 및 응답 처리
- 장치 상태 확인, 드라이버 상태, 기기 제어 명령 관리
- Oil/Water 센서 데이터 추출

---

2 주요 기능

1. **PORT_COMMANDS enum**
    - 장치로 전송할 **명령어(Command)** 정의
    - 각 enum 멤버가 **실제 문자열 명령어(AppCommands 내 정의)와 연결**

- 문자열 → `Uint8List` 변환 (`toUint8List()`)
- 전화번호 기반 명령어 생성 (`getValidPhoneCommand`)

2. **PORT_RESPONSES enum**
   - 장치로부터 수신되는 **응답(Response)** 정의
   - 문자열 기반 매칭 (`byValue`)
   - 일치하는 응답이 없으면 `null` 반환
3. **AppCommands 클래스**
   - **실제 문자열 명령어 상수** 정의
     - 예: `cmdHandshake`, `cmdClose`, `returnOk` 등
   - **정규식 기반 Oil/Water 데이터 추출** (`returnOilWaterFormat`)
   - **문자열 포맷 처리**, 접두사/접미사 정의 (`prefixAnswer`, `validPhonePrefix`)
4. **데이터 파싱**
   - `RegExp r"O(\d+(?:\.\d+)?)W(\d+(?:\.\d+)?).*#?$"`
     - "O7.89W1.23#" 같은 문자열에서 oil=7.89, water=1.23 추출
   - 안전한 double 변환 (`tryParse`)

3️⃣ 구조 분석

```
AppCommands.dart
 ├─ class AppCommands
 |    ├─ 명령어 상수(cmdHandshake, cmdClose, cmdPhone, ...)
 |    ├─ 응답 상수(returnOk, returnFail, ...)
 |    ├─ 정규식(RegExp regExp)
 |    └─ 데이터 파싱 함수(returnOilWaterFormat)
 ├─ enum PORT_COMMANDS
 |    ├─ close, handshake, open, postper, ...
 |    ├─ command: String
 |    ├─ toUint8List()
 |    ├─ toUint8ListByString()
 |    └─ getValidPhoneCommand()
 └─ enum PORT_RESPONSES
      ├─ ok, fail, full, reject, open, ...
      ├─ response: String
      └─ byValue(String) → PORT_RESPONSES?
```

특징:

- **Enum 기반 명령어/응답 관리**로 오타 방지
- **문자열 ↔ Uint8List** 변환을 통합
- **전화번호 명령어 포맷 처리** 내장
- **Oil/Water 데이터 파싱** 포함

---

4 동작 흐름 예시

1. 앱에서 장치와 통신을 위해 `PORT_COMMANDS.handshake.toUint8List()` 호출 → `[CMD]HANDSHAKE` 문자열을 바이트 배열로 변환
2. 장치 응답 수신 → 문자열 비교: `PORT_RESPONSES.byValue("ANS:OK")` → `PORT_RESPONSES.ok`
3. 전화번호 검증 명령어 생성 → `PORT_COMMANDS.getValidPhoneCommand("01012345678")` → `[VALID]01012345678ENDSTR`
4. Oil/Water 데이터 수신 → `AppCommands.returnOilWaterFormat("O7.89W1.23#")` → `(oil:7.89, water:1.23)`

---

5 장점

- **타입 안전성**: 명령어와 응답을 enum으로 관리
- **재사용성**: 문자열 ↔ Uint8List 변환, 전화번호 명령어, 데이터 파싱 함수 포함
- **정규식 파싱 안전**: Null, tryParse 처리로 오류 방지
- **확장성**: 명령어/응답 추가 용이

---

6 단점 / 개선점

1. **문자열 상수 관리**
    - 일부 상수 이름이 일관적이지 않음 (`cmdOpens` vs `cmdOpenb`)
    - 개선: 명령어 타입별 접두사 규칙 통일
2. **데이터 파싱 유연성 부족**
    - 현재 Oil/Water 포맷에 하드코딩(`O\d+W\d+`)
    - 개선: 다른 센서 포맷 지원 가능하도록 파라미터화
3. **유닛 테스트 부족**
    - 명령어 생성, 응답 매핑, Oil/Water 파싱 기능 테스트 필요
4. **중복 코드**
    - `toUint8List()` vs `toUint8ListByString()` 거의 동일
    - 개선: 코드 통합 가능
5. **상태 관리 없음**
    - 현재 단순 상수/파싱 중심, 통신 상태나 에러 처리 기능 없음
    - 개선: 통신 성공/실패 로그, 리트라이 기능 추가 가능

---

7 개선된 사용 예시 (개념)

```
// 장치 핸드셰이크 전송
final handshakeBytes = PORT_COMMANDS.handshake.toUint8List();
sendToDevice(handshakeBytes);
```

```dart

// 장치 응답 확인
final response = PORT_RESPONSES.byValue(receivedString);
if (response == PORT_RESPONSES.ok) {
  print("Handshake OK");
}

// 전화번호 명령어 전송
final phoneCmd = PORT_COMMANDS.getValidPhoneCommand("01012345678");
sendToDevice(Uint8List.fromList(phoneCmd.codeUnits));

// Oil/Water 데이터 추출
final (oil, water) = AppCommands.returnOilWaterFormat("O7.89W1.23#");
print("Oil=$oil, Water=$water");
```

- **안정적인 명령어 전송/응답 매칭** 가능
- **데이터 파싱 후 바로 사용 가능**