ค่า **CRC-16 Modbus** ที่คำนวณได้จากชุดข้อมูลนี้คือ **`0xEA7E`**

เมื่อนำมาต่อท้ายคำสั่งในรูปแบบ ไบต์ต่ำ (Low Byte) ตามด้วย ไบต์สูง (High Byte) จะได้ชุดคำสั่งที่สมบูรณ์ดังนี้ครับ: [1]

🎁 คำสั่งที่ใส่ CRC เรียบร้อยแล้ว

```
uint8_t all_relay_on[] = {0x04, 0x0F, 0x00, 0x00, 0x00, 0x08, 0x01, 0xFF, 0x7E, 0xEA};
```

🔍 ความหมายของแต่ละไบต์ (Modbus RTU)

- **`0x04`** : หมายเลขบอร์ด (Slave Address)
- **`0x0F`** : คำสั่งเขียนข้อมูลหลายจุดพร้อมกัน (Write Multiple Coils)
- **`0x00, 0x00`** : ตำแหน่งเริ่มต้นที่รีเลย์ตัวที่ 1 (Starting Address)
- **`0x00, 0x08`** : จำนวนรีเลย์ที่ต้องการสั่งงานทั้งหมด 8 ตัว (Quantity of Coils)
- **`0x01`** : จำนวนไบต์ของข้อมูลที่จะส่งมาควบคุม (Byte Count)
- **`0xFF`** : ข้อมูลควบคุม (Binary: `11111111` หมายถึงสั่งเปิดครบทั้ง 8 รีเลย์)
- **`0x7E`** : CRC Low Byte
- **`0xEA`** : CRC High Byte [2]

```
def crc16_modbus(data: bytes) -> int:
    crc = 0xFFFF
    for byte in data:
        crc ^= byte
        for _ in range(8):
            if crc & 1:
                crc = (crc >> 1) ^ 0xA001
            else:
                crc >>= 1
    return crc

data = bytes([0x04, 0x0f, 0x00, 0x00, 0x00, 0x08, 0x01, 0xff])
crc = crc16_modbus(data)
low_byte = crc & 0xFF
high_byte = (crc >> 8) & 0xFF
print(f"CRC: {crc:04X}, Low: {low_byte:02X}, High: {high_byte:02X}")
```