โครงการอบรมเชิงปฏิบัติการ Upskill/Reskill

เอกสารพัฒนาเฟิร์มแวร์ UDS4ECU

ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยธรรมศาสตร์

2026-04-23

1. Firmware Requirements (UDS4ECU)

UDS4ECU คือเฟิร์มแวร์สำหรับบอร์ด NUCLEO-C562RE ทำหน้าที่เป็น Electronic Control Unit (ECU) จำลอง โดยใช้โปรโตคอล **ISO 14229 (UDS)** บน **ISO 15765-2 (ISO-TP)** เพื่อตอบสนองต่อคำสั่งอ่านค่าสถานะจากอุปกรณ์ทดสอบ (Tester) ผ่านบัส CAN

1.1 Interface & Protocol Specification

- **Physical Layer:** CAN 2.0B / CAN FD (500 kbps Nominal Rate)
- **Transport Layer: ISO-TP (ISO 15765-2)** สำหรับการจัดการเฟรมข้อมูล (Segmentation) ที่ยาวเกิน 8 ไบต์ในกรณี CAN 2.0B หรือ 64 ไบต์ในกรณี CAN-FD
- **Application Layer: UDS (ISO 14229-1)** เน้นบริการการอ่านข้อมูล (Read Data by Identifier)
- **Addressing Mode:** - **Physical Request ID:** `0x7E0` (คำสั่งจาก PC ไปยัง ECU)
  - **Physical Response ID:** `0x7E8` (คำตอบจาก ECU ไปยัง PC)

1.2 Diagnostic Data Identifiers (DID) Definition

ขอบเขตของการอ่านข้อมูลสถานะใน Workshop นี้จะถูกนิยามผ่าน **Data Identifiers (DID)** ขนาด 2 ไบต์ ดังนี้:

DID (Hex)	ข้อมูลที่อ่าน (Parameters)	ขนาด (Size)	รูปแบบข้อมูล (Data Format)
0xF101	สถานะปุ่มกด (User Button - PC13)	1 Byte	0x00=Released, 0x01=Pressed
0xF102	ค่าแรงดัน ADC ช่อง A0	2 Bytes	0-3300 (mV) แบบ Big-endian
0xF103	ค่าแรงดัน ADC ช่อง A1	2 Bytes	0-3300 (mV) แบบ Big-endian
0xF104	ค่าแรงดัน ADC ช่อง A2	2 Bytes	0-3300 (mV) แบบ Big-endian
0xF105	ค่าแรงดัน ADC ช่อง A3	2 Bytes	0-3300 (mV) แบบ Big-endian
0xF100	**Group Read:** ปุ่มกด + ADC A0-A3	9 Bytes	รวมข้อมูล DID 0xF101-0xF105

1.3 Supported UDS Services

เฟิร์มแวร์ต้องรองรับบริการพื้นฐาน (Mandatory Services) ดังต่อไปนี้:

1. **Diagnostic Session Control (0x10):**
   - รองรับ Default Session (`0x01`) และ Extended Diagnostic Session (`0x03`)

2. **Tester Present (0x3E):**
    - เพื่อรักษาการเชื่อมต่อใน Extended Session
3. **Read Data by Identifier (0x22):**
    - อ่านค่าปุ่มกดและ ADC ตามตาราง DID ด้านบน

1.4 Functional Requirements

1. **Atomic Sampling:** เมื่อได้รับคำสั่งอ่านกลุ่มข้อมูล (DID 0xF100) เฟิร์มแวร์ต้องล็อคค่าจาก ADC และสถานะปุ่มกด เพื่อความสอดคล้องของข้อมูล (Data Consistency)
2. **ISO-TP Management:** เนื่องจากข้อมูล DID 0xF100 มีขนาด 9 ไบต์ (เกิน 8 ไบต์ในกรณี CAN 2.0B) เฟิร์มแวร์ต้องสามารถจัดการการส่งแบบ **First Frame (FF)** และรับ **Flow Control (FC)** เพื่อส่ง **Consecutive Frame (CF)** ได้ถูกต้อง
3. **Error Handling (Negative Response):** - หากมีการขอ DID ที่ไม่มีในสารบบ ต้องตอบกลับด้วย Negative Response Code (NRC) `0x31` (Request Out of Range)
    - หากส่งรูปแบบคำสั่งผิด ต้องตอบกลับด้วย NRC `0x13` (Incorrect Message Length Or Invalid Format)

1.5 Hardware Resource Mapping

ฮาร์ดแวร์	ขาใช้งาน	หน้าที่ในระบบ
Digital Input	PC13	ตรวจสอบสถานะปุ่มกด (Active Low)
Analog Input	PA0, PA1, PA4, PB0	ช่องวัดแรงดัน A0, A1, A2, A3 (ผ่าน ADC1)
FDCAN1	PB8, PB9	รับ-ส่งข้อความ UDS ผ่านบัส CAN
Status LED	PA5	กระพริบตามสถานะ Diagnostic Session

2. Firmware Design

การออกแบบเฟิร์มแวร์ UDS4ECU มุ่งเน้นไปที่การสร้างระบบตอบสนองคำสั่งวินิจฉัย (Diagnostic Response) ที่มีความน่าเชื่อถือ โดยมีเลเยอร์ ISO-TP เป็นตัวจัดการการรับส่งข้อมูลที่มีความยาวผันแปร

2.1 Functional Block Diagram

สถาปัตยกรรมซอฟต์แวร์ถูกแบ่งเป็น 4 เลเยอร์ เพื่อรองรับมาตรฐาน ISO 14229:

1. **Hardware Layer:** จัดการ FDCAN และ ADC Peripherals รวมถึง GPIO ผ่าน HAL API
2. **Transport Layer (ISO-TP):** จัดการการหั่นข้อมูล (Segmentation) และการรวมเฟรม (Reassembly)
3. **Diagnostic Service Layer:** ตีความคำสั่ง UDS (เช่น 0x22, 0x10) และตรวจสอบความถูกต้องของ DID
4. **Application Data Layer:** เชื่อมโยงค่าจากปุ่มกดและ ADC เข้ากับตาราง DID ที่กำหนดไว้

2.2 Behavior Design (Main State Machine)

หัวใจของ UDS คือการจัดการ **Diagnostic Session** ซึ่งควบคุมสิทธิ์ในการเข้าถึงข้อมูล:

สถานะ (State)	รายละเอียด (Description)	สิทธิ์การเข้าถึง (Access Rights)
DEFAULT_SESSION	สถานะเริ่มต้นหลัง Boot	อ่านค่า DID ทั่วไปได้ (0xF101-0xF105)
EXTENDED_SESSION	เข้าถึงผ่านคำสั่ง `10 03`	อ่าน Group DID (0xF100) และปรับแต่งค่าระบบ

สถานะ (State)	รายละเอียด (Description)	สิทธิ์การเข้าถึง (Access Rights)
SAFETY_LOCK	สถานะป้องกันเมื่อเกิด Error	ระงับการตอบสนองคำสั่งชั่วคราว

2.3 ISO-TP Communication Flow (Frame Segmentation)

เพื่อให้บอร์ด ECU สามารถส่งข้อมูล Group DID (0xF100) ที่มีความยาวมากกว่า 8 ไบต์ (CAN 2.0B) หรือ 64 ไบต์ (CAN-FD) กลับไปยังคอมพิวเตอร์ เฟิร์มแวร์จะใช้กลไก **Tx-Segmentation** ดังนี้:

2.3.1 ลำดับการรับส่งข้อมูล (Communication Sequence)

1. **First Frame (FF):** ECU เริ่มต้นการส่งข้อมูลโดยบรรจุค่า PCI ประเภท `0x1` พร้อมระบุความยาวข้อมูลทั้งหมด (9 ไบต์)
2. **Flow Control (FC):** ECU หยุดรอรับการยืนยันจาก Tester โดยต้องได้รับค่าสถานะ `FS=0` (Continue to send) จึงจะดำเนินขั้นตอนถัดไป
3. **Consecutive Frame (CF):** ECU ส่งข้อมูลส่วนที่เหลือโดยระบุ PCI ประเภท `0x2` พร้อมลำดับเฟรม (Sequence Number)

2.3.2 โครงสร้าง PCI สำหรับ Minimal ISO-TP

ประเภทเฟรม	Byte 0 (PCI)	Byte 1	รายละเอียด
First Frame (FF)	`0x10`	`0x09`	`1`=Type, `009`=Total Length (9 bytes)
Flow Control (FC)	`0x30`	`0x00`	`3`=Type (ECU รอรับค่านี้จาก PC)
Consecutive Frame (CF)	`0x21`	Data	`2`=Type, `1`=Sequence Number

หมายเหตุ: หากใช้งานบน **CAN FD** ระบบจะยุบกระบวนการทั้งหมดให้เหลือเพียง **Single Frame (SF)** ขนาดใหญ่เพียงเฟรมเดียว ทำให้ Latency ลดลงอย่างมาก

2.4 Data Packing for ADC Group (DID 0xF100)

ข้อมูลจะถูกบรรจุลงใน Buffer แบบเรียงลำดับ (Serialization) ก่อนส่งออกทาง ISO-TP:

Byte Index	Data Field	Source Variable	Format
0	Button Status	PC13 Status	0x00 / 0x01
1-2	ADC A0	`ADC_Value[0]`	Big-endian (High/Low)
3-4	ADC A1	`ADC_Value[1]`	Big-endian (High/Low)
5-6	ADC A2	`ADC_Value[2]`	Big-endian (High/Low)
7-8	ADC A3	`ADC_Value[3]`	Big-endian (High/Low)

2.4 Diagnostic Data Handling Matrix

ตารางสรุปการประมวลผลคำสั่ง Read Data by Identifier (Service 0x22):

ขั้นตอน (Step)	กิจกรรม (Activity)	ผลลัพธ์ (Outcome)
1. Request	รับข้อมูล `[22] [DID_HI] [DID_LO]`	เข้าสู่กระบวนการตรวจสอบ DID
2. Validation	เช็คว่า DID อยู่ในตารางที่กำหนดหรือไม่	หากไม่มี ตอบกลับ NRC 0x31 (Request Out of Range)
3. Access Check	เช็คว่า Session ปัจจุบันมีสิทธิ์อ่านหรือไม่	หากไม่มี ตอบกลับ NRC 0x7F (Service Not Supported)
4. Data Capture	อ่านค่าจาก PC13 และ ADC1 (Ch A0-A3)	ข้อมูลถูกบรรจุลงใน Response Buffer
5. Response	ส่งข้อมูลกลับในรูปแบบ `[62] [DID_HI] [DID_LO] [DATA...]`	สถานะการอ่านสมบูรณ์

2.5 Timing & Performance Requirements

เพื่อให้เป็นไปตามมาตรฐาน ISO 14229 เฟิร์มแวร์ต้องควบคุมจังหวะเวลาดังนี้:

1. **P2 Server Time (50ms):** ECU ต้องตอบกลับคำสั่งภายใน 50ms หลังจากได้รับ Request
2. **P2* Server Time (500ms):** กรณีที่การประมวลผล ADC ใช้เวลานาน ระบบต้องส่ง Pending Response เพื่อขอเวลาเพิ่ม
3. **S3 Server Timeout (5000ms):** หากอยู่ใน Extended Session แล้วไม่มีคำสั่งใหม่เข้ามาใน 5 วินาที ระบบจะกลับสู่ Default Session อัตโนมัติ

3. Firmware Implementation Design

โครงสร้างเฟิร์มแวร์ UDS4ECU ถูกออกแบบตามสถาปัตยกรรมแบบ Layered เพื่อแยกส่วนการจัดการ Physical Bus ออกจากลอจิกของโปรโตคอลวินิจฉัย ช่วยให้ผู้เรียนสามารถทำความเข้าใจการไหลของข้อมูลจากระดับบิต (CAN) ไปจนถึงระดับความหมายของข้อมูล (UDS DID)

3.1 Firmware Architecture & File Organization

การจัดวางไฟล์โค้ดถูกแบ่งตามหน้าที่ความรับผิดชอบ เพื่อให้ง่ายต่อการขยายผลในกรณีที่มีการเพิ่ม Service หรือ DID ใหม่:

Src files	Layer	Responsibility
`main.c`	Application	จัดการ Main Loop, เรียกใช้ Diagnostic Manager และคุม LED
`uds_service.c/h`	Service	ตีความ UDS Service (0x22, 0x10, 0x3E) และจัดการ Session
`iso_tp.c/h`	Transport	จัดการ ISO-TP State Machine (SF, FF, CF, FC)
`ecu_data.c/h`	Data	แมปค่าจาก ADC และ Button เข้ากับหมายเลข DID
`can_driver.c/h`	Hardware	Wrapper สำหรับ FDCAN HAL เพื่อส่งและรับเฟรมดิบ

3.2 Main Loop Execution Sequence

การทำงานในลูปหลักเน้นการจัดการ “Diagnostic Session” และการรักษาความปลอดภัยของระบบ:

1. **Check ISO-TP Queue:** ตรวจสอบว่ามีข้อความ UDS ที่ประกอบร่างเสร็จสมบูรณ์จาก ISO-TP หรือยัง

2. **UDS Request Dispatcher:** หากมี Request เข้ามา จะส่งไปที่ `uds_service.c` เพื่อตรวจสอบ Service ID
3. **Data Retrieval:** หากเป็นคำสั่งอ่านข้อมูล (0x22) ระบบจะไปดึงค่าล่าสุดจาก `ecu_data.c`
4. **Session Monitoring:** ตรวจสอบ S3 Timer เพื่อสลับกลับสู่ Default Session หากไม่มีกิจกรรม
5. **ISO-TP Transmission:** ส่งผลลัพธ์กลับผ่าน ISO-TP เพื่อหั่นข้อมูลเป็นเฟรมตามความเหมาะสม (SF/MF)

3.3 HW Configuration Summary (Diagnostic Optimized)

การตั้งค่า Peripherals บน STM32CubeMX สำหรับโปรเจกต์ UDS:

Peripheral	Mode	Key Settings	Purpose
FDCAN1	Classic / FD	ID: 0x7E0 (RX Filter), 0x7E8 (TX)	เฉพาะเจาะจงสำหรับ ECU ตัวนี้
ADC1	Continuous	Scan Mode, DMA Enabled	อัปเดตค่าแรงดัน A0-A3 ตลอดเวลา
TIM3	Interrupt	คาบ 1ms (Internal Tick)	ใช้จัดการ ISO-TP Timeout และ S3 Timer
GPIO PC13	Input	Pull-up, Falling Edge	อ่านสถานะ User Button

3.4 Detailed Design (Implementation Details)

3.4.1 ISO-TP State Machine

การส่งข้อมูลที่ยาวเกิน 8 ไบต์ (เช่น DID 0xF100) ต้องจัดการสถานะการส่งข้อมูลแบบ Multi-frame:

```
typedef enum {
    ISOTP_IDLE,
    ISOTP_WAIT_FIRST_FRAME,
    ISOTP_WAIT_FLOW_CONTROL,
    ISOTP_SEND_CONSECUTIVE_FRAME,
    ISOTP_ERROR
} ISOTP_State_t;
```

3.4.2 Diagnostic Identifier (DID) Mapping

การเชื่อมโยงฮาร์ดแวร์เข้ากับโปรโตคอล UDS ทำผ่านโครงสร้างแบบตาราง (Look-up Table) ที่ชี้ด้วย function pointer:

```
typedef struct {
    uint16_t did;              // DID (e.g., 0xF101)
    uint8_t  data_size;        // payload size in bytes
    void* data_ptr;         // Pointer to data
} UDS_DID_Mapping_t;

// mapping struct
const UDS_DID_Mapping_t did_table[] = {
    {0xF101, 1, &user_button_status},
    {0xF102, 2, &adc_value_a0},
    // ...
};
```