Электромобильдерге арналған микроконтроллер жүйелерін бағдарламалау

Дәуренов ӘД

8-дәріс. Микроконтроллерлердің тізбекті интерфейстері

Тақырыпты меңгергеннен кейін студент:

- микроконтроллерге тізбекті интерфейстер не үшін қажет екенін;
- тізбекті және параллель деректер берудің айырмашылығын;
- синхронды және асинхронды байланыстың айырмашылығын;
- UART жұмыс принципін;
- SPI жұмыс принципін;
- I²C жұмыс принципін;
- Master/Slave түсініктерін;
- TX, RX, SCK, MOSI, MISO, SDA, SCL және CS желілерінің қызметін;
- UART, SPI және I²C артықшылықтары мен шектеулерін;
- бұл интерфейстердің электромобильде қайда қолданылатынын;
- DMA не үшін қолданылатынын түсінуі тиіс.

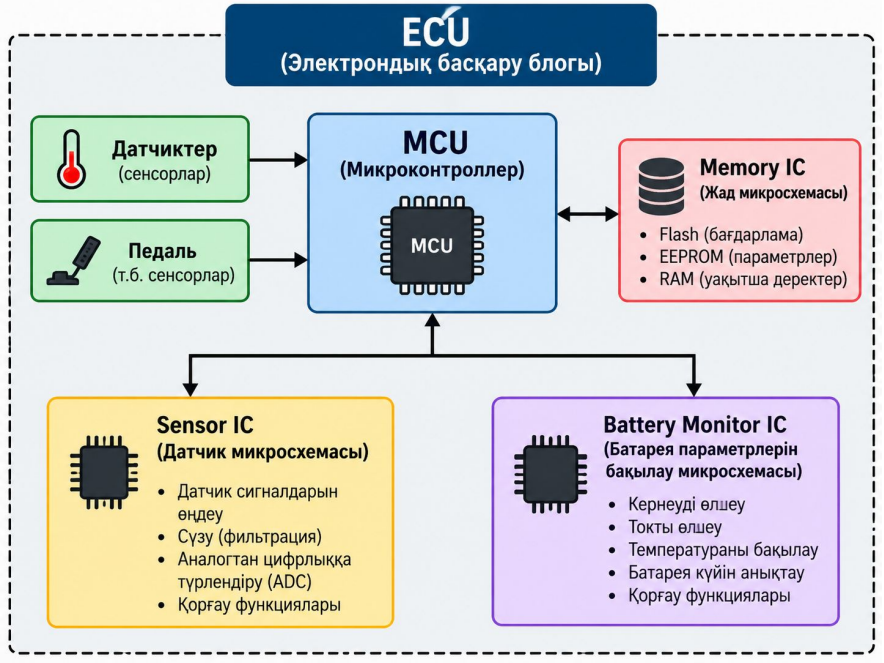
Микроконтроллерге байланыс интерфейстері не үшін қажет?

Алдыңғы тақырыптарда микроконтроллерді көбінесе жеке құрылғы ретінде қарастырдық:

Датчик → ADC → MCU → PWM → Атқарушы құрылғы

Бірақ нақты автомобильдік контроллер жеке жұмыс істемейді.

Бір электрондық ECU блоктың ішінде бірнеше микросхема болуы мүмкін:



Микроконтроллер осы құрылғылармен **цифрлық деректер** арқылы ақпарат алмасуы керек.
Мысалы: MCU → температураны сұрау → Sensor IC

Sensor IC → 42.5°C → MCU

немесе:

MCU → оқу командасы → Memory
Memory → сақталған деректер → MCU

Бұл үшін **байланыс интерфейстері** қолданылады.

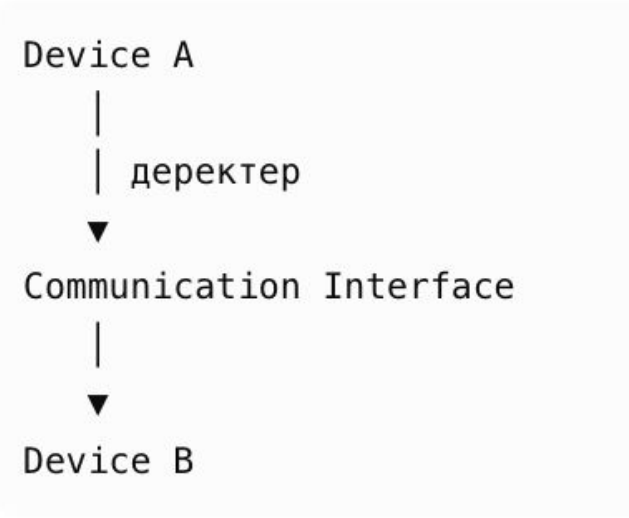
Интерфейс дегеніміз не?

Байланыс интерфейсі — екі немесе бірнеше электрондық құрылғының деректермен қалай алмасатынын анықтайтын электрлік желілер мен ережелер жиынтығы.

Интерфейс мыналарды анықтайды:

- қандай сымдар қолданылады;
- биттер қашан беріледі;
- деректер беру қалай басталады;
- деректер беру қалай аяқталады;
- қай құрылғы алмасуды басқарады;
- қабылдаушы құрылғы қалай анықталады.

Қарапайым түрде:



Цифрлық деректерді берудің екі негізгі тәсілі бар: тізбекті және параллель.

Мысалы, микроконтроллер 10110010 деген 8 биттік ақпаратты басқа құрылғыға жіберуі керек.

1. Тізбекті деректер беру (Serial Transmission)

Деректер бірінен кейін бірі, кезекпен жіберіледі.

Қарапайым мысал: бір есіктен адамдардың бір-бірден шығуы сияқты.

2. Параллель деректер беру (Parallel Transmission)

Бірнеше бит бір уақытта, бөлек желілер арқылы жіберіледі.

Қарапайым мысал: сегіз есіктен сегіз адамның бір уақытта шығуы сияқты.

Электромобильдегі мысал:

STM32 микроконтроллері температура датчигінен I²C арқылы ақпарат алса, бұл — тізбекті деректер беру. Ал микроконтроллердің ішкі параллель деректер шинасында бірнеше бит бір уақытта тасымалдануы мүмкін.

Яғни, ең бастысы: тізбекті — бірінен кейін бірі, параллель — бір уақытта қатар.

5. Параллель деректерді беру

Бірнеше бит бір уақытта беріледі. Мысалы: **10110110**



8 биттің мәні:	
D7	1
D6	0
D5	1
D4	1
D3	0
D2	1
D1	1
D0	0

Параллель берудің артықшылығы мен кемшілігі

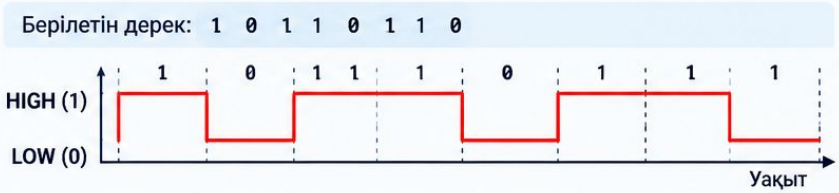
Артықшылығы:
- Жоғары жылдамдық
- Бір уақытта бірнеше бит беріледі

Кемшілігі:
- Көп сым қажет (8 бит үшін 8 желі)
- Ұзақ қашықтықта қолдану қиын

6. Тізбекті деректерді беру принципі

Тізбекті беру кезінде биттер **бірінен кейін бірі беріледі**. Мысалы: 1 → 0 → 1 → 1 → 0 → 1 → 1 → 0





Тізбекті интерфейстер (Serial Interfaces):

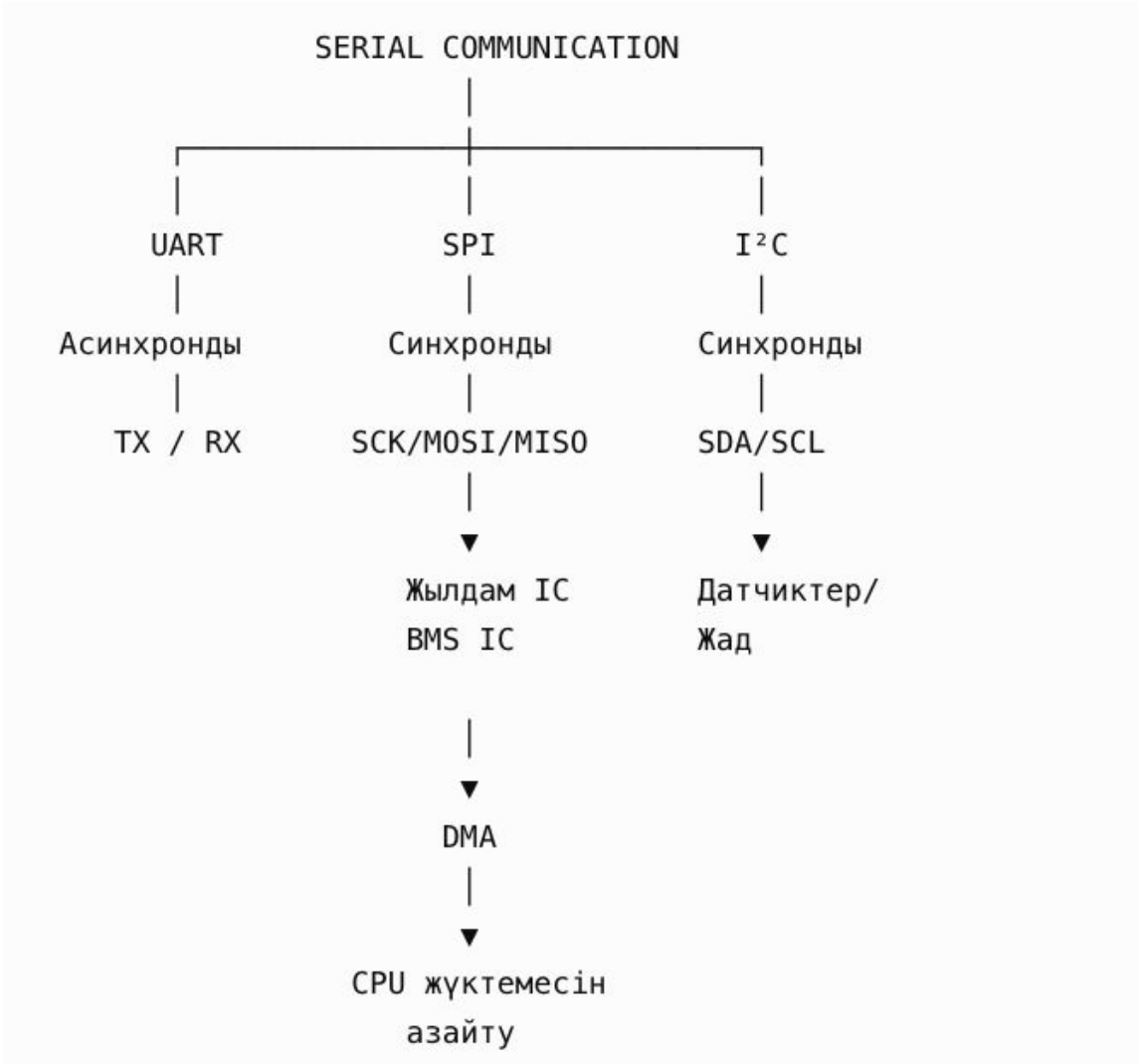
Тізбекті деректерді беру келесі интерфейстерде қолданылады:

UART	SPI	I²C
(асинхронды тізбекті байланыс)	(жылдам тізбекті интерфейс)	(екі сымды тізбекті интерфейс)

Тізбекті берудің артықшылығы мен кемшілігі

Артықшылығы:
- Аз сым қажет (тек бір немесе бірнеше желі)
- Ұзақ қашықтықта қолдануға болады

Кемшілігі:
- Жылдамдығы төменірек
- Биттер бірінен кейін бірі беріледі (кейде кідіріс болады)

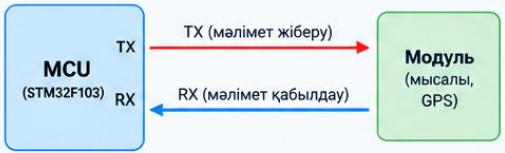

SERIAL COMMUNICATION
UART
SPI
I²C
Асинхронды
Синхронды
Синхронды
TX / RX
SCK/MOSI/MISO
SDA/SCL
Жылдам IC
BMS IC
Датчиктер/
Жад
DMA
CPU жүктемесін
азайту

7. MCU негізгі тізбекті интерфейстері (STM32F103 мысалында)

1 UART (асинхронды)

Екі құрылғы арасында қарапайым тізбекті байланыс. Синхрондау тактілік сигналсыз жүзеге асады.

Негізгі желілер	TX, RX
Синхрондау түрі	асинхронды
Қолданылуы	диагностика, логтау, модульдермен байланыс





2 SPI (синхронды)

Жоғары жылдамдықты тізбекті интерфейс. Бір мастер және бір немесе бірнеше слейв құрылғылармен жұмыс істейді.

Негізгі желілер	SCK, MOSI, MISO, CS
Синхрондау түрі	синхронды
Қолданылуы	сенсорлар, жад микросхемалары, дисплейлер (жоғары жылдамдық)

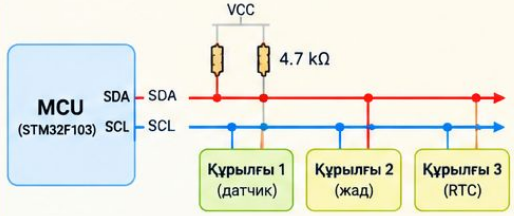


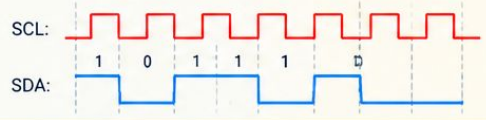


3 I²C (синхронды)

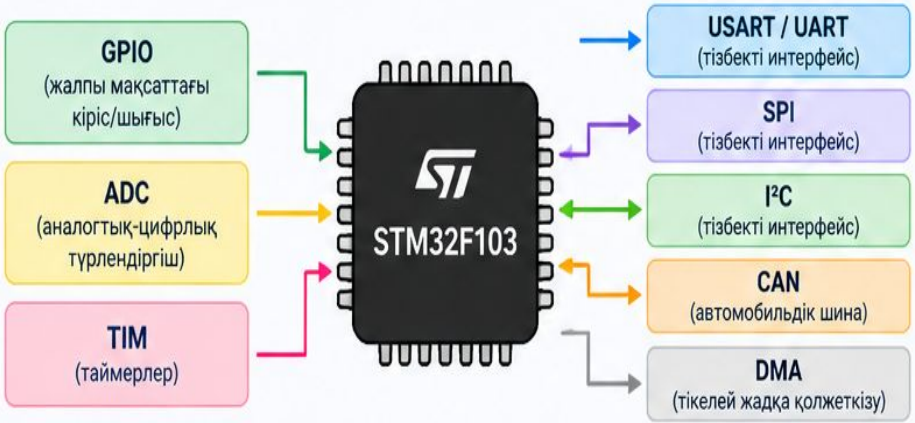
Екі сымды тізбекті интерфейс. Көп құрылғыны бір шинаға қосуға болады.

Негізгі желілер	SDA, SCL
Синхрондау түрі	синхронды
Қолданылуы	датчиктер, EEPROM, RTC, температура, қысым датчиктері





STM32F103 ішіндегі аппараттық модульдер



Интерфейс	Негізгі желілер	Синхрондау түрі
UART	TX, RX	асинхронды
SPI	SCK, MOSI, MISO, CS	синхронды
I²C	SDA, SCL	синхронды



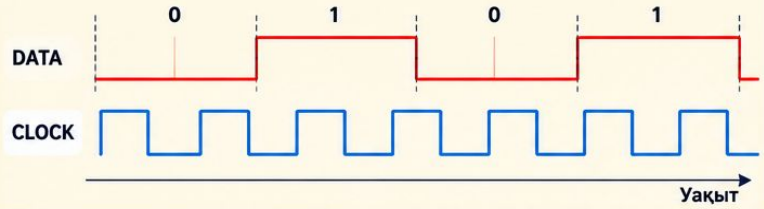
Қорытынды:

- UART — қарапайым, тактілік сигналсыз (асинхронды).
- SPI — жоғары жылдамдықты, тактілік сигналмен (синхронды).
- I²C — екі сымды, бірнеше құрылғыны қосуға болады (синхронды).
- STM32F103 бұл интерфейстердің барлығын аппараттық түрде қолдайды.

8. Синхронды байланыс

Синхронды беру кезінде деректер әр бит ұшін тактілік сигнал (Clock) арқылы синхрондалады.
Мысалы: SPI және I²C – синхронды интерфейстер.

Сигналдардың уақыттық диаграммасы



SPI мысалы (синхронды байланыс)



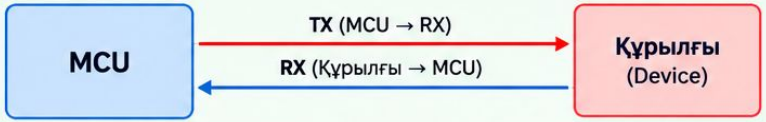
Синхронды байланыстың ерекшеліктері:

- Сағат (Clock) сигналы қолданылады.
- Деректер биттері Clock бойынша оқылады.
- Жоғары жылдамдықты беруге болады.
- Мысал: SPI, I²C.

9. Асинхронды байланыс

Асинхронды беру кезінде бөлек Clock желісі қолданылмайды.
Биттер алдын ала келісілген жылдамдықпен (Baud Rate) беріледі.
Мысалы: UART – асинхронды интерфейс.

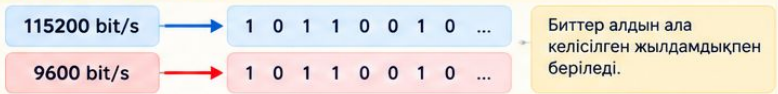
UART қосылу сұлбасы (асинхронды байланыс)



UART деректер кадрының форматы



Baud Rate мысалы



Асинхронды байланыстың ерекшеліктері:

- Clock желісі қолданылмайды.
- Биттер алдын ала келісілген жылдамдықпен (Baud Rate) беріледі.
- Қарапайым құрылымы.
- Мысал: UART.

Синхронды және асинхронды байланыстың айырмашылығы

Параметр	Синхронды байланыс	Асинхронды байланыс
Тактілік сигнал (Clock)	Қолданылады	Қолданылмайды
Деректерді беру тәсілі	Clock бойынша	Алдын ала келісілген жылдамдықпен (Baud Rate)
Жылдамдық	Жоғары	Төменірек
Күрделілігі	Жоғарырақ	Қарапайым
Мысалдар	SPI, I²C	UART

10. UART

UART — Universal Asynchronous Receiver/Transmitter.

Қазақша: **әмбебап асинхронды қабылдағыш-таратқыш.**

UART — екі цифрлық құрылғы арасындағы ең қарапайым байланыс тәсілдерінің бірі.

Негізгі желілер:

TX	—	Transmit	(дерек жіберу)
RX	—	Receive	(дерек қабылдау)
GND	—	ортақ жер	(жер, Ground)

UART жұмыс істеу принципі (1 байттың құрылымы)



UART ерекшеліктері:

- Асинхронды байланыс (Clock сигналы қолданылмайды).
- Екі сым арқылы дерек алмасады (TX, RX).
- Көп құрылғыларда қолданылады (MCU, сенсорлар, модульдер).
- Құрылымы қарапайым.

11. UART қосылуы

1. Екі құрылғыны UART арқылы қосу

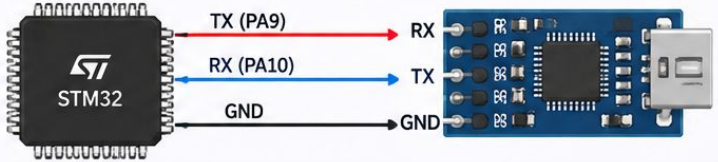


Желі байланысы:

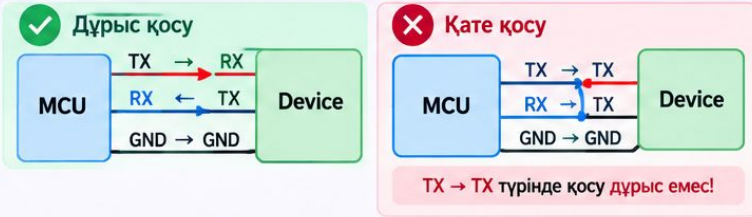
- MCU-дің TX желісі → құрылғының RX желісіне қосылады.
- MCU-дің RX желісі ← құрылғының TX желісіне қосылады.
- GND — ортақ жер.

Яғни: TX → RX және RX ← TX

2. UART қосу мысалы (STM32 ↔ Сыртқы модуль)



3. Қалыпты және қате қосу салыстыруы



12. UART арқылы бір байт беру

Мысалы, келесі деректі беру керек:

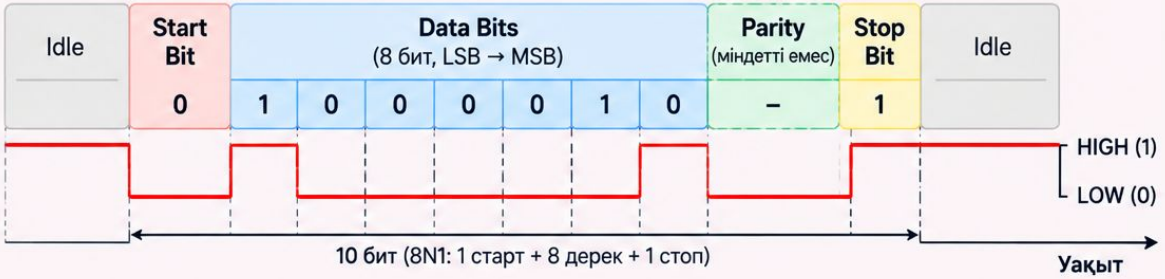
01000001

Бұл ASCII жүйесінде:

‘A’ (0x41)

әрпі.

UART дерек кадрының құрылымы (мысал: 'A' = 01000001)



13. Start Bit және Stop Bit

UART-та жеке **Clock** желісі жоқ.

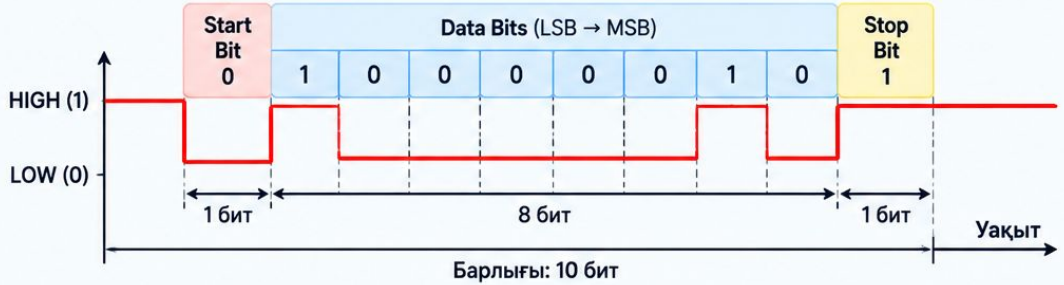
Қабылдаушы құрылғы жаңа байттың басталуын **Start Bit** арқылы анықтайды.

Дерек соңында **Stop Bit** қолданылады.

Кең таралған конфигурация: 8N1

- **8** → 8 data bits (8 дерек биті)
- **N** → No parity (паритет жоқ)
- **1** → 1 stop bit (1 стоп бит)

UART кадрының уақыттық диаграммасы (8N1, мысал: 'A' = 01000001)



Start Bit (басталу биті)

- Әрқашан 0 (LOW).
- Қабылдаушыға деректің басталуын хабарлайды.
- Осы бит арқылы қабылдаушы уақытты синхрондайды.

Stop Bit (тоқтату биті)

- Әрқашан 1 (HIGH).
- Деректің аяқталғанын көрсетеді.
- Келесі байтқа дейін желі бос (Idle) күйінде болады.

14. Baud Rate (деректер беру жылдамдығы)

1. Анықтамасы

Baud Rate — секунд ішінде берілетін биттер саны.

Яғни, бұл бір секундта қанша бит жіберілетінін көрсететін параметр.

$$\text{Baud Rate} = \frac{1}{T_{bit}}$$

мұндағы T_{bit} — бір биттің ұзақтығы (секунд).

2. Мысалдар

Baud Rate (bit/s)	Бір биттің ұзақтығы (T_{bit})	Сипаттамасы
9600	≈ 104 мкс (0.000104 с)	Жиі қолданылады (мысалы, модульдер)
115200	≈ 8.68 мкс (0.00000868 с)	Жоғары жылдамдық (микроконтроллерлер)

3. Bit уақытын есептеу

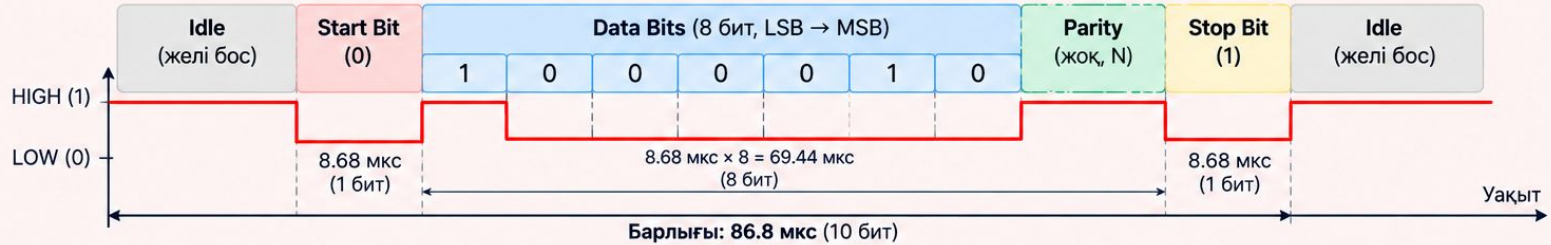
Бағалаған формула:

$$T_{bit} = \frac{1}{\text{Baud Rate}}$$

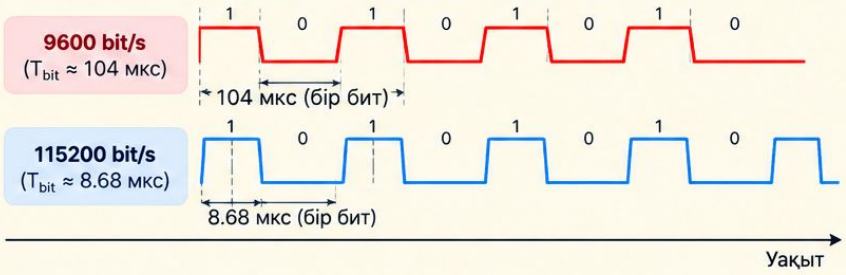
Мысал:

```
Baud Rate = 115200 bit/s
T_bit = 1 / 115200 ≈ 8.68 × 10⁻⁶ s
      = 8.68 мкс
```

4. UART кадрының уақыттық диаграммасы (115200 bit/s, 8N1)



5. Әртүрлі Baud Rate үшін сигналдар салыстыруы



6. Маңызды ескертулер

- Baud Rate жоғары болған сайын, бір биттің ұзақтығы қысқарады.
- Пайдалы деректер көлемі жалпы кадр уақытынан аз, өйткені Start Bit, Stop Bit және қажет болса Parity Bit та беріледі.
- Тарату мен қабылдау құрылғыларында Baud Rate бірдей болуы керек.
- Жиі қолданылатын мәндер: 9600, 19200, 38400, 57600, 115200 bit/s.

15. UART қайда қолданылады?

Негізгі қолдану мысалдары



1 MCU ↔ PC
Деректер алмасу,
бағдарламалау, логтарды өңдеу



2 MCU ↔ Debug adapter
Бағдарламалау және отладка
(SWD/UART)



2 MCU ↔ GPS module
GPS деректерін (NMEA)
қабылдау

```
$GPGGA,123519,
4307.000,N,
07651.000,E,...
```

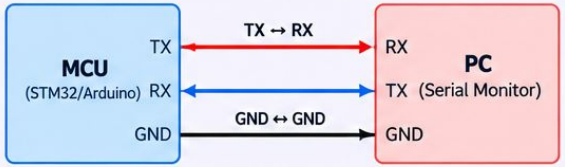


4 MCU ↔ Bluetooth module
Сымсыз байланыс (UART-to-Bluetooth)

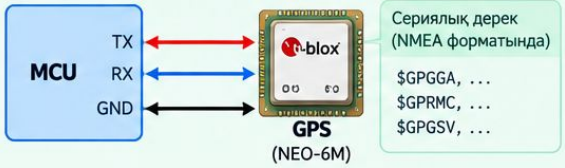


3 MCU ↔ Communication bridge
UART ↔ USB, UART ↔ Ethernet,
және т.б. түрлендіргіштер

Жалпы қосылу сұлбасы (MCU ↔ PC)



Нақты мысал: GPS модулі



Код мысалы (Arduino)

```
void setup() {
  Serial.begin(115200);  // Baud Rate
}

void loop() {
  Serial.println("Hello UART!");
  delay(1000);
}
```

Бұл код Serial Monitor-пен тізбекті байланыс үшін қолданылады.

16. Электромобильдегі UART

UART қолдану мақсаттары



1 ECU ішінде
Диагностика, логтар, конфигурация



2 Бағдарламаны жүктеу кезінде
Bootloader (жаңа бағдарламаны жүктеу)



3 MCU мен қосымша микросхема арасында
Сенсорлар, драйверлер, басқару модульдері

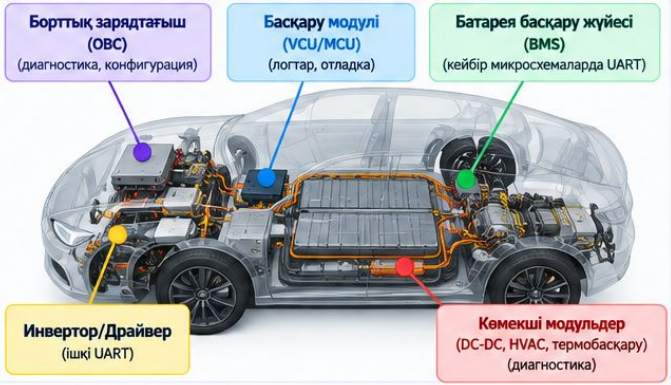


4 Communication bridge арқылы
UART ↔ CAN/LIN/Ethernet түрлендіргіштері



5 Кейбірр BMS микросхемаларында
Батарея басқару жүйесі (диагностика,
конфигурация)

Электромобильдегі болуы мүмкін орындар



ECU арасындағы байланыс түрлері

CAN	Негізгі автомобиль шинасы. Көп ECU арасында қолданылады.
CAN FD	Жоғары жылдамдықты CAN (үлкен деректер көлемі).
LIN	Қарапайым сенсорлар мен қосалқы модульдер үшін.
Automotive Ethernet	Жоғары жылдамдықты деректер алмасу (камералар, ADAS және т.б.).

UART әдетте бүкіл автомобиль үшін негізгі желі ретінде қолданылмайды, бірақ диагностика, бағдарламалау және ішкі модульдермен байланыста кеңінен қолданылады.

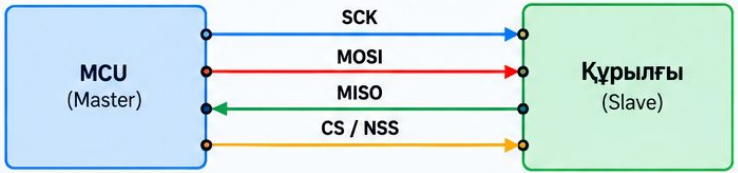
17. SPI

SPI — Serial Peripheral Interface.

Бұл — синхронды тізбекті интерфейс.
SPI әдетте төрт негізгі сигналды қолданады:

Сигнал	Атауы	Бағыты (Master жағынан)	Сипаттамасы
SCK	Serial Clock	Master → Slave	Тактілік сигнал
MOSI	Master Out Slave In	Master → Slave	Мастерден деректер шығады (Slave-ке)
MISO	Master In Slave Out	Slave → Master	Slave-ден деректер шығады (Master-ге)
CS / NSS	Chip Select	Master → Slave	Қажетті құрылғыны таңдау

SPI қосылу сұлбасы



SPI ерекшеліктері:

- Синхронды интерфейс (тактілік сигнал қолданылады).
- Толық дуплексті байланыс (бір мезгілде жіберу және қабылдау).
- Жоғары жылдамдық.
- Көп құрылғыларды қолдайды (әр құрылғы үшін жеке CS).
- Қысқа қашықтыққа қолданылады (әдетте бір плата ішінде).

18. SPI желілері

1. SCK — Serial Clock (тактілік сигнал)

Тактілік сигналды тек Master модулі қалыптастырады.
Әр бір бит деректер осы тактілік сигналмен синхрондалады.



2. MOSI — Master Out Slave In

Мастерден Slave-ке деректер жіберу желісі.



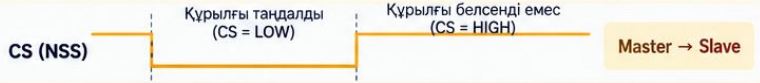
3. MISO — Master In Slave Out

Slave-ден Master-ге деректер жіберу желісі.

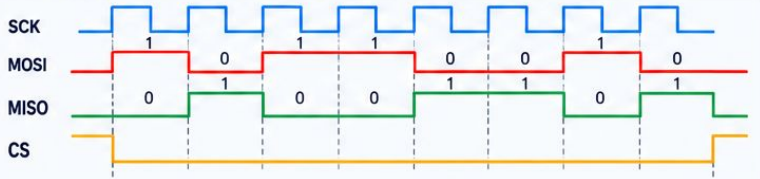


4. CS / NSS — Chip Select

Қажетті құрылғыны таңдау үшін қолданылады. Көп құрылғылар болғанда әр Slave үшін жеке CS желісі болады.



5. SPI деректер алмасуының уақыттық диаграммасы (мысал)



19. Master және Slave

Негізгі түсініктер

Master — байланыс процесін басқаратын және Clock сигналын қалыптастыратын құрылғы.

Slave — Master командасына жауап беретін құрылғы.

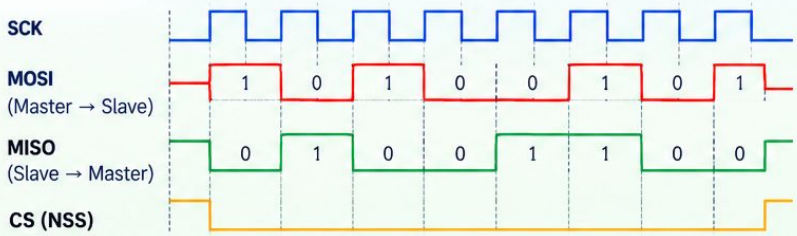
SPI Master ↔ Slave қосылу сұлбасы



Жұмыс принципі

1. Master байланыс сессиясын бастайды.
2. Master тактілік сигналды (SCK) қалыптастырады.
3. Master деректерді MOSI арқылы жібереді.
4. Slave Master командасына жауап беріп, деректерді MISO арқылы жібереді.
5. CS (Chip Select) арқылы қажетті Slave құрылғы таңдалады.

Деректер алмасу уақыты диаграммасы

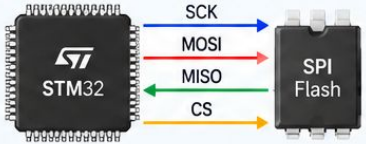


Терминдердің қолданылуы

Термин	Синоним	Қолданылатын мәні
Master	Controller	SPI шинасын басқаратын құрылғы (әдетте MCU, процессор).
Slave	Peripheral / Target	Master командасына жауап беретін құрылғы (сенсор, жад, дисплей т.б.).
Controller	Master	Байланысты басқаратын құрылғы.
Peripheral / Target	Slave	Байланысқа жауап беретін құрылғы.

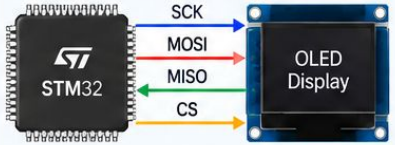
Нақты мысалдар

1 MCU (Master) ↔ Жад (Slave)



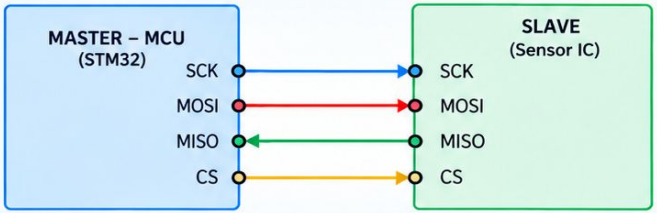
Микроконтроллер (Master) жад микросхемасымен (Slave) байланысады.

2 MCU (Master) ↔ Дисплей (Slave)



Микроконтроллер (Master) дисплей модулімен (Slave) байланысады.

20. SPI қарапайым схемасы



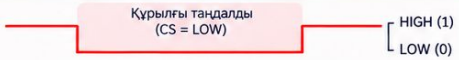
Сигналдардың қызметі

SCK	Тактілік сигнал (Master-ден)
MOSI	**Master → Slave** Деректер/команда (Master-ден)
MISO	**Slave → Master** Жауап/деректер (Slave-ден)
CS / NSS	**Chip Select** Қажетті құрылғыны таңдау (Master-ден, активті LOW)

21. SPI қалай жұмыс істейді?

1 **CS = LOW** (құрылғыны таңдау)

Master қажетті құрылғыны таңдайды (CS сигналын LOW етеді).



2 **SCK** (тактілік сигнал)

Master Clock сигналын қалыптастырады.



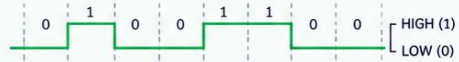
3 **MOSI (Master → Slave)**

Master MOSI арқылы команда/деректер жібереді.



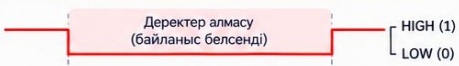
4 **MISO (Slave → Master)**

Slave MISO арқылы жауап/деректер береді.

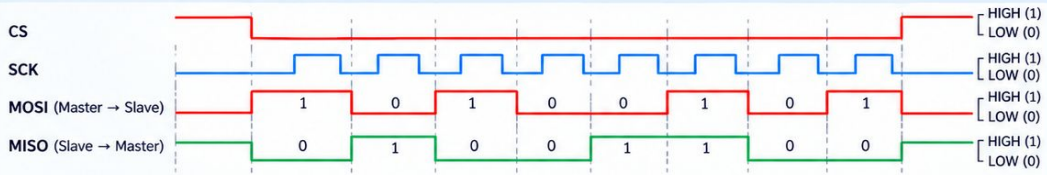


5 **CS = HIGH** (байланысты аяқтау)

Деректер алмасу біткен соң Master CS сигналын HIGH етеді.



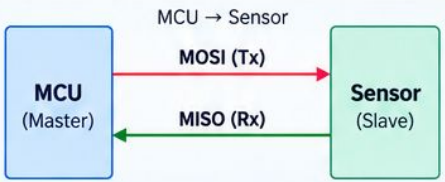
Толық уақыттық диаграмма (мысал, 8 бит)



22. Full Duplex (толық дуплексті байланыс)

Негізгі түсінік

SPI бір уақытта екі бағытта деректер бере алады.

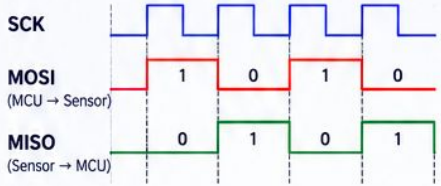


Full Duplex дегеніміз не?

Transmit + Receive
деректер бір уақытта орындала алады.

Full Duplex —
толық дуплексті байланыс деп аталады.

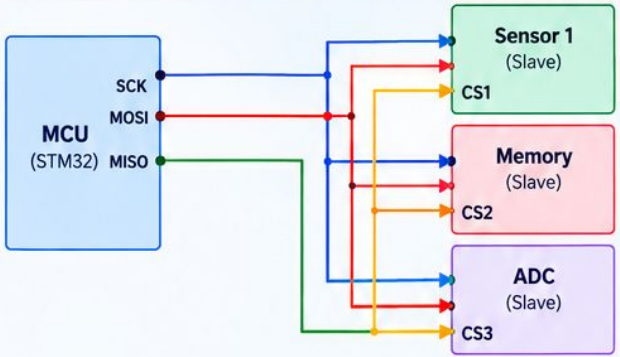
Уақыттық диаграмма (бір уақытта)



MOSI-де дерек жіберу мен MISO-де дерек қабылдау бір уақытта жүреді (әр SCK тактінде).

23. Бірнеше SPI құрылғысын қосу

Жалпы схема



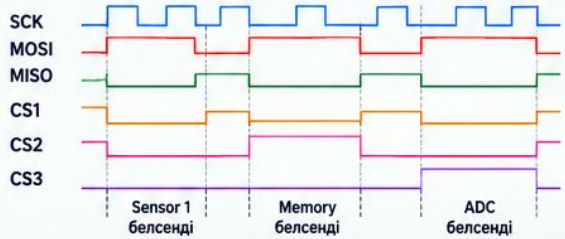
SCK, MOSI және MISO ортақ болуы мүмкін.
Әр құрылғыға жеке CS (Chip Select) қажет.

CS сигналдары

Әр құрылғыны таңдау үшін жеке CS желісі қолданылады.
Белсенді деңгей — LOW (активті LOW).

Құрылғы	CS сигналы	Сипаттама
Sensor 1	CS1	1-құрылғыны таңдау
Memory	CS2	2-құрылғыны таңдау
ADC	CS3	3-құрылғыны таңдау

Уақыттық диаграмма (құрылғыларды таңдау)



24. SPI артықшылықтары

Жоғары жылдамдық
Жоғары жиілікті дерек беру (бірнеше МГц – ондаған МГц).

MCU ↔ IC байланысы
Микроконтроллер мен перифериялық микросхемалар арасында тиімді байланыс.

Екі бағытты деректер алмасу
Full Duplex режимі (MOSI + MISO).

ADC/DAC жұмысы
Аналогтық-цифрлық (ADC) және цифрлық-аналогтық (DAC) микросхемалармен жұмыс.

Жадпен жұмыс
Flash, EEPROM, SD-карта сияқты жад микросхемаларымен байланыс.

Арнайы басқарушы микросхемалар
Дисплейлер, сенсорлар, радиомодульдер және басқа перифериняы басқару.

25. SPI кемшіліктері

1. **Желілер саны артады**
   Әр құрылғыға жеке CS желісі қажет.
2. **Көп құрылғы болғанда күрделенеді**
   Көп CS желілері микроконтроллердің пин санын көбейтеді.
3. **Қашықтық шектеулі**
   Әдетте қысқа қашықтыққа қолданылады (бір плата ішінде).
4. **Автомобильдік негізгі желі ретінде қолданылмайды**
   Автомобильдік ECU арасындағы негізгі байланыс жолында қолданылмайды (оның орнына CAN, CAN FD, LIN, Automotive Ethernet қолданылады).

Құрылғылар саны артқан сайын



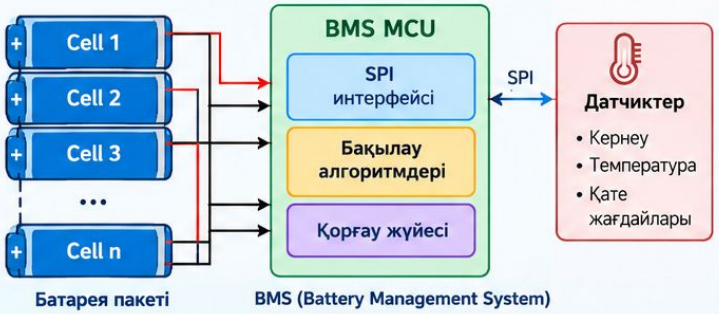
Қолдану аймағы

Қолданылады
- Бір плата ішінде
- Жақын орналасқан микросхемалар арасында
- Сенсорлар, жад, дисплейлер
- Перифериялық құрылғылар

Қолданылмайды (әдетте)
- Алыс қашықтықта
- Автомобильдік ECU арасындағы негізгі байланыс желісі ретінде
- Көп драйверлі ұзақ желілерде
- Электрлік шуды жоғары ортада

26. BMS жүйесіндегі SPI

BMS құрылымы



BMS құрамында:

1. **MCU**
   — SPI арқылы оқшауланған сериялық байланыс.
2. **Батареяны бақылау IC**
   — Әр ұяшықтың кернеуін, температурасын, күйін бақылайды.
3. **Датчиктер**
   — Температура, ток, жалпы кернеу.
4. **Қорғаныс схемалары**
   — Артық кернеу, төмен кернеу, қысқа тұйықталудан қорғайды.

BMS-тің жұмыс істеу принципі

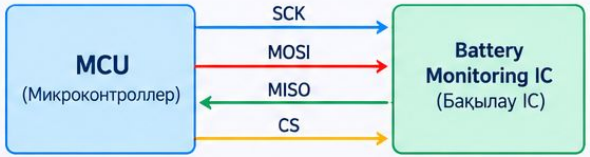
Деректерді оқу
(кернеу, температура, ток)
↓
SPI арқылы MCU-ға жіберу
↓
Шешім қабылдау
(зарядтау/разрядтау, қорғау)
↓
Әрекет ету
(балансировка, ескерту, өшіру)

Бақылау параметрлері

Cell Voltage	Ұяшықтардың кернеуі
Temperature	Температура
Fault Conditions	Қате жағдайлары

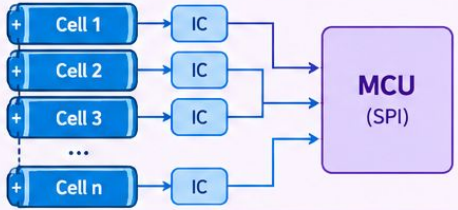
Бұл параметрлер BMS арқылы бақыланып, SPI интерфейсімен беріледі.

BMS және SPI байланысы



SPI — деректерді жоғары жылдамдықпен, қысқа қашықтыққа жеткізетін синхронды интерфейс.

Ұяшықтар массиві



Әрбір ұяшықтың параметрлері жеке бақылау IC арқылы оқылады.

BMS-тің артықшылықтары

- **Қауіпсіздік**
  Жоғары/төмен кернеу, қысқа тұйықталудан қорғайды.
- **Ұзақ қызмет ету**
  Ұяшықтар арасындағы теңгерімді сақтайды (балансировка).
- **Тиімділік**
  Батареяның толық пайдалануын қамтамасыз етеді.
- **Сапа бақылауы**
  Температура мен жұмыс жағдайын бақылайды.
- **Интеллектуалды басқару**
  SPI арқылы деректерді жылдам өңдейді.

29. Құрылғы регистрі дегеніміз не?

Регистр дегеніміз не?

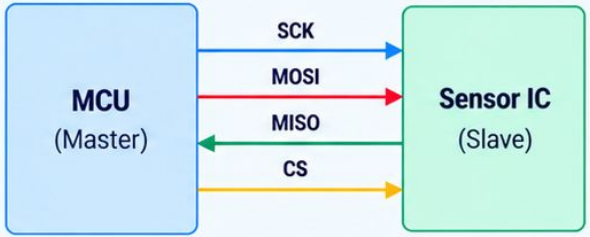
Регистр — микросхема ішіндегі жад ұяшығы, онда белгілі бір параметр немесе күй сақталады.

MCU регистрді оқу немесе жазу арқылы құрылғыны басқарады және оның мәліметтерін алады.

Мысал: Sensor IC регистрлері

Адрес (Hex)	Атауы	Сипаттамасы
0x00	Device ID	Құрылғының идентификаторы
0x01	Configuration	Жұмыс параметрлері
0x02	Temperature	Температура мәні
0x03	Status	Құрылғының күйі
0x04	Fault	Қате жағдайлары

SPI байланысы



Оқу командасы (Read 0x02)

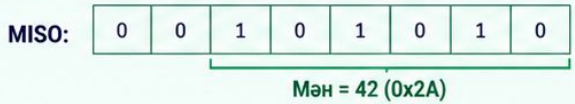
1. MCU CS = LOW (құрылғыны таңдайды)
2. MCU **Read Register** командасын жібереді (адрес = 0x02)
3. Sensor IC 0x02 регистріндегі мәнді қайтарады
4. MCU CS = HIGH (байланысты аяқтайды)

SPI кадрлар мысалы (8-битты адрес + 8-битты дерек)

1. Read командасы (адрес = 0x02)



2. Қайтарылған дерек (Sensor → MCU)



SPI регистрлік операциялары

Операция	MOSI (1-бит)	Мақсаты
Read Register	**1**	Регистрден оқу (құрылғыдан дерек алу)
Write Register	**0**	Регистрге жазу (құрылғыны конфигурациялау)

Практикалық қолданылуы

Параметрлерді оқу
- температура
- кернеу
- ток
- күй ақпараты

Конфигурациялау
- жұмыс режимі
- өлшеу диапазоны
- фильтр параметрлері
- іске қосу/сөндіру

Статус оқу
- дайын/жұмыс істеуде
- ескерту
- қате флагтары

Қате тексеру
- overtemperature
- overcurrent
- sensor fault
- диагностикалық ақпарат

30. I²C (Inter-Integrated Circuit)

I²C дегеніміз не?

I²C — Inter-Integrated Circuit.
Бұл — синхронды тізбекті интерфейс.

Негізгі артықшылығы:
бірнеше құрылғыны қосу үшін екі негізгі сигналдық желі жеткілікті.

```
SDA   — Serial Data
SCL   — Serial Clock
```

I²C артықшылықтары

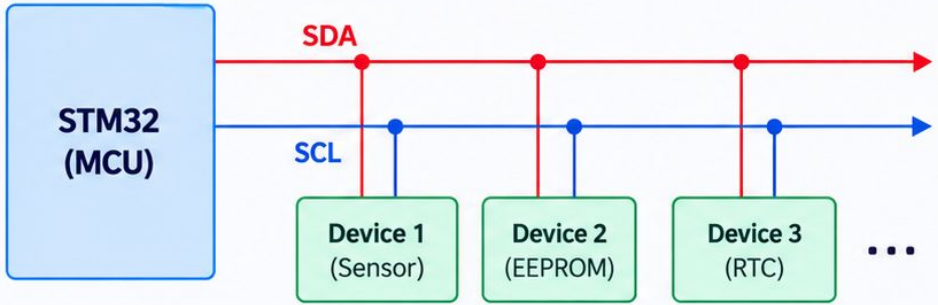
1. **Екі ғана сигналдық желі:** SDA және SCL.
2. **Бір шинаға бірнеше құрылғыны қосуға болады** (адрестеу арқылы).
3. **Қарапайым аппараттық реализация.**
4. **Көптеген микросхемаларда қолданылады** (сенсорлар, EEPROM, RTC, ADC және т.б.).

31. I²C желілері

1. **SDA — Serial Data**
   Деректер желісі.
   `MCU ↔ Device`
   (екі бағытты деректер алмасу)

2. **SCL — Serial Clock**
   Тактілік сигнал желісі.
   `Controller → Devices`
   (MCU тактілік сигналды қапыптастырады)

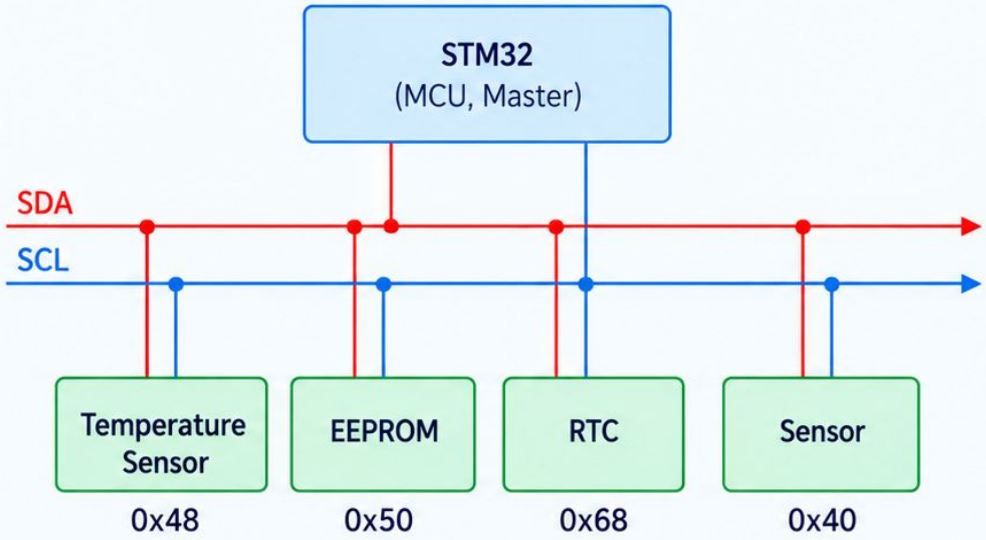
Қарапайым қосылу схемасы



32–33. Бірнеше I²C құрылғысын қосу және I²C адресі

32. Бірнеше I²C құрылғысын қосу

Қарапайым схема:



- Барлық құрылғылар бірдей SDA және SCL желілерінде орналасады.
- MCU нақты құрылғыны оның адресі арқылы анықтайды.

33. I²C адресі

Әр құрылғының өз адресі болады.

Мысал:

Құрылғы	I²C адресі (7-bit, Hex)
Temperature Sensor	**0x48**
EEPROM	**0x50**
RTC	**0x68**
Sensor	**0x40**

MCU мысалы:

```
Address = 0x48
```

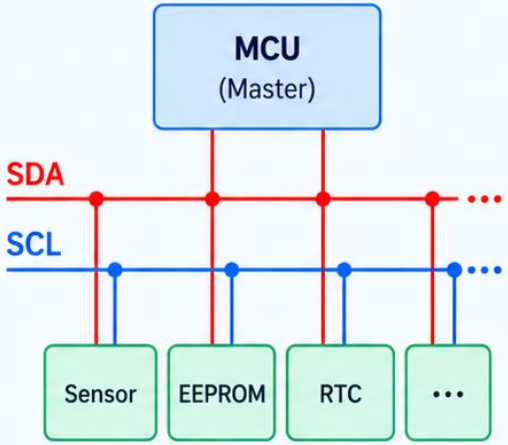
MCU осы адресте орналасқан құрылғыға жауап береді (мысалы, Temperature Sensor).

37–38. I²C қайда қолданылады? SPI және I²C негізгі айырмашылығы

37. I²C қайда қолданылады?

I²C арқылы жиі келесі құрылғылар қосылады:

- Temperature Sensor
- Pressure Sensor
- EEPROM
- RTC
- Accelerometer
- Configuration IC
- Monitoring IC

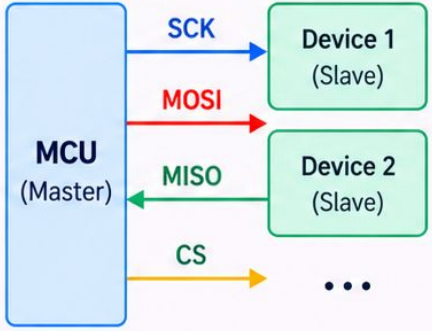


I²C бір платаға бірнеше салыстырмалы түрде баяу перифериялық құрылғыларды қосуға өте ыңғайлы.

38. SPI және I²C негізгі айырмашылығы

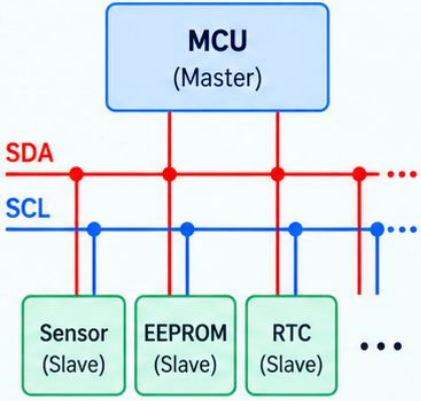
SPI

- жоғары жылдамдық
- көбірек желі
- жеке CS

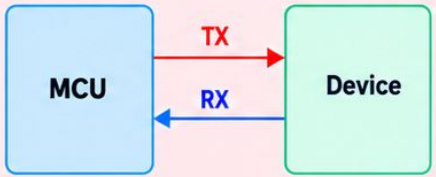
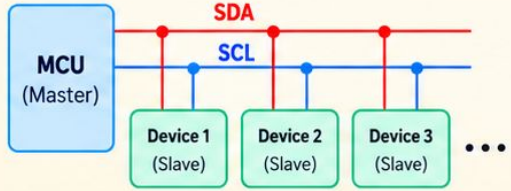


I²C

- SDA + SCL
- құрылғы адресі
- бір желіге бірнеше құрылғы



39. UART, SPI және I²C салыстыру

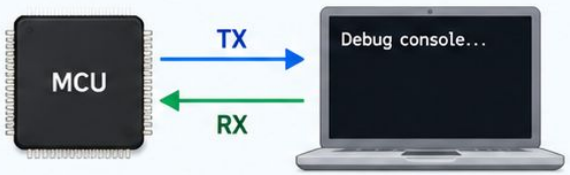
Параметр	UART	SPI	I²C
Байланыс түрі	асинхронды	синхронды	синхронды
Clock	жоқ	бар	бар
Негізгі желілер	TX, RX	SCK, MOSI, MISO, CS	SDA, SCL
Құрылғылар саны	әдетте point-to-point	бірнеше	бірнеше
Адрестеу	әдетте жоқ	CS арқылы	бар
Full Duplex	иә	иә	әдетте ортақ екі бағытты data желісі
Қарапайымдылығы	жоғары	орташа	орташа
Жылдамдығы	орташа	әдетте жоғары	әдетте SPI-ден төмен
Қолданылуы	debug, modules	fast IC, ADC, BMS	sensors, EEPROM
Байланыс схемасы (мысал)			

40. Қай интерфейсті қашан таңдау керек?

1 Debug terminal

Әдетте:

UART



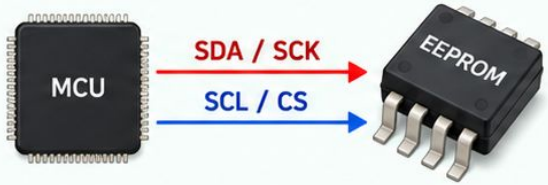
2 Жылдам сыртқы ADC

SPI



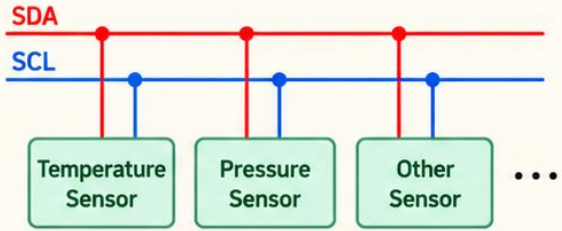
3 EEPROM

I²C немесе SPI



4 Бірнеше датчик

I²C



5 Battery Monitor IC

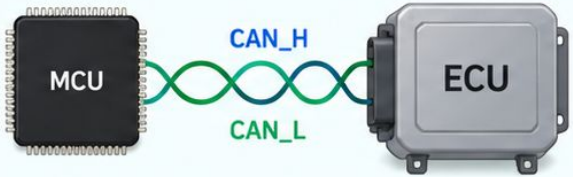
SPI / isolated serial interface



6 Басқа автомобильдік ECU

Әдетте:

CAN / CAN FD



41. Плата ішіндегі интерфейс пен автомобиль желісінің айырмашылығы

Студенттерде мынадай сұрақ туындауы мүмкін:

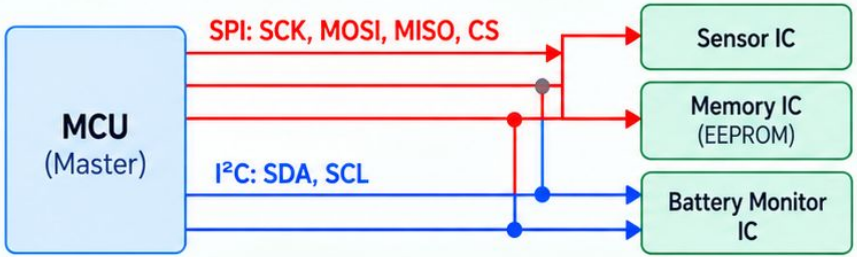
«SPI бар болса, CAN не үшін керек?»

Себебі олардың міндеттері әртүрлі.

SPI / I²C — плата ішінде қолданылады, ал CAN — автомобиль деңгейінде ECU-лар арасында байланыс үшін қолданылады.

1. SPI / I²C — плата ішінде (chip-to-chip)

Бір платаның ішінде MCU мен перифериялық құрылғылар арасында қолданылады.

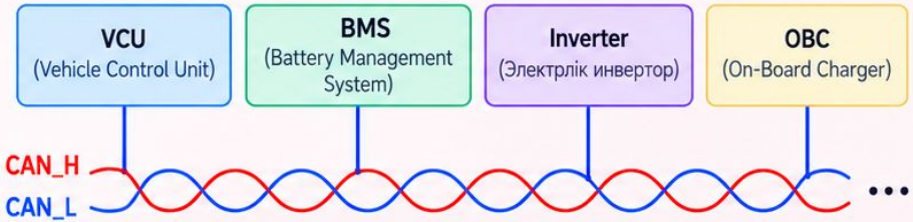


Мысал:

- MCU ↔ Sensor IC
- MCU ↔ Memory IC (EEPROM)
- MCU ↔ Battery Monitor

2. CAN — автомобиль желісі (ECU-to-ECU)

Бірнеше ECU (автомобильдік контроллерлер) арасында ұзақ қашықтықта, кедергілерге төзімді байланыс үшін қолданылады.



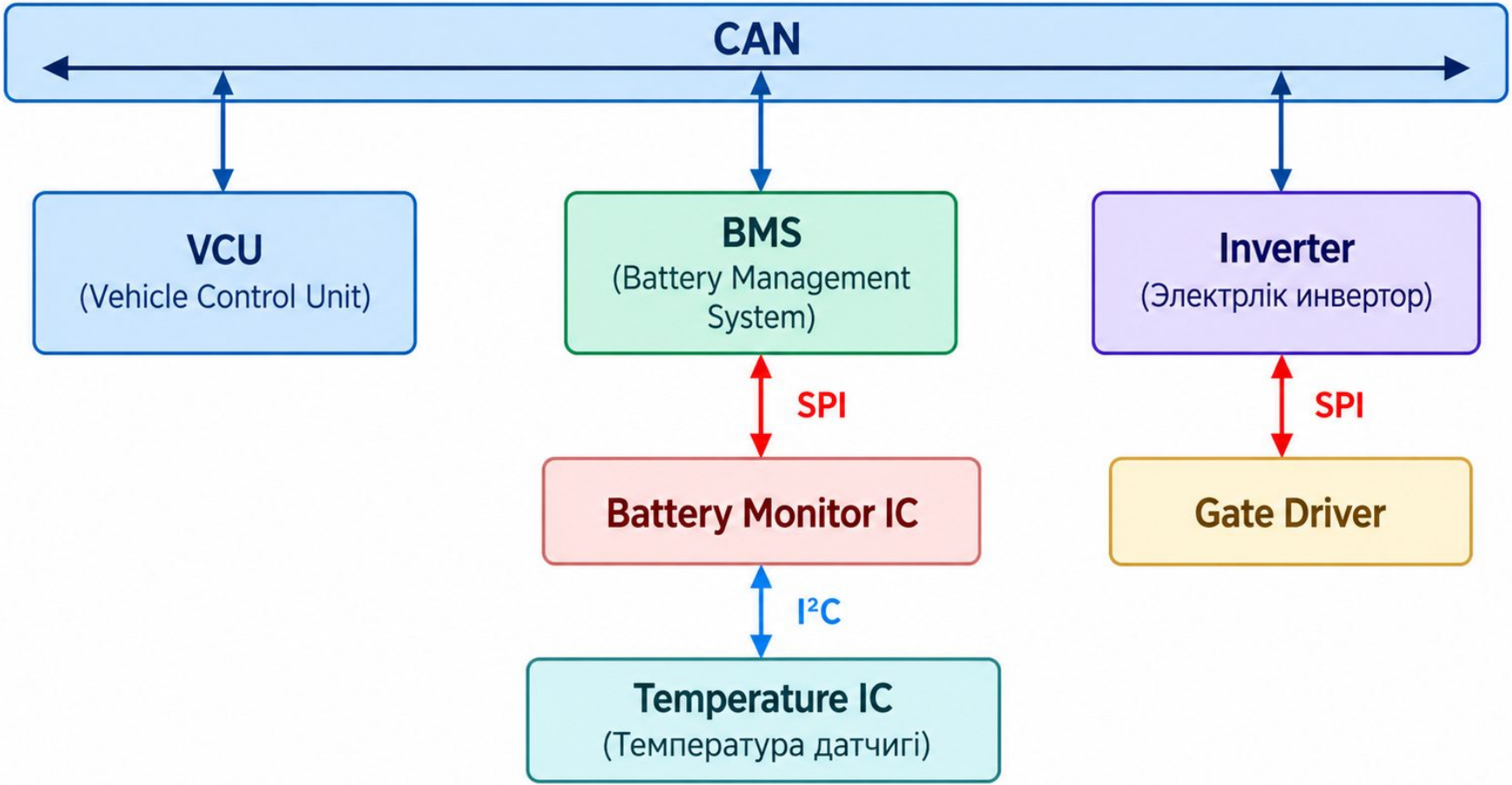
Мысал:

- Автомобильдік контроллерлер арасында:
  VCU ↔ BMS ↔ Inverter ↔ OBC

Қорытынды:

- SPI / I²C — chip-to-chip communication (плата ішінде, қысқа қашықтық, жоғары жылдамдық).
- CAN — ECU-to-ECU communication (автомобиль деңгейінде, ұзақ қашықтық, сенімділік).

42. Электромобиль архитектурасының мысалы



44–46. DMA — Direct Memory Access

44 DMA

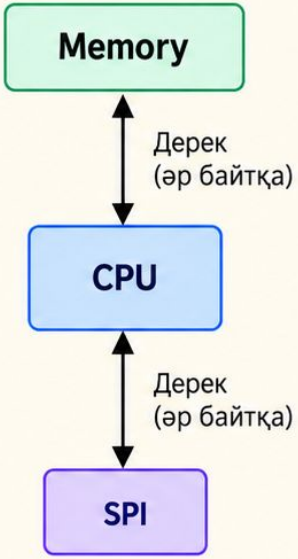
DMA — Direct Memory Access.

Қазақша:

жадқа тікелей қол жеткізу.

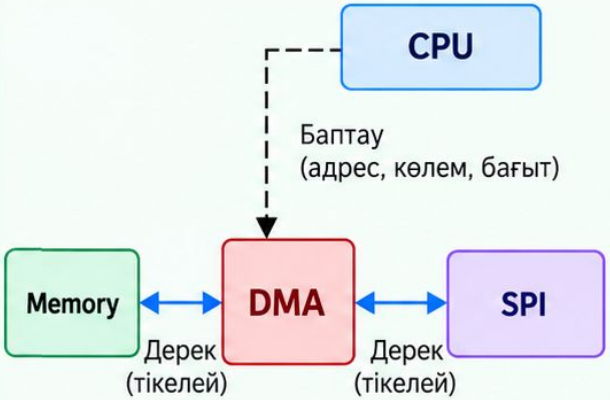
DMA — жад пен перифериялық құрылғы арасында деректерді CPU-дың әр байтқа жеке араласуынсыз тасымалдауға мүмкіндік беретін аппараттық блок.

45 DMA қолданбай дерек беру



Әр байтты CPU өзі басқарады.

46 DMA арқылы дерек беру



CPU DMA-ға:

осы массивті SPI арқылы жібер

деп тапсырма береді.

Одан кейін DMA деректерді өздігінен тасымалдайды.

DMA арқылы дерек беру мысалы

1 Массив

```
uint8_t data[1000];
```

Жіберілетін деректер массиві (1000 байт).

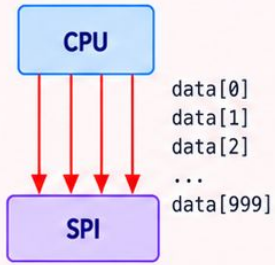
2 DMA-сыз (CPU арқылы)

```
CPU
 ↓
data[0] → SPI

CPU
 ↓
data[1] → SPI

CPU
 ↓
data[2] → SPI

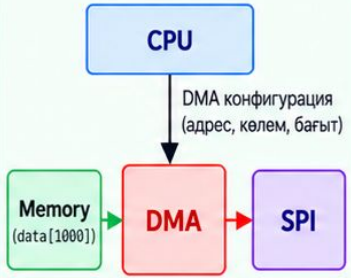
...
```



CPU әр байтты жеке жібереді. Процессор уақыты көп жұмсалады.

3 DMA қолданғанда

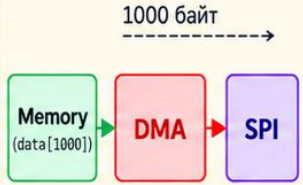
```
CPU
 ↓
DMA баптайды:
"1000 байтты жібер"
 ↓
CPU басқа тапсырмаларды
орындайды
```



DMA өзі деректерді жадтан SPI-ға тасымалдайды. CPU осы уақытта басқа тапсырмаларды орындай алады.

4 DMA арқылы дерек беру

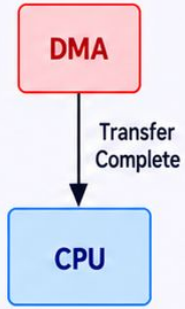
```
DMA:
data[] → SPI
```



DMA жадтағы массивті тікелей SPI-ға жібереді (CPU араласпайды).

5 Тасымалдау аяқталғаннан кейін

```
Interrupt
```



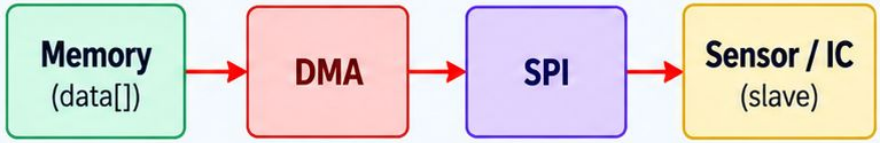
Барлық 1000 байт жіберілгеннен кейін DMA interrupt (аяқталу сигналы) береді.

6 Қорытынды

- DMA — жад пен перифериялық құрылғы арасында деректерді CPU-дың әр байтқа жеке араласуынсыз тасымалдайтын аппараттық блок.
- DMA қолданғанда CPU уақыты босап, басқа тапсырмаларды орындай алады.
- Тасымалдау аяқталғаннан кейін DMA CPU-ға interrupt арқылы хабарлайды.

48. DMA және SPI

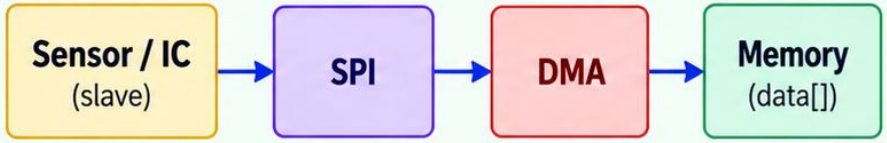
1 Дерек жіберу кезінде (Memory → Sensor)




```
MEMORY
   ↓
  DMA
   ↓
  SPI
   ↓
Sensor / IC
```

- DMA жадтағы деректерді тікелей SPI арқылы сенсорға жібереді.
- CPU бұл процесте қатыспайды.
- Жоғары жылдамдықпен үлкен көлемді дерек жіберуге ыңғайлы.

2 Дерек қабылдау кезінде (Sensor → Memory)

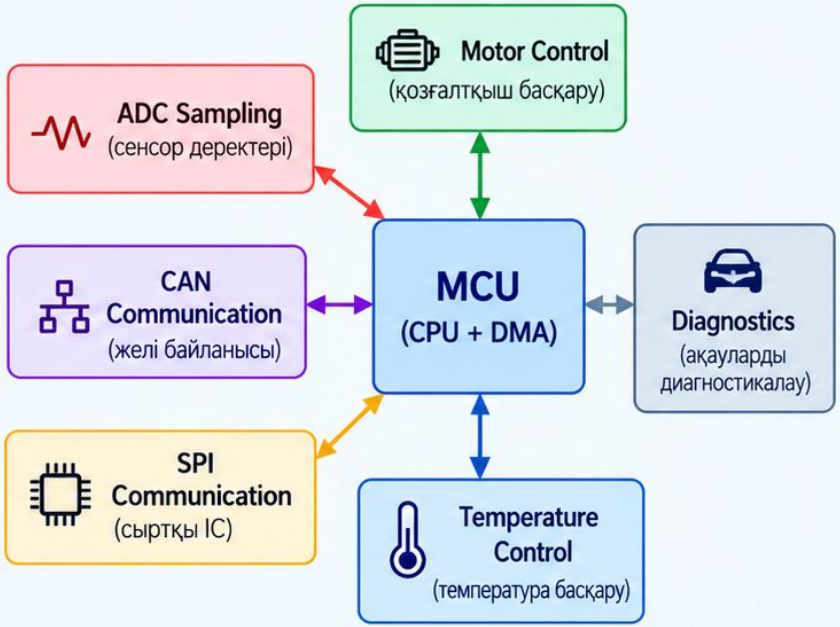



```
Sensor
  ↓
 SPI
  ↓
 DMA
  ↓
Memory
```

- Сенсордан келетін деректер SPI арқылы DMA-ға түседі.
- DMA деректерді жадқа тікелей жазады.
- CPU бұл процесте бос, басқа тапсырмаларды орындай алады.
- АЦП, сенсорлар және үлкен көлемді дерек қабылдау үшін өте тиімді.

49. DMA автомобильдік контроллерде не үшін қажет?

1 Автомобильдік MCU бір уақытта көптеген тапсырмаларды орындайды



Автомобильдік MCU осындай көптеген тапсырмаларды бір уақытта орындай алады.

2 DMA қолданғанда CPU жүктемесі азаяды

CPU – негізгі басқару алгоритмдері

- **Motor Control** (қозғалтқыш басқару)
- **Diagnostics** (ақауларды өңдеу)
- **Control Algorithm** (басқару алгоритмдері)
- **System Logic** (жүйелік логика)

DMA – деректерді тасымалдау

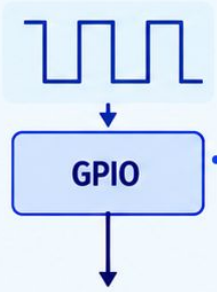
- **ADC Sampling** (сенсор деректерін жадыға көшіру)
- **CAN Communication** (CAN RX/TX буферін өңдеу)
- **SPI Communication** (сыртқы IC-мен дерек алмасу)
- **Memory Transfer** (үлкен көлемді деректерді көшіру)

CPU Load ↑ → **DMA қолданғанда CPU Load ↓**

DMA CPU-дың әр байтты өзі тасымалдау міндетінен босатып, негізгі басқару алгоритмдеріне көбірек уақыт бөлуге мүмкіндік береді.

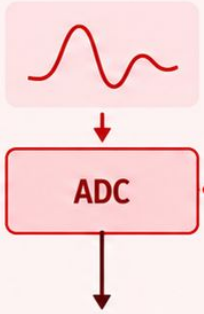
GPIO → ADC → Timer/Interrupt → PWM → Comm → DMA

1 GPIO (Digital Signal)



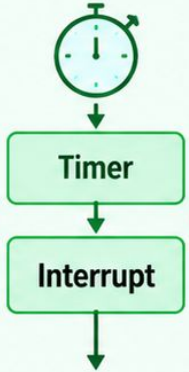
GPIO сыртқы цифрлық сигналды қабылдайды.

2 ADC (Analog Sensor)



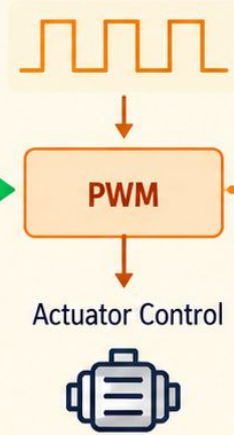
ADC аналогтық сенсор сигналын сандық дерекке түрлендіреді.

3 Timer → Interrupt (Periodic Task)



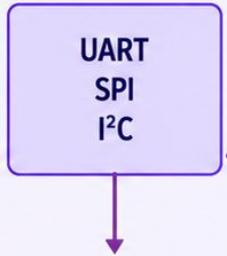
Timer периодтық оқиғаны жасайды, interrupt CPU-ға сигнал береді, периодтық тапсырма орындалады.

4 PWM (Actuator Control)



PWM актуаторды басқару үшін шығыс сигналын қалыптастырады (мысалы, мотор, клапан).

5 UART / SPI / I²C (Communication)



UART / SPI / I²C арқылы басқа микросхемалармен дерек алмасады.

6 DMA (Data Transfer)



DMA деректерді CPU-дың қатысуынсыз жад пен периферия арасында тиімді тасымалдайды.