

5.DMA (Direct Memory Access)

DR.SOMSIN THONGKRAIRAT



life.augmented



DMA (Direct Memory Access)

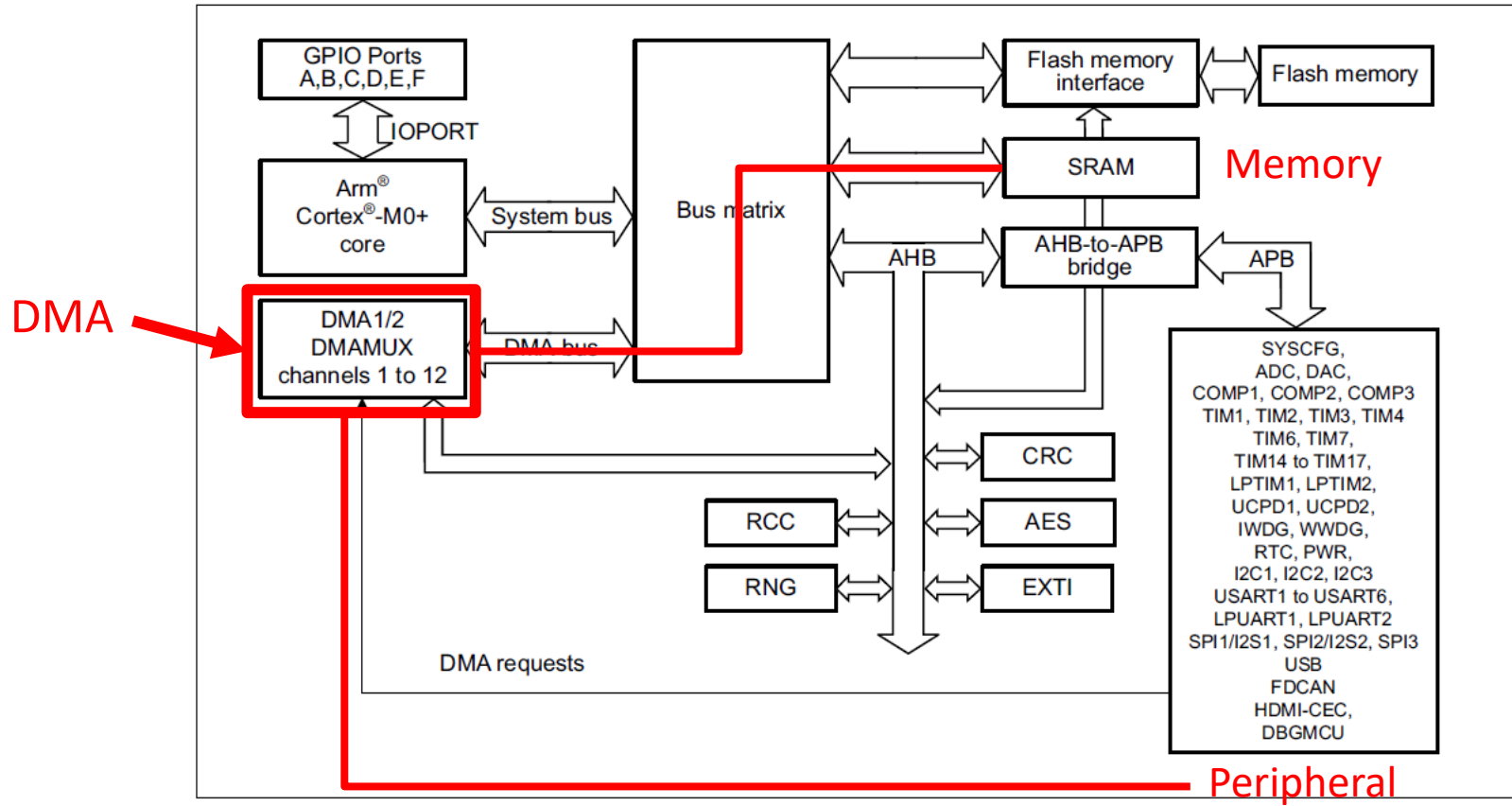
Transfer data directly without CPU

Transfer From Memory to Memory

Transfer From Peripheral to Memory

Transfer From Memory to Peripheral

DMA architecture



Purpose (จุดประสงค์)

การ copy ข้อมูลเป็นงานที่ไม่ซับซ้อน แต่ในระบบปกติเป็นการทำงานที่ขาดไม่ได้

Q: อะไรคือการ Copy ข้อมูล

A: การใช้เครื่องหมาย =

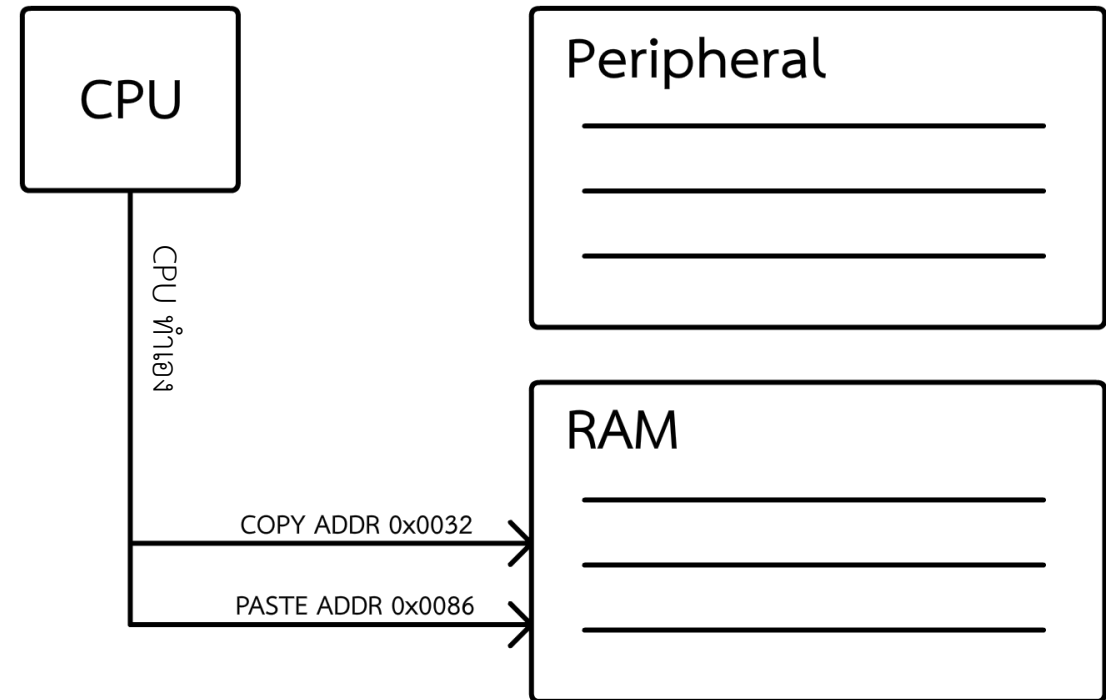
* มีโปรแกรมไหนไม่ใช้เครื่องหมาย = ในการทำงานได้บ้าง

Typical method (mem to mem)

ในกรณีปกติ CPU จะ Copy และ PASTE ข้อมูลทีละ Unit
ในขณะที่ CPU กำลังทำงานดังกล่าวอยู่ CPU จะถือว่าอยู่ใน
State Busy หรือกำลังทำงานอยู่ และไม่สามารถทำอะไรอย่าง
อื่นได้เลย เช่น

```
int a = 3,b = 5;
```

```
int c = a;
```

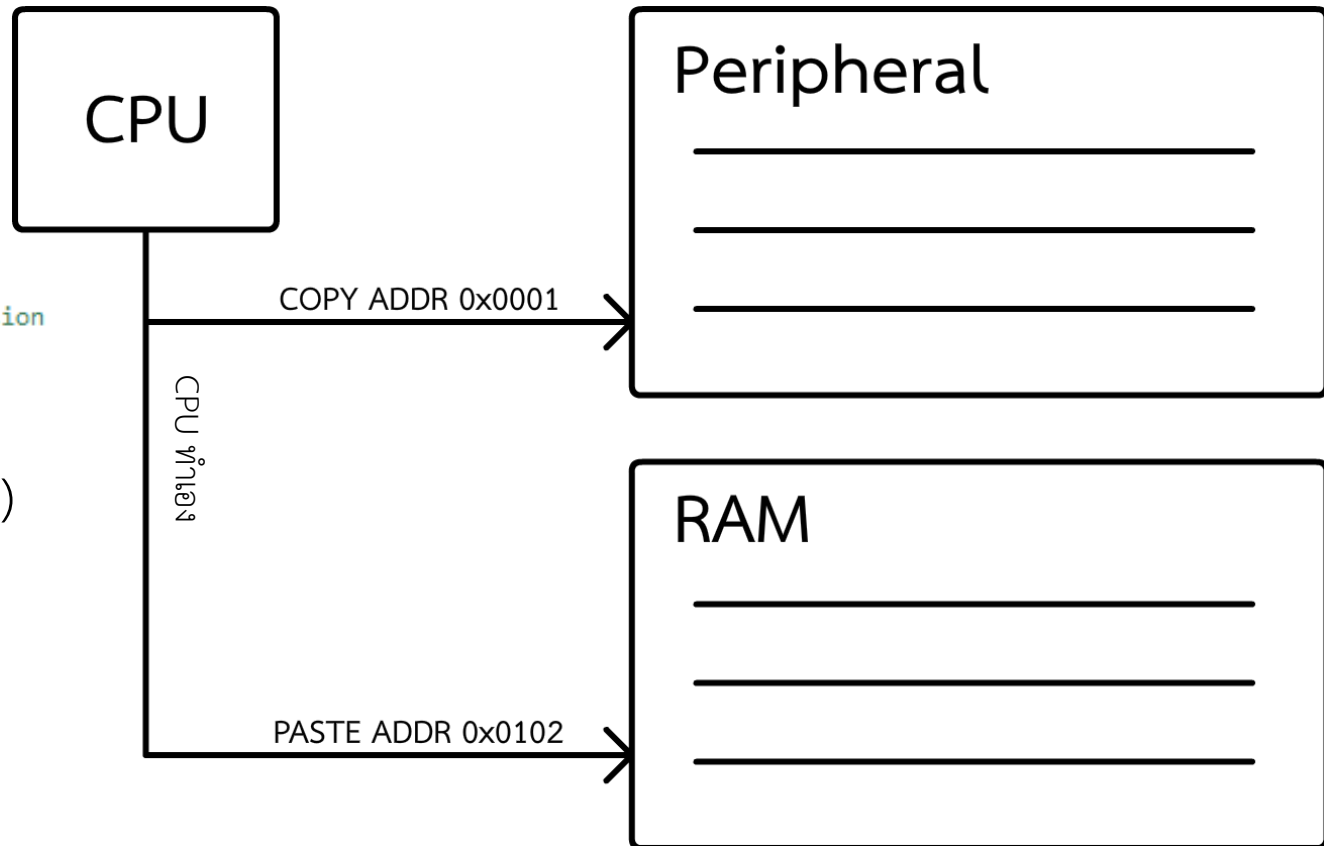


Peripheral to memory

เซ็น ADC

```
HAL_ADC_Start(&hadc1); // start conversion  
HAL_ADC_PollForConversion(&hadc1, 1000); // wait until conversion  
adc_value = HAL_ADC_GetValue(&hadc1); // get ADC result value  
HAL_ADC_Stop(&hadc1); // stop conversion
```

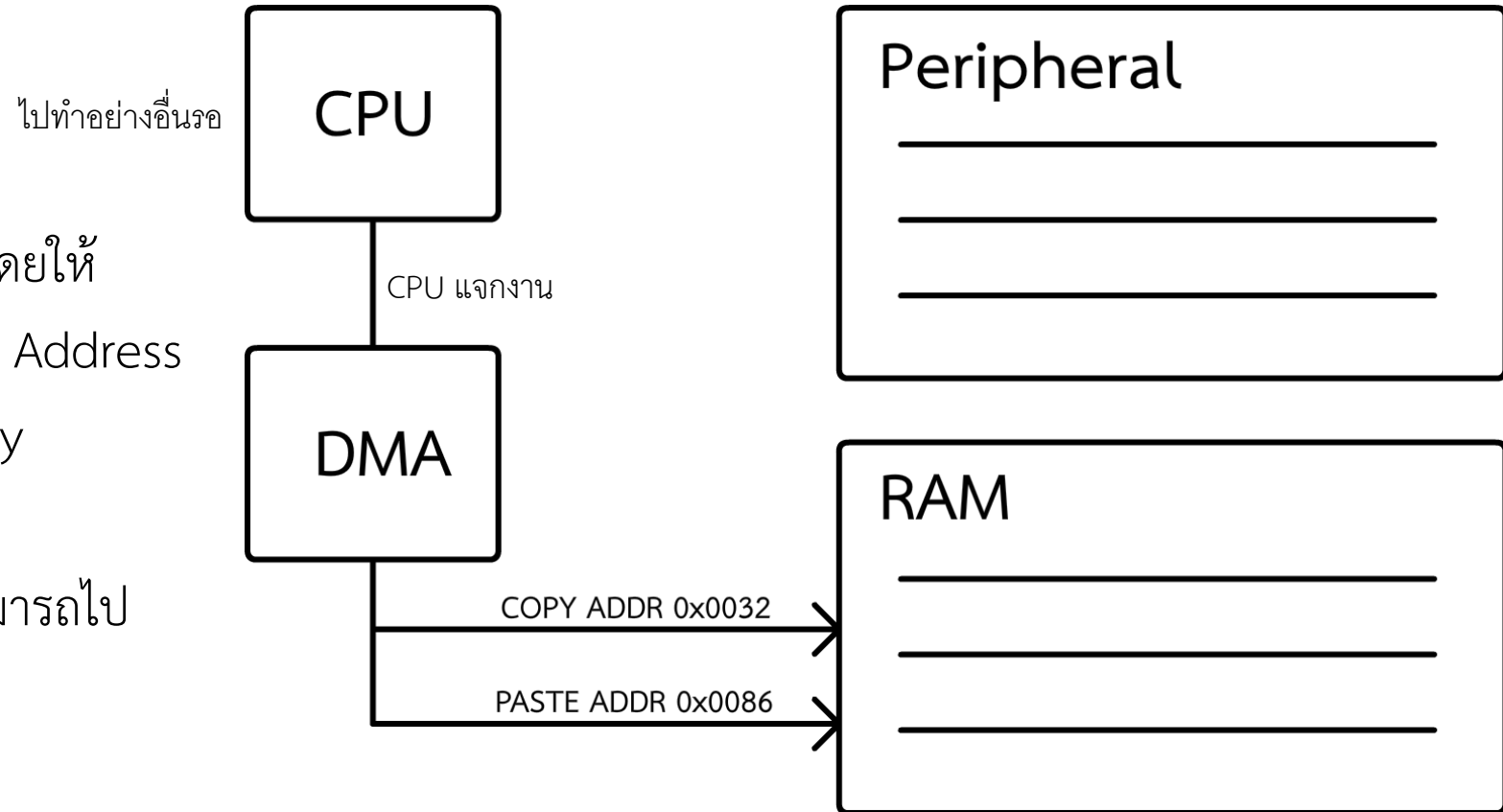
Copy ค่าที่อ่านมาจาก ADC ไปไว้ในตัวแปร(RAM)



DMA Method

CPU แจกงานให้ DMA ทำแทน โดยให้
Source Address , Destination Address
และ จำนวนข้อมูลที่ต้องการ copy

ในขณะที่ DMA ทำงาน CPU สามารถไป
ทำงานอย่างอื่นได้

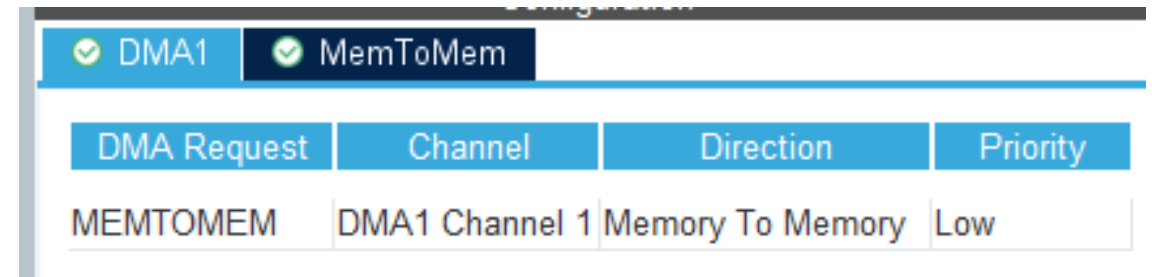
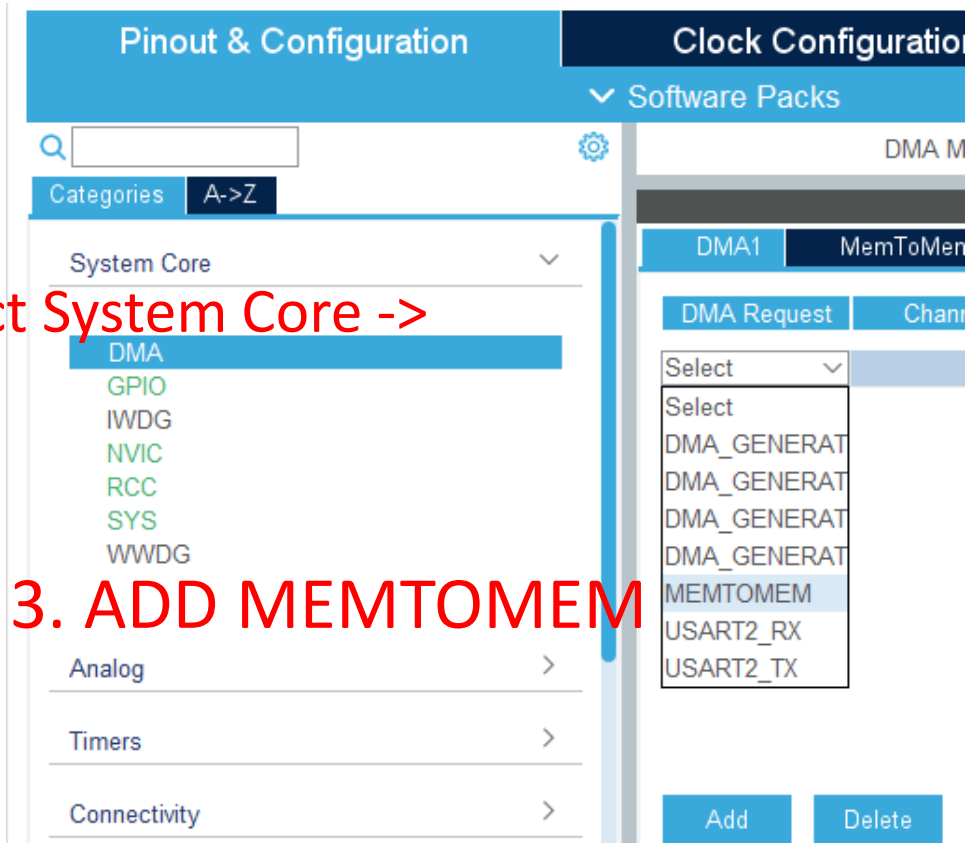


LAB 1 copy array

1. select System Core ->
DMA

3. ADD MEMTOMEM

2. Click ADD



4. Check DMA

Coding

```
/* USER CODE BEGIN 2 */
HAL_UART_Transmit(&huart2, "haruhi DMA\r\n", 13, 1000);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    HAL_DMA_Start (&hdma_memtomem_dma1_channel1, string_source, string_destination, 22);
    while(HAL_DMA_PollForTransfer(&hdma_memtomem_dma1_channel1, HAL_DMA_FULL_TRANSFER, 100) != HAL_OK)
    {
        __NOP();
    }

    string_buffer_size = sprintf(string_buffer, "src = %sdest = %s\r\n", string_source, string_destination);
    HAL_UART_Transmit(&huart2, string_buffer, string_buffer_size, 1000);
    HAL_Delay(2000);
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

```
/* USER CODE BEGIN 1 */
uint8_t string_source[128] = "Somsin Thongkrait\r\n";
uint8_t string_destination[128] = "Thanavit Anuwongpinit\r\n ";
```

```
unsigned char string_buffer[256];
int string_buffer_size = -1;
/* USER CODE END 1 */
```

```
/* USER CODE BEGIN 2 */
HAL_UART_Transmit(&huart2, "haruhi DMA\r\n", 13, 1000);
/* USER CODE END 2 */
```

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
```

```
    HAL_DMA_Start (&hdma_memtomem_dma1_channel1, string_source, string_destination, 22);
    while(HAL_DMA_PollForTransfer(&hdma_memtomem_dma1_channel1, HAL_DMA_FULL_TRANSFER, 100) !=
    HAL_OK)
    {
        __NOP();
    }
```

```
    string_buffer_size = sprintf(string_buffer, "src = %sdest =
    %s\r\n", string_source, string_destination);
    HAL_UART_Transmit(&huart2, string_buffer, string_buffer_size, 1000);
    HAL_Delay(2000);
/* USER CODE END WHILE */
```

Coding

```
/* USER CODE BEGIN 2 */  
HAL_UART_Transmit(&huart2, "haruhi DMA\r\n", 13, 1000);  
/* USER CODE END 2 */
```

```
/* Infinite loop */  
/* USER CODE BEGIN WHILE */  
while (1)  
{
```

```
    HAL_DMA_Start (&hdma_memtomem_dma1_channel1, string_source, string_destination, 22);  
    while(HAL_DMA_PollForTransfer(&hdma_memtomem_dma1_channel1, HAL_DMA_FULL_TRANSFER, 100) != HAL_OK)  
    {  
        __NOP();  
    }
```

```
    string_buffer_size = sprintf(string_buffer, "src = %sdest = %s\r\n", string_source, string_destination);  
    HAL_UART_Transmit(&huart2, string_buffer, string_buffer_size, 1000);  
    HAL_Delay(2000);
```

```
/* USER CODE END WHILE */
```

```
/* USER CODE BEGIN 3 */  
}  
/* USER CODE END 3 */
```

Assign Work to DMA



Wait until copy completed



Print Output



Coding

```
HAL_DMA_Start(&hdma_memtomem_dma1_channel1, string_source, string_destination, 22);
while(HAL_DMA_PollForTransfer(&hdma_memtomem_dma1_channel1, HAL_DMA_FULL_TRANSFER, 100) != HAL_OK)
{
    __NOP();
}

string_buffer_size = sprintf(string_buffer, "src = %sdest = %s\r\n", string_source, string_destination);
HAL_UART_Transmit(&huart2, string_buffer, string_buffer_size, 1000);
HAL_Delay(2000);
/* USER CODE END WHILE */
```

DMA Copy

```
src = Somsin Thongkrait
dest = Somsin Thongkrait
```

```
/*HAL_DMA_Start(&hdma_memtomem_dma1_channel1, string_source, string_destination, 22);
while(HAL_DMA_PollForTransfer(&hdma_memtomem_dma1_channel1, HAL_DMA_FULL_TRANSFER, 100) != HAL_OK)
{
    __NOP();
}*/
string_buffer_size = sprintf(string_buffer, "src = %sdest = %s\r\n", string_source, string_destination);
HAL_UART_Transmit(&huart2, string_buffer, string_buffer_size, 1000);
HAL_Delay(2000);
/* USER CODE END WHILE */
```

DMA not Copy

```
src = Somsin Thongkrait
dest = Thanavit Anuwongpinit
```

DMA Transferring data

string_source



string_destination

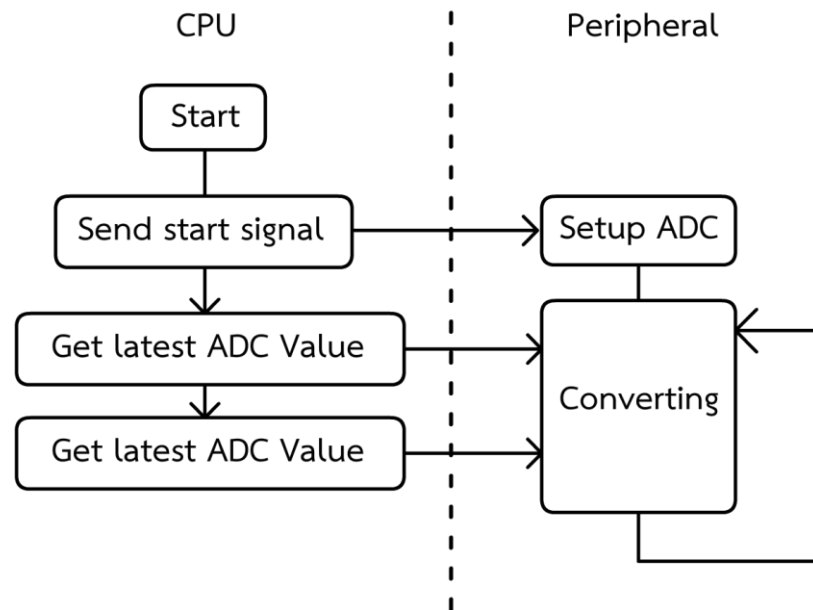


```
for(int i=0;i<22;i++){ // using CPU
    string_destination[i] = string_source[i];
}
```

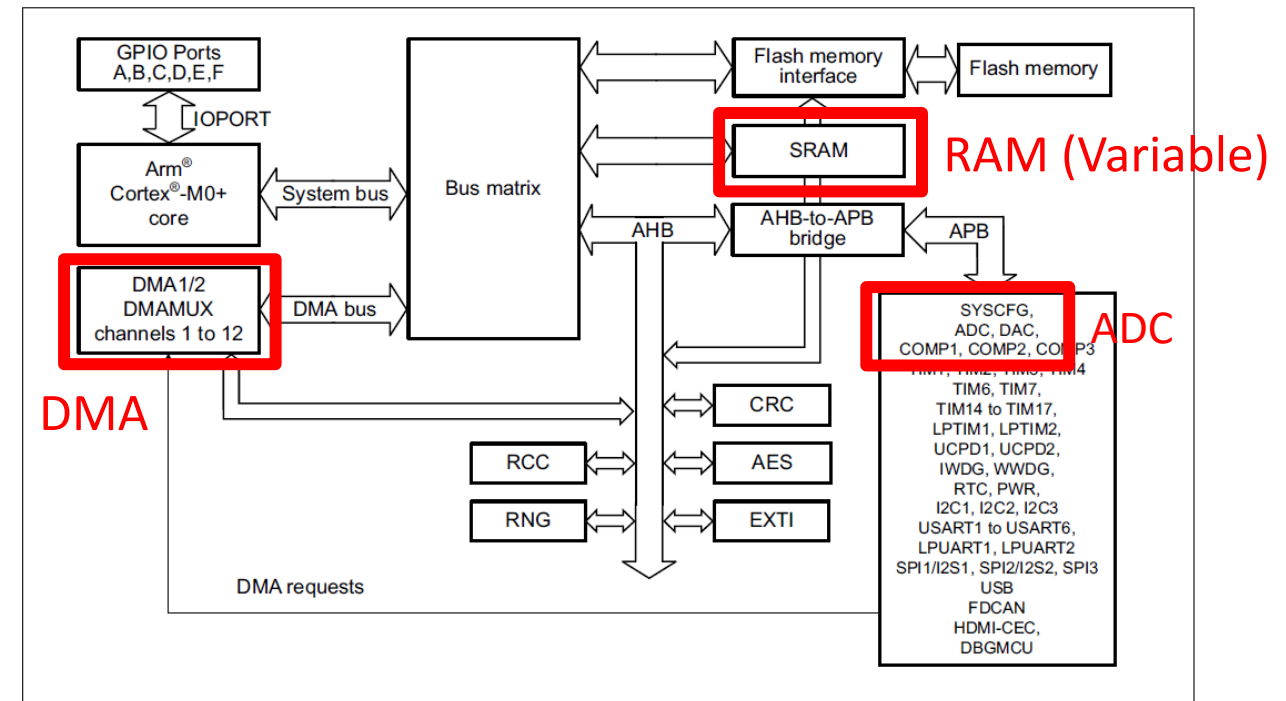
Same Result! But using CPU

Peripheral to memory (ADC to RAM)

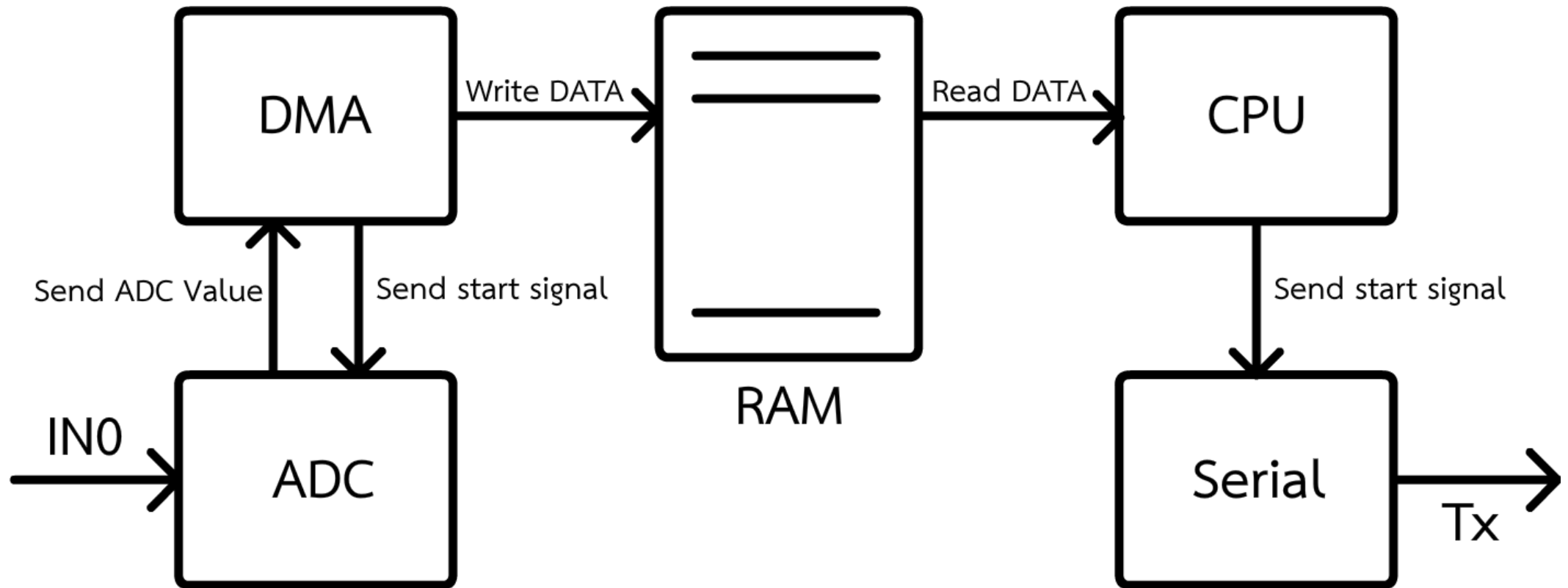
ยังจำ ADC กันได้ใช่ไหมครับ?



CPU Method



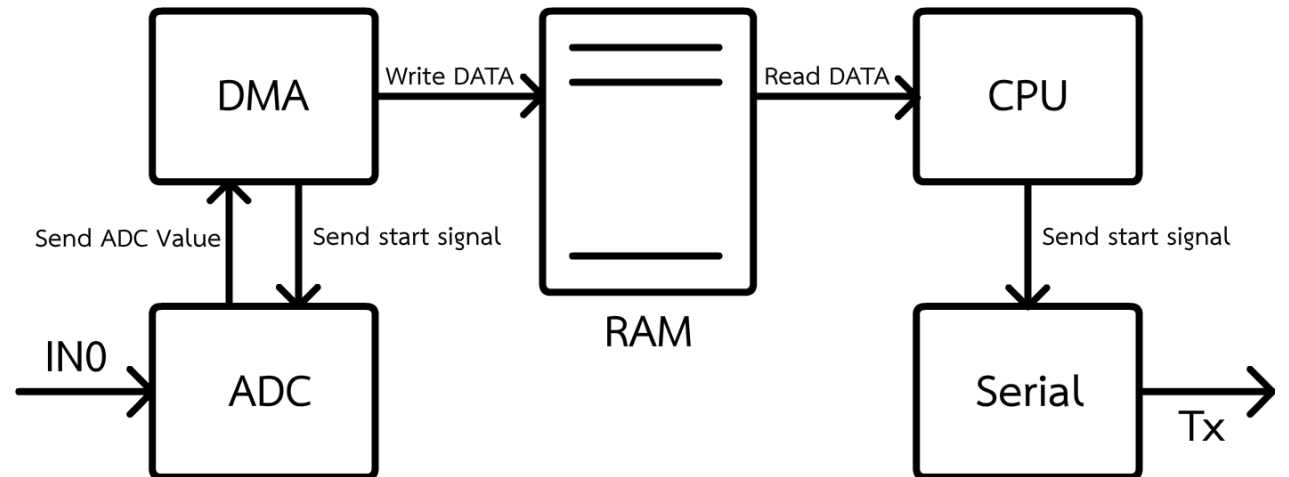
ADC using DMA

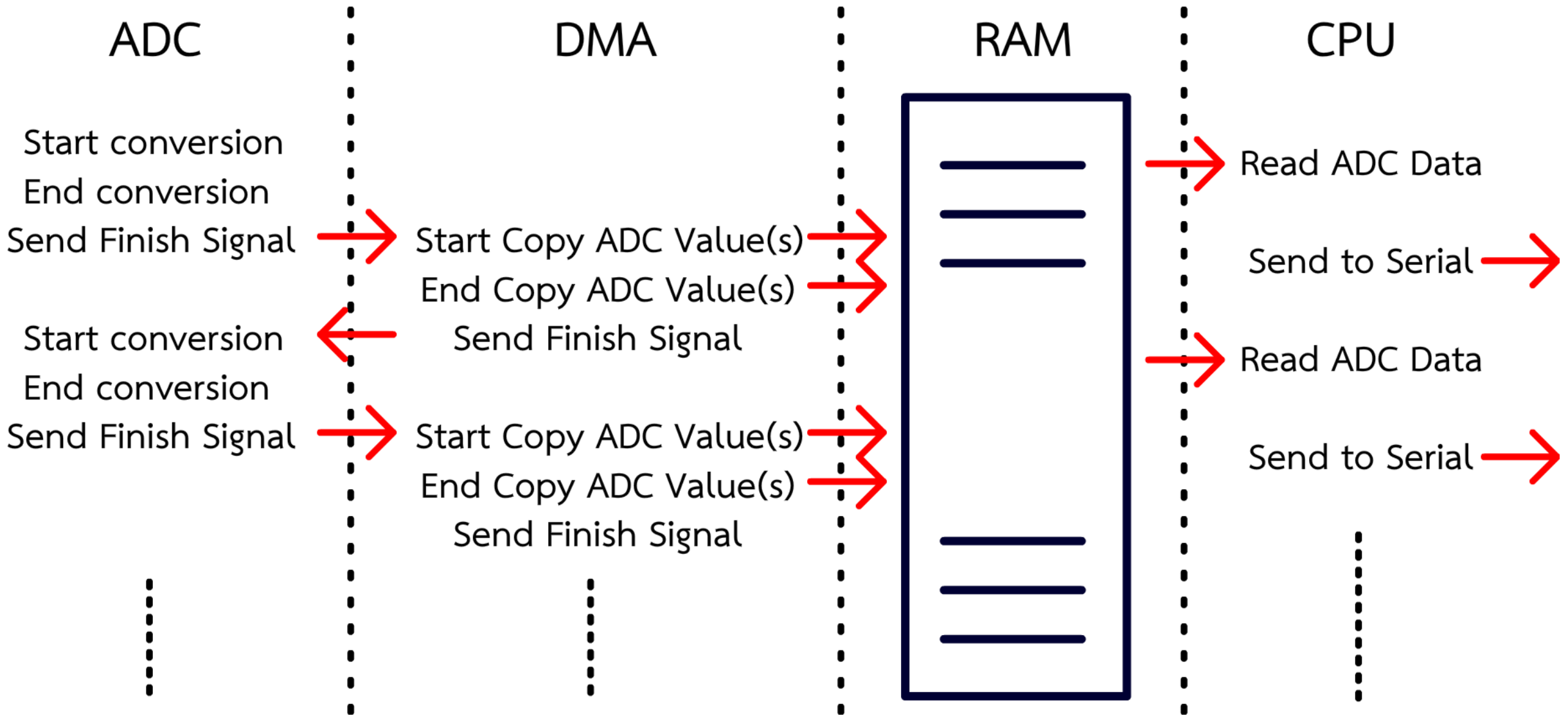


ADC using DMA

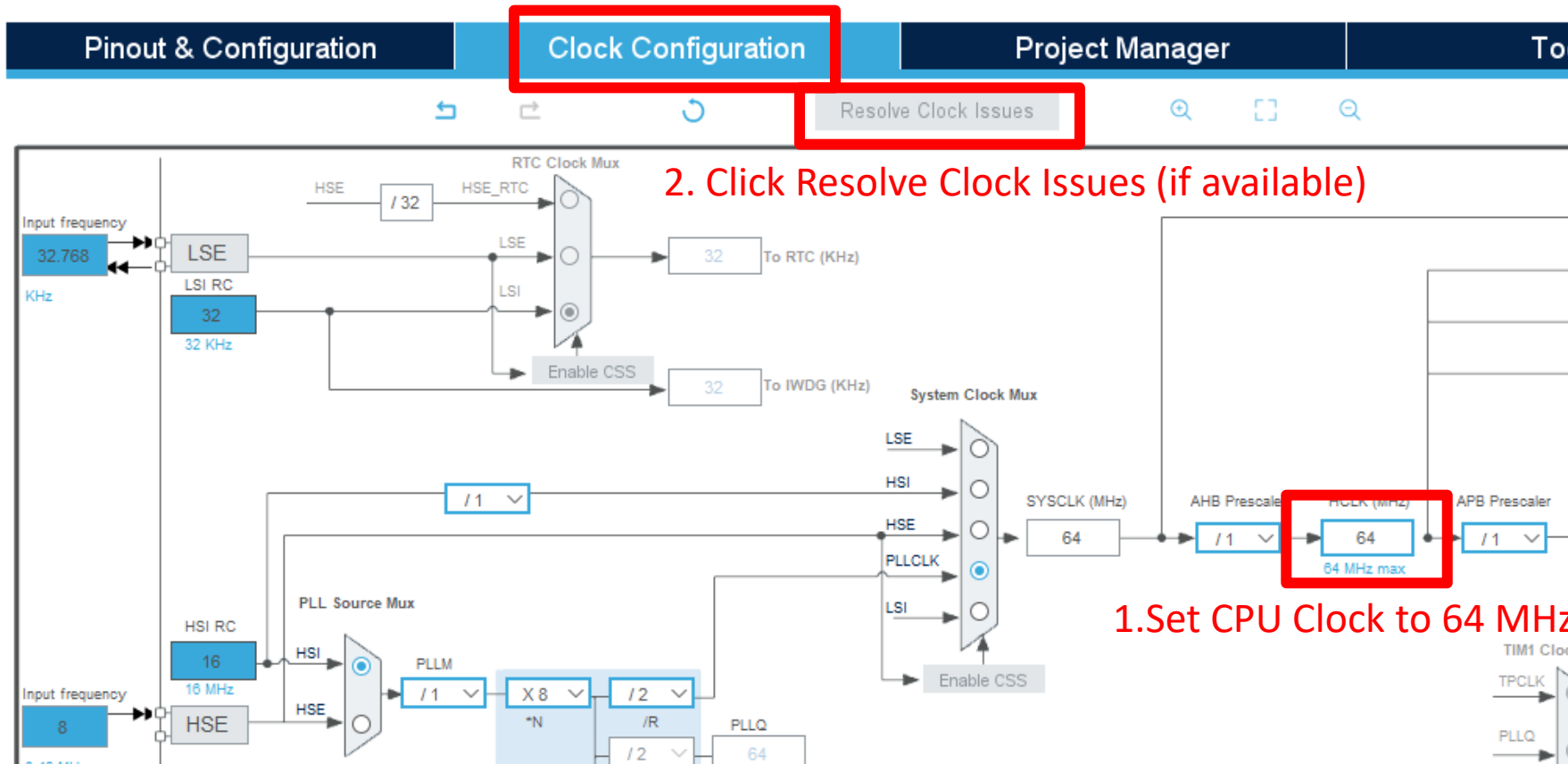
CPU ไม่ต้อง Polling ข้อมูลจาก DMA เหมือน Polling method และสามารถไปทำอย่างอื่นระหว่างที่ DMA และ ADC ทำงานได้

*จะสังเกตว่า CPU ไม่ได้ ติดต่อกับ ADC เลย มีเพียงขั้นตอน Setup (Set resolution , Set sequence การเชื่อมต่อ) เท่านั้น ที่ CPU ติดต่อกับ ADC

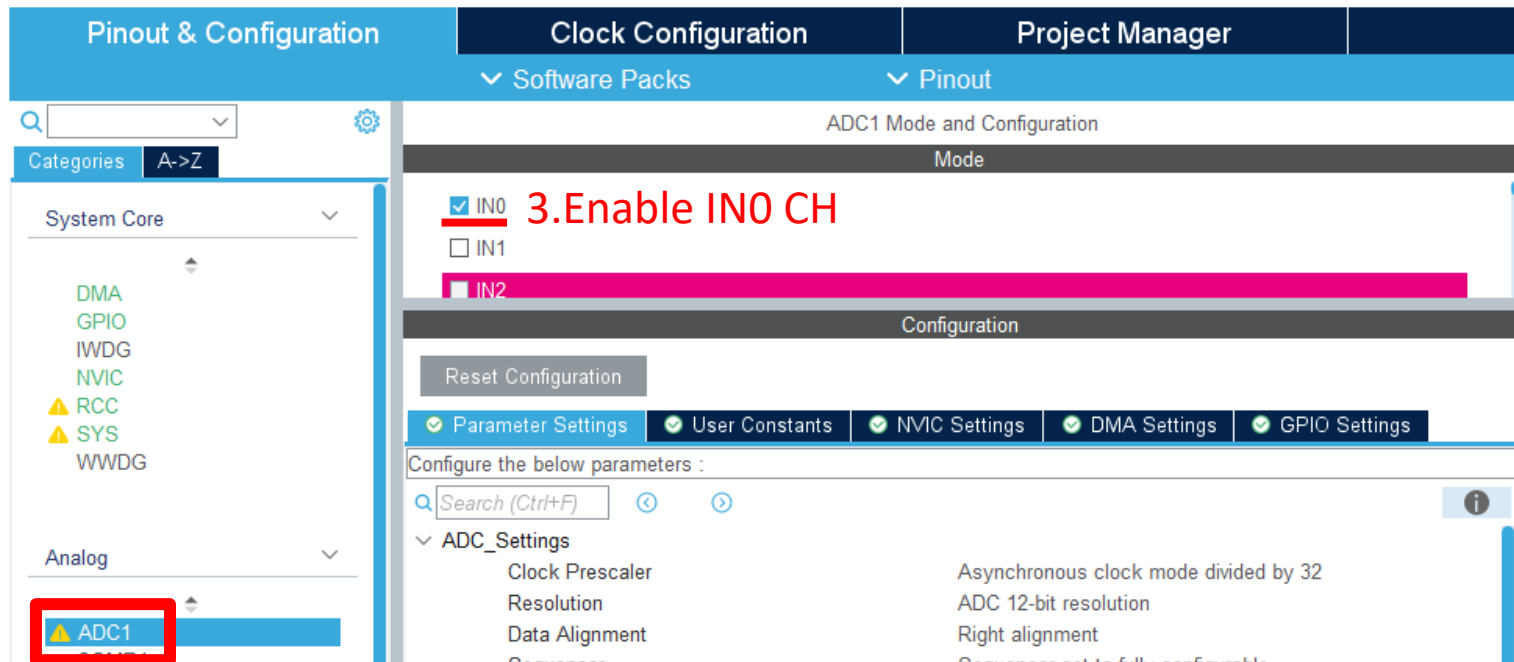




LAB 2 DMA ADC (ยากมากกกก)



LAB 2 DMA ADC (ยากมากกกก)



LAB 2 DMA ADC Setup

Parameter Settings ✓ User Constants ✓ **NVIC Settings ✓** DMA Settings ✓ GPIO Settings ✓

NVIC Interrupt Table	Enabled	Preemption Priority
DMA1 channel 2 and channel 3 interrupts	<input checked="" type="checkbox"/>	0
ADC1, COMP1 and COMP2 interrupts (COMP interrupts through EXTI lines 17 and 18)	<input checked="" type="checkbox"/>	0

4.Enable Interrupt Signal

5.Add DMA to ADC

Parameter Settings ✓ User Constants ✓ NVIC Settings ✓ **DMA Settings ✓** GPIO Settings ✓

DMA Request	Channel	Direction	Priority
ADC1	DMA1 Channel 2	Peripheral To Memory	High

Add Delete

DMA Request Settings

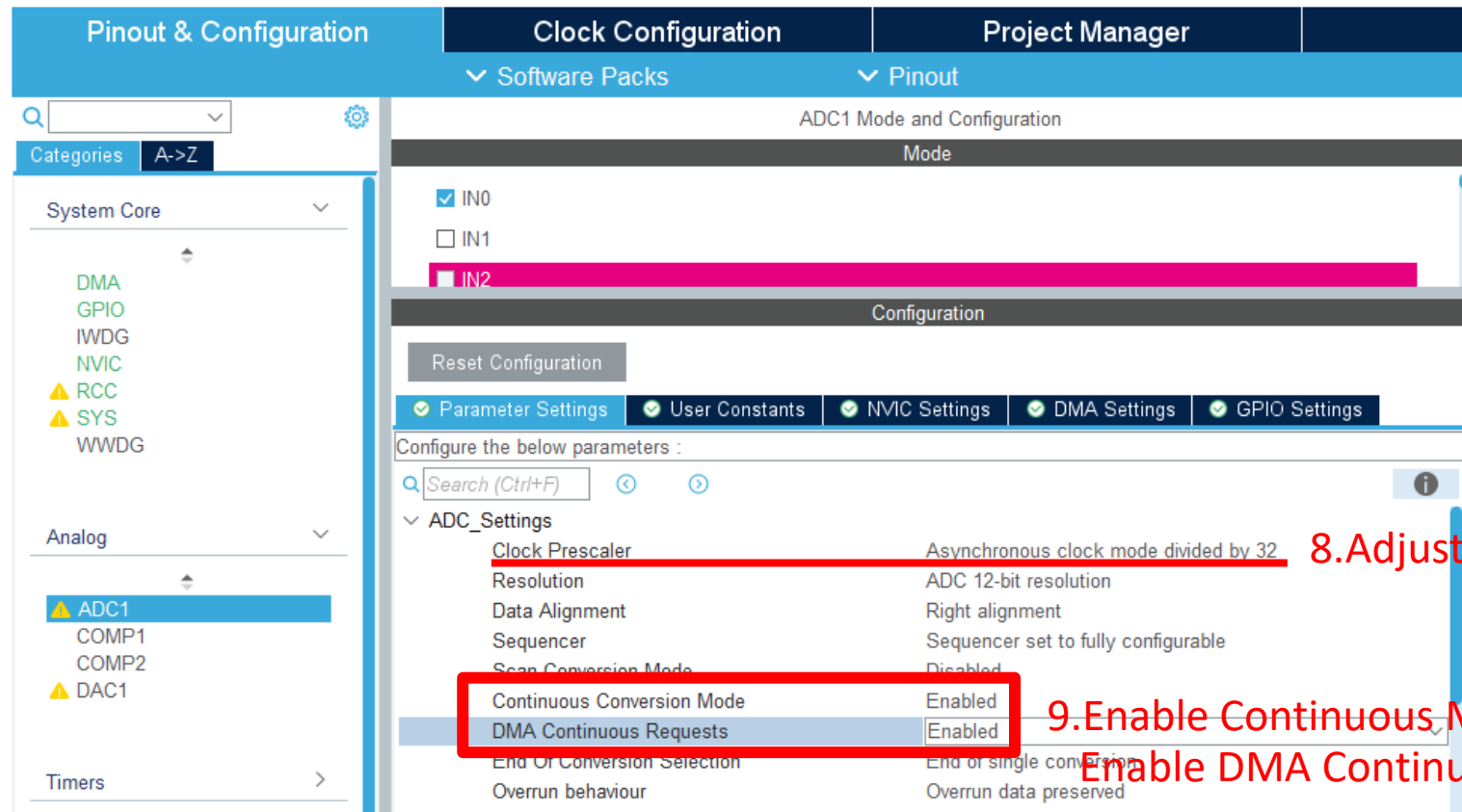
Peripheral		Memory
Mode Circular	Increment Address <input type="checkbox"/>	<input checked="" type="checkbox"/>
Data Width Half Word		Half Word

6.Config DMA Channel
(can be any channel)

7.Config data transfer

- Circular mode
- Incremen Address
- Half word or word should fine

LAB 2 DMA ADC (ยากมากกกก)



8. Adjust ADC Clock Speed

9. Enable Continuous Mode & Enable DMA Continuous Mode

Coding

```
98  MX_ADC1_Init();  
99  /* USER CODE BEGIN 2 */  
100 HAL_ADCEx_Calibration_Start(&hadc1); // Calibrate ADC  
101  
102 unsigned char string_buffer[256];  
103 int string_buffer_size = -1;  
104  
105 volatile unsigned int ADC_value = 0;  
106  
107 HAL_ADC_Start_DMA(&hadc1, &ADC_value, 1);  
108 /* USER CODE END 2 */  
109  
110 /* Infinite loop */  
111 /* USER CODE BEGIN WHILE */  
112 while (1)  
113 {  
114     string_buffer_size = sprintf(string_buffer, "ADC = %d\r\n", ADC_value);  
115     HAL_UART_Transmit(&huart2, string_buffer, string_buffer_size, 1000);  
116     HAL_Delay(1000);  
117     /* USER CODE END WHILE */  
118
```

Setup ADC in DMA Mode



```
/* USER CODE BEGIN 2 */  
HAL_ADCEx_Calibration_Start(&hadc1); // Calibrate ADC  
unsigned char string_buffer[256];  
int string_buffer_size = -1;  
volatile unsigned int ADC_value = 0;  
  
HAL_ADC_Start_DMA(&hadc1, &ADC_value, 1);  
/* USER CODE END 2 */  
  
/* Infinite loop */  
/* USER CODE BEGIN WHILE */  
while (1)  
{  
    string_buffer_size = sprintf(string_buffer, "ADC = %d\r\n", ADC_value);  
    HAL_UART_Transmit(&huart2, string_buffer, string_buffer_size, 1000);  
    HAL_Delay(1000);  
    /* USER CODE END WHILE */
```

Coding

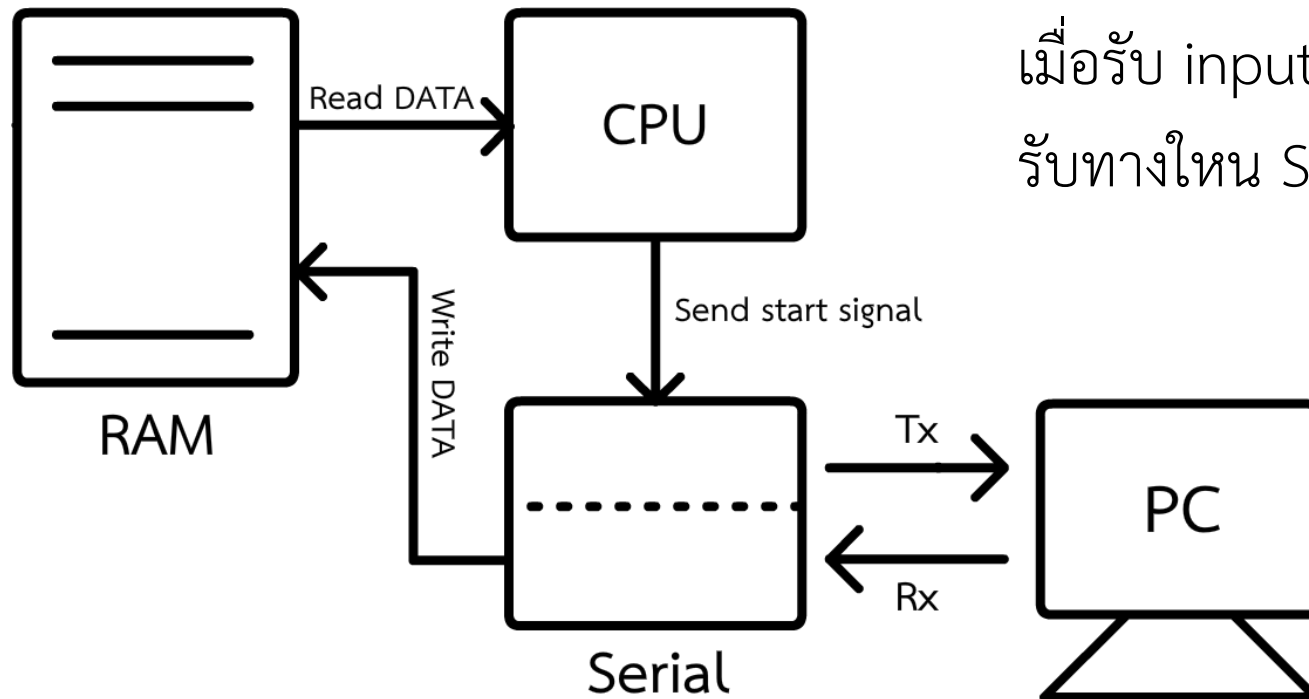
จะสังเกตว่า CPU ทำเพียงแค่การอ่านค่า และนำเข้า function sprintf การเปลี่ยนแปลงค่าของตัวแปร *ADC_value* ถูกเปลี่ยนจากที่อื่น ไม่ใช่ด้วย CPU และ main function

```
ADC = 1246
ADC = 1232
ADC = 1238
ADC = 1220
ADC = 1244
ADC = 1222
ADC = 1238
ADC = 1246
ADC = 1240
ADC = 1233
ADC = 1216
ADC = 1247
ADC = 1233
ADC = 1232
ADC = 1234
ADC = 1254
ADC = 1235
ADC = 1211
```

```
98  MX_ADC1_Init();
99  /* USER CODE BEGIN 2 */
100 HAL_ADCEx_Calibration_Start(&hadc1); // Calibrate ADC
101
102 unsigned char string_buffer[256];
103 int string_buffer_size = -1;
104
105 volatile unsigned int ADC_value = 0;
106
107 HAL_ADC_Start_DMA(&hadc1, &ADC_value, 1);
108 /* USER CODE END 2 */
109
110 /* Infinite loop */
111 /* USER CODE BEGIN WHILE */
112 while (1)
113 {
114     string_buffer_size = sprintf(string_buffer, "ADC = %d\r\n", ADC_value);
115     HAL_UART_Transmit(&huart2, string_buffer, string_buffer_size, 1000);
116     HAL_Delay(1000);
117     /* USER CODE END WHILE */
118 }
```

DMA Keep change *ADC_value* Value

Serial DMA



เมื่อรับ input จาก PC ให้เขียนข้อมูลลงในตัวแปรรับทางไหน Serial ผ่านทาง Keyboard

LAB3 DMA Serial

The screenshot shows the STM32CubeMX Pinout & Configuration window. The left sidebar lists various components, with USART2 selected. The main area displays the USART2 Mode and Configuration settings. The DMA Settings tab is active, showing a table of DMA requests. The table has columns for DMA Request, Channel, Direction, and Priority. The first entry is USART2_RX, DMA1 Channel 1, Peripheral To Memory, and Low priority. Below the table, the DMA Request Settings are configured: Mode is set to Circular, Increment Address is unchecked, Data Width is set to Byte, and Memory is checked. The DMA Request Synchronization Settings section is partially visible at the bottom.

DMA Request	Channel	Direction	Priority
USART2_RX	DMA1 Channel 1	Peripheral To Memory	Low

DMA Request Settings

Mode	Increment Address	Data Width	Memory
Circular	<input type="checkbox"/>	Byte	<input checked="" type="checkbox"/>

DMA Request Synchronization Settings

Enable synchronization ☐

Add ADC

- USART2_RX (ตัวรับ, Receiver)
- Circular Mode
- Byte data (ASCII code)

LAB3 DMA Serial

The screenshot displays the STM32CubeMX configuration tool. On the left, a sidebar lists various peripherals, with 'USART2' selected and highlighted in blue. The main area shows the 'NVIC Settings' tab, which contains the 'NVIC Interrupt Table'. This table lists two interrupts: 'DMA1 channel 1 interrupt' and 'USART2 global interrupt / USART2 wake-up interrupt through EXTI line 26'. The 'Enabled' column for 'USART2' has a blue checkmark, and the 'Preemption Priority' is set to 0. A red underline is placed under the 'Enabled' checkbox for USART2, and a red text annotation 'Enable Interrupt Signal' points to it.

Reset Configuration				
Parameter Settings	User Constants	NVIC Settings	DMA Settings	GPIO Settings
NVIC Interrupt Table		Enabled	Preemption Priority	
DMA1 channel 1 interrupt		<input checked="" type="checkbox"/>	0	
USART2 global interrupt / USART2 wake-up interrupt through EXTI line 26		<input checked="" type="checkbox"/>	0	

Enable Interrupt Signal

Coding

```
94  MX_USART2_UART_Init();
95  /* USER CODE BEGIN 2 */
96  unsigned char string_buffer[256];
97  int string_buffer_size = -1;
98  volatile unsigned char Rx_value = 'H';
99
100 HAL_UART_Receive_DMA(&huart2, &Rx_value, 1);
101 /* USER CODE END 2 */
102
103 /* Infinite loop */
104 /* USER CODE BEGIN WHILE */
105 while (1)
106 {
107     string_buffer_size = sprintf(string_buffer, "Rx = %c\r\n", Rx_value);
108     HAL_UART_Transmit(&huart2, string_buffer, string_buffer_size, 1000);
109     HAL_Delay(1000);
110     /* USER CODE END WHILE */
111
```

```
/* USER CODE BEGIN 2 */
unsigned char string_buffer[256];
int string_buffer_size = -1;
volatile unsigned char Rx_value = 'H';

HAL_UART_Receive_DMA(&huart2, &Rx_value, 1);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    string_buffer_size = sprintf(string_buffer, "Rx = %c\r\n", Rx_value);
    HAL_UART_Transmit(&huart2, string_buffer, string_buffer_size, 1000);
    HAL_Delay(1000);
    /* USER CODE END WHILE */

```

Coding

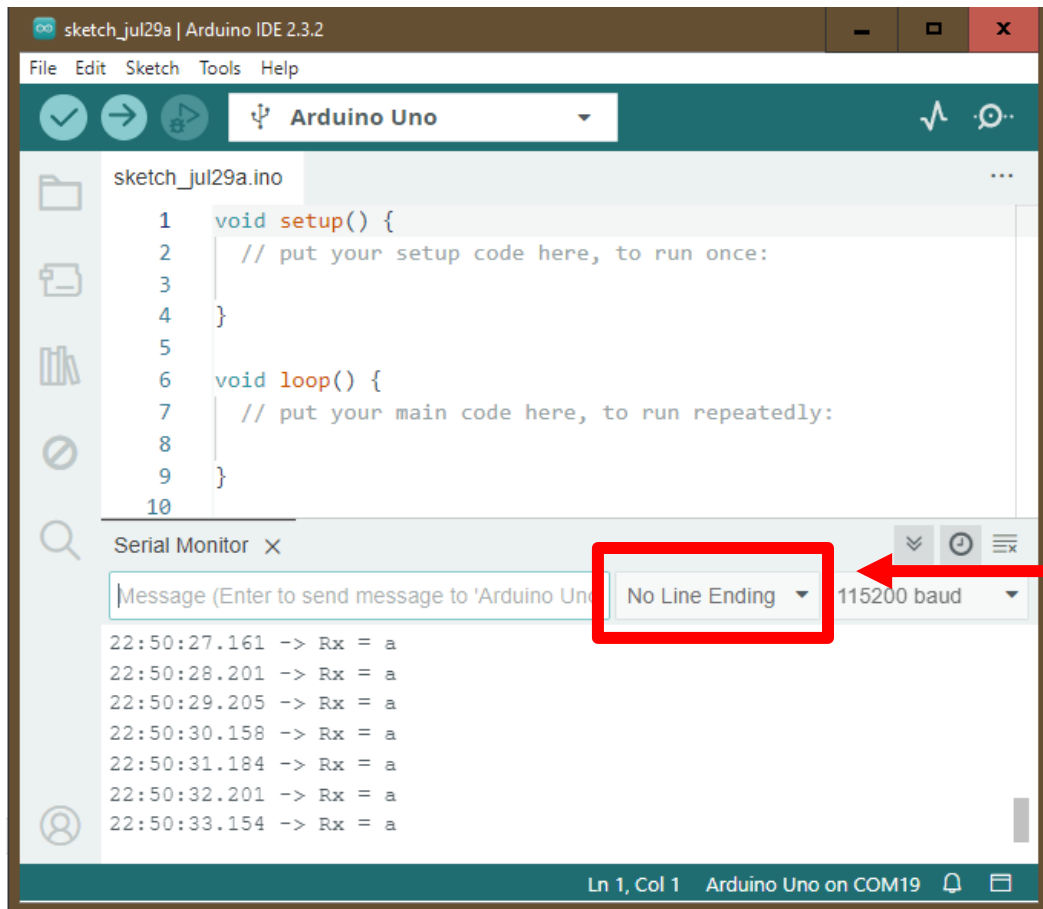
ลองพิมพ์ตัวอย่างที่ console ลองดู

```
Rx = H  
aRx = a  
rRx = r  
uRx = u  
hRx = h  
iRx = i  
Rx = i
```

```
94  MX_USART2_UART_Init();  
95  /* USER CODE BEGIN 2 */  
96  unsigned char string_buffer[256];  
97  int string_buffer_size = -1;  
98  volatile unsigned char Rx_value = 'H';  
99  
100 HAL_UART_Receive_DMA(&huart2, &Rx_value, 1);  
101 /* USER CODE END 2 */  
102  
103 /* Infinite loop */  
104 /* USER CODE BEGIN WHILE */  
105 while (1)  
106 {  
107     string_buffer_size = sprintf(string_buffer, "Rx = %c\r\n", Rx_value);  
108     HAL_UART_Transmit(&huart2, string_buffer, string_buffer_size, 1000);  
109     HAL_Delay(1000);  
110     /* USER CODE END WHILE */  
...
```

DMA Keep change *Rx_value* Value

*สำหรับคนที่ใช้ Arduino



ให้เลือก ending word เป็น No Line Ending

input

output

Rx = H

aRx = a

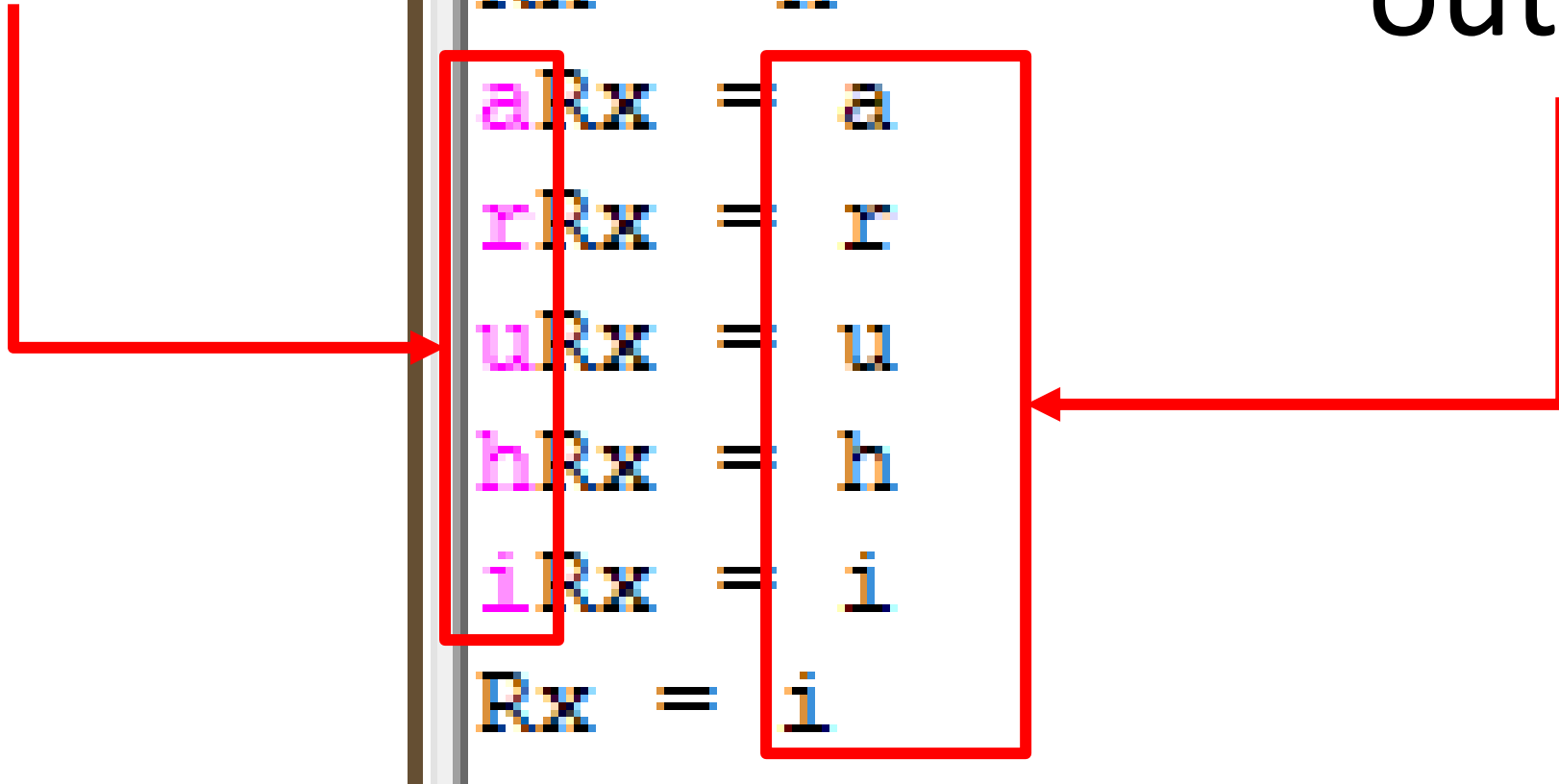
rRx = r

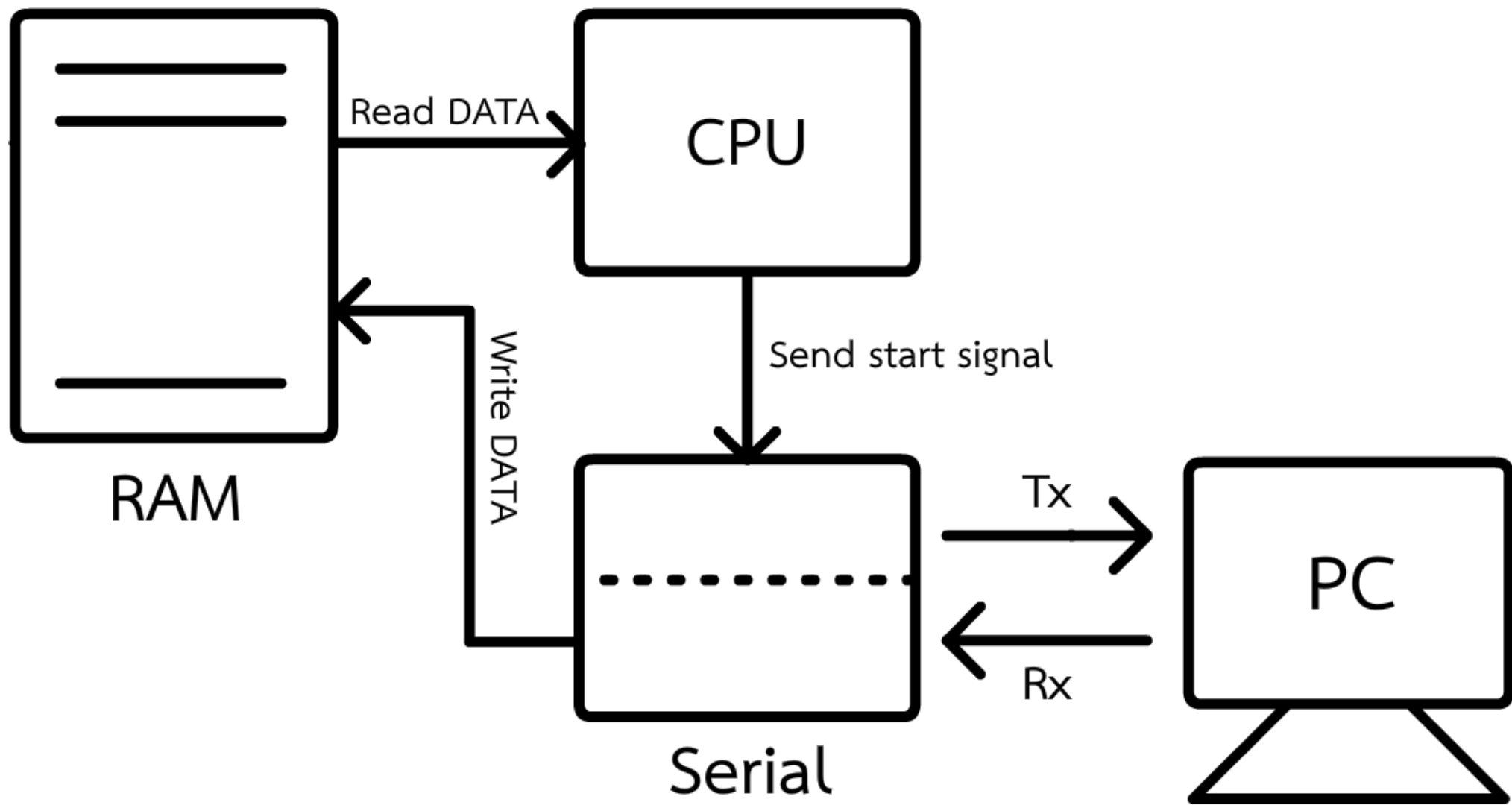
uRx = u

hRx = h

iRx = i

Rx = i





LAB ให้เลือก 2 อย่าง ทำแบบไหนก็ได้

1. ถ้ากด A-Z ที่ keyboard ให้ไฟ LD4 ติด แต่ถ้า กดเลข 1-9 ให้ LD4 ดับ
2. ถ้าจ่าย 0-2 V เข้าที่ขา A0 ให้ LD4 ติด แต่ถ้า จ่าย 2-3.3V ให้ LD4 ดับ

ทำอะไรก็ได้ เลือกมา 1 อย่าง

กัณลีม (Reminder)

สั่งให้ไฟติด

```
HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, 1);
```

สั่งให้ไฟดับ

```
HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, 0);
```