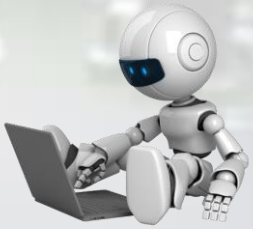
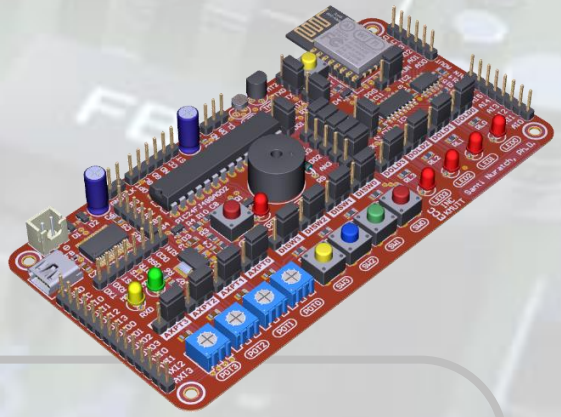
Week #03 – APIs & Functions

INC 362:

Computer-based Control and Monitoring for Modern Industrial Automation Systems





Asst.Prof.Dr.Santi Nuratch

Embedded Computing and Control Lab. @ INC-KMUTT

santi.inc.kmutt@gmail.com

Department of Control System and Instrumentation Engineering,
King Mongkut's University of Technology Thonburi, **KMUTT**

APIs and Functions

All details can be found in the ecc-os-apis.pdf

MCU Functions

1 MCU

1.1 MCU_Init

C	`void MCU_Init(void)`
Description	
กำหนดค่าเริ่มต้นต่าง ๆ ให้กับไมโครคอนโทรลเลอร์	
Parameter	
void	
Return	
void	

LED_Init, LED_On, LED_Set

2 LED

2.1 LED_Init

C	`void LED_Init(void)`
Description	
กำหนดการทำงานของพอร์ต (ขาของไมโครคอนโทรลเลอร์) ที่ต่ออยู่กับ LED ทั้ง 4 ตัวบนบอร์ดให้ทำงานเป็นพอร์ตแบบเอาต์พุต และสั่งให้ LED ทั้ง 4 ตัวบนบอร์ดดับ	
Parameter	
void	
Return	
void	

2.2 LED_On/LED_Set

C	`void LED_On(uint8_t id) / void LED_Set(uint8_t id)`	
Description		
สั่งให้ LED หมายเลขที่กำหนดด้วย id ติด		
Parameter	Details	
id	หมายเลขของ LED มีค่าเท่ากับ 0, 1, 2 หรือ 3 (LED_ID_0, LED_ID_1, LED_ID_2 หรือ LED_ID_3)	
Return		
void		

LEDx_On, LEDx_Set

2.3 LED0_On/LED0_Set

C	`void LED0_On(void) / void LED0_Set(void)`
Description	
สั่งให้ LED หมายเลข 0 ติด	
Parameter	
void	
Return	
void	

2.4 LED1_On/LED1_Set

C	`void LED1_On(void) / void LED1_Set(void)`
Description	
สั่งให้ LED หมายเลข 1 ติด	
Parameter	
void	
Return	
void	

LEDx_On, LEDx_Set

2.5 LED2_On/LED2_Set

C	`void LED2_On(void) / void LED2_Set(void)`
Description	
สั่งให้ LED หมายเลข 2 ติด	
Parameter	
void	
Return	
void	

2.6 LED3_On/LED3_Set

C	`void LED3_On(void) / void LED3_Set(void)`
Description	
สั่งให้ LED หมายเลข 3 ติด	
Parameter	
void	
Return	
void	

LED_Off, LED_Clr

2.7 LED_Off/LED_Clr

C	`void LED_Off(uint8_t id) / void LED_Clr(uint8_t id)`
Description	
สั่งให้ LED หมายเลขที่กำหนดด้วย id ดับ	
Parameter	Details
id	หมายเลขของ LED มีค่าเท่ากับ 0, 1, 2 หรือ 3 (LED_ID_0, LED_ID_1, LED_ID_2 หรือ LED_ID_3)
Return	
void	

2.8 LED0_Off/LED0_Clr

C	`void LED0_Off(void) / void LED0_Clr(void)`
Description	
สั่งให้ LED หมายเลข 0 ดับ	
Parameter	
void	
Return	
void	

LEDx_Off, LEDx_Clr

2.9 LED1_Off/LED1_Clr

C	`void LED1_Off(void) / void LED1_Clr(void)`
Description	
สั่งให้ LED หมายเลข 1 ดับ	
Parameter	
void	
Return	
void	

2.10 LED2_Off/LED2_Clr

C	`void LED2_Off(void) / void LED2_Clr(void)`
Description	
สั่งให้ LED หมายเลข 2 ดับ	
Parameter	
void	
Return	
void	

LEDx_Off, LEDx_Clr, LED_Inv

2.11 LED3_Off/LED3_Clr

C `void LED3_Off(void) / void LED3_Clr(void)`
Description
สั่งให้ LED หมายเลข 3 ดับ
Parameter
void
Return
void

2.12 LED_Inv/LED_Toggle

C `void LED_Inv(uint8_t id) / void LED_Toggle(uint8_t id)`	
Description	
สั่งให้ LED หมายเลขที่กำหนดด้วย id กลับสถานะตัวเอง (ติดเปลี่ยนเป็นดับ หรือ ดับเปลี่ยนเป็นติด)	
Parameter	Details
id	หมายเลขของ LED มีค่าเท่ากับ 0, 1, 2 หรือ 3 (LED_ID_0, LED_ID_1, LED_ID_2 หรือ LED_ID_3)
Return	
void	

LEDx_Inv, LEDx_Toggle

2.13 LED0_Inv/LED0_Toggle

C	`void LED0_Inv(void) / void LED0_Toggle(void)`
Description	
สั่งให้ LED หมายเลขหลายเลข 0 กลับสถานะตัวเอง (ติดเปลี่ยนเป็นดับ หรือ ดับเปลี่ยนเป็นติด)	
Parameter	
void	
Return	
void	

2.14 LED1_Inv/LED1_Toggle

C	`void LED1_Inv(void) / void LED1_Toggle(void)`
Description	
สั่งให้ LED หมายเลขหลายเลข 1 กลับสถานะตัวเอง (ติดเปลี่ยนเป็นดับ หรือ ดับเปลี่ยนเป็นติด)	
Parameter	
void	
Return	
void	

LEDx_Inv, LEDx_Toggle

2.15 LED2_Inv/LED2_Toggle

C `void LED2_Inv(void) / void LED2_Toggle(void)`
Description
สั่งให้ LED หมายเลขหลายเลข 2 กลับสถานะตัวเอง (ติดเปลี่ยนเป็นดับ หรือ ดับเปลี่ยนเป็นติด)
Parameter
void
Return
void

2.16 LED3_Inv/LED3_Toggle

C `void LED3_Inv(void) / void LED3_Toggle(void)`
Description
สั่งให้ LED หมายเลขหลายเลข 3 กลับสถานะตัวเอง (ติดเปลี่ยนเป็นดับ หรือ ดับเปลี่ยนเป็นติด)
Parameter
void
Return
void

LED_Write

2.17 LED_Write

C	`void LED_Write(uint8_t data)`
Description	
เขียนข้อมูลขนาด 1 ไบต์ออกไปยัง LED ทั้ง 4 ตัว (ข้อมูล 4 บิตด้านบน จะไม่ส่งผลกับการทำงานของ LED)	
Parameter	
void	
Return	
void	
Remark	
LED ทั้ง 4 ตัวบนบอร์ดทำงานแบบ Active Low ดังนั้นการเขียนค่าลอจิก 0 ไปยังตำแหน่งบิตที่ตรงกับหมายลข LED คือการสั่งให้ LED ติด เช่น ค่า 0x0E จะทำให้ LED หมายเลข 0 ติดเพียงตัวเดียว	

LED_Get, LEDx_Get

2.18 LED_Get

C `uint8_t LED_Get(uint8_t id)`

Description	
อ่านค่าสถานะของ LED หมายเลขที่กำหนดด้วย id	
Parameter	**Details**
id	หมายเลขของ LED มีค่าเท่ากับ 0, 1, 2 หรือ 3 (LED_ID_0, LED_ID_1, LED_ID_2 หรือ LED_ID_3)
Return	
สถานะของ LED มีค่าเป็น 0 (LED_ON) หรือ 1 (LED_OFF) (LED บนบอร์ดทำงานแบบ Active Low)	

2.19 LED0_Get

C `uint8_t LED0_Get(void)`

Description
อ่านค่าสถานะของ LED หมายเลข 0
Parameter
void
Return
สถานะของ LED หมายเลข 0 มีค่าเป็น 0 (LED_ON) หรือ 1 (LED_OFF) (LED บนบอร์ดทำงานแบบ Active Low)

LEDx_Get

2.20 LED1_Get

```
uint8_t LED1_Get(void)
```

Description
อ่านค่าสถานะของ LED หมายเลข 1
Parameter
void
Return
สถานะของ LED หมายเลข 1 มีค่าเป็น 0 (LED_ON) หรือ 1 (LED_OFF) (LED บนบอร์ดทำงานแบบ Active Low)

2.21 LED2_Get

```
uint8_t LED2_Get(void)
```

Description
อ่านค่าสถานะของ LED หมายเลข 2
Parameter
void
Return
สถานะของ LED หมายเลข 2 มีค่าเป็น 0 (LED_ON) หรือ 1 (LED_OFF) (LED บนบอร์ดทำงานแบบ Active Low)

LEDx_Get, LED_Read

2.22 LED3_Get

C uint8_t **LED3_Get**(void)
Description
อ่านค่าสถานะของ LED หมายเลข 3
Parameter
void
Return
สถานะของ LED หมายเลข 3 มีค่าเป็น 0 (LED_ON) หรือ 1 (LED_OFF) (LED บนบอร์ดทำงานแบบ Active Low)

2.23 LED_Read

C uint8_t **LED_Read**(void)
Description
อ่านข้อมูลขนาด 1 ไบต์จากพอร์ตที่ต่ออยู่กับ LED ทั้ง 4 ตัว (ข้อมูล 4 บิตด้านบนจะเป็น 0)
Parameter
void
Return
void
Remark
LED ทั้ง 4 ตัวบนบอร์ดทำงานแบบ Active Low ดังนั้นถ้าค่าลอจิกที่อ่านมาได้เป็นลอจิก 0 แสดงว่า LED ในตำแหน่งบิตเดียวกันนี้อยู่ในสถานะติด เช่น ค่าที่อ่านมาได้เท่ากับ 0x0E หมายความว่า ณ เวลานี้ LED หมายเลข 0 ติดอยู่เพียงตัวเดียว

LED_SetMode

2.24 LED_SetMode

```
C  void LED_SetMode(uint8_t id, uint8_t mode)
```

Description

กำหนดการทำงานของ LED ที่ระบุด้วย id ให้เป็นแบบปกติ (LED_MODE_NORMAL) หรือ แบบ PWM (LED_MODE_PWM) ตามที่กำหนดด้วย mode

Parameter	Details
id	หมายเลขของ LED มีค่าเท่ากับ 0, 1, 2 หรือ 3 (LED_ID_0, LED_ID_1, LED_ID_2 หรือ LED_ID_3)
mode	โหมด (รูปแบบ) การทำงานของ LED มีค่าเป็น LED_MODE_NORMAL หรือ LED_MODE_PWM การทำงานในโหมดปกติ การติดดับของ LED จะถูกควบคุมโดยฟังก์ชันควบคุม การทำงานในโหมก PWM การติดดับของ LED จะขึ้นอยู่กับค่าพารามิเตอร์ของสัญญาณ PWM ที่กำหนด

Return

void

LED_SetPwm

2.25 LED_SetPwm

```c
void LED_SetPwm(uint8_t id, uint16_t period,
                uint16_t shift_time, uint16_t on_time)
```

Description

กำหนดค่าพารามิเตอร์ของสัญญาณ PWM ซึ่งเป็นตัวกำหนดรูปแบบการกระพริบของ LED หมายเลขที่กำหนดด้วย id

Parameter	Details
id	หมายเลขของ LED มีค่าเท่ากับ 0, 1, 2 หรือ 3 (LED_ID_0, LED_ID_1, LED_ID_2 หรือ LED_ID_3)
period	กำหนดระยะเวลาทั้งคาบของสัญญาณ
shift_time	กำหนดระยะเวลาหน่วงก่อนที่จะทำให้สัญญาณ PWM เปลี่ยนจาก HIGH เป็น LOW (LED ติด)
on_time	กำหนดระยะเวลาของสัญญาณ PWM ช่วงที่เป็น LOW (LED ติด)

Return

void

Remark

ด้วยคุณสมบัติของสัญญาณ PWM คือ ระยะเวลาที่เป็น LOW รวมกับระยะเวลาที่เป็น HIGH จะต้องมีค่าเท่ากันกับระยะเวลาทั้งคาบ (PERIOD) ดังนั้นการกำหนดค่าพารามิเตอร์ต่าง ๆ ทั้ง 3 ตัว จะต้องพิจารณาโดยคำนึงถึงคุณสมบัติของสัญญาณ PWM เพื่อให้ผลลัพธ์ออกมาถูกต้อง

LED ทั้ง 4 ตัวบนบอร์ด ทำงานแบบ Active Low ด้วยเหตุนี้เมื่อสะญญาณ PWM มีสถานะเป็น LOW จะทำให้ LED ติด นั่นหมายความว่า on_time ในที่นี้เป็นตัวกำหนดระยะเวลาช่วงที่เป็น LOW ของสัญญาณ PWM หรือกำหนดระยะเวลาที่ทำให้ LED ติดนั่นเอง

หน่วยวัดเชิงเวลาของพารามิเตอร์ทั้ง 3 ตัวเป็นหน่วยมิลลิวินาที เช่นถ้ากำหนด period มีค่าเท่ากับ 500 หมายความว่าสัญญาณ PWM นี้จะมีระยะเวลาทั้งคาบเท่ากับ 0.5 วินาที และมีความถี่เท่ากับ 2 เฮิรตซ์ (Hz)

LED_SetPwmPeriod

2.26 LED_SetPwmPeriod

```
C  void LED_SetPwmPeriod(uint8_t id, uint16_t period)
```

Description	
กำหนดระยะเวลาทั้งคาบของสัญญาณ PWM ซึ่งเป็นตัวกำหนดรูปแบบการกระพริบของ LED หมายเลขที่กำหนดด้วย id	
Parameter	**Details**
id	หมายเลขของ LED มีค่าเท่ากับ 0, 1, 2 หรือ 3 (LED_ID_0, LED_ID_1, LED_ID_2 หรือ LED_ID_3)
period	กำหนดระยะเวลาทั้งคาบของสัญญาณ
Return	
void	
Remark	
shift-time (phase shift) ของสัญญาณ PWM จะถูกคำนวณใหม่อัตโนมัติเพื่อรักษา on-time ค่าเดิมไว้	

LED_SetPwmShift, LED_SetPwmOnTime

2.27 LED_SetPwmShift

```
void LED_SetPwmShift(uint8_t id, uint16_t shift)
```

Description

กำหนดระยะเวลาหน่วงของสัญญาณ PWM ซึ่งเป็นตัวกำหนดรูปแบบการกระพริบของ LED หมายเลขที่กำหนดด้วย id

Parameter	Details
id	หมายเลขของ LED มีค่าเท่ากับ 0, 1, 2 หรือ 3 (LED_ID_0, LED_ID_1, LED_ID_2 หรือ LED_ID_3)
shift	กำหนดระยะเวลาหน่วงก่อนที่จะทำให้สัญญาณ PWM เปลี่ยนจาก HIGH เป็น LOW (LED ติด)

Return

void

Remark

on-time (ช่วงระยะเวลาที่ LED ติด) จะถูกคำนวณใหม่อัตโนมัติเพื่อรักษา period ค่าเดิมไว้

2.28 LED_SetPwmOnTime

```
void LED_SetPwmOnTime (uint8_t id, uint16_t on_time)
```

Description

กำหนดระยะเวลาหน่วงของสัญญาณ PWM ซึ่งเป็นตัวกำหนดรูปแบบการกระพริบของ LED หมายเลขที่กำหนดด้วย id

Parameter	Details
id	หมายเลขของ LED มีค่าเท่ากับ 0, 1, 2 หรือ 3 (LED_ID_0, LED_ID_1, LED_ID_2 หรือ LED_ID_3)
on_time	กำหนดระยะเวลาของสัญญาณ PWM ช่วงที่เป็น LOW (LED ติด)

Return

void

Remark

shift-time (phase-shift) ของสัญญาณ PWM จะถูกคำนวณใหม่อัตโนมัติเพื่อรักษา period ค่าเดิมไว้

UART1_Init

3 UART

3.1 UART1_Init

```
void UART1_Init(uint32_t baudrate)
```

Description	
กำหนดค่าเริ่มต้นต่าง ๆ ให้กับพอร์ตอนุกรมหมายเลข 1	
Parameter	**Details**
baudrate	กำหนดความเร็วในการรับส่งข้อมูล (จำนวนบิตต่อวินาที) หรือ Baud rate
Return	
void	

UART2_Init, UART_Init

3.2 UART2_Init

C	`void UART2_Init(uint32_t baudrate)`

Description	
กำหนดค่าเริ่มต้นต่าง ๆ ให้กับพอร์ตอนุกรมหมายเลข 2	
Parameter	**Details**
baudrate	กำหนดความเร็วในการรับส่งข้อมูล (จำนวนบิตต่อวินาที) หรือ Baud rate
Return	
void	

3.3 UART_Init

C	`void UART_Init(uint8_t uart_id, uint32_t baudrate)`

Description	
กำหนดค่าเริ่มต้นต่าง ๆ ให้กับพอร์ตอนุกรมหมายเลข 1 หรือ 2 ตามที่กำหนดด้วย uart_id	
Parameters	**Details**
uart_id	กำหนดหมายเลขของพอร์ตอนุกรม มีค่าเป็น 1 (UART_ID_1) หรือ 2 (UART_ID_2)
baudrate	กำหนดความเร็วในการรับส่งข้อมูล (จำนวนบิตต่อวินาที) หรือ Baud rate
Return	
void	

Uartx_AsyncWriteString

3.4 Uart1_AsyncWriteString

```
uint16_t Uart1_AsyncWriteString (const char * message)
```

Description	
เขียนข้อความไปยังพอร์ตอนุกรมหมายเลข 1 แบบ Non-Blocking (ไม่หยุดรอให้การทำงานของฮาร์ดแวร์เสร็จสิ้น)	
Parameters	**Details**
message	ข้อความ (อาร์เรย์ของตัวอักษร) ที่ต้องการเขียนไปยังพอร์ตอนุกรมหมายเลข 1
Return	
จำนวนตัวอักษรที่เขียนไปยัง Queue ของพอร์ตอนุกรมหมายเลข 1 ในกรณีที่ Queue เต็ม หรือพื้นที่ว่างของ Queue ไม่เพียงพอจะคืนค่ามาเป็น 0	

3.5 Uart2_AsyncWriteString

```
uint16_t Uart2_AsyncWriteString (const char * message)
```

Description	
เขียนข้อความไปยังพอร์ตอนุกรมหมายเลข 2 แบบ Non-Blocking (ไม่หยุดรอให้การทำงานของฮาร์ดแวร์เสร็จสิ้น)	
Parameters	**Details**
message	ข้อความ (อาร์เรย์ของตัวอักษร) ที่ต้องการเขียนไปยังพอร์ตอนุกรมหมายเลข 2
Return	
จำนวนตัวอักษรที่เขียนไปยัง Queue ของพอร์ตอนุกรมหมายเลข 2 ในกรณีที่ Queue เต็ม หรือพื้นที่ว่างของ Queue ไม่เพียงพอจะคืนค่ามาเป็น 0	

Uart_AsyncWriteString

3.6 Uart_AsyncWriteString

```
uint16_t Uart_AsyncWriteString (uint8_t uart_id, const char * message)
```

Description

เขียนข้อความไปยังพอร์ตอนุกรมหมายเลข1 หรือ 2 ตามที่กำหนดด้วย uart_id แบบ Non-Blocking (ไม่หยุดรอให้การทำงานของฮาร์ดแวร์เสร็จสิ้น)

Parameters	Details
uart_id	กำหนดหมายเลขของพอร์ตอนุกรม มีค่าเป็น 1 (UART_ID_1) หรือ 2 (UART_ID_2)
message	ข้อความ (อาร์เรย์ของตัวอักษร) ที่ต้องการเขียนไปยังพอร์ตอนุกรม

Return

จำนวนตัวอักษรที่เขียนไปยัง Queue ของพอร์ตอนุกรมหมายเลข 2 ในกรณีที่ Queue เต็ม หรือพื้นที่ว่างของ Queue ไม่เพียงพอจะคืนค่ามาเป็น 0

3.7 Uart1_AsyncWriteBytes

```
uint16_t Uart1_AsyncWriteBytes(const uint8_t * data, uint16_t length)
```

Description

เขียนข้อมูล (byte data) ไปยังพอร์ตอนุกรมหมายเลข 1 แบบ Non-Blocking (ไม่หยุดรอให้การทำงานของฮาร์ดแวร์เสร็จสิ้น)

Parameters	Details
data	ข้อมูล (อาร์เรย์ของ byte data) ที่ต้องการเขียนไปยังพอร์ตอนุกรมหมายเลข 1
length	ความยาวของข้อมูล (จำนวนไบต์) ที่อยู่ใน data

Return

จำนวนไบต์ที่เขียนไปยัง Queue ของพอร์ตอนุกรมหมายเลข 1 ในกรณีที่ Queue เต็ม หรือพื้นที่ว่างของ Queue ไม่เพียงพอจะคืนค่ามาเป็น 0

Uart2_AsyncWriteBytes

3.8 Uart2_AsyncWriteBytes

```
uint16_t Uart2_AsyncWriteBytes(const uint8_t * data, uint16_t length)
```

Description

เขียนข้อมูล (byte data) ไปยังพอร์ตอนุกรมหมายเลข 2 แบบ Non-Blocking (ไม่หยุดรอให้การทำงานของฮาร์ดแวร์เสร็จสิ้น)

Parameters	Details
data	ข้อมูล (อาร์เรย์ของ byte data) ที่ต้องการเขียนไปยังพอร์ตอนุกรมหมายเลข 2
length	ความยาวของข้อมูล (จำนวนไบต์) ที่อยู่ใน data

Return

จำนวนไบต์ที่เขียนไปยัง Queue ของพอร์ตอนุกรมหมายเลข 2 ในกรณีที่ Queue เต็ม หรือพื้นที่ว่างของ Queue ไม่เพียงพอจะคืนค่ามาเป็น 0

พอร์ตอนุกรมแต่ละตัวจะมี Queue-Buffer ที่ใช้ในการรับและส่งข้อมูลเป็นตัวเองแยกออกจากกันอย่างอิสระ กระบวนการของ Queue-Buffer นี้จะทำงานอยู่เบื้องหลังภายใต้ระบบปฏิบัติการและการอินเตอร์รัพท์ของพอร์ตอนุกรม

OS_Init, OS_Start

4 OS

4.1 OS_Init

C void OS_Init(void)
Description
กำหนดค่าเริ่มต้นต่าง ๆ ให้กับระบบปฏิบัติการ
Parameter
void
Return
void

4.2 OS_Start

C void OS_Start(void)
Description
ให้กับระบบปฏิบัติการเริ่มทำงาน เมื่อฟังก์ชันนี้ถูกเรียกใช้ ทั้งระบบจะถูกจัดการด้วยระบบปฏิบัติการอย่างเต็มรูปแบบ
Parameter
void
Return
void

OS_TimeSet, OS_TimeGet

4.3 OS_TimeSet

```
C  void OS_TimeSet(uint8_t hh, uint8_t mm, uint8_t ss)
```

Description

กำหนดหรือตั้งค่าเวลาของระบบใหม่ เมื่อฟังก์ชันนี้ถูกเรียกใช้เวลาในหลัก ms (มิลลิวินาที) และ us (ไมโครวินาที) จะถูกรีเซตเป็น 0 อัตโนมัติ

Parameters	Details
hh	กำหนดเวลาในหลักชั่วโมง (0-23)
mm	กำหนดเวลาในหลักนาที (0-59)
ss	กำหนดเวลาในหลักวินาที (0-59)

Return

void

4.4 OS_TimeGet

```
C  os_time_t OS_TimeGet(void)
```

Description

อ่านค่าเวลาปัจจุบันของระบบ (เวลาที่ระบบทำงานมาจนถึงปัจจุบัน) เวลาของระบบจะเริ่มนับเมื่อฟังก์ชัน OS_Start ถูกเรียกใช้ เวลาของระบบเป็นตัวแปรแบบโครงสร้างประกอบด้วย hh (ชั่วโมง) mm (นาที) ss (วินาที) ms (มิลลิวินาที) และ us (ไมโครวินาที)

Parameter

void

Return

ค่าเวลาปัจจุบันของระบบเป็นเป็นตัวแปรแบบโครงสร้างชนิด os_time_t

OS_TimeHighResolution, OS_TimeMicroseconds

4.5 OS_TimeHighResolution

C	`double OS_TimeHighResolution(void)`
Description	
อ่านค่าเวลาปัจจุบันของระบบแบบความละเอียดสูงในหน่วยนับมิลลิวินาที (mS) ค่าที่ได้จะเป็นค่าจำนวนจริง	
Parameter	
void	
Return	
เวลาปัจจุบันของระบบในหน่วยนับมิลลิวินาที (mS) แบบความละเอียดสูง (เป็นตัวเลขจำนวนจริงที่มีความละเอียดแบบ double precision)	

4.6 OS_TimeMicroseconds

C	`uint32_t OS_TimeMicroseconds(void)`
Description	
อ่านค่าเวลาปัจจุบันของระบบแบบความละเอียดสูงในหน่วยนับไมโครวินาที (uS) ค่าที่ได้จะเป็นค่าจำนวนเต็ม	
Parameter	
void	
Return	
เวลาปัจจุบันของระบบในหน่วยนับไมโครวินาที (uS) เป็นตัวเลขจำนวนเต็มขนาด 32 บิต	

OS_TimeMilliseconds, OS_TickedCheck

4.7 OS_TimeMilliseconds

C `uint32_t OS_TimeMilliseconds(void)`
Description
อ่านค่าเวลาปัจจุบันของระบบแบบความละเอียดสูงในหน่วยนับมิลลิวินาที (mS) ค่าที่ได้จะเป็นค่าจำนวนเต็ม
Parameter
void
Return
เวลาปัจจุบันของระบบในหน่วยนับมิลลิวินาที (mS) เป็นตัวเลขจำนวนเต็มขนาด 32 บิต

4.8 OS_TickedCheck

C `uint32_t OS_TickedCheck(void)`
Description
ตรวจสอบสถานะของระบบปฏิบัติการว่าได้ถูกกระตุ้นด้วย System tick แล้วหรือไม่ เมื่อฟังก์ชันนี้ถูกเรียกใช้ สถาะการถูกกระตุ้นของระบบปฏิบัติการจะถูกรีเซต (ระบบปฏิบัติการจะถูกกระตุ้นทุก ๆ 1 mS)
Parameter
void
Return
ในกรณีที่ระบบปฏิบัติการถูกกระตุ้นแล้วจะคืนค่าเป็น 1 และจะคืนค่าเป็น 0 หากระบบปฏิบัติการยังไม่ถูกกระตุ้น

OS_Sleep, OS_SystemTickSetCallback

4.9 OS_Sleep

C	`void OS_Sleep(void)`

Description
หยุดการทำงานของ time-constrained looper ฟังก์ชันนี้ถูกเรียกใช้ระบบปฏิบัติการจะไม่ตอบสนองกับส่วนต่าง ๆ ของระบบที่ต้องการทำงานตามเวลา เช่น Worker และ Timer แต่เวลาของระบบและฟังก์ชัน callback ต่าง ๆ ยังคงทำงานตามปกติ
Parameter
void
Return
void

4.10 OS_SystemTickSetCallback

C	`void OS_SystemTickSetCallback(uint16_t ticks, os_callback_t callback)`

Description	
กำหนดฟังก์ชัน callback เพื่อให้ระบบปฏิบัติการเรียกใช้ทันทีทันใดเมื่อถึงเวลาที่กำหนดด้วยจำนวน system ticks	
Parameters	**Details**
ticks	จำนวน system tick ที่ต้องการให้ฟังก์ชัน callback ถูกประมวลผล (เช่นถ้ากำหนดมีค่าเท่ากับ 10 ฟังก์ชัน callback จะถูกประมวลผลทุก ๆ 10 mS เป็นต้น)
callback	กำหนดฟังก์ชัน callback เพื่อให้ระบบปฏิบัติการไปประมวลผลทันทีทันใดเมื่อถึงเวลาที่กำหนด
Return	
void	
Remark	
ฟังก์ชัน callback ตัวนี้จะถูกเรียกโดยตรงแบบทันทีจากระบบปฏิบัติการ callback นี้จะไม่ถูกเขียนเข้าไปยัง Callback-Queue ออกแบบมาสำหรับงานที่ไม่ต้องการความคลาดเคลื่อนเชิงเวลา ควรหลีกเลี่ยงการใช้งาน callback ตัวนี้ในการประมวลผลที่ต้องใช้เวลานานกว่า 500 uS เพราะอาจจะส่งผลให้ระบบทำงานผิดพลาดได้	

OS_Uart1SetRxCallback

4.11 OS_Uart1SetRxCallback

C `void OS_Uart1SetRxCallback (os_callback_t callback)`

Description

กำหนดฟังก์ชัน callback เพื่อให้ระบบปฏิบัติการเรียกใช้เมื่อพอร์ตอนุกรมหมายเลข 1 ได้รับข้อมูล (byte data) การรับข้อมูลของพอร์ตอนุกรมจะใช้ Ring-Buffer เป็นตัวเก็บข้อมูล ทำงานร่วมกันกับการอินเตอร์รัพท์และระบบปฏิบัติการ ดังนั้นจึงการันตีได้ว่าข้อมูลที่ถูกส่งเข้ามาด้วยความเร็วสูงและต่อเนื่องจะถูกรับได้ครบ 100% ฟังก์ชัน callback นี้จะถูกเรียกโดยระบบปฏิบัติการผ่านทาง Callback-Queue ข้อมูลที่รับเข้ามาแต่ละไบต์จะถูกส่งผ่านไปให้กับฟังก์ชัน callback เมื่อฟังก์ชัน callback ถูกเรียกใช้

Parameters	Details
callback	กำหนดฟังก์ชัน callback เพื่อให้ระบบปฏิบัติการไปประมวลผลเมื่อพอร์ตอนุกรมหลายเลข 1 ได้รับข้อมูล

Return

void

Remark

ข้อมูลที่ระบบปฏิบัติการส่งผ่านไปให้ฟังก์ชัน callback จะเป็นชนิด uart_event_t ซึ่งมีหมายเลขพอร์ต (id) และข้อมูลขนาด 1 ไบต์ (data) มีค่าเท่ากับข้อมูลที่ได้รับเข้ามา

OS_Uart2SetRxCallback

4.12 OS_Uart2SetRxCallback

```
C  void OS_Uart2SetRxCallback (os_callback_t callback)
```

Description

กำหนดฟังก์ชัน callback เพื่อให้ระบบปฏิบัติการเรียกใช้เมื่อพอร์ตอนุกรมหมายเลข 2 ได้รับข้อมูล (byte data) การรับข้อมูลของพอร์ตอนุกรมจะใช้ Ring-Buffer เป็นตัวเก็บข้อมูล ทำงานร่วมกันกับการอินเตอร์รัพท์และระบบปฏิบัติการ ดังนั้นจึงการันตีได้ว่าข้อมูลที่ถูกส่งเข้ามาด้วยความเร็วสูงและต่อเนื่องจะถูกรับได้ครบ 100% ฟังก์ชัน callback นี้จะถูกเรียกโดยระบบปฏิบัติการผ่านทาง Callback-Queue ข้อมูลที่รับเข้ามาแต่ละไบต์จะถูกส่งผ่านไปให้กับฟังก์ชัน callback เมื่อฟังก์ชัน callback ถูกเรียกใช้

Parameters	Details
callback	กำหนดฟังก์ชัน callback เพื่อให้ระบบปฏิบัติการไปประมวลผลเมื่อพอร์ตอนุกรมหลายเลข 1 ได้รับข้อมูล

Return

void

Remark

ข้อมูลที่ระบบปฏิบัติการส่งผ่านไปให้ฟังก์ชัน callback จะเป็นชนิด uart_event_t ซึ่งมีหมายเลขพอร์ต (id) และข้อมูลขนาด 1 ไบต์ (data) มีค่าเท่ากับข้อมูลที่ได้รับเข้ามา

OS_Uart1SetTxCallback

4.13 OS_Uart1SetTxCallback

C `void OS_Uart1SetTxCallback (os_callback_t callback)`

Description

กำหนดฟังก์ชัน callback เพื่อให้ระบบปฏิบัติการเรียกใช้เมื่อพอร์ตอนุกรมหมายเลข 1 ได้ทำการส่งข้อมูลใน Byte-Queue หมดแล้ว (Queue ว่าง) การทำงานของฟังก์ชัน callback ตัวนี้จะเชื่อมโยงกับฟังก์ชันอื่น ๆ ที่ทำหน้าที่ส่งข้อมูลออกไปยังพอร์อนุกรมหมายเลข 1

Parameters	Details
callback	กำหนดฟังก์ชัน callback เพื่อให้ระบบปฏิบัติการไปประมวลผลเมื่อพอร์ตอนุกรมหลายเลข 1 ได้ทำการส่งข้อมูลใน Byte-Queue หมดแล้ว (Queue ว่าง)

Return

void

Remark

ข้อมูลที่ระบบปฏิบัติการส่งผ่านไปให้ฟังก์ชัน callback จะเป็นชนิด uart_event_t ซึ่งมีหมายเลขพอร์ต (id) และข้อมูลขนาด 1 ไบต์ (data) มีค่าเท่ากับ 0

OS_Uart2SetTxCallback

4.14 OS_Uart2SetTxCallback

C `void OS_Uart2SetTxCallback (os_callback_t callback)`

Description

กำหนดฟังก์ชัน callback เพื่อให้ระบบปฏิบัติการเรียกใช้เมื่อพอร์ตอนุกรมหมายเลข 2 ได้ทำการส่งข้อมูลใน Byte-Queue หมดแล้ว (Queue ว่าง) การทำงานของฟังก์ชัน callback ตัวนี้จะเชื่อมโยงกับฟังก์ชันอื่น ๆ ที่ทำหน้าที่ส่งข้อมูลออกไปยังพอร์อนุกรมหมายเลข 2

Parameters	Details
callback	กำหนดฟังก์ชัน callback เพื่อให้ระบบปฏิบัติการไปประมวลผลเมื่อพอร์ตอนุกรมหลายเลข 2 ได้ทำการส่งข้อมูลใน Byte-Queue หมดแล้ว (Queue ว่าง)

Return

void

Remark

ข้อมูลที่ระบบปฏิบัติการส่งผ่านไปให้ฟังก์ชัน callback จะเป็นชนิด uart_event_t ซึ่งมีหมายเลขพอร์ต (id) และข้อมูลขนาด 1 ไบต์ (data) มีค่าเท่ากับ 0

OS_TimerCreate

5 Timer

5.1 OS_TimerCreate

```c
timer_t *OS_TimerCreate(char *name, uint16_t ticks,
                        timer_mode_t mode, timer_callback_t callback)
```

Description

สร้าง Software Timer พร้อมกำหนดลักษณะการทำงานและกำหนด Callback function ให้กับ Timer

Parameters	Details
name	ชื่อของ Timer
ticks	ระยะเวลาของ Timer ของ Timer หน่วยเป็น mS
mode	รูปแบบการทำงานของ Timer สามารถกำหนดให้เป็น **TIMER_MODE_ONESHORT** (ทำงานครั้งเดียวแล้วหยุด) หรือ **TIMER_MODE_CONTINUEOUS** (ทำงานต่อเนื่อง)
callback	Callback function ของ Timer เมื่อถึงเวลาที่กำหนดโดย **ticks** ส่วนของโปรแกรมที่อยู่ใน Callback function จะถูกประมวลผล

Return

Pointer ที่ชี้ไปยัง Timer ที่ถูกสร้างขึ้น

Remark

ระบบปฏิบัติการนี้ยอมให้สร้าง Timer ได้สูงสุด 5 ตัว แต่ละตัวจะมี ID เป็นของตัวเองคือ TIMER_ID_0, TIMER_ID_1, TIMER_ID_2, TIMER_ID_3, TIMER_ID_4

OS_TimerDelete

5.2 OS_TimerDelete

```
int OS_TimerDelete(timer_t *timer)
```

Description

ลบ Timer ออกจากระบบปฏิบัติการ

Parameters	Details
timer	Pointer ที่ชี้ไปยัง Timer ที่ถูกสร้างไว้ก่อนหน้า

Return

1 หากทำการลบได้สมบูรณ์ม

0 หาก Timer ตัวที่กำหนดยังไม่ได้ถูกสร้างหรือถูกลบออกไปแล้ว

Remark

เมื่อ Timer ไม่ได้ใช้งานควรลบออกไปจากระบบ เพื่อเป็นการลดภาระของระบบปฏิบัติการ และคืนหน่วยความจำให้ระบบ

OS_TimerSetCallback

5.3 OS_TimerSetCallback

```
int OS_TimerSetCallback(timer_t *timer, timer_callback_t callback)
```

Description

กำหนด Callback ให้กับ Timer

Parameters	Details
timer	Pointer ที่ชี้ไปยัง Timer ที่ถูกสร้างไว้ก่อนหน้า
callback	Callback function ของ Timer

Return

1 หากทำการกำหนด callback function ได้สมบูรณ์

0 หากทำการกำหนด callback function ล้มเหลว เช่นกำหนด callback function ให้กับ Timer ที่ได้ถูกลบออกไปจากระบบแล้ว

Remark

ในการณีที่ต้องการยกเลิก callback function ของ Timer สามารถทำได้โดยการกำหนดให้ค่า callback เป็น NULL

OS_TimerSetTicks

5.4 OS_TimerSetTicks

```
int OS_TimerSetTicks(timer_t *timer, uint16_t ticks)
```

Description

กำหนดระยะเวลาการทำให้ให้กับ Timer

Parameters	Details
timer	Pointer ที่ชี้ไปยัง Timer ที่ถูกสร้างไว้ก่อนหน้า
ticks	ระยะเวลาของ Timer หน่วยเป็น mS

Return

1 หากทำการกำหนด ticks ได้สมบูรณ์

0 หากทำการกำหนด ticks ล้มเหลว เช่นกำหนด ticks ให้กับ Timer ที่ได้ถูกลบออกไปจากระบบแล้ว

Remark

ค่า ticks ที่กำหนดโยฟังก์ชั่นนี้จะถูก Update ไปยัง Timer เมื่อ Timer เกิดการ Overflow หรือ ถูกสั่ง Restart

OS_TimerStop

5.5 OS_TimerStop

```
int OS_TimerStop(timer_t *timer)
```

Description

หยุดการทำงานของ Timer

Parameters	Details
timer	Pointer ที่ชี้ไปยัง Timer ที่ถูกสร้างไว้ก่อนหน้า

Return

1 หากทำการหยุดการทำงานของ Timer ได้สมบูรณ์

0 หากไม่สามารถการหยุดการทำงานของ Timer ได้ เช่นพยามหยุดการทำงานของ Timer ที่ได้ถูกลบออกไปจากระบบแล้ว

Remark

การสั่งให้ Timer หยุดทำงานเป็นเพียงหยุดหยุดการทำงานของ Timer เท่านั้น Timer ตัวนี้ยังคงอยู่ในระบบ พร้อมรอรับคำสั่งให้เริ่มทำงานใหม่อีกครั้งโดยฟังก์ชั่น OS_TimerStart

OS_TimerStart

5.6 OS_TimerStart

```c
int OS_TimerStop(timer_t *timer)
```

Description	
เริ่มการทำงานของ Timer	
Parameters	**Details**
timer	Pointer ที่ชี้ไปยัง Timer ที่ถูกสร้างไว้ก่อนหน้า
Return	
1 หากทำการเริ่มการทำงานของ Timer ได้สมบูรณ์ 0 หากไม่สามารถเริ่มการทำงานของ Timer ได้ เช่นพยามเริ่มการทำงานของ Timer ที่ได้ถูกลบออกไปจากระบบแล้ว	
Remark	
การสั่งให้ Timer เริ่มทำงาน ค่าเวลาของ Timer จะถูกรีเซต (เริ่มจับเวลาใหม่)	

PSW_Get, PSWx_Get

6 PSW

6.1 PSW_Get

C uint8_t PSW_Get(uint8_t id)	
Description	
อ่านค่าสถานะของ PSW (Push Button Switch) ที่ถูกกำหนดโดย id	
Parameters	Details
id	Id ของ PSW สามารถมีค่า 0, 1, 2, 3
Return	
1 หาก PSW ถูกกด (ON) 0 หาก PSW ไม่ถูกกด (OFF)	

6.2 PSW0_Get

C uint8_t PSW0_Get(void)	
Description	
อ่านค่าสถานะของ PSW หมายเลข 0	
Parameters	Details
void	
Return	
1 หาก PSW0 ถูกกด (ON) 0 หาก PSW0 ไม่ถูกกด (OFF)	

PSWx_Get

6.3 PSW1_Get

```
uint8_t PSW1_Get(void)
```

Description	
อ่านค่าสถานะของ PSW หมายเลข 1	
Parameters	**Details**
void	
Return	
1 หาก PSW1 ถูกกด (ON) 0 หาก PSW1 ไม่ถูกกด (OFF)	

6.4 PSW2_Get

```
uint8_t PSW2_Get(void)
```

Description	
อ่านค่าสถานะของ PSW หมายเลข 2	
Parameters	**Details**
void	
Return	
1 หาก PSW2 ถูกกด (ON) 0 หาก PSW2 ไม่ถูกกด (OFF)	

PSW_Scan

6.6 PSW_Scan

```
uint8_t PSW_Scan(void)
```

Description	
ตรวจสอบว่า PSW ถูกกดหรือไม่	
Parameters	**Details**
void	
Return	
หมายเลขน้อยสุดของ PSW ตัวที่ถูกกด เช่น หาก PSW1 และ PSW3 ถูกกดในเวลาเดียวกัน จะได้ค่า 1 เป็นต้น และถ้าหากไม่มี PSW ถูกกดเลย จะได้ค่า 0x0F (PSW_ALL_OFF)	

PSW_Read, OS_SwitchSetCallback

6.7 PSW_Read

```
C  uint8_t PSW_Read(void)
```

Description
อ่านค่าข้อมูลของ PSW ทั้ง 4 ตัว

Parameters	Details
void	

Return
ข้อมูลที่แสดงถูกรูปแบบของการกด PSW ทั้ง 4 ตัว เช่น ถ้า PSW0 และ PSW3 ถูกกดในเวลาเดียวกัน ค่าที่ได้จะเป็น 0x06 (0110)

Remark
PSW ทั้ง 4 มีรูปแบบของวงจรเป็น Active-Low คือเมื่อถูกกดค่าลอจิกจะเป็น 0 เมื่อไม่ถูกกด ค่าลิจิกจะเป็น 1

6.8 OS_SwitchSetCallback

```
C  int16_t OS_SwitchSetCallback(uint8_t id, switch_callback_t callback)
```

Description
กำหนด callback function ให้กับ PSW หมายเลขที่กำหนดด้วย id

Parameters	Details
id	หมายเลขของ PSW มีค่าเป็น 0, 1, 2, 3 หรือ PSW_ID_0, PSW_ID_1, PSW_ID_2, PSW_ID_3
callback	Callback function ของ PSW

Return
1 หากการกำหนด callback function ให้กับ PSW เสร็จสมบูรณ์ 0 หากการกำหนด callback function ให้กับ PSW ล้มเหลว

Remark
callback function จะถูกประมวลผลทันทีเมื่อ PSW ถูกกด

Beep, Beep_PowerSet

7 Beep

7.1 Beep

C `void Beep(uint16_t period)`

Description	
ส่งสัญญาณความถี่เสียงออกไปยัง Buzzer	
Parameters	**Details**
period	ระยะเวลาหนวยเป็น mS ที่กำหนดให้มีสัญญาณเสียง Beep
Return	
void	
Remark	
ขนาดความดังและความถี่ของสัญญาณเสียงจะเป็นไปตามค่าที่ได้กำหนดไว้ก่อนหน้าด้วย Beep_PowerSet และ Beep_FreqSet	

7.2 Beep_PowerSet

C `void Beep_PowerSet(float power)`

Description	
กำหนดขนาดความดังของสัญญาณเสียง Beep	
Parameters	**Details**
power	ขนาดความดังของสัญญาณเสียง Beep มีค่า **0.0** (0%) ถึง **1.0** (100%)
Return	
void	
Remark	
เพื่อหลีกเลี่ยงความผิดพลาดเชิงการคำนวณ และความผิดเพี้นของเสียง ควรกำหนดค่าให้มีค่ามากกว่า 0.0 และ น้อยกว่า 1.0	

Beep_FreqSet

7.3 Beep_FreqSet

C	`void Beep_FreqSet(float freq)`
Description	
กำหนดขนาดความถี่ของสัญญาณเสียง Beep	
Parameters	**Details**
freq	ขนาดความถี่ของสัญญาณเสียง Beep
Return	
void	
Remark	
ควรกำหนดค่าความถี่ให้อยู่ในช่วงที่หูได้ยิน	

ADC_Get, ADCx_Get

8 ADC

8.1 ADC_Get

```
C  uint16_t ADC_Get(uint8_t id)
```

Description	
อ่านค่าข้อมูลของ ADC (Analog-to-Digital Converter) ที่ถูกกำหนดโดย id	
Parameters	**Details**
id	Id ของ ADC สามารถมีค่า 0, 1, 2, 3
Return	
ค่าของ ADC (ที่ระบุโดย id) ขนาด 16-bit มีค่า 0 ถึง 1023 (0.0V ถึง 3.3V)	

8.2 ADC0_Get

```
C  uint16_t ADC0_Get(void)
```

Description	
อ่านค่าข้อมูลของ ADC0	
Parameters	**Details**
void	
Return	
ค่าของ ADC0 ขนาด 16-bit มีค่า 0 ถึง 1023 (0.0V ถึง 3.3V)	

ADCx_Get

8.3 ADC1_Get

C uint16_t ADC1_Get(void)	
Description	
อ่านค่าข้อมูลของ ADC1	
Parameters	Details
void	
Return	
ค่าของ ADC1 ขนาด 16-bit มีค่า 0 ถึง 1023 (0.0V ถึง 3.3V)	

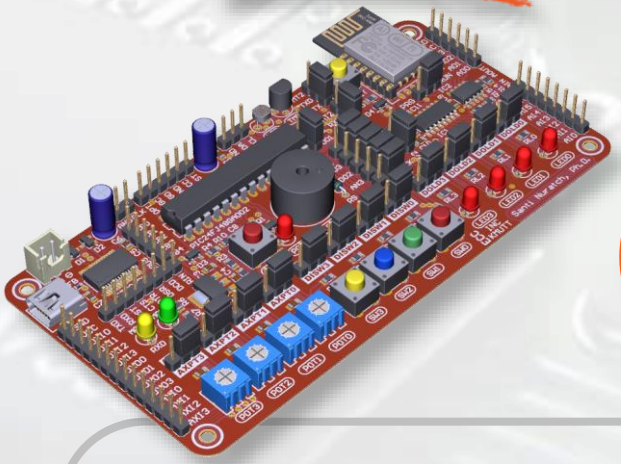
8.4 ADC2_Get

C uint16_t ADC2_Get(void)	
Description	
อ่านค่าข้อมูลของ ADC2	
Parameters	Details
void	
Return	
ค่าของ ADC2 ขนาด 16-bit มีค่า 0 ถึง 1023 (0.0V ถึง 3.3V)	

8.5 ADC3_Get

C uint16_t ADC3_Get(void)	
Description	
อ่านค่าข้อมูลของ ADC3	
Parameters	Details
void	
Return	
ค่าของ ADC3 ขนาด 16-bit มีค่า 0 ถึง 1023 (0.0V ถึง 3.3V)	

THANK YOU!



Santi Nuratch., Ph.D.

Embedded Computing and Control Lab. @ INC-KMUTT

santi.inc.kmutt@gmail.com, santi.nur@kmutt.ac.th

Department of Control System and Instrumentation Engineering,
King Mongkut's University of Technology Thonburi, **KMUTT**