


Embedded Systems Programming on STM32 MCU

การโปรแกรมระบบสมองกลฝังตัวบน


ไมโครคอนโทรลเลอร์ STM32

ครั้งที่ 3 : ระบบปฏิบัติการทันเวลา


 โพรเซสและเธรด และการทำงานแบบหลายภารกิจ

 โครงสร้างของโพรเซส

 การทำงานแบบหลายภารกิจ และระบบปฏิบัติการหลายภารกิจ

 เธรดในโพรเซส และ ความแตกต่างระหว่างโพรเซสกับเธรด

 ระบบปฏิบัติการทันเวลา

 คุณสมบัติของระบบปฏิบัติการทันเวลา

 การใช้งานระบบปฏิบัติการทันเวลาบน STM32

 การใช้งาน FreeRTOS ผ่าน CMSIS บน STM32 ในเบื้องต้น



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS

RTOS Basics

โครงสร้างของโปรเซส (Process)

- 🌿 โปรแกรม (Program) คือชุดคำสั่งที่ถูกบันทึกไว้ในหน่วยความจำสำรองที่พื่อนนำมาใช้งาน
 - 💡 ระบบปฏิบัติการต้องอ่านโปรแกรมเข้ามาในหน่วยความจำหลัก และจัดโครงสร้างหน่วยความจำหลักและทรัพยากรอื่นๆ เสียก่อนที่จะเริ่มทำงาน
- 🌿 โปรเซส (Process) คือองค์ประกอบของโครงสร้างหน่วยความจำและข้อมูลประกอบต่างๆ ที่ใช้เพื่อการดำเนินการในหน่วยหนึ่งๆ
 - 💡 โครงสร้างหน่วยความจำภายในประกอบไปด้วย ส่วนชุดคำสั่ง (text section) ส่วนตัวแปรส่วนกลาง (data section) ส่วนพื้นที่สแต็ก (stack)(ตัวแปรเฉพาะที่ เลขที่อยู่ตำแหน่งกระโดดกลับจากฟังก์ชันและอาร์กิวเมนต์) และส่วนฮีป (heap) (พื้นที่สำหรับอ้างอิงหน่วยความจำที่ขอเพิ่มเติมขณะโปรเซสทำงาน)
 - 💡 โครงสร้างข้อมูลสำหรับบันทึกสถานะและข้อมูลต่างๆ ของโปรเซส (Process Control Block-PCB) และรายการทรัพยากรที่เป็นเจ้าของ
- 🌿 ตัวอย่างเช่น ซอฟต์แวร์แก้ไขเอกสาร ตัวโปรแกรมและตัวเอกสารจะถูกอ่านขึ้นมาบนหน่วยความจำเกิดเป็นหนึ่งโปรเซส การแก้ไขเอกสารหลายตัวจะดำเนินการในลักษณะการแยกเป็นหน่วย (instance) ของโปรเซสอิสระจากกัน

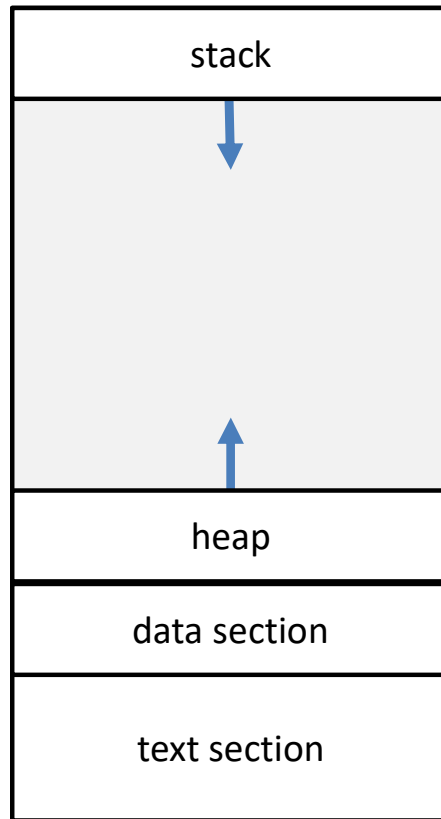


Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

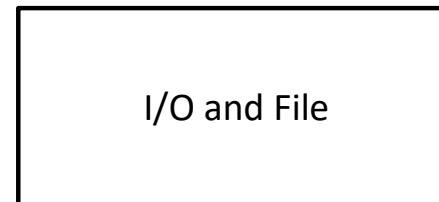
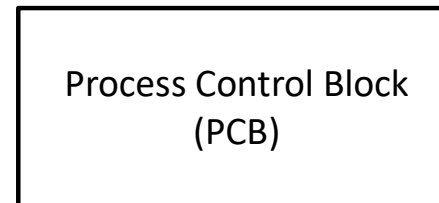
TODAY TOPIC IS

RTOS Basics

โปรเซส (Process)



Logical Address space



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS

RTOS Basics

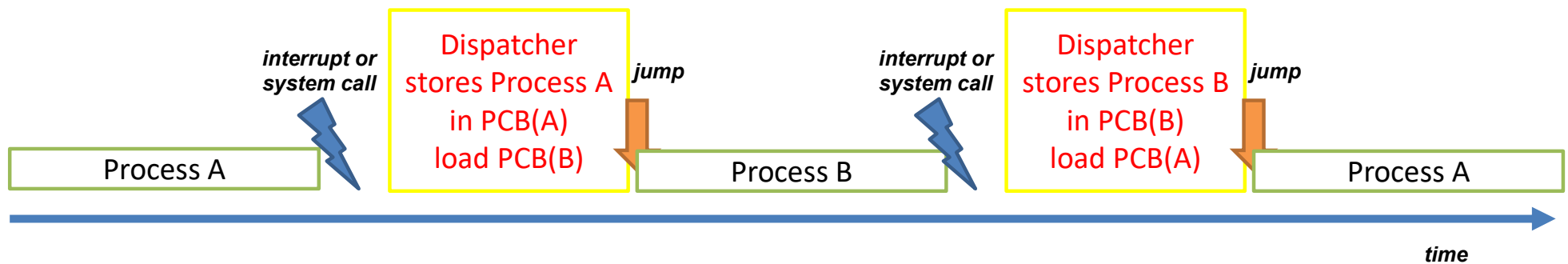
การทำงานแบบหลายภารกิจ (Multitasking)

🌱 ระบบปฏิบัติการทำหน้าที่สลับสับเปลี่ยนโปรเซสต่างๆ เข้ามารันบนซีพียู

💡 อาศัย dispatcher ซึ่งเป็นโปรเซสของระบบปฏิบัติการ ทำงานทุกๆ คาบเวลาที่กำหนด หรือถูกเรียกทางอ้อมผ่าน system call (บริการของระบบปฏิบัติการ) จากโปรเซสที่กำลังทำงาน

🎨 ตัวอย่างเช่น การเรียกใช้ฟังก์ชัน `delay()` การใช้งาน `printf()` หรือ `scanf()` ในโปรแกรมภาษาซี เป็นต้น

🎨 ระบบปฏิบัติการจะบันทึกสถานะต่างๆ ของโปรเซสปัจจุบันที่กำลังรัน (ค่าเรจิสเตอร์ทั้งหมดใน CPU และอื่นๆ) เก็บลง PCB แล้วไปโหลดสถานะต่างๆ ของโปรเซสถัดไปจาก PCB (ประจำโปรเซสถัดไป) แล้วกระโดดไปทำงานจากตำแหน่งที่ค้างอยู่



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS
RTOS Basics

การพัฒนาซอฟต์แวร์แบบหลายภารกิจ vs. การขัดจังหวะ

🌱 สำหรับอุปกรณ์หรือองค์ประกอบทางฮาร์ดแวร์ใน MCU ที่รองรับการขัดจังหวะ ผู้พัฒนาอาจเขียนโปรแกรมในลักษณะของการขับเคลื่อนด้วยการขัดจังหวะ (Interrupt driven) เพื่อทำงานตอบสนองการกระตุ้นจากฮาร์ดแวร์

- 💡 สามารถออกแบบโครงสร้างโปรแกรมออกเป็นองค์ประกอบแยกตามการตอบสนองต่อเหตุการณ์/การขัดจังหวะจากฮาร์ดแวร์ ทำให้การควบคุมอุปกรณ์แบ่งเป็นส่วนส่วนจากกันอย่างชัดเจน
- 💡 เพิ่มเติมองค์ประกอบของโปรแกรมตามฮาร์ดแวร์ที่เพิ่มเติม/เปลี่ยนแปลง ได้โดยง่าย
- 💡 เลือกที่จะทำหรือไม่ทำเหตุการณ์ต่าง ได้ตามต้องการ (enable/disable interrupt)

🌱 ปัญหาการเขียนโปรแกรมอาศัยการขัดจังหวะ

- 💡 ไม่เหมาะสมกับระบบปฏิบัติการแบบหลายผู้ใช้ (multiuser) หรือระบบปฏิบัติการสมัยใหม่โดยทั่วไป เพราะโปรเซสผู้ใช้งานอาจก่อปัญหาต่อการจัดสรรทรัพยากรระบบโดยระบบปฏิบัติการ
- 💡 งานบางประเภทต้องการการทำงานพร้อมกัน (concurrent) โดยธรรมชาติ และไม่มีเหตุการณ์ (event/interrupt) รองรับกลไกการกระตุ้นให้เริ่มทำงาน ไม่อาจจะใช้การขัดจังหวะเพื่อเข้ามาสลับงานได้อย่างสะดวก



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS

RTOS Basics

ระบบปฏิบัติการแบบหลายภารกิจ (Multitasking OS)

- 🌿 ระบบปฏิบัติการสมัยใหม่ถูกออกแบบมาเพื่อให้รองรับการทำงานของหลายๆ โพรเซสพร้อมกัน
- 🌿 การจัดการ I/O และ File ถูกโอนหน้าที่ให้ระบบปฏิบัติการทำงานให้แทน
 - 💡 โพรเซสผู้ใช้ เรียกผ่าน system call ของระบบปฏิบัติการ
 - 💡 การจัดการอุปกรณ์ต่างๆ ระบบปฏิบัติการจัดการกับการขัดจังหวะแทนโพรเซสผู้ใช้ โดยกำหนดฟังก์ชันเรียกกลับ (Callback function) ให้ผู้ใช้เขียนจัดการกับเหตุการณ์แทนการเข้าถึงกลไกทางฮาร์ดแวร์โดยตรง
- 🌿 แต่ละโพรเซสมีทรัพยากรที่ถูกจัดสรรมาใช้งานแยกกันเป็นอิสระ
- 🌿 ระบบปฏิบัติการสมัยใหม่ ยังรองรับการทำงานแบบหลายเธรด (Multithread) ได้อีกด้วย
 - 💡 รองรับโพรเซสที่สามารถแตกเธรดได้หลายเธรด





Asst.Prof. Thanwa SRIPRAMONG
PRESENTER


TODAY TOPIC IS


RTOS Basics


เธรด (thread)


 เธรด เป็นองค์ประกอบการทำงานย่อยภายในโปรเซส

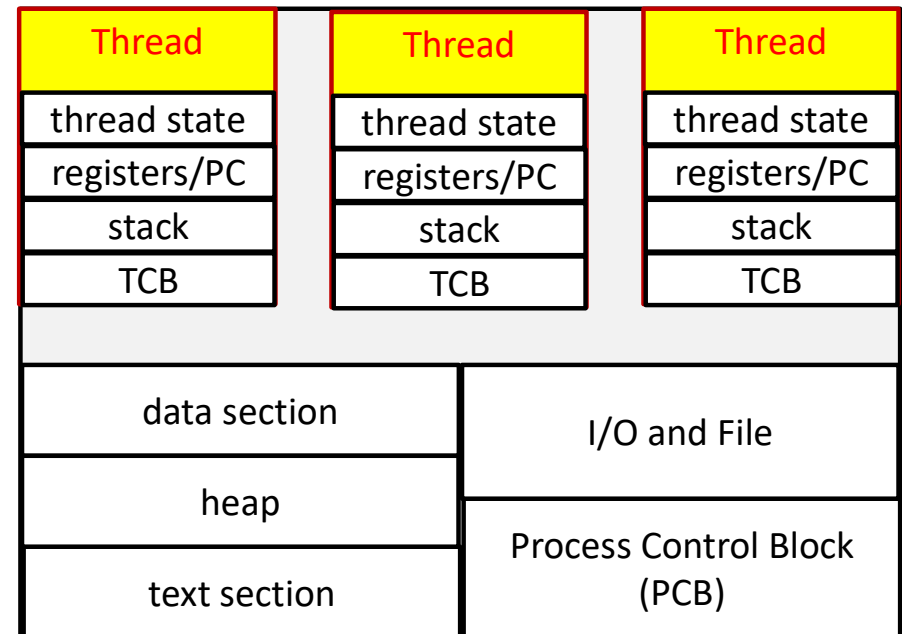
 แต่ละเธรดทำงานถึงอิสระจากกัน (ทำงานไปพร้อมกัน/ถูกสลับเวลาใช้ CPU แบบ multitasking เช่นเดียวกับการทำงานแบบหลายโปรเซส)

 ใช้คำสั่ง (ฟังก์ชันร่วมกัน) ใช้ตัวแปรส่วนกลาง I/O และไฟล์ ร่วมกันระหว่างทุกเธรด

 แยกตัวแปรเฉพาะที่และสแต็กออกจากกัน

 การเขียนโปรแกรมจะอยู่ในรูปฟังก์ชันของเธรด ที่สามารถเรียกทำงานพร้อมๆ กันได้

 จินตนาการเหมือนมีคนหลายๆ คน ทำงานพร้อมๆ กันแต่ใช้สิ่งของต่างๆ ร่วมกัน





Asst.Prof. Thanwa SRIPRAMONG
PRESENTER


TODAY TOPIC IS


RTOS Basics


การพัฒนาโปรแกรมแบบหลายเธรด

 โปรแกรมที่ถูกพัฒนาขึ้นแบบดั้งเดิม สามารถปรับมาเป็นการเขียนแบบหลายเธรด (Thread parallelism) ได้ในสองรูปแบบ

 Data parallelism การแบ่งข้อมูล(ขนาดใหญ่) ออกเป็นหลายๆ ส่วนเพื่อให้แต่ละเธรด (ที่มีฟังก์ชันทำงานเหมือนกัน) ทำงานกับข้อมูลแต่ละส่วน

 ตัวอย่างเช่น มีข้อมูลภาพขนาดใหญ่ จึงแบ่งภาพขนาดใหญ่ออกเป็นส่วนๆ เพื่อให้ฟังก์ชันจัดการภาพ จัดการกับแต่ละส่วนของภาพไปพร้อมๆ กันหลายเธรด (ในกรณีที่ CPU มีหลายคอร์ จะช่วยเพิ่มความเร็วในการจัดการได้อย่างมาก)

 Task parallelism การแบ่งการทำงานที่หลากหลายออกเป็นส่วนๆ เพื่อให้แต่ละเธรด (ที่มีฟังก์ชันการทำงานแตกต่างกัน) ทำงานกับงานแต่ละส่วน

 ตัวอย่างเช่น มีการจัดการกับอุปกรณ์ I/O หลายตัวพร้อมกัน จึงแบ่งฟังก์ชันการทำงานออกเป็นหลายฟังก์ชัน และให้แต่ละเธรดทำงานกับฟังก์ชันแต่ละตัวแยกกันไป ส่งผลทำให้เขียนโปรแกรมได้ง่ายขึ้นกว่าการใช้ฟังก์ชันเดียวจัดการกับ I/O ทุกตัว



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS

RTOS Basics

ประโยชน์ของการทำงานแบบหลายเธรด

- 🌱 ประหยัดทรัพยากรมากกว่าการเขียนโปรแกรมแบบหลายโปรเซส
 - 💡 สามารถกำหนดตัวแปรส่วนกลาง (global variables) เพื่อส่งผ่านข้อมูลระหว่างเธรดได้ง่าย
 - 💡 สามารถแชร์ฟังก์ชันใช้งานร่วมกันระหว่างเธรดได้ (กรณีสร้างเธรดหลายเธรดแต่ทำงานลักษณะเดียวกัน)
- 🌱 ช่วยให้งานคอมพิวเตอร์ที่มีหลายคอร์ได้อย่างคุ้มค่า
 - 💡 CPU และ MCU ปัจจุบันมีการพัฒนาให้มีชิพียูหลายหน่วย(คอร์-core) การแตกเธรดจะเกิดการกระจายการทำงานของเธรดไปยังคอร์ต่างๆ ได้ ทำให้ใช้งาน CPU ได้คุ้มค่า
- 🌱 แบ่งการทำงานออกเป็นส่วนๆ (modularity) สร้างความสะดวกในการจัดการกับองค์ประกอบของระบบที่มีหลากหลาย
 - 💡 ตัวอย่างเช่น การจัดการกับเซ็นเซอร์และแอ็กชูเอเตอร์ (actuator) ต่างๆ โดยสร้างเธรดสำหรับอุปกรณ์แต่ละตัวแยกจากกัน




Asst.Prof. Thanwa SRIPRAMONG
PRESENTER


TODAY TOPIC IS


RTOS Basics


ข้อดีของการทำงานแบบหลายเธรด

 การพัฒนาโปรแกรมยุ่งยากขึ้นกว่าเดิม


 เนื่องจากการขาดประสิทธิภาพ ไปจนถึงการต้องเลือกว่างานส่วนไหนจะแยกการทำงานออกจากกัน หรือจะแบ่งข้อมูลส่วนใดออกไปทำงานในแต่ละเธรด

 การสร้างเธรดหลายเธรดเพื่อช่วยประมวล อาจไม่ได้ทำให้ความเร็วเพิ่มขึ้น (แม้บน CPU ที่มีหลายคอร์)

 เนื่องจากการประมวลข้อมูลขนาดใหญ่อาจจะต้องมีการใช้ผลการคำนวณจากองค์ประกอบข้อมูลในส่วนอื่นเข้ามาเกี่ยวข้องกับข้อมูลอีกส่วน การคำนวณในแต่ละเธรดจึงต้องรอผลลัพธ์จากเธรดอื่นให้ทำงานเสร็จในขั้นตอนหนึ่งๆ ก่อน เธรดปัจจุบันจึงจะทำงานต่อได้

 ดูเรื่องกฎของแอมดาห์ล (Amdahl's law)

 อาจสร้างปัญหาเรื่องสถานะแข่งขัน (race condition)

 รายละเอียดเพิ่มเติมในเนื้อหาครั้งที่ 6








Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS

RTOS Basics

ปัญหาของระบบปฏิบัติการหลายภารกิจโดยทั่วไป

-  การจัดสรรเวลาให้กับโปรเซสแต่ละตัว อาจจัดสรรเวลาแบบไม่กำหนดคาบเวลาที่ชัดเจน หรืออาจจัดสรรเวลาแบบชัดเจน (กรณีหลังมักใช้ round-robin)
-  หากมีโปรเซส/เธรดจำนวนมากทำงานพร้อมกัน อาจส่งผลทำให้บางโปรเซส/เธรดไม่มีโอกาสได้ทำงาน หรือได้สัดส่วนเวลาทำงานน้อยกว่าที่ควรจะเป็น หรือไม่สามรถระบุเป็นสัดส่วนเวลาที่แน่นอน
-  อาจมีโปรเซสบางตัวที่ใช้เวลานานมาก และอาจทำงานโดยที่โปรเซสอื่นๆ คาดไม่ถึง ทำให้โปรเซสอื่นอาจไม่สามารถทำงานเสร็จได้ทันตามกำหนด
-  ตัวอย่างเช่นในระบบปฏิบัติการวินโดวส์ ในขณะที่ผู้ใช้กำลังทำงาน ระบบปฏิบัติการอาจจะตรวจสอบไฟล์เพื่อหาไวรัสอยู่เบื้องหลัง ซึ่งการอ่านฮาร์ดดิสก์ที่มีขนาดไฟล์ใหญ่มากๆ ส่งผลทำให้ผู้ใช้งานอาจรู้สึกช้าซอฟต์แวร์ที่กำลังใช้งานกระตุก (หยุดทำงานเป็นระยะๆ)
-  การแตกเธรดเพื่อจัดการ I/O หลายๆ ตัวพร้อมๆ กัน ผู้พัฒนาอาจเขียนโค้ดที่ใช้เวลาทำงานนานเกินไป และจัดสรรลำดับความสำคัญไม่เหมาะสม อาจส่งผลทำให้เธรดที่ต้องจัดการ I/O บางตัวอาจทำงานไม่ทัน



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS

RTOS Basics

ระบบปฏิบัติการทันเวลา (Real-time Operating System)

🌿 ระบบปฏิบัติการที่ถูกออกแบบมาให้แต่ละโปรเซส/เธรด ที่ทำงานอยู่บนระบบ สามารถได้รับโอกาสทำงานได้อย่างเหมาะสม

💡 พิจารณาจากความสามารถในการประเมินช่วงเวลาระหว่างการรับข้อมูลเข้า นำไปคำนวณ จนส่งข้อมูลออก ต้องทำให้ได้ภายในคาบเวลาที่ (ผู้พัฒนา) กำหนด (response time ในค่าที่กำหนด)

🌿 ลักษณะการจัดสรรเวลาให้กับโปรเซส/เธรดแต่ละตัว อาจใช้กลไก

💡 Round-robin (กำหนด timer interrupt มีคาบเวลาตามที่กำหนด และเปลี่ยนงานเมื่อครบคาบเวลา)

💡 Event driven (กำหนด interrupt ทางฮาร์ดแวร์ หรืออื่นใด สำหรับโปรเซส/เธรด ที่มีความสำคัญ ที่จำเป็นต้องทำงานให้เสร็จภายในเวลาที่กำหนด สามารถเข้ามาแย่งเวลาทำงานโปรเซสอื่นที่กำลังทำอยู่ได้)

🌿 Hard real-time vs Soft real-time

💡 Hard real-time แต่ละงานมีการจัดแบ่งเวลาสลับทำงานภายในระดับมิลลิวินาทีหรือน้อยกว่านั้น

💡 Soft real-time การจัดสรรเวลางานอาจจะมีโอกาสคลาดเคลื่อนได้ในระดับร้อยมิลลิวินาที สำหรับระบบงานที่ไม่ต้องการความเที่ยงมากนัก หรือ response time มีค่าสูงในระดับวินาทีหรือมากกว่า



Asst.Prof. Thanwa SRIPRAMONG

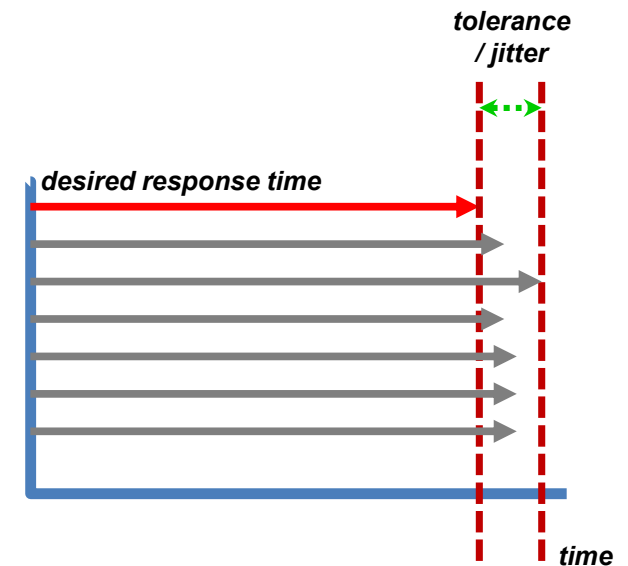
PRESENTER

TODAY TOPIC IS

RTOS Basics

คุณสมบัติที่สำคัญของระบบปฏิบัติการทันเวลา

- 🌱 การได้ผลการทำงานที่ถูกต้องแม่นยำ (Determinism)
 - 💡 ผลการทำงานของระบบจะต้องให้คำตอบที่มีความถูกต้องแม่นยำ (input -> output เช่นเดิมเสมอ)
 - 💡 ความผิดพลาดในระดับที่กำหนด
 - 💡 ได้ output ภายในเวลาที่กำหนด (latency ภายในเวลาที่กำหนด)
 - 🌍 มีการกำหนดค่า jitter ไว้ชัดเจนสำหรับการทำงานหนึ่งๆ
- 🌱 ประสิทธิภาพการทำงานที่สูง (High performance)
 - 💡 ต้องทำงานได้อย่างรวดเร็ว และตอบสนองทันเวลา
 - 🌍 เมื่อเทียบกับระบบปฏิบัติการโดยทั่วไป









Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS

RTOS Basics

คุณสมบัติที่สำคัญของระบบปฏิบัติการทันเวลา(ต่อ)

-  ความทนทานต่อความผิดพลาดของระบบและความปลอดภัยสูง
 -  ระบบมักถูกนำไปใช้ในงานวิกฤติ (critical systems) ที่ต้องมีการป้องกันการถูกโจมตีหรือเจาะระบบ และต้องทำงานต่อไปหากองค์ประกอบบางส่วนเกิดความเสียหาย
-  ความสามารถในการควบคุมงานที่มีลำดับความสำคัญให้ทำงานภายในเวลาที่กำหนด
 -  ในบางสภาพของการทำงานของระบบ อาจส่งผลทำให้การวนกลับมาทำงานของบางงานไม่ทันตามเวลา (เช่น จากการรอคอย I/O ที่นานผิดปกติของบางงาน) ในกรณีเช่นนี้จะต้องมีกลไกลดการทำงานของโปรเซส/เธรดที่สำคัญ (high priority) ให้มาทำงานก่อน (พักงานอื่นไว้ในภายหลัง) เพื่อรับประกันเวลา response time ของโปรเซส/เธรด ที่สำคัญดังกล่าวให้ได้ภายในคาบเวลาที่กำหนด
-  ใช้พื้นที่หน่วยความจำน้อยและทรัพยากรระบบเท่าที่จำเป็น
 -  ระบบปฏิบัติการทันเวลามักถูกนำไปใช้ในระบบสมองกลฝังตัวที่มีพื้นที่หน่วยความจำ/ทรัพยากรที่จำกัด ดังนั้นระบบปฏิบัติการทันเวลาที่ถูกออกแบบมาในลักษณะนี้จะต้องมีขนาดเล็ก โดยยังให้ขีดความสามารถเท่าที่จำเป็นต่อการทำงาน



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS

RTOS Basics

FreeRTOS

- 🌿 เป็นระบบปฏิบัติการทันท่วงทีหนึ่งที่ยิมนำมาใช้ในงานสมองกลฝังตัว
 - 💧 มีขนาดเล็ก และฝังตัวรวมอยู่กับโปรแกรมผู้พัฒนาในรูปแบบของไลบรารี
 - 💧 Open source ไม่ต้องเสียค่าใช้จ่าย และเปิดเผยโค้ด
- 🌿 STM32CubeIDE รองรับ FreeRTOS ในฐานะเป็น Middleware
 - 💧 มีไลบรารีติดตั้งมาพร้อมใช้งานได้ทันที
 - 💧 สามารถเรียกใช้ฟังก์ชัน (API) ตามรูปแบบของ FreeRTOS หรือเรียกใช้ API ตามรูปแบบ CMSIS
 - 🌈 Common Microcontroller Software Interface Standard (CMSIS) เป็นมาตรฐานการเรียกใช้ API ของ CPU ในตระกูล Arm Cortex (STM32 ใช้ CPU ที่เป็นหัวใจในการทำงานในตระกูล Arm Cortex M)



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS

RTOS Basics



สรุปหัวข้อ

- 🌿 หลักการทำงานพื้นฐานของระบบหลายงาน (multitasking) คือการสลับเอาชุดคำสั่งของแต่ละงานขึ้นมาทำเป็นเวลาด้านๆ ก่อนเปลี่ยนสลับให้งานอื่นขึ้นมาทำบ้าง
- 🌿 การเขียนโปรแกรมแบบหลายเธรด เป็นการกำหนดให้ฟังก์ชันย่อยที่กำหนดขึ้นในโปรเซส ทำงานไปพร้อมๆ กัน (ตามหลักการของ multitasking)
- 🌿 ระบบปฏิบัติการทันเวลา เป็นระบบปฏิบัติการที่มีการรับประกันว่าแต่ละโปรเซส/เธรด จะต้องทำงานได้เสร็จในเวลาที่กำหนดไว้ (ตามที่ออกแบบ/ขีดความสามารถของฮาร์ดแวร์) โดยไม่มีสภาพที่ไม่คาดฝันที่ทำให้บางงานเสร็จไม่ทันตามเวลา
- 🌿 FreeRTOS เป็น(ไลบรารี)ระบบปฏิบัติการทันเวลาที่นิยมใช้ใน STM32 เพื่อใช้ในการสร้างเธรดหลายตัวให้ทำงานพร้อมกัน



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS

RTOS Basics