Inter-Task Communication and Synchronisation

1. Inter-Task Communication and Synchronisation

1.1 Three Paradigms

- **Task-owned facilities:** attributes the RTOS gives a task for input; the classic example is a signal.
- **Kernel objects:** stand-alone RTOS facilities – event flags, mailboxes, queues/pipes, semaphores and mutexes.
- **Message passing:** the RTOS lets you create message objects that can be sent from one task to another or to several tasks; fundamental to message-passing RTOS designs.

2. Mechanisms

2.1 Shared Variables / Shared Memory

Multiple tasks access the same memory location: one writes, another reads. Access must be synchronised with locks.

- **Advantages:** simple, fast, minimal memory overhead.
- **Limitations:** risk of data corruption, needs a locking mechanism, poor for large transfers.

2.2 Signals

Bit flags that notify a task of an event. Any task can set signal bits but only the owner task reads them; reading usually clears the signal.

- **Advantages:** very fast, low memory, suitable for event notification.
- **Applications:** task synchronisation, interrupt notification.

2.3 Mutex (Mutual Exclusion)

Protects a shared resource. Only one task can own it at a time, and the task that took it must release it, preventing simultaneous access.

- **Advantages:** prevents race conditions, keeps data intact, safer than a binary semaphore for resource protection.
- **Applications:** shared memory, shared peripherals, file access.

2.4 Event Flag Groups

A collection of event flags shared among tasks. Any task can set or clear flags, and tasks wait until a required combination occurs.

- **Advantages:** handles several events together, supports blocking waits, flexible synchronisation.
- **Applications:** coordinating multiple tasks, event-driven systems.

2.5 Semaphores

A synchronisation object controlling access to a shared resource: a task obtains it, uses the resource, then releases it.

- **Binary semaphore:** value 0/1, used for signalling.

- **Counting semaphore:** tracks multiple available instances of a resource.
- **Advantages:** prevents resource conflicts, supports synchronisation, allows blocking until the resource is free.

2.6 Mailboxes

Transfer a single message between tasks: the sender places a message, the receiver reads it, and the mailbox is empty afterwards.

- **Advantages:** simple, suits small messages, supports blocking send/receive.
- **Applications:** command passing, status messages.

2.7 Queues

A message buffer storing several messages in FIFO order: added at the rear, removed from the front.

- **Features:** FIFO, multiple messages, blocking send/receive, and message priority through "jamming" (inserting an urgent message at the front).
- **Applications:** sensor data collection, producer–consumer systems.

3. Summary of ITC Mechanisms

Mechanism	Purpose	Best used for
Shared memory	Data sharing	Fast communication
Signals	Event notification	Interrupt handling
Event flags	Multiple-event synchronisation	Task coordination
Semaphore	Resource control	Shared resources
Mailbox	Single message transfer	Commands and notifications
Queue	Multiple message transfer	Sensor/communication data
Mutex	Mutual exclusion	Protecting critical resources