Final Presentation

Real-time operating system for ARM microcontrollers

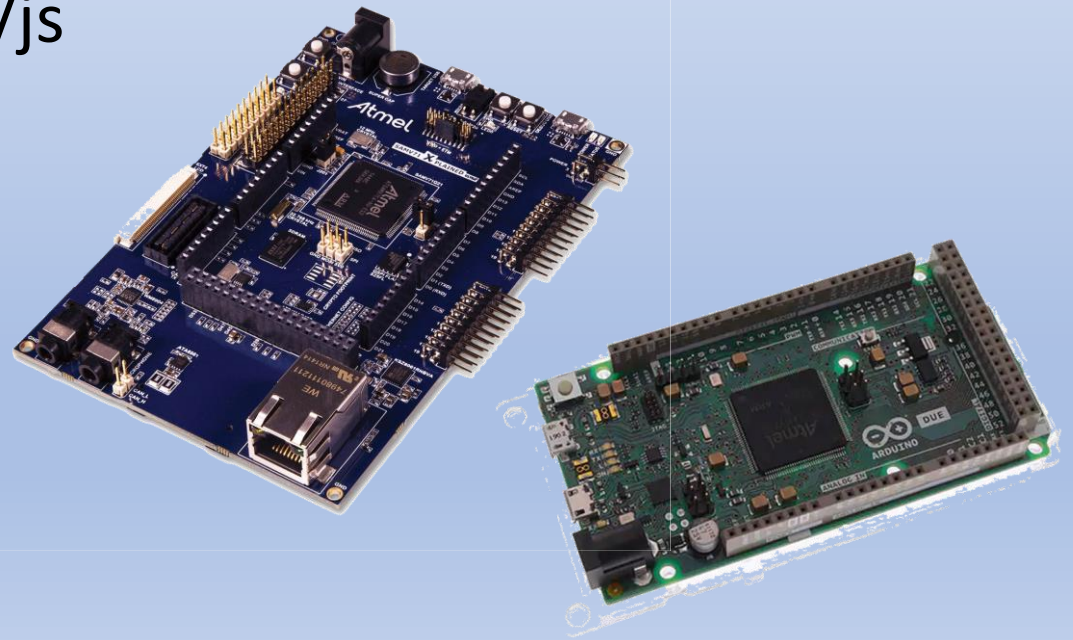
ESTEC Contract Number: **4000135473/21/NL/GLC/js**

1st December 2022 12:30

Michael Ryan, CTO, O.C.E.Technology Ltd

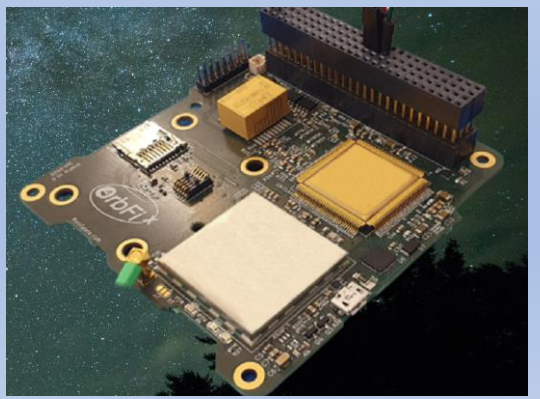
Summary

- RTOS for ARM Cortex-M with unique features (named OCEOS)
- Tested to ECSS Category B safety standard
- Tested for Microchip Cortex-M3 SAM3X8ERT & Cortex-M7 SAMRH71
- GSTP contract 4000135473/21/NL/GLC/js
- Duration: 13 months
- Budget: ESA €300k Co-funding €145k
- ESA Technical Officer: Piotr Skrzypek
- OCEOS development kit now on-sale



Embedded System Characteristics (1)

- **Fixed Code Base** *Software not added during system life*
  => can use physical addresses, RTOS doesn't need page tables etc.
- **Code must be robust** *So an RTOS design should…*
  => ensure certain failure modes are impossible
  - e.g. unbounded priority inversion, deadlocks,…
  => allow behaviour policing by the application (white box)
  - performance data recorded and checkable at any time
  => automatically check for problems
  - e.g. stack overrun, missed deadlines…
  => automatically trigger application problem handlers
  => provide calls to deal with problems
  - e.g. kill task, disable task, …



Embedded System Characteristics (2)

- **But things can go wrong** *So an RTOS design must provide*
  => **F**ault **A**nticipation, **D**etection, **I**solation, **R**eporting, **R**ecovery
  - make it easy for the application to police the system:
    - min time between task start requests
    - max execution times, deadline misses
    - max pre-emptions, max stack usage
  - automatic checks of key components
    - memory area sentinels, stack space
  - automatic logging of anomalies
  - triggers for user defined problem handler functions
  - kill tasks, disable tasks, …

OCEOS: For Embedded Systems

- **Based on ‘Stack Resource Policy’ (Baker 1991)**
  => single system stack per CPU (not stack per task)
- **Deterministic**
  => behaviour predictable
  => memory statically allocated
  => timing overheads minimized and quantifiable
- **Application task timing recorded for analysis**
  => maximum execution times, maximum times to completion……
  => missed deadlines trigger application defined action
- **Timed actions independent of scheduling**
  => output at specific time, task start request at specific time

OCEOS: RTOS (1)

- **Fixed priority**
  => task priorities fixed based on task importance
- **Pre-emption threshold**
  => pre-emption only by tasks with higher priority than threshold
- **Multiple execution instances**
  => multiple same task ‘jobs’ can be queued for execution typically using different data
- **Timed actions independent of scheduling**
  => data output at specific time
  => task start request at specific time

OCEOS: RTOS (2)

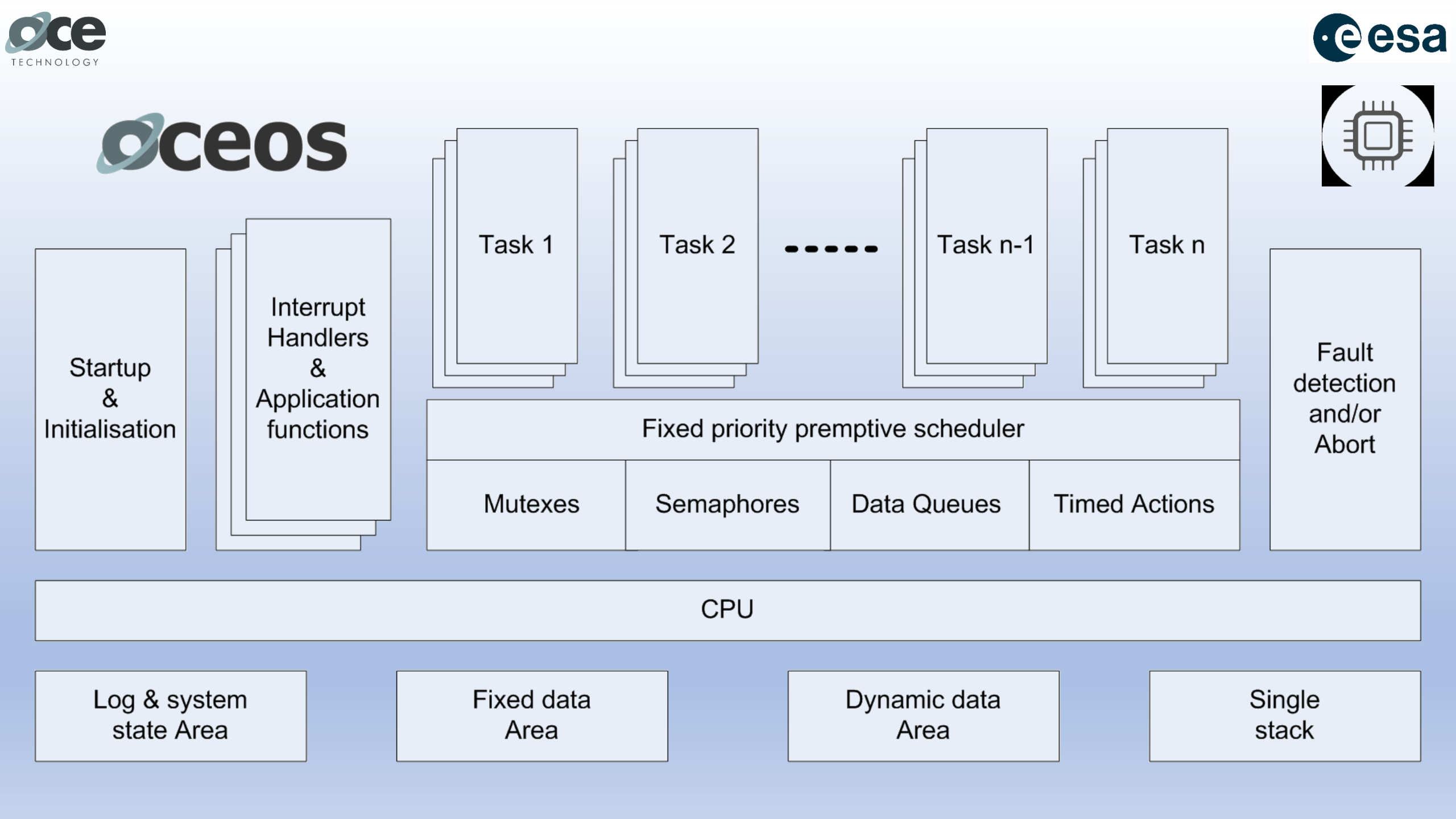
- **Mutexes**
  => unbounded priority inversion cannot occur
  => deadlocks cannot occur
- **Counting semaphores**
  => allow wait with timeout
- **Data queues**
  => allow read with timeout

OCEOS: RTOS (3)

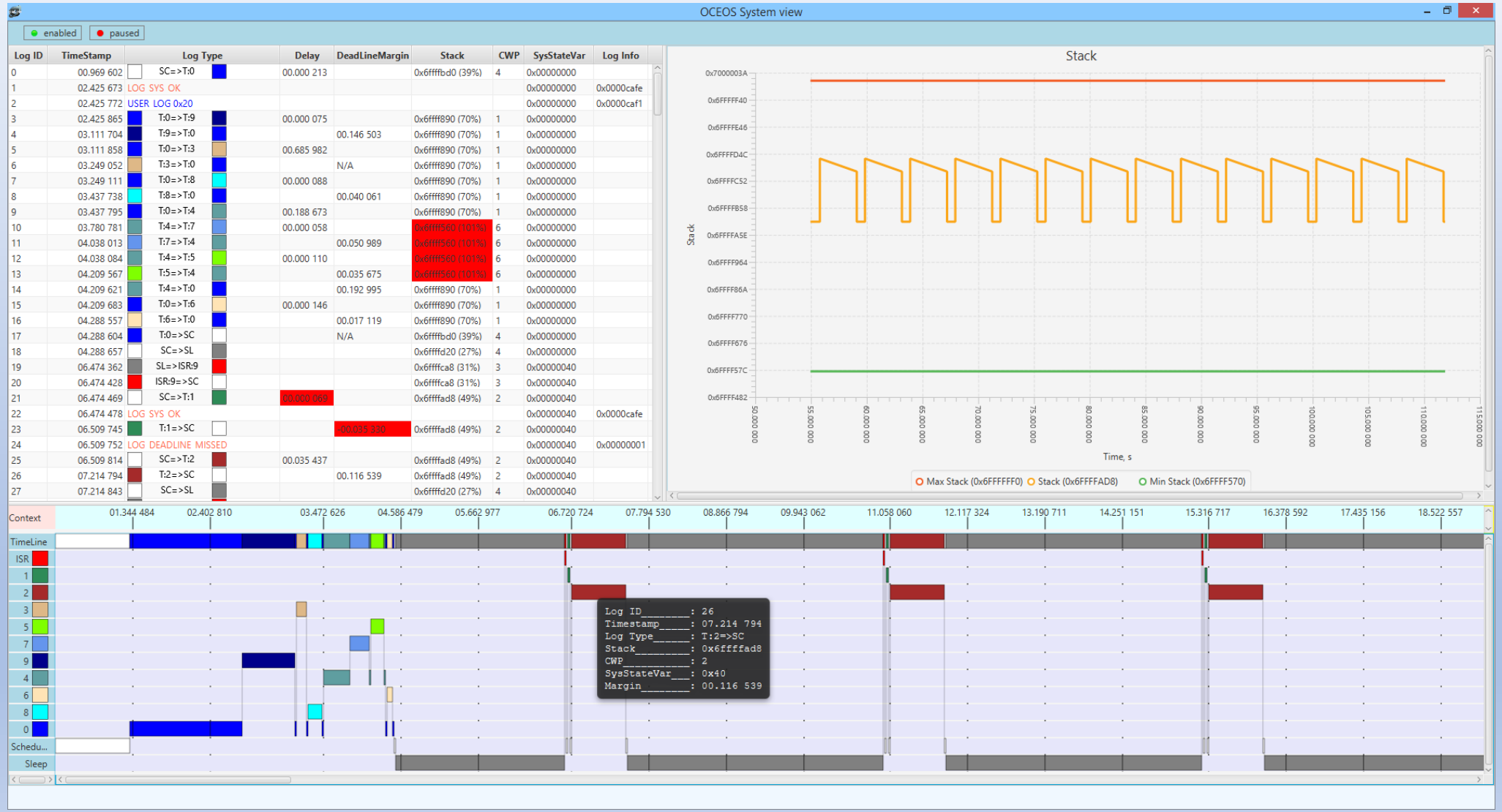
- **System time**
  => in microseconds, 64 bit
- **Context switch timing**
  => context switch time minimized
- **Interrupts**
  => interrupt disabled timing is minimized
  => high priority timer interrupt reserved for timed actions
- **Some numbers**
  255 tasks, 15*255 execution instances (jobs),
  63 mutexes, 63 semaphores, 63 data queues

OCEOS: RTOS (4) – USING IT

- **Library** - components not used not linked into the executable
- **Servant not Master** – started by application main()
- **Step 1 : Create application configuration, pass to *oceos_init()***
  what stack space, log entries
  how many tasks, jobs per task, timed actions,
  how many mutexes, semaphores, data queues
- **Step 2: Create corresponding tasks, mutexes, etc.**
  using *oceos_task_create()* etc.
- **Step 3: Use oceos_init_finish() to complete fixed data and checksum**
- **Step 4: Pass fixed data and initial task (if any) to *oceos_start()***
  dynamic data area is set up
  scheduling begins

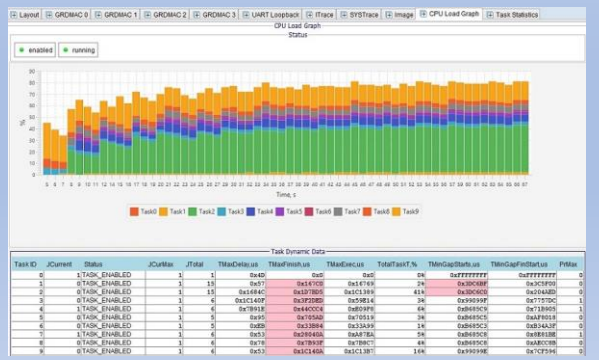

Task 1
Task 2
Task n-1
Task n
Startup & Initialisation
Interrupt Handlers & Application functions
Fault detection and/or Abort
Fixed priority premptive scheduler
Mutexes
Semaphores
Data Queues
Timed Actions
CPU
Log & system state Area
Fixed data Area
Dynamic data Area
Single stack

Debug support - DMON



Current Status

- **OCEOS (single core)**
  - SPARC and ARM versions complete
    (with additional support for GR716 microcontroller)
    ESA Flight Level B qualification ready
- **OCEOSmp (multicore)**
  - Multicore SPARC & RISC-V in initial test, ARM pending
  - Example test results using SPARC quad core Gaisler GR740:
    - 1001 task starts even distribution: Per CPU 251,250,250,250
    - 4096 sample FFT (one task, four jobs in parallel): Speedup factor 3.7
- **Availability**
  - OCEOS - single-core development kit on-sale
  - OCEOS - multicore beta evaluations available soon



Finally

- **Thanks to ESA for their support**
- **Thank you for listening**
- **Any Questions?**

michael.ryan@ocetechnology.com

www.ocetechnology.com