ERIKA Enterprise Configuration details

Daniele Alessandrelli
d.alessandrelli@sssup.it
Slides by Paolo Gai
summary

- ERIKA Enterprise features
- comparison of the various versions
- OIL definition for Microchip dsPIC® DSC
ERIKA Enterprise
supported API
- OSEK OS (BCC1, BCC2, ECC1, ECC2)
- OSEK OIL 1.4.1
- OSEK ORTI 2.1.1 for Lauterbach Trace32

support for
- basic (with stack sharing) / extended tasks
- resources
- events
- hooks
- alarms
erika enterprise

currently available for
• Microchip dsPIC
• ARM7TDMI (Samsung KS32C50100, UniBo MPARM)
• AVR
• Nios II (with multicore support)

also available for
• ARM7TDMI (Triscend A7, ST Janus, ST STA2051)
• Tricore 1
• PPC 5xx (PPC 566EVB)
• Hitachi H8 (RCX/Lego Mindstorms)
• C167/ST10 (Ertec EVA 167, tiny/large mem. model)
ERIKA is distributed under the GPL with linking exception license (also known as GNU Crosstool license)

ERIKA Enterprise is available together with the RT-Druid IDE code generator

- integrated into Eclipse
- code generation for ERIKA Enterprise
**Comparison**

**Conformance classes**
- BCC1, BCC2, ECC1, ECC2

**Startup/Shutdown**
- StartOS, application modes, StartupHook, autostartSystem
- ShutdownOS and ShutdownHook

**FP, EDF**
- FP (similar to BCC2, or ECC2 if multistack) and EDF
- No, the main is already the main thread!
- No
### comparison (2)

#### Error Handling and Hooks
- error codes, standard and extended status
- support for ErrorHook and macros
- No
- No

#### PreTaskHook / PostTaskHook
- Support for PreTaskHook and PostTaskHook / nothing
- No
- No

#### ORTI
- Yes (Nios II)
- Yes (FRSH on Nios II)
<table>
<thead>
<tr>
<th>Task</th>
<th>Informations on tasks</th>
<th>Basic / extended tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>- TerminateTask and ChainTask</td>
<td>- GetTaskID and GetTaskState</td>
<td>- Basic and Extended Tasks</td>
</tr>
<tr>
<td>- No (less RAM!)</td>
<td>- No (monostack does not have a task state!)</td>
<td>- blocking primitives to be called within tasks with a private stack</td>
</tr>
</tbody>
</table>
comparison (4)

Number of pending activations

- BCC1 and ECC1 = only one pending activation.
  BCC2 and ECC2 = more than one (in OIL file), activations of tasks with same priorities in FIFO order

- the number of pending activations as an integer value, maximum value is implementation dependent. No FIFO order.

Events

- Yes, in ECC1 and ECC2

- No
Blocking / non-blocking semaphores

- ECC1/ECC2 Blocking and non blocking semaphores
- BCC1/BCC2 non blocking semaphores

Primitives for disabling interrupts

- Yes
- Semaphore primitives only in multistack configuration.
- No
erika enterprise

- OSEK BCC1, monostack, 2 Tasks, 1 resource, dsPIC

Code footprint (24-bit instructions): 379 (1137 bytes)
- ISR2 stub (for each IRQ) 27
- IRQ end 36
- kernel global functions 99
- ActivateTask 57
- GetResource 12
- ReleaseResource 41
- StartOS 26
- Task end (TerminateTask) 81

Data footprint (bytes)
- ROM 18
- RAM 52
erika enterprise

- FP kernel, monostack, 4 tasks, 1 resource, dsPIC

Code footprint (24-bit instructions): 244 (732 bytes)
- ISR2 stub (for each IRQ) 24
- IRQ end 23
- kernel global functions 67
- ActivateTask 43
- GetResource + ReleaseResource 42
- Task end 45

Data footprint (bytes)
- ROM 26
- RAM 42
ERIKA Enterprise supports the following boards:

- **Evidence / Embedded Solutions FLEX board**
  supported devices: LEDs, various external devices using add-on boards

- **Microchip Explorer 16**
  both PIC33 and PIC24
  supported devices: LEDs, Buttons, LCD, Analog

- **Microchip dsPICDEM 1.1 Plus**
  supported devices: LEDs, Buttons, LCD, Analog, Audio (tbd)
OIL for EE

- the OIL presented in the following slides is a subset of the OSEK OIL standard
- it is a quick tutorial to the OIL definition which can be used for ERIKA Enterprise on the Microchip dsPIC® DSC
- two columns
  - the first column contains the definition
  - the second column contains examples
- it does not include EDF OIL details
OIL (OS object)

definition

OIL_VERSION = "2.4";
IMPLEMENTATION ee {
OS {
    STRING EE_OPT[];
    STRING CFLAGS[];
    STRING ASFLAGS[];
    STRING LDFLAGS[];
    STRING LDDEPS[];
    STRING LIBS[];
    BOOLEAN USERESSCHEDULER = TRUE;
    […]
}

example

CPU mySystem {
    OS myOs {
        EE_OPT = "DEBUG";
        EE_OPT = "MYDEFINE";
        CFLAGS = "-IC:/.../scicos";
        USERESSCHEDULER = FALSE;
    }
}
OIL (OS object : CPU data)

**definition**

```c
ENUM [
    [...] PIC30 {
        STRING APP_SRC[];
        BOOLEAN [ 
            TRUE {
                BOOLEAN [ 
                    TRUE {
                        UINT32 SYS_SIZE;
                    },
                    FALSE [ IRQ_STACK;
                },
                FALSE [ MULTI_STACK = FALSE;
            }
            BOOLEAN ICD2 = FALSE;
            BOOLEAN ENABLE_SPLIM = TRUE;
        }
    }
    CPU_DATA[];
}
```

**example**

```c
CPU_DATA = PIC30 {
    APP_SRC = "code.c";
    MULTI_STACK = FALSE;
    ICD2 = TRUE;
};
```

```c
CPU_DATA = PIC30 {
    APP_SRC = "code.c";
    MULTI_STACK = TRUE {
        IRQ_STACK = TRUE {
            SYS_SIZE=64;
        };
        IRQ_STACK = TRUE {
            SYS_SIZE=64;
        };
    };
    ICD2 = TRUE;
    ENABLE_SPLIM = TRUE;
};
```
OIL (OS object : MCU data)

```c
enum [ 
  PIC30 { 
    enum [ 
      custom { 
        string model;
        string linkerscript;
        string dev_lib;
        string include_c;
        string include_s;
      },
      PIC24FJ128GA006, 
      PIC24FJ128GA008, 
      [...] 
    ] model;
  } 
] MCU_DATA;
```

```c
MCU_DATA = PIC30 {
  model = PIC33FJ256GP710;
};
```

```c
MCU_DATA = PIC30 {
  model = custom {
    linkerscript = "p33FJ256GP710.gld";
    dev_lib = "libp33FJ256GP710-elf.a";
    include_c = "p33FJ256GP710.h";
    include_s = "p33FJ256GP710.inc";
  }
};
```
OIL (OS Object: board data)

ENUM [
    NO_BOARD,
    EE_FLEX {
        BOOLEAN USELEDS;
    },
    MICROCHIP_EXPLORER16 {
        BOOLEAN USELEDS;
        BOOLEAN USEBUTTONS;
        BOOLEAN USELCD;
        BOOLEAN USEANALOG;
    },
    MICROCHIP_DSPICDEM11PLUS {
        BOOLEAN USELEDS;
        BOOLEAN USEBUTTONS;
        BOOLEAN USELCD;
        BOOLEAN USEANALOG;
        BOOLEAN USEAUDIO;
    }
]

BOARD_DATA =
    MICROCHIP_EXPLORER16 {
        USELEDS = TRUE;
        USEBUTTONS = TRUE;
        USELCD = TRUE;
        USEANALOG = TRUE;
    };

BOARD_DATA = EE_FLEX {
    USELEDS = TRUE;
};

BOARD_DATA =
    MICROCHIP_DSPICDEM11PLUS {
        USELEDS = TRUE;
        USEBUTTONS = TRUE;
        USELCD = TRUE;
    };

BOARD_DATA = NO_BOARD;
OIL (OS Object: libraries and kernel type)

```plaintext
ENUM [  
    ENABLE {  
        STRING NAME;  
    }  
] LIB;

ENUM [  
    FP {  
        BOOLEAN NESTED_IRQ;  
    },  
    BCC1,  
    BCC2,  
    ECC1,  
    ECC2  
] KERNEL_TYPE;

LIB = ENABLE {  
    NAME = SCICOS;  
};

KERNEL_TYPE = FP;
```

www.evidence.eu.com
TASK {
    UINT32 PRIORITY;
    UINT32 ACTIVATION = 1;
    ENUM [NON, FULL] SCHEDULE;
    TYPE RESOURCE[];
    ENUM [
        SHARED,
        PRIVATE {
            UINT32 SYS_SIZE;
        }
    ] STACK = SHARED;
};

TASK TaskFlash {
    PRIORITY = 1;
    STACK = SHARED;
    SCHEDULE = FULL;
};

TASK Producer {
    PRIORITY = 2;
    STACK = PRIVATE {
        SYS_SIZE = 64;
    }
    SCHEDULE = FULL;
};
RESOURCE { 
    ENUM [ 
        STANDARD { 
            STRING APP_SRC[]; 
        }, 
        [...] 
    ] RESOURCEPROPERTY; 
};

TASK LowTask { 
    RESOURCE = "myResource"; 
    [...] 
};

RESOURCE myResource { 
    RESOURCEPROPERTY=STANDARD; 
};
COUNTER { 
    [...] 
};

ALARM { 
    COUNTER_TYPE COUNTER;
    ENUM [ 
        ACTIVATETASK { 
            TASK_TYPE TASK;
        },
        [...] 
        ALARMCALLBACK { 
            STRING 
            ALARMCALLBACKNAME;
        } 
    ] ACTION;
};

COUNTER myCounter;

ALARM AlarmFlash { 
    COUNTER = "myCounter";
    ACTION = ACTIVATETASK { 
        TASK = "TaskFlash";
    };
};