
C LANGUAGE FOR EMBEDDED & REAL-TIME APPLICATIONS

Ref : 002603A

Duration : 5 days

OBJECTIVES

- The course shows how to parameterise the linker according to a board mapping
- Access to peripheral registers by using unions and bitfield structures is explained
- An interrupt handler enabling nesting is developed in C
- The course shows how to connect `read()` and `write()` functions to a low level driver
- The course shows how to design a proprietary task scheduler
- Communication between tasks is highlighted
- Implementation of the ST I/O library is explained

RELATED COURSES

- Trainings specific to a microcontroller (see our catalogue)
- Linux – Kernel programming, Device Drivers, BSP (Ref.003154A)
- Linux – User Programming (Ref.003346A)

PREREQUISITES

- Basic knowledge of C or other programming language is recommended even if the C syntax is revised at the beginning of the course
- This is not a C course for beginners

PRACTICAL LABS

- For on-site courses, labs can be run under the following environments :
 - GNU/Lauterbach
 - Keil μ Vision
 - IAR Workbench
- For open courses, labs are run under IAR Workbench

Contact

Tel : +33 (0)5 62 13 52 32
Fax : +33 (0)5 61 06 72 60
training@mvd-fpga.com

Course also available
customized

C

Next sessions, see : <http://www.mvd-fpga.com/en/formationsCalend.html>

TOPICS

PARAMETERIZING THE COMPILER SUITE

- Becoming familiar with the development environment
- Organization of a C program
- Compiler parameterizing
- Linker parameterizing
- Sections automatically generated by the compiler
- Creation of custom sections
- Guidelines for processors containing MMU and cache
- The linker script file, declaration of memory areas, group command
- Cstart file

THE PRE-PROCESSING

- Inclusion of header files
- Macros
- Assembly macros, volatile vs non-volatile registers
- Conditional compilation

TYPES AND OPERATORS

- Storage classes : `register`, `extern`, `static` and `auto`
- Local variables vs global variables
- C types
- Constants
- Implicit and explicit casting
- Operators, priority between operators and evaluation ordering
- `volatile` attribute
- `const` attribute

STATEMENTS

- Selective statements : `if...else`, ternary operator, `switch...case`
- Iterative statements : `while()`, `do...while()`, `for()`
- Control statements : `break`, `continue`, `return`

POINTER AND ARRAY

- Addressing modes

- Pointers : declaration, initialisation
- Single dimension arrays
- Multi dimension arrays

STRUCTURE AND UNION

- Structure : declaration, initialisation, access to fields, nested structures
- Array of structures
- Bitfield structures, endian sensitivity
- Benefit of typedef
- Enum type
- Union : declaration, initialisation, access to field
- Modelling IO registers through bitfield structures and unions
- Padding, implicit alignment of fields

FUNCTIONS

- Declaration in C ANSI and in C K&R
- static and extern attributes
- Function call, argument passing
- Return value
- `main()` function, reset to main sequence
- Pointer to function
- C/assembly interface : description of the stack frame
- Inline assembly

DYNAMIC ALLOCATION AND LINKED LISTS

- Benefit of the heap compared to static data sections
- Related functions : `malloc()`, `calloc()`, `realloc()` and `free()`
- Description of the memory allocation first fit algorithm
- Linked lists : single-linked, double-linked, tree lists
- Basic operations : item insertion and deletion

SPECIFIC INPUT / OUTPUT DRIVERS

- Exception management in C, the vector table of the ARM core
- Level-triggered vs edge-triggered interrupts
- Introduction to STR712 timers

- Generic drivers
- Libraries, creation of an archive file
- Description of the UARTs present in the STR712
- Studying the serial driver developed by MVD
- Flashing a program into ROM

STANDARD INPUT / OUTPUT FUNCTIONS

- Connection to the specific serial input and output functions
- Standard input and output functions, definition of `stdin`, `stdout` and `stderr`
- Buffered inputs outputs
- Formatted input and output functions `printf()` and `scanf()`
- File input output functions, description of `fopen()` and `fclose()`

TASK SCHEDULING

- Definitions : thread vs process, preemptive vs non preemptive RTOS
- Task states and transition between states
- Multitask and real time
- Task switch
- Determinism
- Implementation of a task scheduler : definition of a scheduling algorithm
- Management of task descriptor lists
- Task suspend and resume mechanism
- Inter task communication

DOCUMENTATION

Training manuals will be given to attendees during training in print.