# ac6

# The C++ Language for Embedded Systems

## Objectives

- Master the C++ language
- Use C++ Template (generic code) in Embedded Systems
- Master the C++ Advanced aspects such as polymorphism, single and multiple inheritances.
- Learn to redefine the C++ operators for dynamic memory allocation in embedded applications
- Make C++ objects persistent in programmable flash
- Manage C++ exceptions for Secure Embedded applications
- Use C++ objects to handle serial transmission / reception of character strings

## Equipment

- Training manuals and software exercises
- One PC for two trainees
- One target board with ARM Cortex M4 microcontroller (STM32F)
- Eclipse environment and GCC compiler

## Prerequisite

- C Language knowledge (see for example our L2 - C language for Embedded MCUscourse)

## Course Environment

- Theoretical course
    - PDF course material (in English) supplemented by a printed version.
    - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance.
- Practical activities
    - Practical activities represent from 40% to 50% of course duration.
    - Code examples, exercises and solutions
    - One PC (Linux ou Windows) for the practical activities with, if appropriate, a target board.
    - ▶ One PC for two trainees when there are more than 6 trainees.
    - For onsite trainings:
    - ▶ An installation and test manual is provided to allow preinstallation of the needed software.
    - ▶ The trainer come with target boards if needed during the practical activities (and bring them back at the end of the course).
- Downloadable preconfigured virtual machine for post-course practical activities
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

## Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

## Evaluation modalities

- The prerequisites indicated above are assessed before the training by the technical supervision of the traineein his company, or by the trainee himself in the exceptional case of an individual trainee.

- Trainee progress is assessed in two different ways, depending on the course:
  - For courses lending themselves to practical exercises, the results of the exercises are checked by the trainer while, if necessary, helping trainees to carry them out by providing additional details.
  - Quizzes are offered at the end of sections that do not include practical exercises to verifythat the trainees have assimilated the points presented
- At the end of the training, each trainee receives a certificate attesting that they have successfully completed the course.
  - In the event of a problem, discovered during the course, due to a lack of prerequisites by the trainee a different or additional training is offered to them, generally to reinforce their prerequisites,in agreement with their company manager if applicable.

## Plan

## *First Day*

## Introduction to C++ for industrial systems

- Introduction to object oriented programming
- History and definition
- Overview on C++ standards
- Modern C++ objectives
- Switch from C to C++
- Embedded C++ specification ( EC++)
- How to write optimized embedded code

***Exercise:*** *Understand function mangling*
***Exercise:*** *Function inlining*
***Exercise:*** *Volatile variable handling*

## C++ and embedded systems

- Object Oriented Programming in C++
  - Encapsulation
  - Classes and objects
  - Attributes and member functions
  - Object construction and destruction
  - Construction parameters
  - Copy constructor
  - Object composition and container
  - Scope qualifier operator

***Exercise:*** *Declaring classes and methods*
***Exercise:*** *Working with default, copy and parameterized constructors*
***Exercise:*** *Understand the differences between composition and aggregation*

## *Second Day*

## C++ and embedded systems

- Operator Overloading
  - Optimizing parameter object passing
  - Overloading operators by member functions
  - Overloading operators by friend functions
  - Memory management operators overloading

***Exercise:*** *The assignment operator*
***Exercise:*** *Overloading operators*

- Simple Inheritance
  - Specialization by addition and substitution
  - Derivation and access rules
  - Construction during inheritance

- o Inheritance polymorphism
- o Virtual methods

*Exercise: Understand inheritance*

- Persistent and flashable objects
  - o Constant and partially constant objects
  - o Persistent objects
  - o Flashable objects

*Exercise: Creating constant, mutable, persistent and ROMable objects*

- Enhancing security with exceptions
  - o Launching, capturing and handling exceptions
  - o Retriggering exception
  - o Exceptions specifications
  - o Handling unexpected exception
  - o Exception objects of the C++ standard library

*Exercise: Handle errors using exceptions*
*Exercise: Unexpected exceptions management*

## *Third Day*

## *C++ Advanced Techniques*

- I/O streams
  - o C++ language standard streams
  - o C++ standard librairies streams
  - o Standard IO redirection by friend functions

*Exercise: Redefine operators '    >' to read/write objects on an IO stream*

- Member pointers
- Generic objects and templates
  - o Classes and generic functions
  - o Templates overloading
  - o Specializing templates
  - o STL (Standard Template Library)
  - o Templates in embedded systems

*Exercise: Generic classes and functions*

- Polymorphic objects
- Virtual objects and abstract classes
- Specializing objects by simple inheritance
  - o Building derivate objects
  - o Access control rules for inherited objects
  - o Specializing objects by multiple inheritance
  - o Conflicts resolution by scope operator
  - o Virtual inheritance

*Exercise: Understand virtual methods by subclassing a generic Device class*
*Exercise: Understand multiple inheritance and virtual bases*

## Renseignements pratiques

### Inquiry :  3 days