**Ford C Programming Academy 2022**

**Course Overview**

During this intensive academy, students will learn how to read, write, compile, execute, and debug programs in the Standard C Programming Language. Emphasis will be on coding for automotive embedded systems, and producing clear, correct, testable, and safe code. We will consider the constraints of embedded microprocessors - i.e, limited program and data memory, varying word sizes and supported data types, and potential lack of FPU and other computational features. We will discuss tradeoffs in programming vis a vis execution speed, code size, and memory usage. Compliance with selected MISRA C 2012 rules will be discussed in some detail. Students will spend about 70% of their time coding in instructor-led and independent exercises. A variety of projects will be completed, with a focus on understanding datatypes and algorithms used in embedded environments.

**Audience**

* Engineering staff who have self-selected to pursue a career in embedded software
* Students will have previously completed training in using MATLAB and Model-Based Design to generate C code
* Programming experience in any language will be helpful, but is not a prerequisite

**Purpose**

Skilling Up

**Course Duration**

3 weeks

**Productivity Objectives**

Upon exiting this program, students will be able to:

* Read Standard C source code generated by MATLAB tools
* Write clear, correct, and functional C code to implement numerical algorithms
* Interface with standard and custom libraries
* Follow best practices for conforming to MISRA and other coding guidelines

**Course Outline**

**Week 1: C Language Programming**

* The C coding environment
  + Hosted vs standalone implementations
  + IDEs and Toolchain overview
    - The command shell and the file system
    - Preprocessor, Compiler, Linker
  + Using and customizing Microsoft Visual Studio
  + Managing code with git and GitHub
* Writing C Programs
  + Foundations: keywords, syntax, punctuation
  + Using the #include directive to access libraries
  + Comments
  + Identifiers
  + Variables and Data Types
  + Writing and calling functions
    - Parameters / arguments / pass by value
    - Return data
* Operators and expressions
  + Syntax, Constraints, and Semantics
  + Arithmetic type conversions
* Arithmetic data types and considerations
  + Integral and floating-point types
  + Symbolic constants (literals and macros)
  + Numeric limits and edge cases
  + Overflow/Underflow
* Using Libraries
  + Standard libraries
    - Math, Algorithms, I/O
    - Complex/Imaginary types
  + User Libraries
    - Ford, MATLAB, etc.
* Controlling the flow of execution
  + Type modifers - const, unsigned
  + Conditionals
    - if/else, switch/case
  + Loops
    - while, do/while, for
* Input/Output
  + Streams and console I/O
  + File I/O
* Structured Data Types
  + struct
  + union
* Pointers and arrays
  + NULL-terminated strings
  + Passing function arguments by reference

**Week 2: The C Toolchain, datatypes, algorithms and memory management**

* Data Types - A Deep Dive
  + Object representations
    - Size, Alignment, Endianess
  + Type conversions
    - Implicit conversions and casting
    - Promotion/truncation
* Organizing code
  + Structuring code for reuse / refactoring
  + Scoping variables - Local, File, Global scopes
  + Writing libraries and header files
* Memory management
  + Automatic variables and the stack
  + Static memory allocation
  + Dynamic memory allocation and the heap
  + Techniques for reusing memory
  + Special storage classes - volatile, register, etc.
* Asynchronous events
  + Traps, Interrupts and Signals
  + Threads and synchronization
  + Atomic variables
* The Toolchain - a Deep Dive
  + Preprocessor, compiler, assembler, linker, loader
    - Examining the output of different tools
  + Preprocessor directives
    - Conditional compilation and guards
    - Object-like macros
    - Function-like macros
  + The Compiler
    - Compilation phases and products
    - Typical compiler switches
      * Output, debug, optimization
      * Library and Include paths
    - Cross-Compiling
* Debugging with Visual Studio
  + The call stack
  + Breakpoints, watchpoints, and stepping through code
  + Examining the value of variables

**Week 3: Testing, Code metrics and Safety Compilance with MISRA**

* Embedded Programming Challenges
  + Processor types and limitations
  + Address space and memory types - Flash/PROM, ROM, RAM
  + Real-time constraints
* Code metrics and tradeoffs
  + Code size
  + Memory utilization
  + CPU usage and efficiency
  + Effects of optimization and data type choices
* Useful algorithms and techniques for embedded systems
  + Preprocessor math
  + Fixed-point math with integral types
* Safety and security in embedded automotive applications
  + MISRA C 2012 Compliance
  + Specific rules for safe and secure C coding
* Clean Code and Programming Style
  + Writing for readability
  + Using comments effectively
  + Designing for reuse
* Ensuring code quality through testing
  + Static Code Analysis and Testing
  + Unit Testing with Unity / CMock
  + Profiling, logging, instrumentation