Embedded C

From Wikipedia, the free encyclopedia

Embedded C is a set of language extensions for the C Programming language by the C Standards committee to address commonality issues that exist between C extensions for different embedded systems. Historically, embedded C programming requires nonstandard extensions to the C language in order to support exotic features such as fixed-point arithmetic, multiple distinct memory banks, and basic I/O operations.

In 2008, the C Standards Committee extended the C language to address these issues by providing a common standard for all implementations to adhere to. It includes a number of features not available in normal C, such as, fixed-point arithmetic, named address spaces, and basic I/O hardware addressing.

Embedded C uses most of the syntax and semantics of standard C, e.g., main() function, variable definition, datatype declaration, conditional statements (if, switch, case), loops (while, for), functions, arrays and strings, structures and union, bit operations, macros, etc.

A Technical Report was published in 2004[1] and a second revision in 2006.[2]

Necessity

During infancy years of microprocessor based systems, programs were developed using assemblers and fused into the EPROMs. There used to be no mechanism to find what the program was doing. LEDs, switches, etc. were used to check for correct execution of the program. Some ‘very fortunate’ developers had In-circuit Simulators (ICEs), but they were too costly and were not quite reliable as well. As time progressed, use of microprocessor-specific assembly-only as the programming language reduced and embedded systems moved onto C as the embedded programming language of choice. C is the most widely used programming language for embedded processors/controllers. Assembly is also used but mainly to implement those portions of the code where very high timing accuracy, code size efficiency, etc. are prime requirements.

As assembly language programs are specific to a processor, assembly language didn’t offer portability across systems. To overcome this disadvantage, several high level languages, including C, came up. Some other languages like PLM, Modula-2, Pascal, etc. also came but couldn’t find wide acceptance. Amongst those, C got wide acceptance for not only embedded systems, but also for desktop applications. Even though C might have lost its sheen as mainstream language for general purpose applications, it still is having a strong-hold in embedded programming. Due to the wide acceptance of C in the embedded systems, various kinds of support tools like compilers & cross-compilers, ICE, etc. came up and all this facilitated development of embedded systems using C.[3] Assembly language seems to be an obvious choice for programming embedded devices. However, use of assembly language is restricted to developing efficient codes in terms of size and speed. Also, assembly codes lead to higher software development costs and code portability is not there. Developing small codes are not much of a problem, but large programs/projects become increasingly difficult to manage in assembly language. Finding good assembly programmers has also become difficult now a days. Hence high level languages are preferred for embedded systems programming.

Advantages

- It is small and simpler to learn, understand, program and debug.
- Compared to assembly language, C code written is more reliable and scalable, more portable between different platforms.
- C compilers are available for almost all embedded devices in use today, and there is a large pool of

experienced C programmers.

- Unlike assembly, C has advantage of processor-independence and is not specific to any particular microprocessor/microcontroller or any system. This makes it convenient for a user to develop programs that can run on most of the systems.
- As C combines functionality of assembly language and features of high level languages, C is treated as a 'middle-level computer language' or 'high level assembly language'.
- It is fairly efficient.
- It supports access to I/O and provides ease of management of large embedded projects.
- Java is also used in many embedded systems but Java programs require the Java Virtual Machine (JVM), which consumes a lot of resources. Hence it is not used for smaller embedded devices.

Other High-level programming language like Pascal, FORTRAN also provide some of the advantages.[3]

References

1. Information Technology — Programming languages, their environments and system software interfaces — Extensions for the programming language C to support embedded processors (http://www.open-std.org/JTC1/SC22/WG14/www/docs/n1021.pdf), ISO/IEC JTC1 SC22 WG14 N1021, Date: 2003-09-24, Reference number of document: ISO/IEC DTR 18037
2. Information Technology — Programming languages - C - Extensions to support embedded processors (http://www.open-std.org/JTC1/SC22/WG14/www/docs/n1169.pdf), ISO/IEC JTC1 SC22 WG14 N1169, Date: 2006-04-04, Reference number of document: ISO/IEC TR 18037
3. Embedded C, What is Embedded C, Difference between C and Embedded C Programming (http://www.engineersgarage.com/tutorials/emebedded-c-language), By Akshay Daga, EngineersGarage

Retrieved from "https://en.wikipedia.org/w/index.php?title=Embedded_C&oldid=684398038"

Categories: C (programming language) | C programming language family