INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

LANGUAGES COMPATIBILITY WITH THE EMBEDDED SYSTEM

TANWIN ASHRAFI[1], NUSRAT ANSARI[2], TAHEREEN MOMIN[3], IMRAN MOMIN[4], ANIS CHODHARY[5]

1. M.Sc.(Electronic), NET, Assistant Professor, Dpt I.T, G. M. Momin College.
2. M.C.A, Assistant Professor, Dpt I.T, G. M. Momin College.
3. MSc.(I.T), Assistant Professor, Dpt I.T, G. M. Momin College.
4. M.C.A, Assistant Professor, Dpt I.T, G. M. Momin College.
5. HOD, Dpt I.T, G. M. Momin College.

Accepted Date: 16/08/2014; Published Date: 01/09/2014

Abstract: Embedded system is a small platform to perform the specified task set by the programmer. It is combination of hardware and software. Hardware complexities required to have software simplification to program the device for that specific task. Devise architecture information need to have before setting it to program in a header file. Since the same program cannot be perform on every device as Assemble language are different for every device but with the advance software development tool, we can work on high level language it give some relief on programming. High level language such as C gives a good platform to embedded system but there are many other languages such as java, synERJY and Ada. Our task is to elaborate the programming difficulties of the embedded system.

Keywords: Embedded system, Assembly language, high level language, C language, java, synERJY, Ada

Corresponding Author: PROF. TANWIN ASHRAFI

Access Online On:

www.ijpret.com

How to Cite This Article:

Tanwin Ashrafi, Nusrat Ansari, Tahereen Momin, Imran Momin, Anis Chodhary; IJPRET, 2014; Volume 3 (1): 43-50

PAPER-QR CODE

INTRODUCTION

Writing software for embedded system is certainly different from writing software for applications running on general-purpose desktops and mainframes. While writing software for the embedded system, programmer need to cognizant of the constraints imposed by the embedded systems- namely, the limited processing power, limited memory, and limited input/output. Most common programming languages for embedded systems are C, BASIC and assembly languages. To produce the most efficient machine code, the programmer must not only create an efficient high level design, but also pay attention to the detailed implementation. One of the best features of C is that it is not tied to any particular hardware or system. This makes it easy for a user to write programs that will run without any changes on practically all machines. C used for embedded systems is slightly different compared to C used for general purpose (under a PC platform) programs for embedded systems are usually expected to monitor and control external devices and directly manipulate and use the internal architecture of the processor such as interrupt handling, timers, serial communications and other available features.

OBJECTIVES

1. Factors to be consider for selecting languages.
2. Different languages used for programming in embedded.
3. Programming difficulties in Assembly.

RESEARCH METHODOLOGY

a) Factors to be consider for selecting languages:

There are many factors to consider when selecting languages for embedded systems

- Efficiency - Programs must be as short as possible and memory must be used efficiently.
- Speed - Programs must run as fast as possible.
- Ease of implementation
- Maintainability

- Readability
- Security
- Reliability

b) Different languages used for programming in embedded:

Focus on the following languages along with Assembly Language: C, Java, Ada and synERJY.

WHY C?

► C is much more flexible than other high-level programming languages:

• C is a structured language.

• C is a relatively small language.

• C has very loose data typing.

• C easily supports low-level bit-wise data manipulation.

• C is sometimes referred to as a “high-level assembly language”.

► When compared to assembly language programming:

• Code written in C can be more reliable.

• Code written in C can be more scalable.

• Code written in C can be more portable between different platforms.

• Code written in C can be easier to maintain.

• Code written in C can be more productive.

► These are some of the common issues that we encounter when considering moving to the C programming language:

• Big and inefficient code generation

• Fat code for the standard IO routines (printf, scanf, strcpy, etc…)

• The use of the stack is not so direct in C

• Data declaration in RAM and ROM

• Difficulty writing Interrupt Service Routines[6,7]

WHY JAVA?

- ► The four top reasons for using Java in embedded systems:
- Java in embedded systems: Write application once, run it on any other hardware platform
- Object-Oriented Programming (OOP): A revolutionary concept that makes developers' life easier
- Standard class libraries: Don't spend time rewriting existing functionalities.
- J2MicroEdition(J2ME) has Wireless Toolkit helps in debugging capabilities and also it has its own emulators.
- ►Comparison with Assembly
- Working with java will give code portable between different platforms.
- Easy to work on multiple processor/ controller.
- Easy to maintain the code for long time.
- ► Common issues that encounter when considering Java programming:
- Before starting out with an embedded Java project there are a great deal of information that need to consumed.
- It is not enough to understand the Java language. Java can't directly control Hardware.
- The JVM/KVM should be understood together with the boot process etc.,just as the C runtime system is understood.

- Lots of profiles and configurations should be studied.[4,5]

WHY ADA?

The Ada language was designed as a common high-order language for programming large-scale and real-time systems. From a real-time application developer's viewpoint, the Ada language addresses the embedded system problem domain.[8]

➢ Features:

- Multitasking: Since real-time software typically must handle an unpredictable sequence of operations, decisions about task rendezvous may have to be made at runtime.
- Time control: Ada offers three pragmas for reducing execution time.
- Input/output: Ada provides all the capabilities I/O operations to and from hardware devices and handle device interrupts.
- Internal Representation: Ada also defines an optional representation pragma named PACK for influencing a compiler's mapping of entities (either arrays or records) onto the underlying machine.
- Error Handling: The Ada language provides a mechanism for dealing with errors or other exceptional situations during program executions.

➢ Compression with Assembly:

- Ada give a good environment working on high platform.
- Support to the Multitasking provide the easy code generation.
- Good library provide the easy way to work on the hardware.
- Error handling is one of the best feature with shows error before implementing on the hardware.

➢ Issues:

- Which languages can be interfaced with Ada?

- Implementation technique for an interrupt entry call.
- Ada runtime systems that will affect embedded systems software.
- Ada is a scoped language like Pascal but has more data types of greater complexity. Memory management is quite complex because of the nature of the data types.
- Ada's tasking facilities may not meet some of the performance requirements of an embedded system where time criticality is top priority, unless some customizing and optimizing are carried out or certain programming paradigms are used
- The Ada provides no explicit facilities to address implementation on multiprocessors.[2]

WHY synERJY?

A team at Fraunhofer Institute for Autonomous Intelligent Systems has developed and implemented 'synERJY', a Java-like synchronous object-oriented language and programming system for the efficient design of embedded systems.

Highlights of synERJY:

- It provides a smooth embedding of the reactive behavior into the object-oriented data model offers fine-grained integration of synchronous formalisms such as the imperative style of ESTEREL, the dataflow-based style of LUSTRE, hierarchical state machines in the style of STATECHARTS.
- It generates efficient code in space and time even for small microcontrollers.

There is little need to discuss the merits of object-oriented design. There are, however a couple of notable benefits to be gained from the combination of object-oriented design with synchronous programming.

- Due to the underlying broadcast mechanism of synchronous programming, reactive objects (ie objects that exhibit reactive behavior) behave like components, in that they may be 'plugged in' without changing their behavior, regardless of the environment.
- It extends the causality analysis of other synchronous languages, time races between method calls can be detected enabling deterministic scheduling at compile time including the scheduling of all data actions.

- The component-based approach supports hiding of hardware details such as irrelevant properties of, for example, specific CAN hardware.[11]

CONCLUSION

Language is the base or says start point of the embedded system designing. Language has to be selected before starting the project. The paper thus highlights the different languages which provide the support for the embedded systems. Many IDE tools are easily available for different language and also support different languages.

C languages always remain the grate language to be work on. It is middle language that means it support high level and low level language. It can handle hardware interfacing with software very much easily. Ada provides a good environmental platform for the 4^{th} generation of the embedded systems. Ada consists of huge amount of library functions which make it much easier to work on. Multitasking being the need of the future, Ada supports it extensively. It even reduces execution time.

Before starting out with an embedded Java project there are great deals of information that need to consumed. The JVM/KVM should be understood together with the boot process etc., just as the C runtime system is understood. Lots of profiles and configurations should have to be studied. Java cannot directly control the hardware.

The first generation of Ada runtime systems for embedded targets is not sufficiently compact, tailorable, or efficient for embedded real-time applications that operate under stringent memory and throughput constraints.

SynERJY is a relatively new but by now stable language, the development of which has been supported by the ESPRIT Projects SYRF and CRISYS. At present it is mainly used in-house for robotics applications and externally for teaching purposes.

REFERENCES

1. www.hyphen-innovation.com.

2. Nelson H. Weiderman, Mark W. Borger, “Ada for Embedded Systems: Issues and Questions” December 1987.

3. Reinhard Budde, Axel Poigné and Karl-Heinz Sylla, “ synERJY: a High-Level Language for Embedded Controllers” ERCIM News No. 52, January 2003.

4. KV Shibu ,"Introduction to Embedded Systems", McGrawHill,©2009.

5. Raj Kamal,"Embedded Systems",2nd Edi, McGrawHill,©2008

6. Dr. Rajiv Kapadia, "8051 Microcontroller & Embedded Systems", 7th edi., Jaico Publi. House, © 2009.

7. M.A. Mazidi, J.G. Mazidi,R.D.McKinlay.,"The 8051 Microcontroller And Embedded Systems", 2nd edi., Pearson Education, ©2009.

8. http://www.adahome.com/History/Steelman/steeltab.htm.

9. Tammy Noergaard,"Embedded Systems Architecture",2nd edi., Newnes, ©2013.

10. HORIBA Scientific ,"User's Guide for SynerJY® Software Version 3", © 2009