Low Cost Control and Monitoring Motion Control ICs

João Moutinho
Faculdade de Engenharia
Porto, Portugal.

Rui Esteves Araújo
Faculdade de Engenharia
Porto, Portugal.

Vicente Leite
Instituto Politécnico de Bragança
Bragança, Portugal.

Abstract—**For a long time, controlling AC motors in an efficient way, was a task only in the reach of some very specific electronic design Engineers. Now, with the up rise of Integrated Platforms it is possible to implement, with almost no effort, the desired application, customizing almost everything at the reach of a simple and user friendly software interface. However, the simplicity in control and monitoring has a major settle back. As expected, such products, very desirable by the market, are only sold in conjunction with an evaluation platform. Sometimes this platform does not fit the consumer needs and buying the full package is not a satisfactory solution, especially if it is just to get the control and monitoring software.**

In this paper, it will be shown that low cost alternatives are possible. Using free tools and manufacturer's documentation, it is simple to implement Motor Control with the desired IC by means of a very intuitive and complete Graphical Unit Interface (GUI), built attending our special needs.

***Index Terms*—AC Motor Control, Low Cost AC Motor Control, Integrated Solutions for AC Motor control, Interfacing Motor Control, Java.**

I. INTRODUCTION

THIS paper intends to demonstrate the possibility of implementing a low cost, well adapted and simple alternative to the control and monitoring software provided by the Digital Control IC's manufacturers, with no need of their evaluation platforms acquisition.

Usually, the hardware designer has two options: he can buy the Core IC or the whole package (that includes some kind of software to control and/or monitor). The second option is more appreciated if the designer thinks that he can not spend time learning the IC, but it is just a matter of time until he will have to open the datasheet and dig a little further. The first option is not so appreciable, but in the end, it will end saving time.

Nowadays, with the evolution of Motion Control IC's and the high energy efficiency, required for today's motor control applications, vector control algorithms [1] are no longer difficult to implement, and the final solution becomes compact, robust and with relatively low expenses. Building a complete servo system was never so simple and cheap.

One of the products of this evolution has come from International Rectifier, recognized as a leading supplier of power semiconductors and systems solutions. They have developed a specific application Design Platform comprising of a flexible, mixed signal chip set, which integrates analog, digital and power electronics to provide a fast time-to-market, high performance, and a cost effective solution for specific applications.

However, like it was previously mentioned, two options must be taken into account. For this example we will choose the first one and buy the Core IC, the IRMCK201. It includes:

- An adjustable FOC algorithm with host CPU access to all important drive parameters and monitoring points, with the ability to make some of the most common changes to the algorithm structure without changing code;
- High bandwidth torque loop response up to 5 kHz;
- Adjustable speed loop bandwidth, typically up to 400 Hz;
- Loss minimization space vector PWM with maximum frequency up to 70 kHz;
- Current loop execution time less than 6µs;
- Flexible drive configuration (PMAC or induction motor);
- Quadrature encoder interface;
- RS232C/RS422 and fast 6-MHz SPI interface;
- Built-in 8-kByte trace buffer memory for diagnostics and monitor function;
- Parallel interface for microcontroller expansion port;
- Express the algorithm design in a clear block-diagram format for understandability (see figure 1).

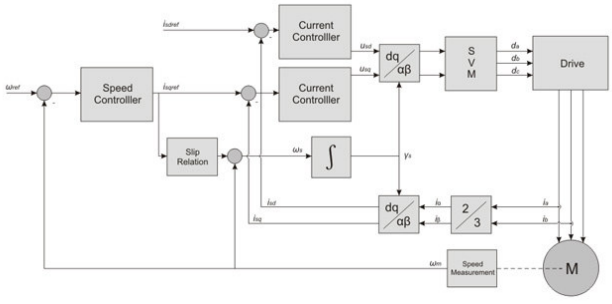

Figure. 1. Detailed block diagram of IRMCK201.

Based on several innovating characteristics, this IC is the right start to an AC Motor Control application. Since it is assumed that we do not want to buy the expensive evaluation platform, let us now focalize on how to build an interface in the most simple and quickest way, so that every characteristic of this very complete IC can be used.

This work has been partially funded by the Department of Electrical and Computer Engineering of the Faculty of Engineering of the University of Porto (FEUP), under contract DEEC-ID/2/2005.

II. INTERFACE DEVELOPMENT

The first issue is to define which are the objectives of having an interface program to interact with the IC:

- Establishment of the status state;
- Control the input variables;
- Monitoring of the output variables;
- Intuitive control;
- Portability;
- Support.

To accomplish all these characteristics, and since one of the greatest interests is to implement a low cost solution, it is necessary to use a software design platform that fulfills our needs. In this research, Java with the Eclipse design platform was our choice.

Java has gained enormous popularity since its first appearance. Its rapid ascension and wide acceptance can be traced by its design and programming features, particularly the fact that you can write a program once, and run it anywhere. As stated in Java language white paper by Sun Microsystems: "Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, multithreaded, and dynamic." [2].

No language is simple, but using the object-oriented programming language Java is considered much simpler and easier than using the popular programming language C++. Partially modeled after C++, Java has replaced the complexity of multiple inheritance in C++ with a simple structure called interface, and also has eliminated the use of pointers.

The reason why Java is much simpler than C++ is because Java uses automatic memory allocation and garbage collection while C++ requires the programmer to allocate memory and to collect garbage. Also, the number of language constructions in Java is small for such a powerful language. The clean syntax makes Java programs easy to write and read. Java is Object-oriented and models the real world. Everything in the world can be modeled as an object. For example, a circle is an object, a person is an object, and an IC is an object. Even a serial port can be perceived as an object. Java is object-oriented because programming in Java is centered on creating objects, manipulating objects, and making objects work together.

One of the most compelling reasons to use Java is its platform independence. Java runs on most major hardware and software platforms, including all Windows versions, Macintosh, and several varieties of UNIX and Linux. Java applets are supported by all Java-compatible browsers. By moving existing software to Java, you are able to make it instantly compatible with these software platforms. Java programs become more portable. Any hardware and operating system dependencies are removed. In C and C++, each implementation decides the precision and storage requirements for basic data types (*short, int, float, double*, etc.). This is a major source of porting problems when moving from one kind of system to another, since changes in numeric precision can affect calculations and assumptions about the size of structs can be violated. Java defines the size of basic types for all implementations; an *int* in one system has the same size (and can represent the same range of values) as in any other system. It does not permit the use of arbitrary pointer arithmetic, so assumptions about struct packing and sizes can not lead to non-portable coding practices.

Although C and C++ are supported on all platforms that support Java, these languages are not supported in a platform-independent manner. C and C++ applications that are implemented on one operating system platform are usually severely intertwined with the native windowing system and OS-specific networking capabilities. Moving between OS platforms requires recompilation, as a minimum and significant redesign, in most cases. With this, one of the most important barriers has fallen: no more a platform will be an impediment.

The main disadvantage of Java is speed. Although Java's ability for producing portable and architecturally neutral code is desirable, the method used to create this code is inefficient. Java code is compiled into byte code and an interpreter called a Java Virtual Machine, specifically designed for a computer's architecture, runs the program. Why is this a problem? Java, being an interpreted system, is currently an order of magnitude slower than C. Unlike natively compiled code, which is a series of instructions that correlate directly to a microprocessors' instruction set, an interpreter must first translate the Java binary code into the equivalent microprocessor instruction. Obviously, this translation takes some amount of time and, no matter how small a length of time is this, it is inherently slower than performing the same operation in machine code.

It is simple to conclude that for exigent performance applications Java may not be the right choice. But for Motor Control and Monitoring, Java's performance is probably more than enough.

To develop our Java application we used Eclipse. It is a popular free Integrated Development Environment (IDE) for writing Java.

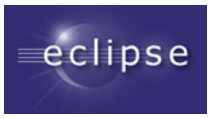

Figure. 2. Logo of the Eclipse IDE java development.

The Eclipse Project is an open source project of eclipse.org. It is an enormous project encompassing more than just an IDE and it allows plug-ins for extra functionality. It uses its own compiler that lets it do incremental compiles and hot swapping of code during debugging. And let us not forget that it is free!

Other alternative to Java development is the NetBeans IDE, another Java IDE that also assumes great importance in the programming world.

III. Application Example

As it was previously mentioned it is currently in progress a development of a Control and Monitoring application to the IRMCK201 Motion Control IC [3]. Totally developed in Java using Eclipse, here is an example of how developing your own software, avoiding buying those expensive and specific solutions from IC manufacturers. In this case, IR sells IRMCS2011 - Complete Encoder Based Servo Drive Design Platform iMOTION Development System [4]. And since our application is not for the same power application, and the hardware around is not the one we want to use, buying the developing system is an expensive and time consuming luxury that we do not have.

Within this philosophy, controlIT (figure 2) is being currently developed. This tool allows the user to fully control the motor, while monitoring several parameters.

Figure. 3. Splash screen of controlIT 2.0.

This software interface allows controlling the IRMCK201. This IC is capable of executing an advanced field oriented control (FOC) and PWM algorithm loop in less than 4 microseconds, while a high performance programmable DSP-based control loop takes as much as 15 to 20 microseconds [5]. Simplicity is the key word. Normally, all control elements regarding FOC are typically implemented by software code in a motion-control DSP or microcontroller. In a real-time control environment with a DSP and a microcontroller, current-control loops are implemented as high-priority software tasks. Intensive knowledge of real-time control is required if sequential execution of each control element is to be computed within the allowable specified time. The execution of these control elements, often driven by hardware events and interrupts, requires precise sequencing of instruction coding to manipulate hardware at specific times for the motor to be controlled properly. With this IC you can skip the (sometimes) obvious and jump to new innovative things.

In addition, the IRMCK201 is designed to be compatible with the rest of the iMotion chipset, comprising the IR2175 linear current-sensing IC, the IR2136 three-phase inverter-driver IC and the IRAM 6 to 20A integrated power modules.

Interfacing these external parts is very simple, and the final result is very satisfactory.

Having the objective of building a hardware platform that can support tests relating Energy Optimal Control (one of the objectives of this application), it has become necessary to quickly implement motor control with access to the main variables in the motor control algorithm that really intervenes in the energy optimization process.

Understanding the communications and control protocol is a necessary step for building the application. In this example, it is shown this case's protocol (figure 4).

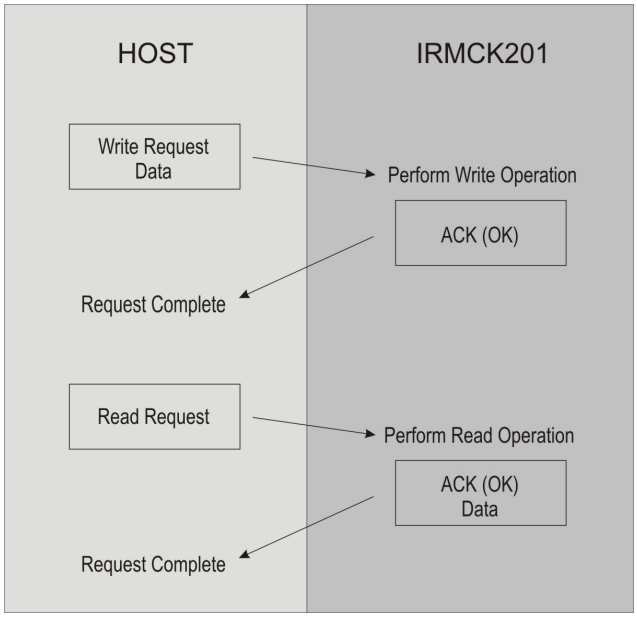

Figure. 4. Write and Read operation exchange, respectively.

Either the write or the read operations are performed with this simple protocol. A Register write operation consists of a command/address byte, byte count, register data and checksum. When the IRMCK201 receives the register data, it validates the checksum, writes the register data, and transmits and acknowledgement to the host. In a similar way a register read operation consists of a command/address byte, byte count and checksum. When the IRMCK201 receives the command, it validates the checksum and transmits the register data to the host. Considering this protocol, it is possible to read/write all the important registers like magnetizing currents, PI gains, among others.

Building a user friendly application that allow us to control the motor, trough the IRMCK201 and the surrounding hardware, is only a few programming lines away.

This application (controlIT) was completely developed in Eclipse very quickly, and it is a good example of how the design engineers can avoid buying those expensive evaluation platforms that usually end up with no application. Due to the object-oriented nature of Java, this application is very customizable, and can be easily converted if a new version of hardware is released or even if the application requirements change (and they will, we are sure).

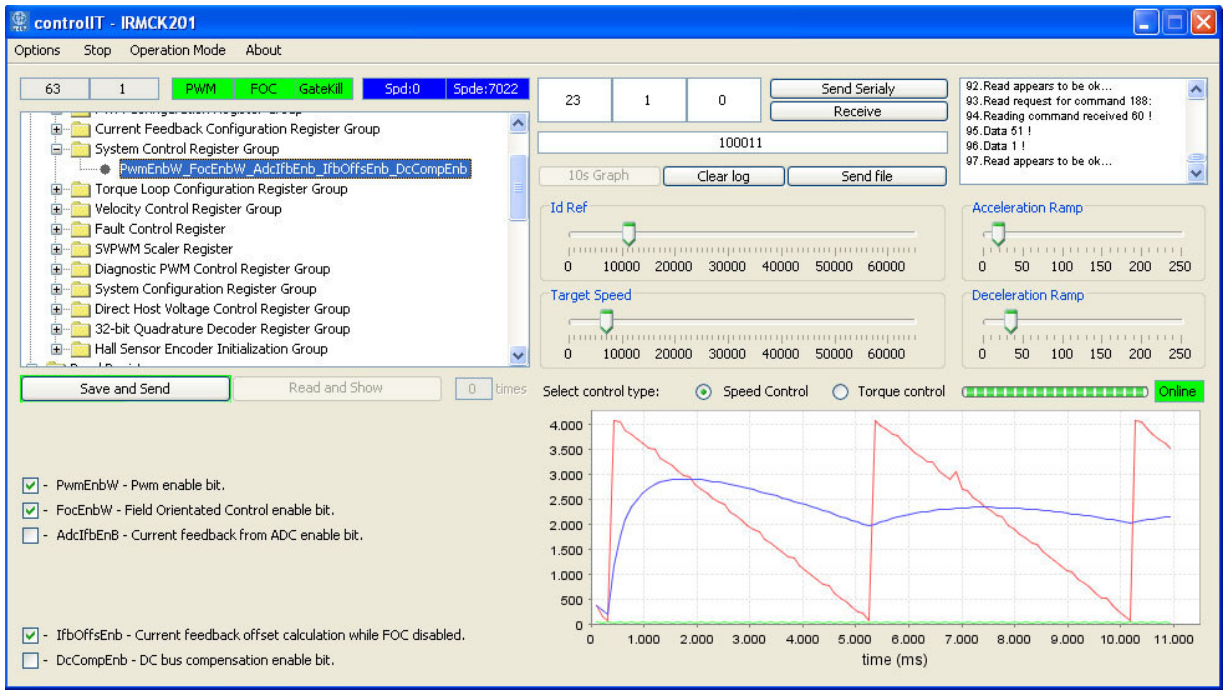

Figure. 5. Screenshot of controlT's GUI in a Windows XP platform. Other OS can be used.

As it can be seen, the GUI is very intuitive and the motor control can be done by a normal or an expert user, by selecting it in the "Operation Mode" menu. In figure 5, expert mode is selected and the user can access all the IRMCK201's registers. It is also possible to see the chart plotting ability of the desired register. Trough sliders it is possible to set the reference input values either in "Speed Control" or in "Torque Control" mode. And at all time, "status", "speed" and "speed error" are monitored in order to provide useful information about the condition of the system, either if everything is running or something is faulty.

Another aspect is the system configuration. It is desirable that this system operates with several motor types. Considering this, an Excel sheet was built that automatically generates a config file that can be sent trough the application, inserting only a few characteristics of all the parts that constitutes the system (pwm frequency, motor characteristics, encoder details, etc.). This function (button) can be seen in figure 5 and it is called "Send file". It works like a preset configuration that is defined in Excel due to the simplicity of formula building and easiness of use. The reason for this mixed solution is again related with the need to minimize time in the development. Excel with all its features allows that everything is done automatically, by just inserting the required information in a few fields and exporting this file that can be imported by controlIT later on. After this, it is just necessary to control the motor, without concerning about configuration, and only intervening on the aspects that are really of interest regarding the application, that in this case is Energy Optimal Control.

IV. Conclusion

The industrial and appliance machine builders must decide between remaining competitive in the marketplace and addressing regulatory requirements for safer, more efficient and yet feature-rich products. From a time-to-market standpoint, the traditional approach of buying off-the-shelf prepackaged motor drives, then using a system integrator to put together the system, is no longer a viable option. And, with the growing complexity of the motor drive, the option of bringing the entire development in-house is also no longer a viable option. It is no longer necessary to spend some much time implementing motor control, if it is possible to use integrated solutions like the one mentioned previously.

With these new tools, motor control has never been so simple and effective. It is now time to start thinking a little bit further. Even if a more complex interface is needed, as it was seen, it is possible to implement software tools in a quick, inexpensive and very satisfactory way.

V. References

[1] Leonhard, Werner, "Control of Electrical Drives", Springer, 3rd ed., 2001, ISBN: 3-540-41820-2.

[2] Peter van der Linden, "Just Java 2", Prentice Hall, June 2004, ISBN: 0131482114.

[3] International Rectifier, "High Performance Configurable Digital AC Servo Control IC", Data Sheet No. PD60224 Rev.B, January 2004.

[4] International Rectifier, "Complete Encoder Based Servo Drive Design Platform iMOTION Development System", Reference Design, Mars 2004.

[5] D. Tam, "Integrated Platform for Motion Control Design", in "Power Electronics Technology, August 2003.