The XBC: a Modern Low-Cost Mobile Robot Controller*

Richard LeGrand[1], Kyle Machulis[2], David P. Miller[2,4], Randy Sargent[3] & Anne Wright[3]
[1]Charmed Labs, [2]KISS Institute for Practical Robotics, [3]NASA Ames/QSS Group, Inc. & [4]Univ. of Oklahoma

***Abstract*— Much of robotics research is carried out using either PICs and processors that are a decade or more out of date The alternative is custom built electronics that is expensive and/or must be reinvented every time a new project is begun. The XBC is a new design for a robot controller merging a modern ARM processor with an FPGA that allows high performance – especially in vision processing and motor control – for a cost similar to controllers with a fraction of its capabilities. Additionally, the XBC uses a new, and still free, software development system, already in wide use. The XBC is being *mass* produced (at least in research hardware terms) so it is readily available and does not require computer hardware or electronics skills in order to be obtained. This paper describes the system, its capabilities and some potential applications.**

***Index Terms*— robot controller, back-emf, color tracking, robot programming environment**

I. The XBC/IC System

For the past half-century, Moore's law has described how general purpose computing has risen in capability while the cost has declined. Along with those changes, advances in operating systems, graphics and user interfaces have lowered the technical barrier for entry to the point where more households in the US have computers than do not [2].

However, this has not been the trend for robotics in the hobbyist and research market. As general purpose machines have advanced, their ability to interface to the physical world in a straightforward manner has often declined. While the number of embedded processors has skyrocketed in recent years [12], the equipment, software and required knowledge for entry into using those processors has also skyrocketed. The robot controllers powering most home-brew robots ten years ago (the Basic STAMP [3] and the Handy Board [7]) are still the controllers for many robots today, and have been displaced in numbers only by the RCX [9] – which while easy to use, has fewer practical capabilities than the systems it displaces.

The XBC/IC system (see Figure 1) is an easy to use low-cost general purpose robot controller. The system provides powerful hardware (an FPGA linked to a commodity ARM processor) combined with the easy to use and very popular Interactive C programming environment. The resulting system has vision, control and interface capabilities that far exceed previous robot controllers for this market. The XBC uses Interactive C, the easy to use C programming environment already used by tens of thousands of robotics researchers, students and hobbyists. Together, the XBC and IC allow easy entry into advanced robotics applications.

*This work was supported in part by the KISS Institute for Practical Robotics

Fig. 1. *The XBC Robot Controller*

II. The XBC Hardware

The XBC's unique hardware uses a Gameboy Advance (GBA) as the main processor. We chose the GBA because of its powerful industry-standard ARM processor, integrated TFT color display, low cost and widespread availability.

The GBA also adds a certain "fun" element to an educational robotics platform. The XBC employs custom robot hardware that plugs into the GBA where the game cartridge is inserted. As expected, this hardware includes connectors, H-bridges and flash memory. The most unique component is an inexpensive but powerful FPGA from Xilinx. The FPGA allows us to create customized peripheral controllers that are specifically useful for robotics by using a palette of logic gates.

The underlying technology for the XBC was first explored in the creation of the Xport [6]. The GBA is an attractive computing platform for robotics, but it is also a proprietary system with no programing or interface documentation for supporting third party hardware. The Xport primarily consists of an FPGA and flash memory. The flash is used to store the GBA's firmware program, and the FPGA is used to implement custom logic peripherals

that are useful for many different applications including robotics. The Xport essentially turns the GBA into a useful embedded system with 4Mbytes of flash memory and 64 I/O signals.

The XBC extends the capabilities of the Xport by adding the following features:

- Four closed-loop back-emf PID motor controllers
- Color vision system capable of recognizing three color models simultaneously and finding connected components, centroid, and moments for multiple targets per color at frame-rate (50 frames/sec)
- 8 analog sensor inputs
- 16 digital sensor inputs
- 4 hobby servo ports
- Onboard battery charger and voltage monitor

A. Back-emf motor control

Existing controllers such as the RCX and Handy Board offer open-loop motor control in the interest of reducing cost. However, if a robot is expected to move accurately within its environment and deal effectively with random factors such as bumps or inclines, closed-loop motor control is immensely useful. Additionally, closed-loop control is a classic robotics problem that provides useful context for an educational platform.

For these reasons, we chose closed-loop motor control for the XBC's motors. However, closed-loop motor control requires some form of motion feedback, which typically results in extra mechanical complexity and cost (e.g. opto-mechanical encoders). But there is another motion feedback method that is often overlooked and doesn't suffer from these drawbacks.

Back-emf feedback is based on the principle that a permanent magnet motor is also a generator producing a voltage that is in direct proportion to its velocity. This voltage, known as back-emf voltage, is observed by periodically removing power from the motor and measuring the resulting voltage with an analog to digital converter [1].

$$V_{back-emf} = K_{back-emf} \frac{d\theta_{motor}}{dt} \qquad (1)$$

Where, $K_{back-emf}$ is the back-emf constant of the motor, θ_{motor} is the motor shaft position, and $V_{back-emf}$ is the resulting voltage generated by the motor, known as the back-emf voltage.

Thus, by measuring the back-emf voltage, the motor velocity can be inferred. Additionally, integrating the back-emf voltage measurements over time yields the motor position.

$$\theta_{motor} = \frac{1}{K_{back-emf}} \int V_{back-emfmeas.} dt \qquad (2)$$

Where, $V_{back-emfmeas.}$ is the measured back-emf voltage.

Integrating the back-emf voltage over time can be easily accomplished by a microprocessor, as we have done with the XBC.

The XBC uses its FPGA to implement a custom four-axis back-emf controller. This controller talks directly to a serial 12-bit A/D converter, which converts back-emf voltages into measurements. To achieve the best signal to noise ratio, the controller takes dozens of back-emf measurements per measurement period per axis and stores them in buffer RAM. At the end of the period, the back-emf measurements are presented to the ARM processor on the GBA. The ARM then reads all of the measurements and calculates the updated motor position for all four axes.

This process takes place 200 times per second. A PID compensator and trajectory generator running on the ARM provide smooth, accurate closed-loop position control at practically no extra cost.

B. Color vision system

To give the robot the ability to detect and track objects in its environment, a custom color vision system was designed for the XBC. Color vision requires relatively modest computational overhead. Additionally, the emergence of low-cost color CMOS imagers with integrated A/D converters has contributed to the affordability and practicality of these systems.

The CMUCam [10] is an example of such a system. It uses a dedicated 8-bit processor to process the camera's stream of pixels. Here, each pixel is passed through a non-temporal filter to determine if the color is within the color model or not. It does this by performing maximum and minimum thresholds on all three components of YUV[1] pixels. It then calculates a centroid of all pixels in the image that fall within the model thresholds. The drawbacks of the CMUCam are its overly simplified color models, which do not handle lighting changes gracefully, and its inability to identify multiple color blobs.

The Cognachrome [5] is a more capable color tracking system. The video hardware on the Cognachrome allows three separate color models to be tracked simultaneously. It uses a Motorola 68332 microprocessor not to compute the centroid of the relevant pixels, but to segment them into connected components. The centroids and moments are then calculated for each resulting blob. The major drawbacks of the Cognachrome is its price (over two thousand dollars), and its size (approximately four by two by ten inches.

The XBC, uses lookup tables inside its FPGA to perform color pixel filtering. One lookup table is used for each of the three color models. The lookup table provides more flexibility when describing the color models. This flexibility allows the user to specify color models in an HSV color space which is much more useful and intuitive than using RGB or YUV. For example, it allows the user to specify a narrow hue range while also specifying a broad luminance range. As a result, the color models are more robust with respect to lighting changes. The FPGA

[1]YUV is an alternate but similar color model to RGB, not to be confused with HSV. It is a common misperception that YUV separates a color into brightness (Y) and chroma information (UV), but in fact YUV is simply a linear reprojection of the red, green and blue values. Unfortunately this means all three values, Y, U and V will change in direct proportion to a pixel's brightness.

also stores a compressed representation of color model membership within an image as run-length sequences. These run-length sequences are retrieved and assembled into blobs by employing a connected components algorithm running on the ARM processor. This algorithm calculates both the centroid of each blob, its mass (size) and moments (major/minor axis and angle). Thus the XBC by leveraging the FPGA with a very modest micro-processor is able to surpass the performance of the Cognachrome at less than a fifth of its price.

III. The Interactive C Programming Environment

Interactive C (IC) is a version of the C programming language made specifically for robotics controllers and computer programming education. It contains all of the basic concepts (arrays, variables, pointers, structures, etc.) and control structures of C (including if/else blocks, for loops and while loops). To reduce programming errors commonly made by users new to the C language, IC removes all pointer arithmetic and integrates run-time array bounds checking into its execution system.

Interactive C was originally developed in 1991 to support the MIT 6.270 robot contest [8]. In 1993, a simple GUI was added to the system to aid in editing and downloading. In 1996 a commercial version of IC was released by Newton Labs. In 2002 KIPR released a free version of IC complete with IDE. Version 5 of IC by KIPR is designed to support many processors, including the XBC. Its modular library design allows support of new boards without changing the core system.

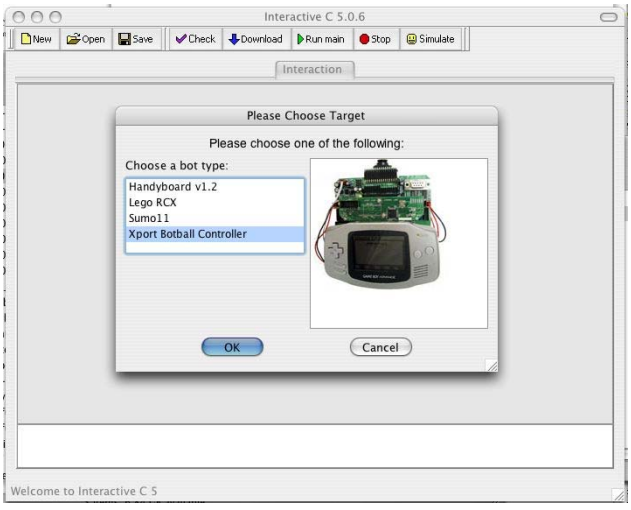

Fig. 2. *IC5 Target Processor Selection*

The root of IC's portability lies in the way it is compiled and executed. Instead of being compiled directly to native machine code, it is compiled to a set of instructions known as pseudo-code (or p-code), which runs on a specialized stack machine. The p-code is interpreted by firmware which implements the stack machine specifically for the controller it is running on. By having this extra layer of code between IC and the controller boards, it is possible for the same IC program to compile and run on many different controller boards (see Figure 2) with little to no changes needed by the user. Since the execution is handled on a layer above the processor, features like bounds checking and graceful error handling are available.

Multithreading is also integrated directly into the language. Because the pseudo-code is fully stack-based, a process's state is defined solely by its stack and program counter. Thus it is easy to task-switch simply by loading a new stack pointer and program counter. This task-switching is handled by the runtime module.

The multithreading and p-code interpreter allow IC to be interactive. C statements or blocks may be entered into the interface and will be evaluated immediately on the processor. This allows function calls and simple loops to be tested before being integrated into a program. This is a tremendous boon to the novice and experienced programmer alike, yielding almost instantaneous feedback and allowing for quick tests of hardware hookups, motor polarity, etc. Thanks to this feature, data can be stored on the board and retrieved for analysis on computers using spreadsheets or scientific software. Users can also test portions of their downloaded programs without having to write test stubs into their code.

As of version 5, IC now includes a pseudo-code simulator. This is an implementation of the stack machine built into the IC compiler that lets users run their code on the same machine IC is running on. It reduces the need to have a controller board present in order to test code (for applications which do not require loops to be tightly closed in the real world). The simulator can be configured to have the same features as the controller board it is simulating – including print types, numbers of analog/digital sensors, and special features of each board (buttons, knobs and other non-standard parts).

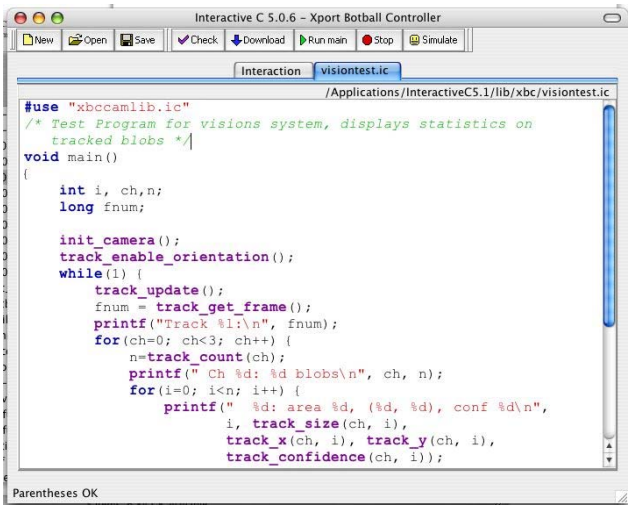

Fig. 3. *The IC5 GUI with Edit Pane*

Interactive C is distributed as both a command line utility, and a full featured GUI. The GUI (see Figure 3)

is designed with the beginning programmer in mind by providing syntax highlighting and coloring, inline function descriptors, bracket/parenthetical checking, error highlighting and help features. It is distributed for free by the KISS Institute.

When combined with the LCD screen and control system of the XBC, the Interactive C firmware provides a full on-board control and status system to users. IC programs can be stopped and started (and in the simulator, paused) using the control system available on the board. A status bar provides navigation and usage instructions as well as status messages about the state of the firmware and the bot itself. It updates multiple times per second with information about sensors, motor position and speed (see Figure 4), servo status and battery power. Icons on the screen alert the user to the status of IC programs currently loaded, whether a program is running on the bot, and the state of any communications between the controller and the PC.

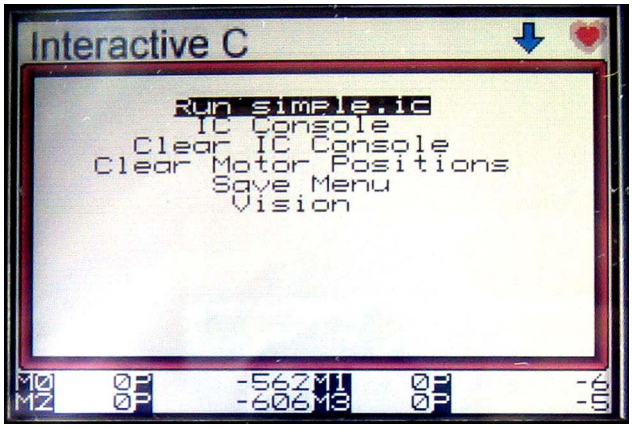

Fig. 4. *The IC Interface on the XBC with Motor Status*

The Interactive C print buffer has more capabilities due to the screen size and video memory of the GBA. Since users are no longer restricted to a 32 (or fewer) character LCD typically found on robot controllers, they can create full multi-page menu GUIs within the Interactive C language, increasing the levels of control and depth their programs can have.

The IC firmware on the XBC allows access and control of the color tracking system. A GUI that runs directly on the XBC displays live video, allows color selection from an HSV palette (see Figure 5 and 6) and can display processed video and/or sprites indicating blob positions and size.

IV. Unique Features of the XBC/IC

The XBC/IC system has several features that make it unique among low-priced robot controllers.

A. FPGA for handling sensors, motor control and vision pre-processing

Many robot designs employ multiple peripheral processors to distribute the typically large computational burden across multiple processors. For example, dedicated processors are often used by robots to perform motor control

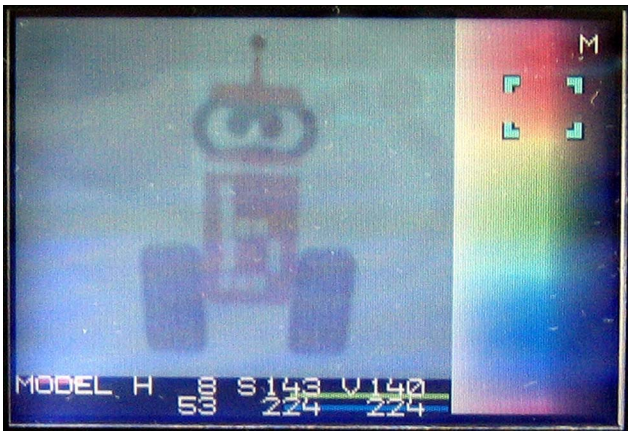

Fig. 5. *The Color Selection Palette with Live Video*

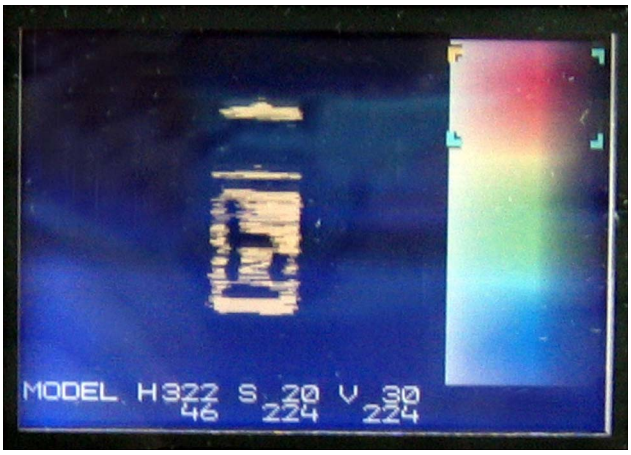

Fig. 6. *The Color Selection Palette with Processed Video*

and image processing. The disadvantage to this approach is that multiple processors are expensive and consume large amounts of power. Furthermore, a typical peripheral processor often uses slow serial communication between it and the main processor. This can create data latency, which can adversely affect system performance.

The XBC is tightly coupled with each peripheral implemented in the FPGA and the ARM processor (GBA). This makes hardware/software codesign possible. Both the back-emf controller and the color vision controller were designed using hardware/software codesign techniques. The main premise for this type of design is recognizing that peripheral hardware can perform some tasks more quickly and more inexpensively than processors (running software) can.

By distributing time-critical tasks across hardware peripherals on the FPGA, approximately 150 million operations per second (MIPS) required by the color vision system and motor controller are offloaded from the GBA's main processor. The GBA's processor is only capable of approximately 10 MIPS. Thus, the XBC's FPGA eliminates the need to move to a faster CPU or multiple-processor design. The resulting cost and power savings is significant.

B. Built in Camera

The built in camera provides video and color tracking features in a compact form factor. An extension cable can be used to reorient the camera as needed.

C. Game Boy as processor

It would simply be impossible to design and manufacture a computing platform that provides similar functionality to the GBA's color LCD, ARM processor graphics hardware and PCM sound at a similar price. Currently, the GBA is available used through gaming retailers for only $29. This has created a unique opportunity that we have taken advantage of by integrating the GBA into the XBC.

D. Ease of Programming

The IC5 XBC firmware, extensive library of robot functions and IC IDE make programming straightforward. No intimate knowledge of the hardware is needed to access sensors or motors.

V. Summary & Application Areas for the XBC/IC System

The XBC was initially designed for the Botball Educational Robotics Program [4], [11]. XBC stands for Xport Botball Controller. The XBC is currently one of two processors used in the Botball Program. As part of the standardized Botball kit, the XBC is distributed to teachers across the country and used by students to create original robots to participate in regional and national Botball tournaments. Educators, mentors and students participate in regional hands-on workshops and learn to program the XBC using IC. Because the XBC is so powerful and versatile, teachers are encouraged to use this controller in a myriad of other ways as part of classes and extracurricular activities throughout the year. In spring of 2005, approximately 2500 students distributed among 300 middle and high schools will be working with the XBC; programming it using Interactive C. Their initial task (as part of the Botball program) is to make a small autonomous robot that maneuvers through a series of obstacles to locate, acquire, sort and place a number of colored objects distributed across and above the game board.

While the Botball task is a game, the techniques used to play it have many applications. Color tracking and vision-based manipulation are a staple of many industrial and assembly processes. Servo control of motors for positioning is used in robot systems from *Roomba* to the *MER* rovers. The XBC is already finding its way into university research labs as a quick and cost effective way to prototype capable robot systems.

The XBC's powerful combination of features is unmatched by *any* other integrated robot controller available today at any price. It is a board that brings together many of the features currently available to researchers only through custom PC-104 boards or similar ad-hoc systems. Vision systems and accurate motor control systems have been developed in academic labs or in large industrial centers for years, but never made it into the mass market, available as a tool for both research and education. With the combination of XBC and Interactive C, academic institutions and home users can experiment and learn with a platform that can also be used in professional and scientific grade applications. With students still in middle school being able to harness the power of the robot throughout their education, their abilities once they reach college level will be higher than ever before. The XBC takes advantage of existing consumer and commodity hardware to create a cost effective robotics solution that inspires as well as motivates learning. The XBC is an innovation to spark innovation.

References

[1] Electro Craft Corp. *DC Motors, Speed Controls, Servo Systems: an Engineering Handbook*. Electro-Craft Corp, 5th edition edition, 1980.

[2] Martin Crutsinger. Computers in half of u.s. homes. *Washington Post*, page E17, October 17 2000.

[3] Parallax Inc. Basic stamps. `http://www.parallax.com/html_pages/products/basicstamps/basic_stamps.as%p`, 2005.

[4] KIPR. Botball robotics education. `http://www.botball.org`, 2005.

[5] Newton Research Labs. Cognachrome vision system. `http://www.newtonlabs.com/cognachrome/index.html`, 2001.

[6] Richard LeGrand. Xport 2.0 user guide. Technical report, Charmed Labs, 2003.

[7] F. Martin. The handy board. `http://www.handyboard.com/techdocs/`, 2002.

[8] 6.270 Organizers. 6.270 - mit's autonomous robot design competition. `http://web.mit.edu/6.270/`, 2005.

[9] Kekoa Proudfoot. Reverse engineering the lego rcx. `http://graphics.stanford.edu/~kekoa/rcx/talk/`, 1998.

[10] Anthony Rowe, Chuck Rosenberg, and Illah Nourbakhsh. A low cost embedded color vision system. In *Proceedings of IROS 2002*. IEEE Press, 2002.

[11] C. Stein. Botball: Autonomous students engineering autonomous robots. *Computers in Education Journal*, (2), June 2003.

[12] F. Vahid and T. Givargis. *Embedded System Design A Unified Hardware/Software Introduction*. John Wiley & Sons, 2002.