

AVR Robot Controller 1.1 Sample Code

Introduction

The following series of programs contain elements needed to successfully control various aspects of the ARC 1.0 board. Included are elements of libraries to implement common functions as well as a sophisticated example of a precision foreground/background multi-tasking code.

All programs are written to the BASCOM compiler and compile on the free demo version available from <http://www.mcselec.com>. All have been tested on the ARC 1.1 board, which is described at <http://www.barello.net/ARC>.

Support for using the ARC 1.0 board and any sample code can be had at <http://www.yahogroups.com/groups/AVRRobotControl>

Background information

CPU Setup

The CPU supplied, the ATmega163, has a variety of internal options that must be set properly so it runs the ARC 1.1 sample code. These options are set prior to shipping, but may need to be reset if you purchase an upgrade CPU. You can set these options using the SAMPLE programmer that comes with BASCOM. The options are:

1. 4.0v BOD
2. BOD Enabled
3. External ceramic resonator with BOD enabled.

A basic understanding of the BASCOM language and the underlying peripheral hardware of the AVR processor are essential to successfully writing high performance programs. At a minimum it will prevent hours of frustrating experimentation trying to make things work. Here is a suggested reading list

1. BASCOM Help: *Language Fundamentals*
In particular pay attention to *Casting*. An easy mistake to make is mixing data types and getting unexpected arithmetic results.
2. BASCOM Help *Newbie Problems* has a very quick description of how the AVR I/O ports work.
3. In general, when looking at a *CONFIG XXX* command (say, **config timer0**), find the corresponding section about internal hardware in the BASCOM help files.
4. Get a reference manual from www.atmel.com/atmel/products/prod200.htm for the AT90S8535. It will be a large PDF file. Alternatively find out who your local rep is and get a bound version or a complete image of the Atmel web site on a CD-rom.
5. Most programs use the PRINT statement to print debug information to a terminal program off the serial port of the board. The default code is set to 19.2 Kbaud.

AVR Robot Controller 1.1 Sample Code

Libraries

The following five files (ARC_10XXX.bas) are not complete programs but are libraries of useful functions. They are used by inserting a **\$include** command at the beginning or end of your main program file. NB: Any compiler directives like **\$regfile =** needs to be in your MAIN program file at the very beginning before any **\$include** directives.

ARC_10.bas

This is not a program, but an **\$include** file that defines all the hardware bits of the ARC 1.0 board. This file also declares any functions or subroutines defined in the other library files, below. This file is included in the main file near the top, before any BASIC code.

ARC_10_Encoder.bas

This **\$include** file uses interrupts to implement a high speed 2x decoding of quadrature inputs. Please refer to the comments or www.barello.net/Papers/Motion_control for more information.

ARC_10_polled_encoder.bas

This **\$include** file uses polling to implement a full 4x decoding of quadrature input. NB: the polling rate must be 2x your maximum edge rate of the encoder. Edge rate is ~4x the slots * revolutions/second. I say approximately because with home made encoders the edges might be closer or farther apart as the encoder revolves if it is not precisely made.

Another way to say this is that the polling period must be $\frac{1}{2}$ the minimum period between two edges.

ARC_10_PWM.bas

This **\$include** file contains two routines for setting PWM values in the left and right motors. The code takes care of the tricky parts like inverting the PWM signal when in reverse. Pay careful attention to the bit length needed when initializing the PWM hardware. This file needs to be included **after** the main BASIC code loop.

ARC_10_servo.bas

This **\$include** file contains the library routines for outputting the servo pulse. The actual routines are encased in a critical section (interrupts off) so that other interrupt activity will not affect their timing. This does, however, shut down the CPU for the duration of the pulse, which can last as long as 2 ms. ARC_10_servo.bas needs to be included **after** the main BASIC code loop.

AVR Robot Controller 1.1 Sample Code

Sample Programs

ADC.bas

This program configures the Analog to Digital converter hardware, reads the right and left sharp ports and prints the results to the serial interface

ADCtoPWM.pwm

Similar to the program ADCtoServo.bas, this one sets the PWM output based upon the ADC input. It illustrates, among other things, how to set alternate PWM control sizes and the math needed to convert 10-bit ADC to 8 & 9 bit values.

ADCtoServo.bas

A slightly more sophisticated program that uses the library functions and the various hardware aliases to translate ADC inputs to servo outputs. It is nice for checking out sensors as the input voltage of 0-5v is translated to servo rotation.

Encoder.bas

This example uses ARC_10_polled_encoder.bas to read and display the quadrature encoder inputs of the ARC 1.0 board.

FlashLed.bas

A very simple program that simply blinks the program LED on and off. I/O ports are configured directly in this case

FlashLedAlias.bas

This is the same as FlashLed.bas with the additional use of the ALIAS command to make code more readable.

Minisumo.bas

This is a complete, mini-sumo application that uses the PWM output to drive motors (not servos). The algorithm is very simple: there is a loop that checks the floor and proximity sensors. If the floor sensor detects a white boundary, the robot backs up, turns away and proceeds forward. If one of the proximity sensors detects something, the robot turns towards it. Modulating the PWM drive for fixed amounts of time controls turning.

Minisumo_servo.bas

This is the same as Minisumo.bas with the exception that it uses R/C pulse signals to control modified R/C servos rather than PWM and gear-head motors.

AVR Robot Controller 1.1 Sample Code

Multi_task_sumo.bas

This is a complete mini-sumo application that is quite sophisticated compared to the previous two examples. It implements seven tasks that are layered in priority where the highest priority task always takes precedence. This example uses rear proximity sensors to detect targets behind the robot and take evasive action.

Pid_example.bas

This sample implements a basic PID loop algorithm. When combined with quadrature encoder input and PWM output it can control shaft position. This is a common and very effective technique to servo an output to a control signal. This sample does not have any input, it just servo's the output to hold the zero position of the encoder.

Another interesting thing about this sample is that it implements a crude form of memory structure so that the same PID code can be re-used for any number of control loops. Each control loop has it's own piece of structured memory that is used to maintain control.

Polled_encoder.bas

This is an example of how to use the polled quadrature encoder library.

Serial_hello_world.bas

A simple program to print out "Hello World" and flash the LED

Sophisticated.bas

This program is a collection of a variety of advanced topics. There is an interrupt handler that manages the winking light in the background and a foreground timer used to regulate how often the ADC is read and servo and PWM values are set.

Most foreground loop, interrupt handler applications have a variable foreground loop rate. The rate is dependent upon how much time is spent in the interrupt handlers. This application shows how to use a high-speed interrupt handler to control the timing of the foreground loop, thus implementing precision timing in both the foreground and background tasks.

Switch.bas

This simple program reads an input bit and sets an output bit.

SwitchAlias.bas

This is a modified version of Switch.bas that uses ARC_10.BAS include file to make the code smaller and more readable. Note that ARC_10.bas replaces the various \$ directives found at the beginning of the previous examples.