YAHOO! MAIL Classic

CHIPS Newsletter Vol 9 Thursday, October 7, 2010 11:08 PM

From: "chips@elproducts.net" <chips@elproducts.net>
To: "Chuck Hellebuyck" <chuck@elproducts.com>

CHIPs newsletter

Vol 9 / October 7, 2010

Official Newsletter of *Chuck Hellebuyck's* Electronic Products

http://www.elproducts.com/

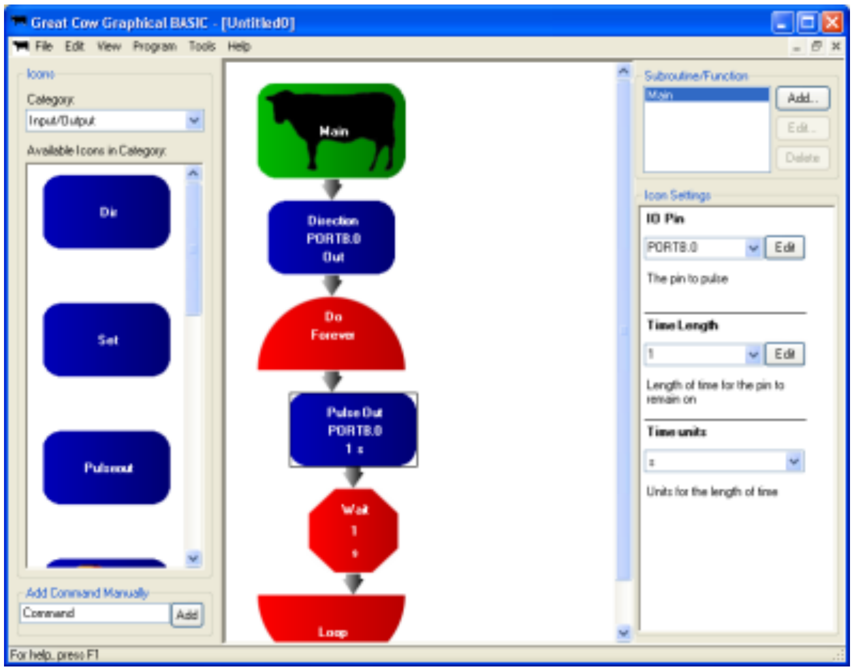
Seems like I just wrote a newsletter.... Oh wait I did.
I agreed to help write some of the documentation for the CHIPINO module and during that time I was re-introduced to several programming options that work well with CHIPINO. I'll show you those along with some updates on what I'm working on. First off lets look at some graphical compiler options for CHIPINO.

Great Cow Graphical Basic

This compiler started out as a text based compiler and continues with that theme today but added another optional graphics layer on top of that. I find this graphical interface very interesting. The documentation is limited but after a little bit of trial and error, I was able to get many projects running on a CHIPINO board using this software. You can easily switch between graphical mode and text mode so that is a really nice feature. There is a great forum on this compiler and many users so help is out there. The best part is this compiler is completely free to use. No code limitations or up sell to a pro version. What a great gift to the hobbyist this compiler is.

The site lists the main developer as Hugh Considine. He's active on the forum and has answered emails to me quickly. I want to publically thank Hugh for a great compiler.

The syntax of Great Cow BASIC is based on that of QBASIC/FreeBASIC, though I find its very similar to PBASIC and PICBASIC as well. How much can you change embedded BASIC for PICs is the point I guess. Great Cow BASIC will allow you to program most 8 bit PIC microcontrollers including 10F, 12F, 16F, and 18F chips. The graphical basic IDE allows you to easily use a PICkit 2 programmer with the software so programming the CHIPAXE or CHIPINO modules is easy.



Category: Input/Output

IO Pin: PORTB.0
The pin to pulse

Time Length: 1
Length of time for the pin to remain on

Time units: s
Units for the length of time

Great Cow Graphical BASIC screen shot

Here are some of the features listed on their website for this compiler:

- Standard BASIC flow control statements - If, Select Case, Do, For, Goto
- Support for multiply, divide, add, subtract, boolean operations and comparisons.
- Bit, Byte, Word and String data types, in addition to byte arrays.
- Subroutines and Functions
- Inline assembly, in most cases without any special directives
- Data tables
- Automatically recalculates all delay commands depending on the clock speed of the chip
- Generates standard MPASM compatible assembly code for PIC

I/O capabilities:

- Standard 2x16 LCD routines
- Routines for on-chip A/D, PWM, SPI, USART, EEPROM and Timers
- RS232 communications - rates between 300 and 19200 bps with user configurable parity, start and stop bits.
- PS/2 keyboard reading
- 4x4 Keypad

Notice the feature that allows you to write functions. I don't know of many BASIC compilers that allow that. With this feature you can easily create your own custom commands.

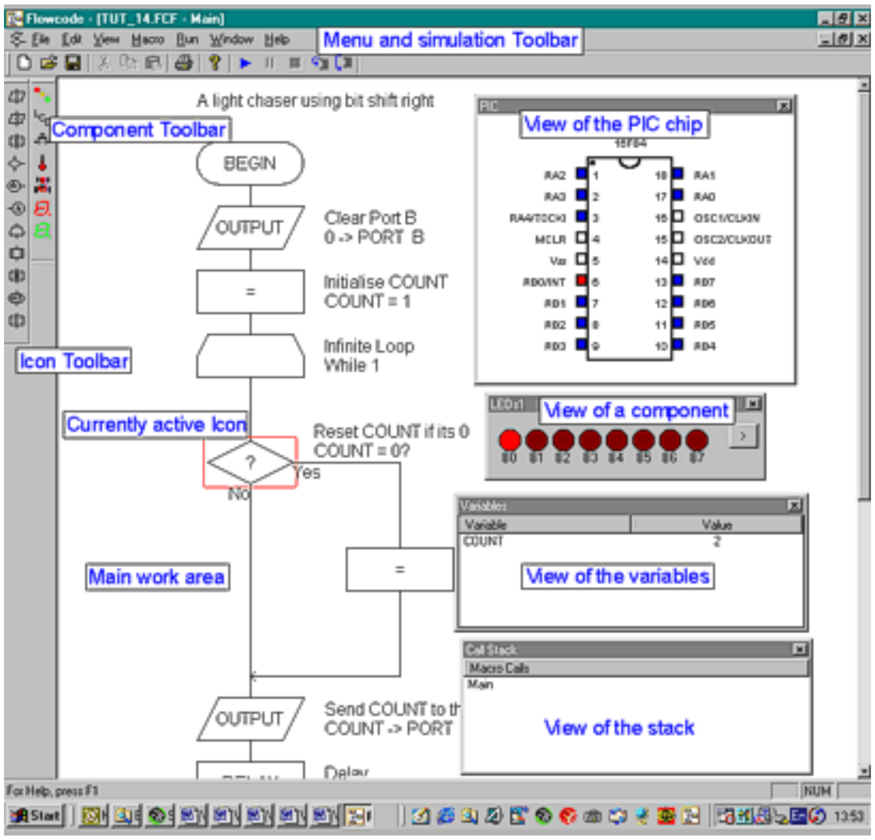
Another great feature of this compiler appears during installation of the compiler. Near the end of the installation it offers you the opportunity to install it on a flash drive instead of your computer's hard drive. This gives you the opportunity to easily create a portable version of this compiler. If you save all your files on the flash drive, then you can take your GCGB with you and work on any computer. How handy is that?!!!

I'm still learning this compiler but I know the Chipaxe guys are seriously looking at switching from PICBASIC PRO to this compiler which puts my books out on a limb but I'm also working on converting some of my PICBASIC books to this compiler. Stay tuned for more on this.
Visit the Great Cow Basic website at:
http://gcbasic.sourceforge.net/index.html

Flowcode
If you would rather write C code in graphical flow chart then consider the flowcode compiler from Matrix Multimedia. It's

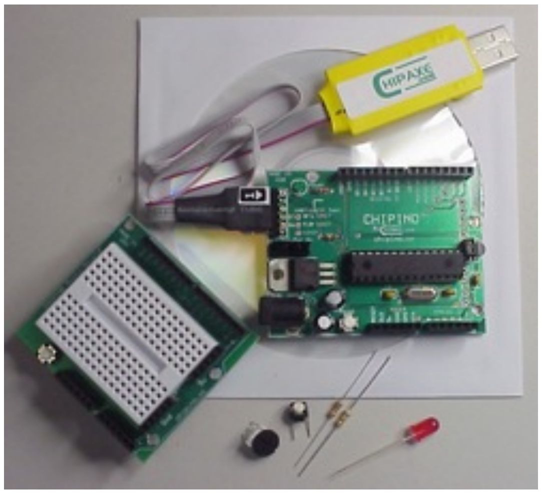
not free but you can get a low cost version if you are a hobbyist or student. There is also a sample version that limits memory and supports only a few parts but this will allow you to test it out.

Flowcode is a nice front end to the BoostC compiler that does all the compiler work. Flowcode can also be setup to use the HI-TECH C compiler rather than BoostC. Flowcode can easily work with the PICkit 2 programmer so once again the Chipaxe and CHIPINO will work well with this. They also offer their own breadboard style modules that use a bootloader. One of the nice features Flowcode offers is a built in graphical simulator. With this you can create your project without any hardware to prove it out. I've tried Flowcode but found it a little more difficult to use than Great Cow Basic but it's also more feature rich. I know of one person developing CHIPINO projects with this compiler so you may see sample code someday at the CHIPINO.cc website.



Flowcode Screen

CHIPINO Update
This leads me into an update on my new favorite development platform, the CHIPINO. The team launched a central site at chipino.cc to create a central location for information. Chipaxe.com now has product in stock and ready to use. The first official CHIPINO shield is also available called the CHIPINO Proto-Shield. I've had a few people ask if they can use an Arduino shield with the CHIPINO and the answer is yes.


Deluxe CHIPINO Starter Kit

The CHIPINO is directly compatible with most Arduino shields. The ADC, Tx/Rx, SPI, PWMs are all lined up on the CHIPINO to match the Arduino. The board outline, mounting holes and connector headers all match the Arduino perfectly. The only areas where they differ are the I2C pins are different locations and the Arduino has a few extra PWMs. This is due to the differences between the Microchip PIC used in CHIPINO and the Atmel part used in Arduino. Therefore it's not 100% compatible but darn close. Therefore you can use many of the Arduino shields with the CHIPINO.

What I think gets missed with the few people I've talked about this is the CHIPINO uses a direct programming method rather than a bootloader like Arduino. It also uses the MPLAB IDE to handle all the interface. What this gives the user is lots of options to work with. The IDE is not only a free software interface to write your code and program with a one click option but you can also debug your code by single stepping through it command by command. You can also simulate your code with the MPLAB simulator and then move to the CHIPINO hardware. CHIPINO bridges the gap between hobbyist and professional by making it easy to get started but also allows the user to grow with the development and not have to switch hardware. You see MPLAB is the same IDE used by professional Microchip PIC developers.

The CHIPINO pricing has been set and the SimpleC library is ready for the initial release. I have to get cranking on my Volume 3 book which shows how I built the SimpleC library and how a reader can expand it with their own functions.

SimpleC Library Update
As I mentioned the library is far enough along to be useful so I released it to the Chipaxe guys to include on the included CHIPINO CD. I took liberties with the C language rules to make the library with the interest of making it easier for someone to learn and contribute. It's an un-written rule in C to not include C code inside of a header file. The SimpleC library directly violates this as the whole library is made up of functions packed in a header file. What I found was this worked really well at simplifying the process of including pre-written functions and also made it easy to see the code for reference. It has some drawbacks as far as optimization and debugging as neither works on code in a header file but the idea is to make it simpler for beginners so I made the decision and its working really well. I know I'll get bashed for that decision but not by any beginner.

Here are the current functions in the SimpleC library:

Digital I/O

- high(*pin no.*)
- low(*pin no.*)
- *var* = input(*pin no.*)

Analog I/O

- adcin(*pin no.*)
- pulseout(*pin no.*, *period*)

Delay (val = 1 to 65535)

- pause (*val*)
- pauseus (*val*)

Hardware PWM (1khz to 32khz)

- pwmout1(*duty cycle,freq*)
- pwmout2(*duty cycle,freq*)

Hardware SPI

- spi_init( )
- spi_out(*byte*)
- wr_byte(*address,data*)
- var = rd_byte(*address*)
- wr_MCP23S08(*register,data*)
- *var* = rd_MCP23S08

Hardware I2C

- i2c_init( )
- i2c_start( )
- i2c_repstart( )
- i2c_stop( )
- *var* = i2c_read( )
- i2c_waitforidle( )
- i2c_write(*data*)
- i2c_eeout(*address,data*)
- *var* = i2c_eein(*address*)

Hardware USART
(2400 or 9600 baud)

- init_serial(*baud*)
- putch(*char*)
- *var* = getch( )

LCD (2x16)

- lcd_init( )
- lcd_clear( )
- lcd_goto(*position*)
- lcd_text("*characters*")
- lcd_symbol(*char*)
- lcd_number(*3digit int*)

Conclusion

This newsletter covered a lot of topics and since you should have received two newsletters in a short time, you have a lot to review. If you have any questions or comments send them to me at chuck@elproducts.com