**Cùng Duyphi bắt đầu học lập trình AVR**

Vâng! đây là chuyên mục mà Phi tôi mong muốn được chia xẻ với anh chị em,với những người chưa làm việc trên dòng vi điều khiển này - hôm nay muốn nghiên cứu về nó thì hãy cùng DUYPHI tôi bắt tay vào nghiên cứu và từng bước tiến hành thí nghiệm lập trình và thiết kế một số ví dụ, ứng dụng từng bước,từng bước 1.  
**Nội dung:**  
1- Xoay quanh nghiên cứu AVR với anh chị em từ những cái nho nhỏ nhất  
2- Tiến hành chọn lựa 1 vài con vi điều khiển đặc trưng để thí nghiệm  
3- Bắt đầu viết chương trình ứng dụng từng bước từ dễ đến khó dần.  
  
Đây là lần đầu tiên DuyPhi tôi tiếp cận AVR, nên xem như chúng ta bắt đầu từ BIT 0,hi hi... còn anh em nào đã thành thạo về AVR thì DuyPhi tôi xin thật lòng lắng nghe những lời chỉ giáo góp ý.  
  
**Chân thành cảm ơn:**  
- Các bạn trong diễn đàn đã quan tâm tham gia vào chuyên mục này  
- Chân thành cảm ơn các bạn - các anh chị em đã nhiệt tình góp ý đóng góp ý kiến cho tôi trên bước đường tiếp cận với AVR.

*Rất ủng hộ anh Phi cái mục này. Để làm việc với AVR thì đầu tiên cần có Kit nạp cho AVR, em đang tìm hiểu và tìm kiếm để làm cái mạch nạp giao tiếp USB với máy tính cho nó tiện lợi. Anh Phi có làm cái này chưa, nếu làm rồi thì post lên cho anh em tham khảo nha.*

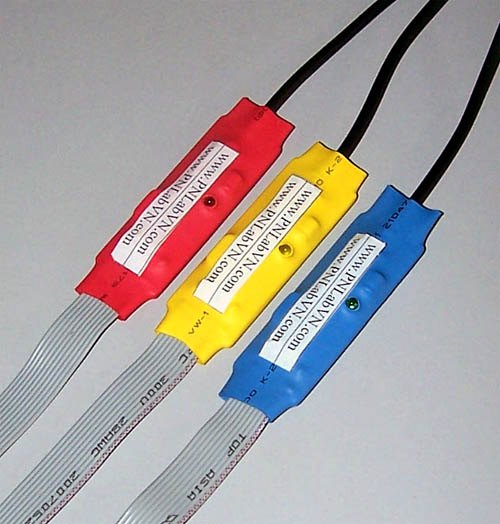
Cái này anh mới tậu được cái AVR910- USB, nếu TIN có rảnh thì ghé qua chỗ của PHI- Phi đưa cho 1 số tư liệu rồi giúp Phi nhân bản cái mạch nạp USB này ra, giúp anh em có cái để dùng, mọi chi phí khác Phi sẽ tài trợ, chẳn hạn vẽ ra 100 bo mạch rồi tặng anh em nào muốn học về AVR thì mình share cho họ, để không tốn thời gian làm mạch nữa! TIN thấy thế nào??!!

*Nếu bạn là người mới bắt đầu làm quen với dòng AVR thì coi như mình xin đóng góp ý kiến vì mình có làm về dòng này được 2 năm rồi.Mình bắt đầu với 1 starter kit ATmega8, development tool là****AVR studio****và tool nạp SPI dùng ctrinh****pony prog****.2 cái này bạn đều có thể tải trên mạng.Bạn có thể dùng tài liệu này để bắt đầu lập trình (*[*http://www.avr-asm-download.de/beginner\_en.pdf*](http://www.avr-asm-download.de/beginner_en.pdf)*) . Theo ngu ý của mình thì khi mới bắt đầu 1 dòng VDK mới, việc lập trình bằng asm giúp bạn hiểu sâu về cấu trúc của nó, điều này thuận lợi cho bạn sau khi chuyển sang lập trình bằng C.  
  
Về startkit thì bạn có thể dùng các ngoại vi cơ bản: button, display led, Rs232, LCD mình nghĩ là đủ cho các bạn hiểu về nó.  
Đôi dòng góp ý!*

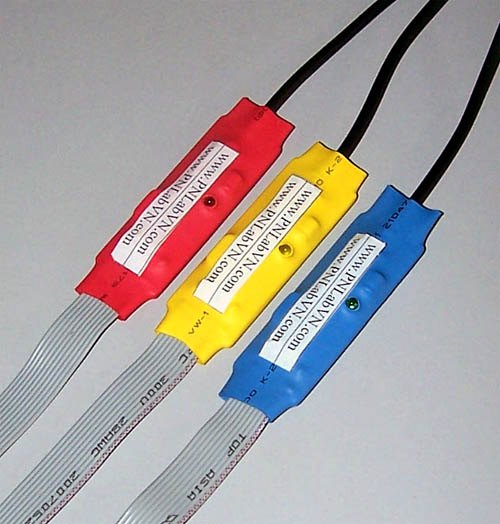
Vâng, DuyPhi xin ghi nhận.  
Để bắt đầu học về AVR Phi cũng bắt tay với lập trình hợp ngữ, vì thấy lập trình C ngay vào AVR thì quá dễ, e rằng mất đi cái hay của nó, nên dùng ASM để bắt đầu.

*E cũng muốn học cái này nè, nhưng đang tìm bộ KIT, vừa rồi vào trang của 1 thầy ở trường giới thiệu thấy cái Easy...gì đó giá...1,2tr, hoảng. Thầy cho email của 1 anh nào đó để nhờ làm giúp nhưng dạo này kinh phí e hơi hẻo nên cũng chưa liên lạc nữa. Ko có bộ KIT thì chẳng học hành gì được.*

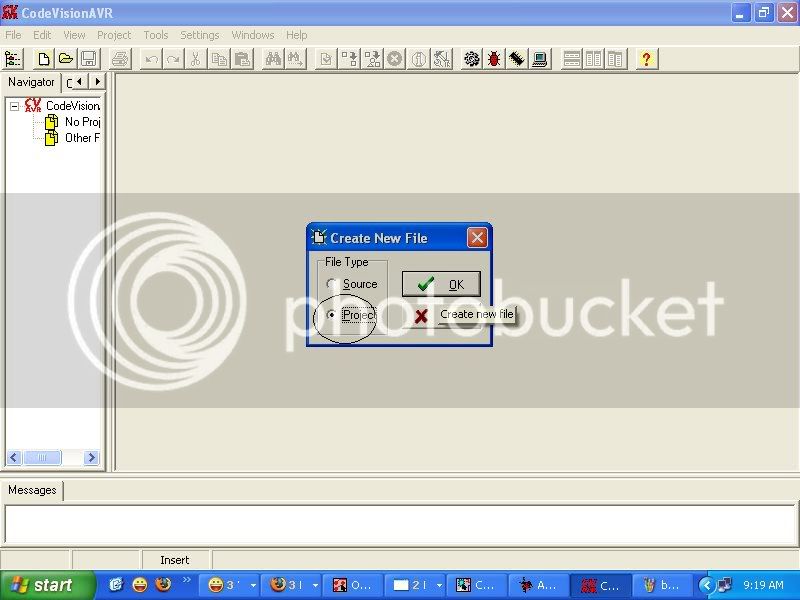
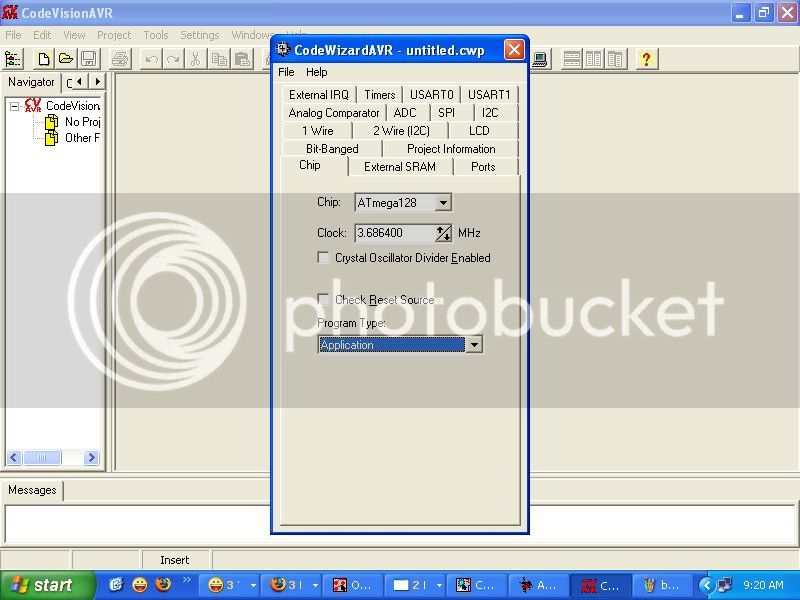
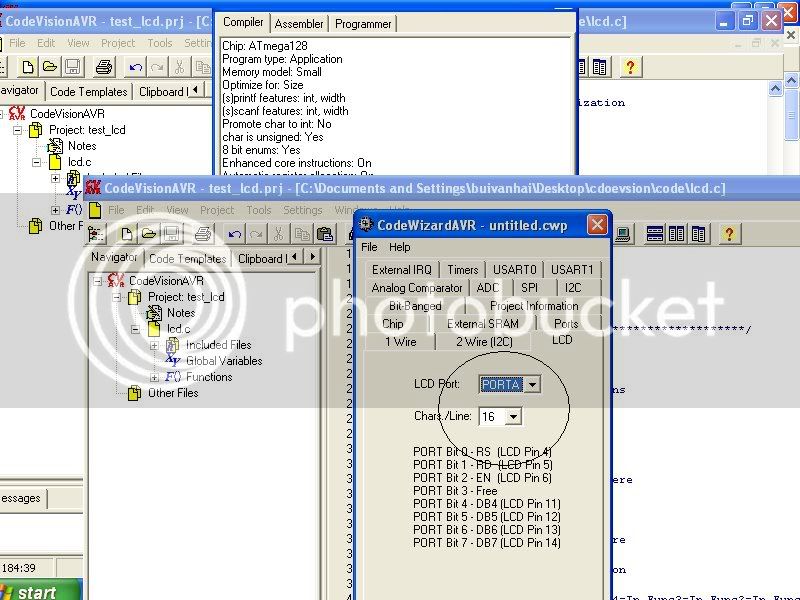
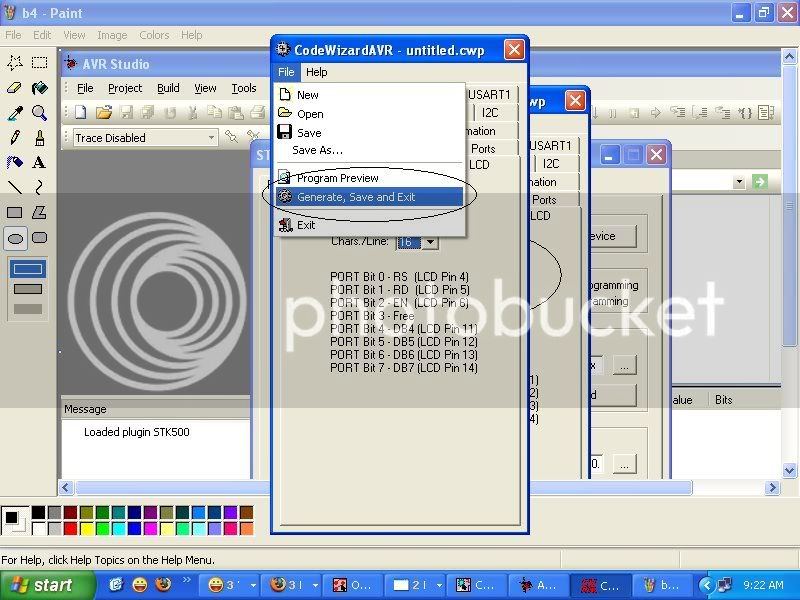
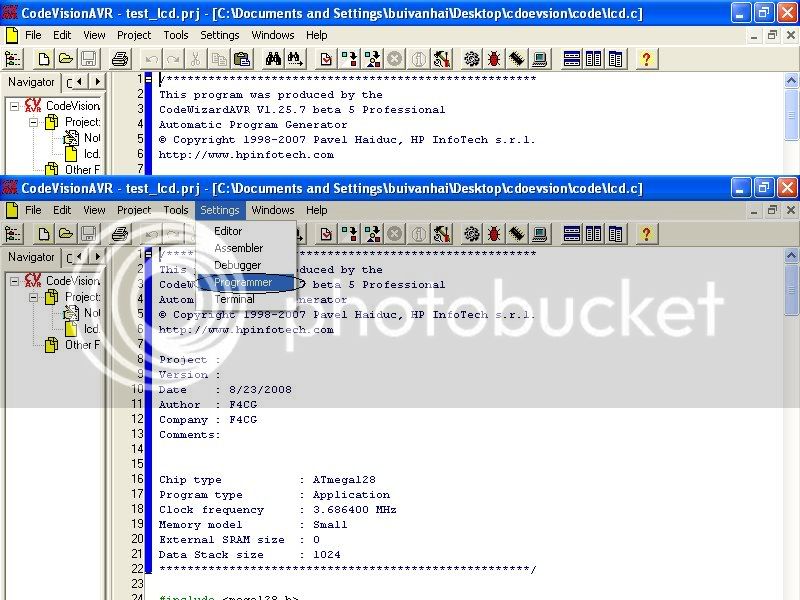
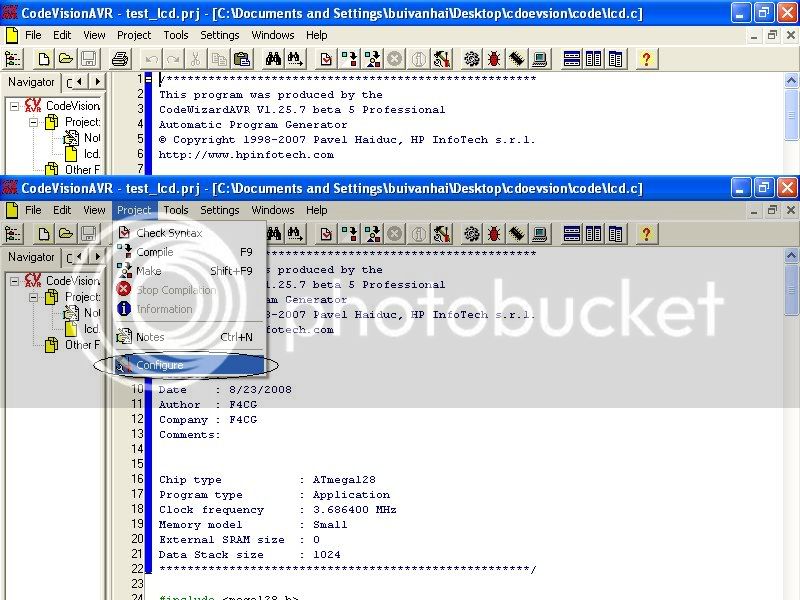
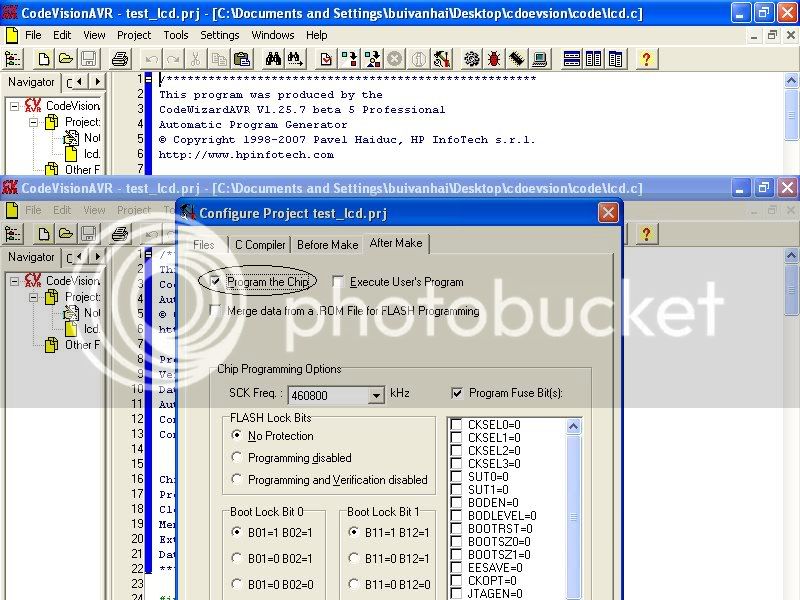
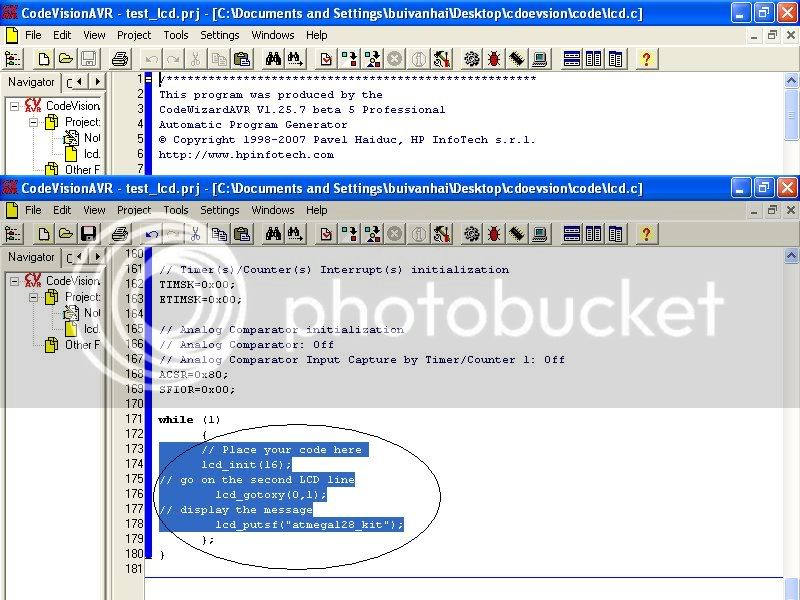
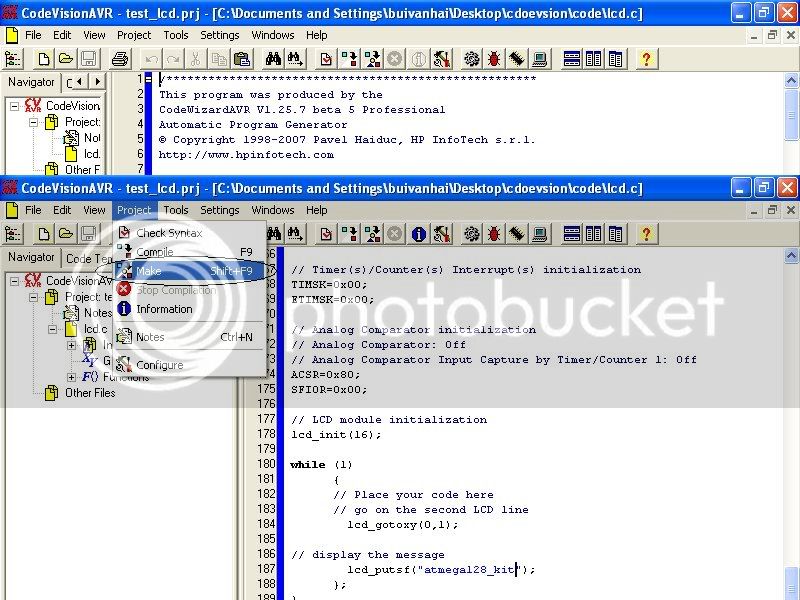
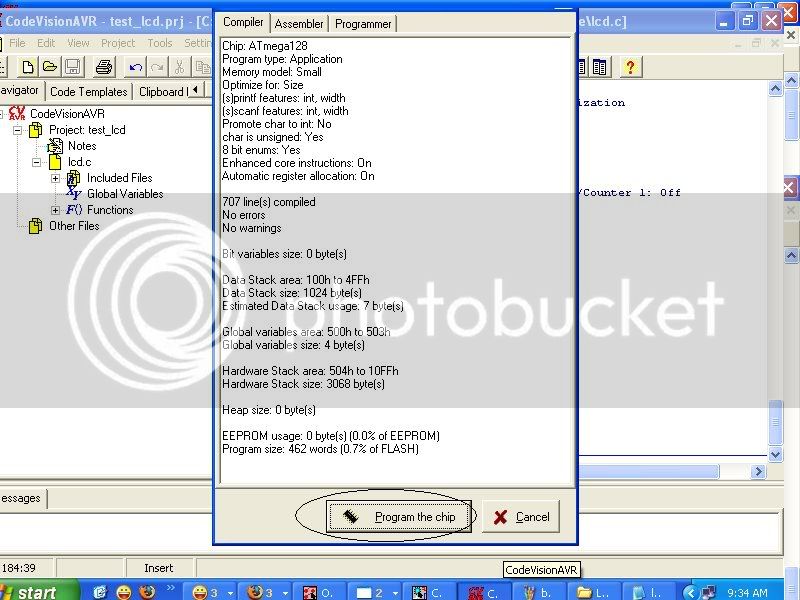
Cái EASY AVR chính hãng của MikroE -DuyPhi hiện đang có, nhưng giá quá cao, còn hàng nhái tên EASY trong nước làm thì cũng tầm trên 1tr đồng, tạm thời Phi chưa có kit demo, sẽ suy nghĩ vài hôm rồi đặt mạch, post lên cho anh em xài chung.  
Bạn cứ an tâm, Phi sẽ cố gắng mọi cách thuận tiện nhất trong khả năng có thể.

AVR 910 USB Programmer \_PNLAB  
  
Hiện tại có một số nhà cung cấp mạch nạp USB-AVR  
Sau đây là thông tin mà DUYPHI đã sưu tầm được từ PNLAB các bạn có thể truy cập vào website sau để xem:  
[http://www.pnlabvn.com/pnlab/index.p...mart&Itemid=29](http://www.pnlabvn.com/pnlab/index.php?page=shop.product_details&category_id=1&flypage=shop.flypage&product_id=45&option=com_virtuemart&Itemid=29)  
  
  
**1.Tên gọi sản phẩm :**  
AVR 910 USB Programmer Cable (cáp nạp các chíp AVR của Atmel chỉ với một kết nối USB) hoạt động tốt trên các hệ điều hành windows 2000, XP và Vista.  
  
**2. Các hỗ trợ chính:**  
Nạp được hầu hết các dòng AVR và một số chíp 89S của Atmel (xem danh sách phía dưới)  
Kiểm tra lỗi sau khi nạp  
Hỗ trợ khóa chíp và lập trình fuse bit  
Header chuẩn ISP cho kết nối thuận tiện  
Tốc độ nạp cao, sử dụng được với hầu hết các trình biên dịch  
Mạch siêu nhỏ gọn, bọc cách điện thuận tiện cho di chuyển, sử dụng (nhiều mầu sắc để lựa chọn!)  
*Cực kì đơn giản trong kết nối, cài đặt và sử dụng*  
  
**3. Đặc điểm:**  
Kích thước: dài 1.5m  
Nguồn cung cấp: 5V từ cổng USB  
Giao tiếp Virtual COM qua USB  
Tốc độ truyền tối đa: 115200bps, 8 bit, no parity, 1 stop, no flow control  
Định dạng file: Intel 8-bit HEX  
Phần mềm: CodeVisionAVR, AVRStudio, AVRDude, AvrOsp II, Bascom AVR.  
**4.Chức năng của sản phẩm :**  
Sản phẩm có khả năng nạp cho hầu hết các loại vi điều khiển AVR có mặt tại Việt Nam và một số chip 89S.  
  
  
  
  
  
  
  
**Giá theo như thông tin trên mạng của PNLAB là: 100 ngàn đồng.**

Attached Files

* [](http://www.dientuvietnam.net/forums/filedata/fetch?id=1333415)
* [](http://www.dientuvietnam.net/forums/filedata/fetch?id=1333416)

DuyPhi quả là tốt bụng  
mình cũng xin góp một chút ít tài liệu cho các bạn mới làm quen với AVR  
đầu tiên là trình biên dịch hay phần mền lập trình  
  
AVR studio 4  
  
  
[http://www.4shared.com/file/60021059...udio4b528.html](http://www.4shared.com/file/60021059/5142f316/aStudio4b528.html)  
[http://www.4shared.com/file/60026988...dio413SP1.html](http://www.4shared.com/file/60026988/1edbefb/AVRStudio413SP1.html)  
[http://www.4shared.com/file/60023647...dio413SP2.html](http://www.4shared.com/file/60023647/1655b69/AVRStudio413SP2.html)  
đầu tiên cài **aStudio4b528**  
tiếp theo cài **AVRStudio413SP1**  
và cài **AVRStudio413SP2**  
  
CodeVisionAVR C Compiler  
  
[http://www.4shared.com/file/60023769...nAVR\_1246.html](http://www.4shared.com/file/60023769/d5297edb/CodeVisionAVR_1246.html)  
cài xong nhớ update License Professional.dat  
Và mạch nạp dùng cổng USB  
  
chạy trên cả hai phần mền  
có sơ đổ, mạch in, file hex, hướng dẫn cài đặt trong file nap.rar  
  
chúc các ban thành công

Thêm 1 hướng dẫn cho codevision.  
Mình dùng mạch nạp stk500\_usb và kít atmega128 do mình thiết kế.  
b1: mở code vision lên.  
  
b2:check radio button project.  
  
b3: chon chip atmega128 va thach anh 8mhz.  
  
b4: chon tab lcd chon portA. Line 16.  
  
b5: file -> generate,....va lưu soure file ,file project,...  
  
b6: chon setting ->progammer->stk500, com3.  
  
b7: project-> configure.  
  
  
b8: viết code.  
  
b9: sau khi viết song chọn. project->make. nếu ko có lỗi bấm ok. mach sẽ tự động nạp chương trình sau khi nap song sẽ chạy chương trinh trên kit.  
  
  
kết quả trên kit.  


kit phát triển các loại:  
Đã có LPC2103 kit, AT91SAM7S kit và J\_Link V.7.

Vâng! xin thay mặt anh em, xin chân thành cảm ơn anh Dũng Và anh Huy AKC.  
DUYPhi sẽ biên tại file của anh DŨNG một tí và bỗ sung thêm một số thông tin khác về mạch nạp USB-AVR rồi chúng ta sẽ gát chúng lại để chuyển sang đề tài khác.  
Thưa anh Dũng, Phi mượn tạm 2 file của anh đi đặt mạch nhé. Chắc đặt 20 PBC để cho các anh em khác dùng chung luôn. Nếu không có gì bất tiện thì đầu tuần tới Phi sẽ đi đặt.  
Thân ái!

ám ơn anh Chip đã quan tâm đến chuyên mục này!  
Như đã nói DuyPhi cũng bắt đầu với AVR trong lúc này là: Từ Bit 0, do đó chúng ta sẽ ko có gì ngại khi đưa ra những vấn đề mà chúng ta còn khúc mắc, mỗi người biết đến đâu thì chia xẻ cho nhau đến đó vậy!  
AVR tốc độ xử lý cực cao do đó đủ để thực hiện một số công việc mà trước một số họ vdk khác không làm được, AVR ngày nay không dùng COM ảo như trước đây nữa, vì thực sự nó vẫn là giao thức RS232. Thay vào đó AVR sẽ thực hiện việc điều khiển vào gtiep với PC thông qua chuẩn USB trực tiếp, trên VB nếu muốn lập trình thao tác được với AVR qua cổng USB trực tiếp thì chúng ta cần thông qua 1 file hỗ trợ gtiep qua USB là thư viện liên kết động: \*.dll. Việt lập trình file dll cần có 1 kiến thức tương đối tốt và chuyên môn hơn về lập trình ứng dụng trên máy tính, chúng ta sẽ không nên đi sâu vào file này, do đó chúng ta sẽ dùng tạm file này bởi 1 số người tốt bụng, hi hi... anh nên vào mạng và search 1 cái là có cái file này, sau đó hãy nhúng nó vào chtrinh của mình có thể là VB, VC++.....  
Đó là những hiểu biết chút ít của DuyPhi trong mấy ngày qua khi tiếp cận với AVR.  
Thú thật chuyện điều khiển AVR thông qua USB thật sự là Phi tôi thích thú vô cùng, thấy mấy cái chuyện đơn giản như sau: PC SẼ ĐIỀU KHIỂN 8 CON LED TẮT CHỚP THÔNG QUA AVR- GTIEP QUA CỔNG USB, Chỉ đơn giản vậy thôi mà gần cả tháng nay vẫn chưa thể hiểu và nắm bắt hết được, hi hi hi...  
***Đúng thật là khi bắt đầu thì dù là 1 chuyện đơn giản cũng ko hề giản đơn.***- *Lê Duy Phi*

*cái mạch nạp NPLab đẹp quá. mà lại rẻ nữa. mình đang sử dụng mạch của TMe xài cũng ổn định lắm, nếu không mình sẽ sử dụng mạch này. mình cũng đang tập làm quen với AVR, có nhiều cái còn chưa hiểu rỏ lắm, đúng lúc luồng này của bác Duyphi mở ra. hy vọng chúng ta sẽ trao đổi nhiều hơn để cùng nhau phát triển. nói nhỏ nhé bác Duyphi là thần tượng của mình đó   
mình đang chuẩn bị học cách giao tiếp USB, bác có thể giải thích giúp mình vài điều này được không? mình định viết trên PC phần mềm giao tiếp với AVR qua cổng USB nhưng chưa biết làm thế nào, có phải là mình cài COM ảo rồi sử dụng giống như COM bình thường không? còn phần mạch AVR thì phải viết phần mềm truyền nhận thế nào? mình chưa roe lắm, mong các cao thu chỉ giúp.*

Để sử dụng giao tiếp USB với AVR cách đơn giản nhất là bác sử dụng Com ảo, sau khi cài đặt cho nó thì sử dụng như COM bình thường ...  
để sử dụng COM ảo bác có thể sử dụng mạch chuyển từ USB -> sang USART em gửi dưới file đính kèm!

Một kinh nghiệm nhỏ khi bắt đầu bằng một project LED chớp tắt, khi lập trình và nạp cho AVR , các bạn nhớ set FUSE cho AVR để AVR chạy với thạch anh ngoài ( bits SUT, CKSEL và CKOPT ), tùy thuộc vào thạch anh bên ngoài => có các set FUSE khác nhau ( xem trong datasheet ).

Cảm ơn các bạn đã nhiệt tình giúp đỡ mọi người làm quen với vi điêu khiển.  
Theo tôi để mọi người có thể nắm bắt nhanh và dễ hiểu nhất đó là các ví dụ cụ thể, từ dễ đến khó. Các anh em không nên đưa ra bài viết hay những câu hỏi lan man... Vì vậy rất mong anh em trên diễn đàn đưa ra các ví dụ cụ thể thay vì nói "suông" để các bạn mới làm quen dễ hiểu nhất, có thể thực hành được ngay.  
Để bắt đầu mình xin được đóng góp một số ví dụ mà hầu như ai mới làm quen VĐK đều làm qua. Ví dụ đơn giản nhất là làm cho cả 8 led nối vào PORTA sáng nhấp nháy.  
  
#include <mega16.h>  
#include <delay.h>  
  
void main(void)  
{  
// Input/Output Ports initialization  
// Port A initialization  
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out  
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0  
PORTA=0xFF; //PORTA xuất ra mức 1 (5V)  
DDRA=0xFF; //Chiều của PORT là chiều xuất dữ liệu ra.  
while (1)  
{  
PORTA = ~PORTA; //đảo tất cả các bit của PORTA, tạo hiệu ứng sáng nhấp nháy  
delay\_ms(500); // trễ 500ms để ta cảm nhận được led sáng nhấp nháy  
  
};  
}

Một ví dụ nữa là làm led sang từ phải qua trái và tắt dần từ trái qua phải.  
  
//cac led noi vao PORTA, cuc duong cua led noi vao tro 330R va noi len VCC  
//doan code nay lam led sang tu phai qua trai va tat dan tu trai qua phai  
  
#include <mega16.h>  
#include <delay.h>  
  
void main(void)  
{  
unsigned char i,data;  
PORTA=0xFF; //port A co tro keo ben trong  
DDRA=0xFF; //Huong cua du lieu = xuat du lieu ra  
while (1)  
{  
data = 0xFF; //tat tat ca cac led  
for(i= 0;i<=7;i++) // lap lai 8 lan tuong ung 8 den led sang lan luot  
{  
data <<=1; // dich sang trai 1 bit de den sang tang dan len  
PORTA = data; // xuat du lieu ra dieu khien den  
delay\_ms(500); // led sang lan luot cac nhau 500ms  
}  
data = 0xFF; // tat cac led  
for(i = 0;i<=7;i++) // lap lai 8 lan, led tat lan luot cho den het  
{  
data >>= 1; // dich bit sang phai 1 bit  
PORTA = ~data; // dao du lieu tu trang thai sang thanh tat va xuat ra dieu khien den  
delay\_ms(500);  
}  
};  
}

Ơ, lập trình bằng assembly... Thôi cũng được bởi dù sao bác Phi cũng bỏ luồng này rồi, mình tự nghiên cứu vậy  
Bác lập dùng trình dịch C nào thế?

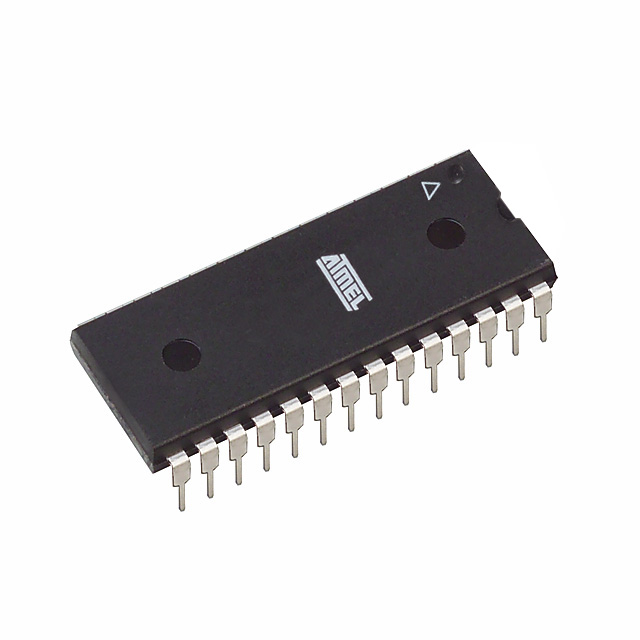
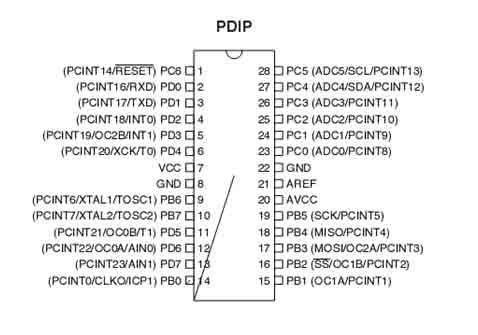
*Một ví dụ nữa là làm led sang từ phải qua trái và tắt dần từ trái qua phải.*

Bác có thể làm cho một led chạy lần lượt từ trái qua phải rồi từ phải qua trái ( không phải là sáng dần và tắt dần)??  
Bác cho em hỏi ý nghĩa của lệnh này như thế nào:  
Lưu ý là SPE và MSTR, SPR0 là các bit của một thanh ghi thuộc thanh ghi SPCR của AVR và bit SPIF thuộc thanh ghi SPSR.  
Trong Assembly:  
**ldi r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)**  
**out SPCR, R17**  
và Trong C:  
***while(!(SPSR & (1<<SPIF))*** ôi, cái lệnh while ấy thật là khó hiểu.  
và  
**SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);**

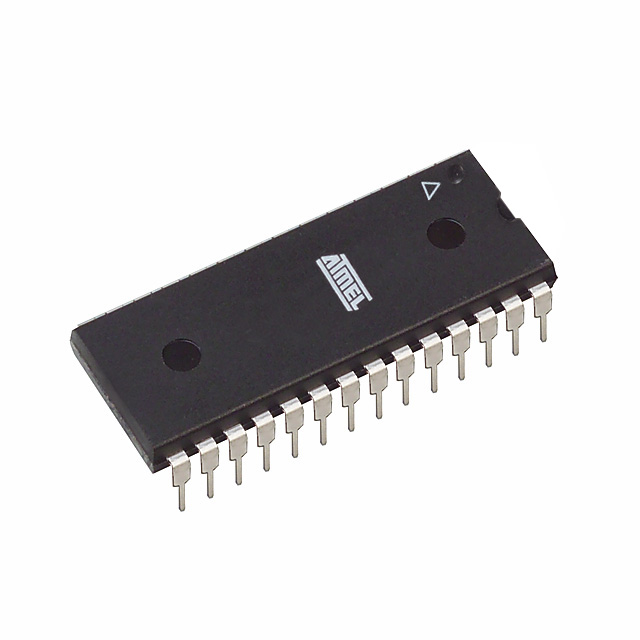
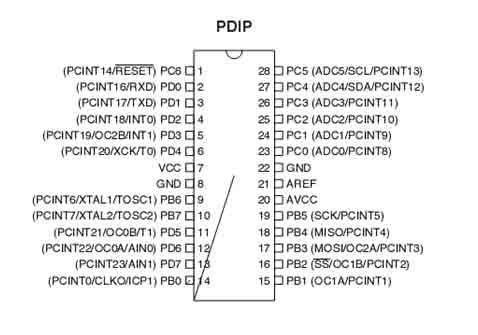
**Tóm tắt lại một số vấn đề trước khi chuyển sang chương khác!**  
**1- Mạch nạp:**  
Hiện nay có các nhà sản xuất:  
- PNLAB- giá 100/ mạch USB-AVR  
  
  
  
  
- Tme: 180 ngàn/ mạch USB-AVR  
  
  
**- Phúc Thịnh- Giá 120 ngàn:**  
  
  
  
  
  
Đặc biệt mạch nạp miễn phí có file đính kèm tự làm rất dễ của anh DŨNG :  
  
  
  
Link download về tự làm:  
[http://dientuvietnam.net/forums/atta...6&d=1219415233](http://dientuvietnam.net/forums/attachment.php?attachmentid=10656&d=1219415233)  
  
Ưu điểm là dùng cáp cổng COM- nhưng khuyết điểm là không tương thích với 1 số mạch trên thị trường do quy cách cáp không theo chuẩn.(góp ý)

**chọn vi điều khiển để tiến hành nghiên cứu**

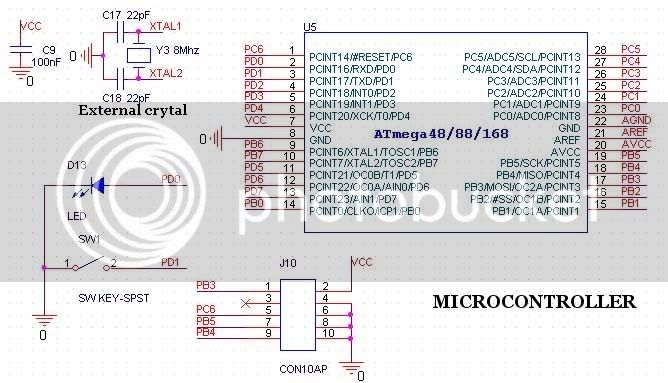
Để chọn ra 1 con CHIP để nghiên cứu lý ra phải xin hội ý kiến của anh chị em, tuy nhiên để khỏi mất thời gian thì tôi sẽ dựa trên ý kiến chủ quan và tình hình khách quan CHIP hiện có bán ở TpHCM mà chọn ra nhá :  
- **Chọn con ARV-- Atmega8**: Bởi vì nó có tích hợp nhiều thứ như ADC, SPI,....tần số dao động của nó cũng ngang bằng với các con mạnh khác trong họ hàng AVR. Vả lại giá của nó khá rẻ, dễ làm bo mạch, và tự làm....  
- **Chọn con AVR- Atmega128:** Con này khá mạnh đầy đủ các tính năng và thuộc hạng cao cấp hơn trong họ hàng về nhiều mặt như số lượng PORT, số lượng ngắt,.....tuy nhiên nó không dễ dàng tự làm được vì nó chỉ có kiểu chân dán,đổi lại vì đó là chiến lược nghiên cứu lâu dài, các bác có 1 con này thì học avr dài dài vẫn còn chuyện để để mà ngâm cứu.

Để đơn giản thì việc đầu tiên tôi sẽ chọn ra 1 trong số 2 con đó để tiến hành làm.  
Tôi chọn con **Atmega8**  
  
Thông tin datasheet của nó, anh em có thể tải về tại:  
  
  
  
  
  


Attached Files

* [](http://www.dientuvietnam.net/forums/filedata/fetch?id=1333484)
* [](http://www.dientuvietnam.net/forums/filedata/fetch?id=1333485)

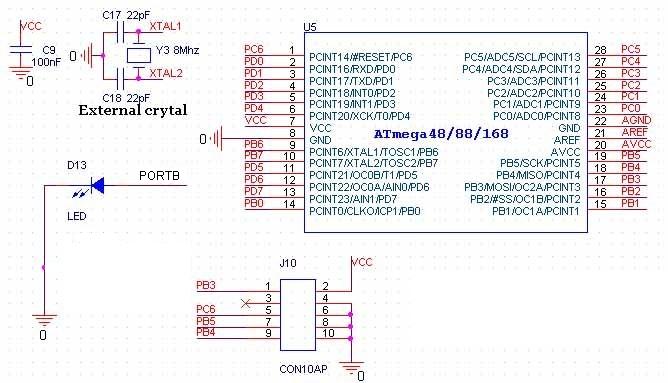
**XÂY DỰNG SƠ ĐỒ MẠCH TIẾN HÀNH THÍ NGHIỆM**

Đây là sơ đồ mạch tối thiểu cần dùng đến như sau:  
  
  
Sơ đồ này tải từ trang của ***Vgam***.

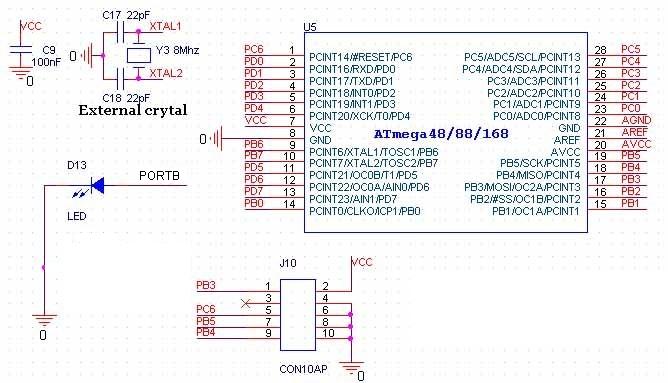
Không biết có anh em nào rảnh design nó thành 1 cái PCB mạch không nhỉ???  
Vả lại vào 1 số trang của anh em cũng không thấy cái PCB TEST nào re rẻ để mua về học. Anh em nào làm thương mại nhớ ghi lại ý kiến góp này của DuyPhi.

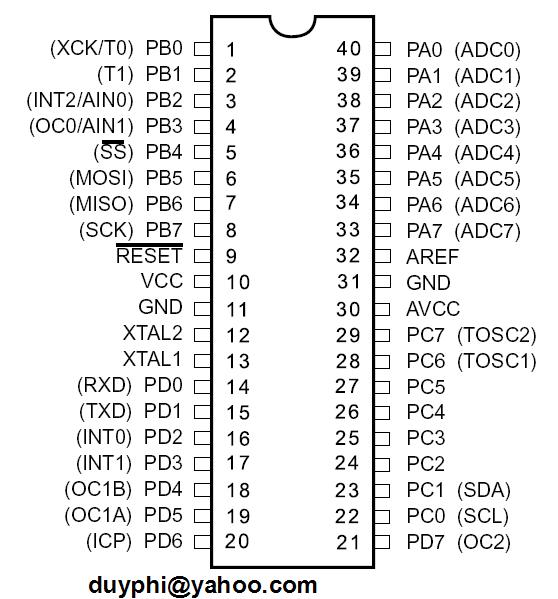
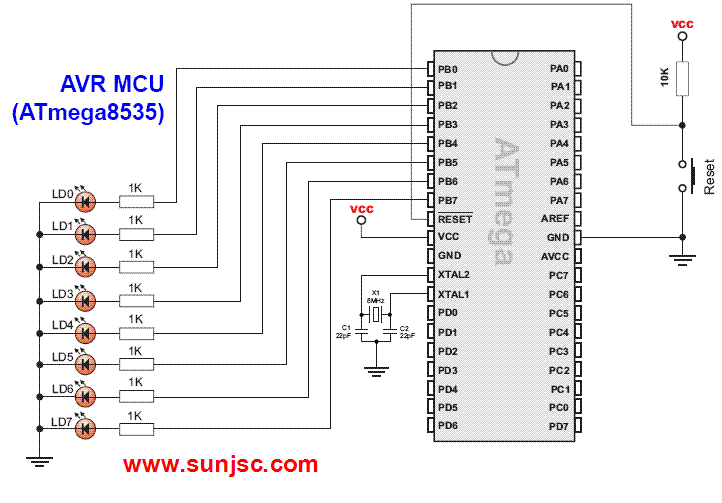
**TẠO DỰ ÁN ĐẦU TIÊN TRONG AVRSTUDIO CHO AVR  
SỬ DỤNG NGÔN NGỮ ASM**

**Các vấn đề mà chúng ta sẽ nghiên cứu:**  
***1- Mục đích  
2-Xây dựng phần cứng  
3- Sử dụng phần mềm Avrstudio  
4- Tạo ví dụ  
5- Viết code  
6- Test chương trình***  
  
  
**Mục đích:**  
*Trong hướng dẫn đơn giản này, chúng ta sẽ tạo một ví dụ ứng dụng mới, và viết một vài dòng code và biên dịch nó trong AVRstudio ASM cho AVR và kiểm tra kết quả.  
Ví dụ của chúng ta sẽ làm cho LED quang sáng tắt, do đó chúng ta rất dễ dàng kiểm tra trên vi điều khiển AVR*

**2-Xây dựng phần cứng**  
  
Đây là sơ đồ kết nối mà bạn sẽ cần để kiểm tra code chương trình cho vi điều khiển. Tôi sẽ sử dụng 3 loại vi điều khiển khác nhau của AVR trong ví dụ này đó là:  
**- atmega8  
- atmega8535  
- atmega128**  
Và dĩ nhiên anh chị em vẫn có thể các con chip khác, tuy nhiên DuyPhi tôi không có nhiều chip để test thử và cũng như các biết là tôi cũng đang bắt đầu từ Bit 0, do đó kết quả thử nghiệm trên các chip khác tôi không chắc lắm.  
  
Led được kết nối vào PORTB, tuy nhiên các bạn có thể mở rộng ví dụ này trên các port khác, ok!  
  
  


Attached Files

* [](http://www.dientuvietnam.net/forums/filedata/fetch?id=1333523)
* [](http://www.dientuvietnam.net/forums/filedata/fetch?id=1333526)

  
  
  
  
  
**CODE CHƯƠNG TRÌNH CHO VI ĐIỀU KHIỂN:**

Code:

.org $000

Batdau:

ldi r16,low(RAMEND)

out SPL,r16 ;init Stack Pointer

ldi r16,high(RAMEND)

out SPH,r16

LDI R16, 255

out DDRB,R16

main:

ldi r16,255

out PORTB,r16

rcall delay

ldi r16,0

out PORTB,r16

rcall delay

rjmp main

;-------------------------------------------------------

;Delay 500ms voi thach anh su dung: 8mhz

;-------------------------------------------------------

delay500ms:

ldi R23,0x6B

\_wg500loop0: ldi R24,0x46

\_wg500loop1: ldi R25,0xB1

\_wg500loop2: dec R25

brne \_wg500loop2

dec R24

brne \_wg500loop1

dec R23

brne \_wg500loop0

pop R25

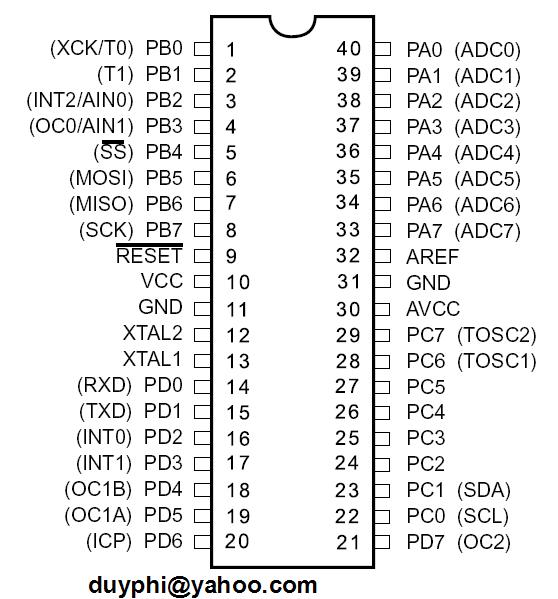
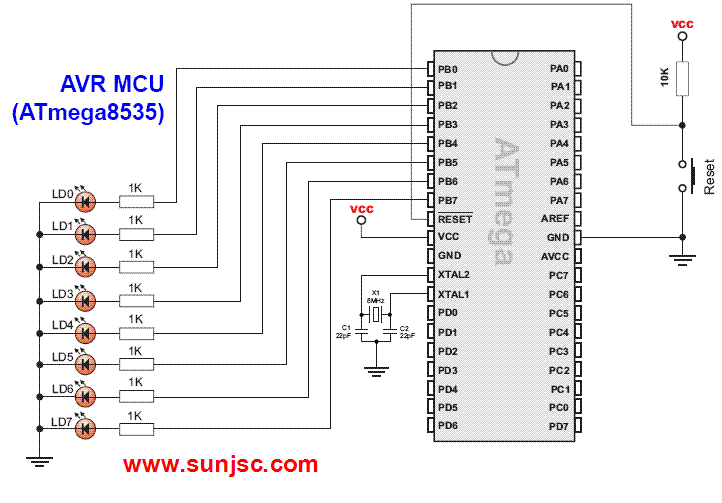
pop R24

pop R23

ret

;end

Attached Files

* [](http://www.dientuvietnam.net/forums/filedata/fetch?id=1333524)
* [](http://www.dientuvietnam.net/forums/filedata/fetch?id=1333525)

**GIẢI THÍC CHƯƠNG TRÌNH:**  
  
Chúng ta phân tích đoạn chương trình đầu tiên:  
*.org $000  
Batdau:  
ldi r16,low(RAMEND)  
out SPL,r16 ;init Stack Pointer  
ldi r16,high(RAMEND)  
out SPH,r16*  
  
  
**Dòng ORG $000** khai báo cho chương trình dịch biết điểm địa chỉ bắt đầu của chương trình, ở đây là $000.  
Dấu**$**là biểu diễn con số sau nó là dạng số HEX.  
Nhãn: "**batdau**" là do lập trình viên tự định nghĩa, hay tự đặt,không liên quan đến cấu trúc lệnh của chương trình. Việc đặt tên 1 nhãn như vậy sẽ làm cho chtrinh của chúng ta thêm trong sáng và dễ hiểu. Dĩ nhiên anh chị em có thể thay thế nhãn "batdau" này bằng 1 cái tên khác như: Begin, reset....  
  
  
Tiếp theo, câu lệnh:  
**ldi r16,low(RAMEND)**  
**ldi** là lệnh của chương trình, có nghĩa là load data input, nạp dữ liệu vào.  
**ldi r16,low(RAMEND)** có nghĩ là nạp dữ liệu địa chỉ byte thấp của RAMEND cho thanh ghi R16.  
**RAMEND????** nó là tên đã được định nghĩa sẵn trong file header. Nó được gắn tại địa chỉ cuối cùng của vùng ram của vi điều khiển.  
***Tại sao chúng ta phải gõ mấy dòng lệnh này???*** các bạn hãy đợi 1 chút, sau khi phân tích vài dòng nữa tôi sẽ quay lại trả lời câu hỏi này.  
  
**Câu lệnh KếTiếp theo:**  
  
**out SPL,r16 ;**  
**OUT** là 1 lệnh trong chtinh, nó có nghĩa là nạp dữ liệu từ R16 vào cho thanh ghi **SPL**  
  
SPL và SPH là 2 thanh ghi của stack pointer. Việc đầu tiên mà chúng ta bắt đầu trước khi lập trình là thiết lập lại stack pointer.  
Khác hơn 8051, chúng ta chỉ cần thiết lập Stack point ở định dưới của vùng ram và nó chỉ có độ lớn là 8bit, tức 1 byte.  
Trong khi đó, AVR có nhiều dòng từ trung bình trở lên mạnh đều có 2 byte cho stack poiter do đó SP của AVR có thêm SPH và SPL tức là độ lớn của nó lên đến 16bit.  
  
**Về cách thức hoạt động của 8051 và AVR cũng có nhiều sự khác nhau:**  
1- Với 8051 thì mỗi khi đẩy vào SP thì địa chỉ của nó sẽ tăng lên 1 và mỗi khi lấy ra khỏi SP thì địa chỉ của nó giảm đi 1.  
2- Với AVR thì ngược lại các bạn ah, mỗi khi đẩy vào SP thì địa chỉ của nó sẽ giảm 1 và mỗi khi lấy ra khỏi SP thì địa chỉ của nó tăng thên 1.  
3- 8051 thì SP của nó chỉ có 8bit, AVR có 16bit-2 byte

**2 Câu lệnh tiếp theo:**  
  
**ldi r16,high(RAMEND)  
out SPH,r16**  
  
2 câu này cũng như đã nói ở trên, lấy byte cao của RAMEND nạp vào r16 rồi đẩy vào SPH.  
Tới đây chúng ta đã kết việc thai lập SP cho vi điều khiển.  
  
**Kết luận:**  
Trước khi viết chương trình cho vi điều khiển, cho dù là AVR hay 8051 đi nữa thì chúng ta cũng phải thiết lập lại trước SP rồi mới tiến hành lập trình tiếp.  
  
**Mở rộng vấn đề:**1- hãy cho biết địa chỉ RAMEND của các AVR sau; atmega2313, atmega8, atmega128.  
2- Thiết lập SP cho AVR ở tại các địa chỉ khác mà không dùng như cách trên.

Code:

LDI R16, 255

out DDRB,R16

Đây là đoạn code thiết lập cổng cho AVR, việc ghi giá trị vào cho thanh ghi DDRB sẽ thiết lập cho PORTB là nhập hoặc xuất. Ví dụ, 1 bit nào của DDR=1 thì bit tương ứng của PORTB sẽ là bit xuất, còn =0 là nhập  
  
**Lưu ý:** *Điều này trái ngược với vi điều khiển PIC, với PIC thì việc cho thanh ghi TRISB=0 thì bit tương ứng trên PORTB sẽ là xuất, và =1 thì lại là nhập. Còn với 8051 thì việc này hoàn toàn được làm tự động hết, nên không có khái niệm thiết lập PORT trong 8051*.

Code:

main:

ldi r16,255

out PORTB,r16

rcall delay

ldi r16,0

out PORTB,r16

rcall delay

rjmp main

Ở đây **MAIN** cũng là nhãn như nhãn ở ""batdau" ở trên.  
Ở đoạn này chỉ có thêm 1 lệnh khác là : **rjmp**. Đây là lệnh nhãy cũng giống như ở 8051 có lện LJMP hay JMP cũng vậy.  
  
**Kết luận:**  
Trong đoạn này đa số là giống ở 8051, một số còn lại thì giống PIC như hơi trái ngược với PIC 1 chút về quy định cách thức nhập xuất của PORT.

Code:

;-------------------------------------------------------

;Delay 500ms voi thach anh su dung: 8mhz

;-------------------------------------------------------

delay500ms:

ldi R23,0x6B

\_wg500loop0: ldi R24,0x46

\_wg500loop1: ldi R25,0xB1

\_wg500loop2: dec R25

brne \_wg500loop2

dec R24

brne \_wg500loop1

dec R23

brne \_wg500loop0

pop R25

pop R24

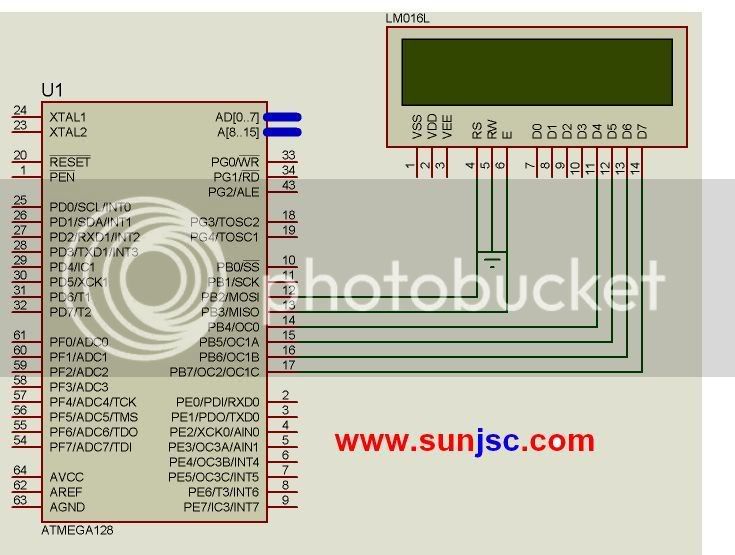
pop R23

ret

;end

Trong chương trình này chúng ta dùng Xtal 8Mhz, nên chu kỳ lệnh Ti=1/fosc=1/8Mhz=0.125us.  
Và đoạn chương trình này chúng ta sẽ có 1 vòng lặp là 500ms, cách tính toán thời gian đáp ứng của đoạn code trên như sau:  
**(((3\*0XB1)+3)\*0x46+3)\*0x6B)\*Ti=500ms.**

Tiếp theo là một ví dụ khác về ứng dụng AVR trong việc điều khiển LCD.  
Các bạn tham khảo, chtrinh chạy trên AVR Atmega128.

Sơ đồ mạch kết nối:  
  
  


Chương trình CODE và hex file của chườn trình  
Các bạn tải về để tham khảo,  
Chúc thành công tốt đẹp!

Attached Files

*Hoan hô bác Duy Phi, em thấy rất dễ hiểu.  
  
Em trả lời thế này:  
1- Địa chỉ RAMEND là điạ chỉ cuối cùng của bộ nhớ RAM của AVR, như vậy, đối với ATMEGA8 thì nó có giá trị là 45Fh (1Kbyte internal Ram +32 R +60 I/O), với Atyny90s2313 là DFh (128 byte internal RAM), với ATmega128 là 10FFh( 4Kbyte RAM).  
2- Thiết lập cách khác : vd cho ATmega128  
LDI R16,0x10  
OUT SPH,R16  
LDI R16,0xFF  
OUT SPL,R16  
Thế có đúng không ạ?*

Vâng!Rất chính xác. Anh em có thể tham khảo cách này của **mrcuongcon** thì dễ hiểu hơn nữa.  
chúc bạn có ngày lễ vui vẻ

*Bác Duy Phi ơi, bây giờ em muốn lấy các kí tự từ trong một bảng .db nhưng không lấy lần lượt  
mà lấy bất kỳ, thì làm thế nào. Trong 8051 thì đại để nó thế này: movc a,@a+dptr.  
Trong avrstudio thì nó chỉ có lệnh LDD Rd,Z+q với q là một hằng số và q mang giá trị 0-63. Ý của em là là sao thế chỗ của q bằng một biến (vd như biến .def count R2) Em muốn làm về led 7 thanh mà mắc chố này.*

oh, anh em chạy nhanh quá đi mất! Mãi lo làm kiếm tiền trả tiền phòng nên ko chạy kịp với anh em rồi. hi hi  
  
Ok, với 8051 thì **MOVC A,@A+DPTR** với A là con số tham chiếu đến 1 địa chỉ nào đó trong bảng thì với AVR chúng ta sẽ làm như sau:  
  
***LDI ZH,HIGH(BANG DATA CUA CHUNG TA)  
LDI ZL,LOW(BANG DATA CUA CHUNG TA)  
ADD ZL,R23***<----------*Ở đây giá trị của R23 tương đương với thanh ghi A trong 8051*  
***LMP***  
...............  
...............  
Xong, giá trị trả về sẽ nằm trong thanh ghi R0.  
  
Chúc vui!

Các bạn lưu ý: Trong giai đoạn này chúng ta chỉ trao đổi về LẬP TRINHH HỢP NGỮ ASM mà thôi. Nếu các bạn muốn trao đổi về lập trình các ngôn ngữ khác thì xin mở thêm luồng mới, DuyPhi sẽ hỗ trợ và xin được trao đỗi với các bạn ở luồng đó, để luồng này được tập trung hơn vào ASM cho AVR.

Ngày mai là bắt đầu nghĩ 2/9, anh em nên nghĩ ngơi vài hôm, sau đó hãy làm việc. Sau 2/9 chúng ta sẽ tiếp tục diễn giải cấu trúc chương trình và cách thức hoạt động của từng câu lệnh trong chương trình trên.  
Đồng thời chúng ta sẽ mở rộng vấn đề lên qua nhiều ví dụ cụ thể khác.  
Chúc anh em có ngày lễ vui vẻ!

**Đây là 1 số các ví dụ về LED:**

Code:

/\* Moving LED example

Chip: ATmegaS8515

Memory Model: SMALL

Data Stack Size: 128 bytes

8 LEDs are connected between the PORTC

\*/

// I/O register definitions for AT90S8515

#include <mega8515.h>

// quartz crystal frquency [Hz]

#define xtal 3686400

// moving LED frequency [Hz]

#define fmove 2

// the LED on PORTC output 0 will be on

unsigned char led\_status=0xfe;

// TIMER1 overflow interrupt service routine

// occurs every 0.5 seconds

interrupt [TIM1\_OVF] void timer1\_overflow(void)

{

// preset again TIMER1

TCNT1=0x10000-(xtal/1024/fmove);

// move the LED

led\_status<<=1;

led\_status|=1;

if (led\_status==0xff) led\_status=0xfe;

// turn on the LED

PORTC=led\_status;

}

void main(void)

{

// set the I/O ports

// all PORTC pins are outputs

DDRC=0xff;

// turn on the first LED

PORTC=led\_status;

// Timer/Counter 1 initialization

// Clock source: System Clock

// Clock value: 3.600 kHz

// Mode: Normal top=FFFFh

// OC1A output: Discon.

// OC1B output: Discon.

// Noise Canceler: Off

// Input Capture on Falling Edge

// Timer 1 Overflow Interrupt: On

// Input Capture Interrupt: Off

// Compare A Match Interrupt: Off

// Compare B Match Interrupt: Off

TCCR1A=0x00;

TCCR1B=0x05;

TCNT1=0x10000-(xtal/1024/fmove);

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x00;

OCR1AL=0x00;

OCR1BH=0x00;

OCR1BL=0x00;

// External Interrupt(s) initialization

// INT0: Off

// INT1: Off

// INT2: Off

MCUCR=0x00;

EMCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization

TIMSK=0x80;

// global enable interrupts

#asm

sei

#endasm

// the rest is done by TIMER1 overflow interrupts

while (1);

}

Vì 1 số bạn gởi thư về hỏi tôi về chuyên mục AVR này. Thật sự thì tôi cũng như nhớ rõ là có luồng này, do đó nếu các bạn có thắc mắc thì email rõ địa chỉ để tiện việc tôi trả lời cho các bạn.  
  
Tiếp theo:  
  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Project : Clignotement d'une Led  
Date : 12/12/2003  
Auteur : Angeliz  
Signe particulier : Autodidacte  
  
Microcontrôleur : ATmega8535  
Quartz : 8,000000 MHz  
Memory model : Small  
External SRAM size : 0  
Data Stack size : 128  
Compilateur : CodeVisionAvr  
  
Led sur bit 0 PORTB anode au 5V  
résistance de 1K en série.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
#include <mega16.h>  
  
#define TOIE0 0  
#define Led PORTB.0  
#define bit\_set(var,bitno) ((var) |= 1 << (bitno))  
#define bit\_clr(var,bitno) ((var) &= ~(1 << (bitno)))  
  
  
// Declare your local variables here  
unsigned char cTime, cSignal;  
  
// Timer 0 interruption de débordement de TCNT0  
// Interruption toutes les 32768mS  
// cSignal passe à 1 toute les 524mS (16\*32768)  
interrupt [TIM0\_OVF] void timer0\_overflow(void)  
{  
if(cTime++ >=15) cSignal=1;  
}  
  
void main(void)  
{  
PORTB=0x00; // configuration du portB à l'état haut  
DDRB=0x01; // configuration du portB en sortie  
TCCR0=0x05; // pas de TCNT0 à 128µS Clock value: 7,813 kHz  
  
PORTA=0xFF;  
DDRA=0xFF;  
  
// Global interruption activé  
#asm("sei")  
bit\_set(TIMSK,TOIE0); // configure TOIE0 à 1, autorise interruption  
cSignal=0; // variable indiquant que le débordement doit être traité si 1  
cTime=0; // variable de temps 1 unité = 32768mS  
  
while (1)  
{  
if(cSignal)  
{  
cSignal=0;  
cTime=0;  
if(Led) Led=0; // Led ON  
else Led=1; // Led OFF  
}  
}  
}

LCD counter!  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Compiler : CodeVision  
Hardware : AVR-Ctrl (with Display)  
Chip type : AT90S8535  
Clock frequency : 8,000000 MHz  
Memory model : Small  
Internal SRAM size : 512  
External SRAM size : 0  
Data Stack size : 128  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
#include <mega16.h>  
#include <stdio.h>  
#include <delay.h>  
  
// Alphanumeric LCD Module functions  
#asm  
.equ \_\_lcd\_port=0x1b  
#endasm  
#include <lcd.h>  
  
// Declare your global variables here  
unsigned int Counter = 0;  
  
// Timer 0 overflow interrupt service routine  
interrupt [TIM0\_OVF] void timer0\_ovf\_isr(void)  
{  
// Place your code here  
Counter++; // incr.  
  
}  
  
  
void main(void)  
{  
// Declare your local variables here  
char text[16]; // Buffer  
  
// Input/Output Ports initialization  
// Port A  
PORTA=0x00;  
DDRA=0x00;  
  
// Port B  
PORTB=0x00;  
DDRB=0x00;  
  
// Port C  
PORTC=0x00;  
DDRC=0x00;  
  
// Port D  
PORTD=0x00;  
DDRD=0x00;  
  
// Timer/Counter 0 initialization  
// Clock source: System Clock  
// Clock value: 7,813 kHz  
// Mode: Output Compare  
// OC0 output: Disconnected  
TCCR0=0x05;  
TCNT0=0x00;  
  
// Timer/Counter 1 initialization  
// Clock source: System Clock  
// Clock value: Timer 1 Stopped  
// Mode: Output Compare  
// OC1A output: Discon.  
// OC1B output: Discon.  
// Noise Canceler: Off  
// Input Capture on Falling Edge  
TCCR1A=0x00;  
TCCR1B=0x00;  
TCNT1H=0x00;  
TCNT1L=0x00;  
OCR1AH=0x00;  
OCR1AL=0x00;  
OCR1BH=0x00;  
OCR1BL=0x00;  
  
// Timer/Counter 2 initialization  
// Clock source: System Clock  
// Clock value: Timer 2 Stopped  
// Mode: Output Compare  
// OC2 output: Disconnected  
TCCR2=0x00;  
ASSR=0x00;  
TCNT2=0x00;  
OCR2=0x00;  
  
// External Interrupt(s) initialization  
// INT0: Off  
// INT1: Off  
// INT2: Off  
MCUCR=0x00;  
MCUCSR=0x00;  
  
// Timer(s)/Counter(s) Interrupt(s) initialization  
TIMSK=0x01;  
  
// Analog Comparator initialization  
// Analog Comparator: Off  
// Analog Comparator Input Capture by Timer/Counter 1: Off  
ACSR=0x80;  
  
// LCD module initialization  
lcd\_init(16);  
  
// Global enable interrupts  
#asm("sei")  
  
lcd\_putsf("Displaytest");  
  
  
while (1)  
{  
sprintf(text, "Counter = %i", Counter); // fill buffer  
lcd\_gotoxy(0,1); // cursor at 0,1  
lcd\_puts(text); // output text  
lcd\_gotoxy(15,0);  
lcd\_putchar(Counter/5); // display ASCII  
  
};  
}

Viết bằn CodeVision C đi.  
  
Code chạy tốt.  
  
void Write\_EEPROM(unsigned int dchi, unsigned char dlieu)  
{  
while(EECR.1);  
EEAR = dchi;  
EEDR = dlieu;  
EECR.2 = 1;  
EECR.1 = 1;  
}  
  
//doc 1 byte du lieu tu dia chi cua eeprom//////////////////////////  
unsigned char Read\_EEPROM(unsigned int dchi)  
{  
while(EECR.1);  
EEAR = dchi;  
EECR.0 = 1;  
return EEDR;  
}