Programmation en C
Pour avr 8 bits

Jérémy Cheynet – INTech
Yann Sionneau – MiNET

www.club-intech.fr
clubcode.minet.net
intlab.minet.net
github.com/leroilion/avr

14 octobre 2010

Les différents outils

Les différents outils

- avr-gcc (pour la compilation)

Les différents outils

- avr-gcc (pour la compilation)
- avr-objcopy (pour créer le fichier hex)

Les différents outils

- avr-gcc (pour la compilation)
- avr-objcopy (pour créer le fichier hex)
- avrdude (pour flasher)

avr-gcc

avr-gcc

```
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -c
monfichier1.c
```

avr-gcc

```
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -c
monfichier1.c
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -c
monfichier2.c
```

avr-gcc

```
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -c
monfichier1.c
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -c
monfichier2.c
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -o
monappli monfichier1.o monfichier2.o
```

avr-gcc

```
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -c
monfichier1.c
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -c
monfichier2.c
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -o
monappli monfichier1.o monfichier2.o
```

avr-objcopy

avr-gcc

```
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -c
monfichier1.c
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -c
monfichier2.c
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -o
monappli monfichier1.o monfichier2.o
```

avr-objcopy

```
avr-objcopy -O ihex -R .eeprom monappli main.hex
```

avr-gcc

```
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -c
monfichier1.c
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -c
monfichier2.c
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -o
monappli monfichier1.o monfichier2.o
```

avr-objcopy

```
avr-objcopy -O ihex -R .eeprom monappli main.hex
```

avrdude

avr-gcc

```
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -c
monfichier1.c
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -c
monfichier2.c
avr-gcc -Wall -mmcu=atmega328p -DF_CPU=16000000 -o
monappli monfichier1.o monfichier2.o
```

avr-objcopy

```
avr-objcopy -O ihex -R .eeprom monappli main.hex
```

avrdude

```
sudo avrdude -P /dev/ttyUSB0 -c stk500v1 -p m328p -b 57600 -D
-U flash :w :main.hex
```

Télécharger les sources

Télécharger les sources

http ://github.com/leroilion/avr
Et télécharger dans les exemples le code blink.c

Télécharger les sources

http ://github.com/leroilion/avr
Et télécharger dans les exemples le code blink.c

Compiler les sources

Télécharger les sources

http ://github.com/leroilion/avr
Et télécharger dans les exemples le code blink.c

Compiler les sources

avr-gcc -Wall -mmcu=atmega328p -O2 -DF_CPU=16000000 blink.c -o blink.out

Télécharger les sources

http ://github.com/leroilion/avr
Et télécharger dans les exemples le code blink.c

Compiler les sources

avr-gcc -Wall -mmcu=atmega328p -O2 -DF_CPU=16000000 blink.c -o blink.out

Créer le fichier hexadécimal

Télécharger les sources

http ://github.com/leroilion/avr
Et télécharger dans les exemples le code blink.c

Compiler les sources

avr-gcc -Wall -mmcu=atmega328p -O2 -DF_CPU=16000000 blink.c -o blink.out

Créer le fichier hexadécimal

avr-objcopy -O ihex -R .eeprom blink.out blink.hex

Télécharger les sources

http ://github.com/leroilion/avr
Et télécharger dans les exemples le code blink.c

Compiler les sources

avr-gcc -Wall -mmcu=atmega328p -O2 -DF_CPU=16000000 blink.c -o blink.out

Créer le fichier hexadécimal

avr-objcopy -O ihex -R .eeprom blink.out blink.hex

Flasher l'arduino

Télécharger les sources

http ://github.com/leroilion/avr
Et télécharger dans les exemples le code blink.c

Compiler les sources

avr-gcc -Wall -mmcu=atmega328p -O2 -DF_CPU=16000000 blink.c -o blink.out

Créer le fichier hexadécimal

avr-objcopy -O ihex -R .eeprom blink.out blink.hex

Flasher l'arduino

sudo avrdude -P /dev/ttyUSB0 -c stk500v1 -p m328p -b 57600 -D -U flash :w :blink.hex
sudo avrdude -c usbtiny -p m328p -U flash :w :blink.hex

3 registres

3 registres

- Le registre DDRx
  Registre de configuration du port

3 registres

- Le registre DDRx
  Registre de configuration du port
- Le registre PORTx
  Registre de sortie du port

3 registres

- Le registre DDRx
  Registre de configuration du port
- Le registre PORTx
  Registre de sortie du port
- Le registre PINx
  Registre de lecture du port

Ecrire un 1 logique

Ecrire un 1 logique

```
monport |= ( 1 << monbit );
```

Ecrire un 1 logique

```
monport |= ( 1 << monbit );
#ifndef sbi
#define sbi(port,bit) (port) |= (1 << (bit))
#endif
```

Ecrire un 1 logique

```
monport |= ( 1 << monbit );
#ifndef sbi
#define sbi(port,bit) (port) |= (1 << (bit))
#endif
```

Ecrire un 0 logique

Ecrire un 1 logique

```
monport |= ( 1 << monbit );
#ifndef sbi
#define sbi(port,bit) (port) |= (1 << (bit))
#endif
```

Ecrire un 0 logique

```
monport &= ~( 1 << monbit );
```

Ecrire un 1 logique

```
monport |= ( 1 << monbit );
#ifndef sbi
#define sbi(port,bit) (port) |= (1 << (bit))
#endif
```

Ecrire un 0 logique

```
monport &= ~( 1 << monbit );
#ifndef cbi
#define cbi(port,bit) (port) &= ~(1 << (bit))
#endif
```

Configuration en sortie du port

Configuration en sortie du port

```
DDRB |= ( 1 << PORTB5 );
```

Configuration en sortie du port

```
DDRB |= ( 1 << PORTB5 );
```

Ecriture sur un port

Configuration en sortie du port

```
DDRB |= ( 1 << PORTB5 );
```

Ecriture sur un port

```
PORTB |= ( 1 << PORTB5 ); //Pour mettre le bit 5 du port B à 1
```

Configuration en sortie du port

```
DDRB |= ( 1 << PORTB5 );
```

Ecriture sur un port

```
PORTB |= ( 1 << PORTB5 ); //Pour mettre le bit 5 du port B à 1
PORTB &= ~( 1 << PORTB5 ); //Pour mettre à 0
```

Configuration du port en entrée

Configuration du port en entrée

```
DDRx &= ~( 1 << PORTxN );
```

Configuration du port en entrée

```
DDRx &= ~( 1 << PORTxN );
```

Lecture du port

Configuration du port en entrée

```
DDRx &= ~( 1 << PORTxN );
```

Lecture du port

```
PORTx |= ( 1 << PORTxN ); //Pour activer le pull-up
PORTx &= ~( 1 << PORTxN ); //Pour désactiver le pull-up
```

Configuration du port en entrée

```
DDRx &= ~( 1 << PORTxN );
```

Lecture du port

```
PORTx |= ( 1 << PORTxN ); //Pour activer le pull-up
PORTx &= ~( 1 << PORTxN ); //Pour désactiver le pull-up
uint8_t etat = ( PINx & ( 1 << PINxN ));
```

Objectif :

Faire un programme qui éclaire une LED si un bouton est poussé.

Objectif :

Faire un programme qui éclaire une LED si un bouton est poussé.

Détails techniques :

Utiliser le PORTB5 en sortie (PORTB7 pour arduino mega), et le PORTB0 en entrée.

Objectif :

Faire un programme qui éclaire une LED si un bouton est poussé.

Détails techniques :

Utiliser le PORTB5 en sortie (PORTB7 pour arduino mega), et le PORTB0 en entrée.

Attention

Ne pas oublier le int main() dans le fichier principal.

Penser à rajouter l'include standard io.h

```
1  #include <avr/io.h>
2  int main( void )
3  {
4          DDRB |= ( 1 << PORTB5 );
5          DDRB &= ~( 1 << PORTB0 );
6          PORTB |= ( 1 << PORTB0 );
7          while(42)
8          {
9                  if( (PINB & ( 1 << PORTB0 )) )
10                         PORTB |= ( 1 << PORTB5 );
11                 else
12                         PORTB &= ~( 1 << PORTB5 );
13         }
14         return 0;
15 }
```

C'est un octet en mémoire

C'est un octet en mémoire
qui permet de configurer le microcontrôleur.

C'est un octet en mémoire
qui permet de configurer le microcontrôleur.

C'est un octet en mémoire
qui permet de configurer le microcontrôleur.

SREG – AVR Status Register

C'est un octet en mémoire
qui permet de configurer le microcontrôleur.

SREG – AVR Status Register

- I (bit 7) – Global interrupt enable

C'est un octet en mémoire
qui permet de configurer le microcontrôleur.

SREG – AVR Status Register

- I (bit 7) – Global interrupt enable
- T (bit 6) – Copy storage

C'est un octet en mémoire
qui permet de configurer le microcontrôleur.

SREG – AVR Status Register

- I (bit 7) – Global interrupt enable
- T (bit 6) – Copy storage
- H (bit 5) – Half carry

C'est un octet en mémoire
qui permet de configurer le microcontrôleur.

SREG – AVR Status Register

- I (bit 7) – Global interrupt enable
- T (bit 6) – Copy storage
- H (bit 5) – Half carry
- S (bit 4) – Sign bit

C'est un octet en mémoire
qui permet de configurer le microcontrôleur.

SREG – AVR Status Register

- I (bit 7) – Global interrupt enable
- T (bit 6) – Copy storage
- H (bit 5) – Half carry
- S (bit 4) – Sign bit
- V (bit 3) – Overflow bit

C'est un octet en mémoire
qui permet de configurer le microcontrôleur.

SREG – AVR Status Register

- I (bit 7) – Global interrupt enable
- T (bit 6) – Copy storage
- H (bit 5) – Half carry
- S (bit 4) – Sign bit
- V (bit 3) – Overflow bit
- N (bit 2) – Negative bit

C'est un octet en mémoire
qui permet de configurer le microcontrôleur.

SREG – AVR Status Register

- I (bit 7) – Global interrupt enable
- T (bit 6) – Copy storage
- H (bit 5) – Half carry
- S (bit 4) – Sign bit
- V (bit 3) – Overflow bit
- N (bit 2) – Negative bit
- Z (bit 1) – Zero bit

C'est un octet en mémoire
qui permet de configurer le microcontrôleur.

SREG – AVR Status Register

- I (bit 7) – Global interrupt enable
- T (bit 6) – Copy storage
- H (bit 5) – Half carry
- S (bit 4) – Sign bit
- V (bit 3) – Overflow bit
- N (bit 2) – Negative bit
- Z (bit 1) – Zero bit
- C (bit 0) – Carry

Objectif :

Faire un programme qui fait clignoter une led en utilisant le TIMER1 sur 16 bits.

Pour cela, on fera :

Objectif :

Faire un programme qui fait clignoter une led en utilisant le TIMER1 sur 16 bits.

Pour cela, on fera :

- On activera les interruptions d'overflow du TIMER1

Objectif :

Faire un programme qui fait clignoter une led en utilisant le TIMER1 sur 16 bits.

Pour cela, on fera :

- On activera les interruptions d'overflow du TIMER1
- On fera compter le TIMER1 pour avoir une interruption toutes les secondes.

Objectif :

Faire un programme qui fait clignoter une led en utilisant le TIMER1 sur 16 bits.

Pour cela, on fera :

- On activera les interruptions d'overflow du TIMER1
- On fera compter le TIMER1 pour avoir une interruption toutes les secondes.
- On regardera l'état de la PIN associée à la LED pour le changer.

http ://github.com/leroilion/avr

http ://github.com/leroilion/avr

TCCR1A – TCCR1B (page 132 – 134)

Mode normal (WGMx = 0), Source d'horloge (CSx = 101), Pas de comparaison (COMx = 0)

http ://github.com/leroilion/avr

TCCR1A – TCCR1B (page 132 – 134)

Mode normal (WGMx = 0), Source d'horloge (CSx = 101), Pas de comparaison (COMx = 0)

TCNT1H – TCNT1L (page 136)

Incrémentation toute les 64μs

http ://github.com/leroilion/avr

TCCR1A – TCCR1B (page 132 – 134)

Mode normal (WGMx = 0), Source d'horloge (CSx = 101), Pas de comparaison (COMx = 0)

TCNT1H – TCNT1L (page 136)

Incrémentation toute les 64μs $\Longrightarrow$ Compter jusqu'à 15625

http ://github.com/leroilion/avr

TCCR1A – TCCR1B (page 132 – 134)

Mode normal (WGMx = 0), Source d'horloge (CSx = 101), Pas de comparaison (COMx = 0)

TCNT1H – TCNT1L (page 136)

Incrémentation toute les 64μs $\Longrightarrow$ Compter jusqu'à 15625 $\Longrightarrow$ Mettre 49910 = 65535 - 15625 dans TCNT.

http ://github.com/leroilion/avr

TCCR1A – TCCR1B (page 132 – 134)

Mode normal (WGMx = 0), Source d'horloge (CSx = 101), Pas de comparaison (COMx = 0)

TCNT1H – TCNT1L (page 136)

Incrémentation toute les 64μs $\Longrightarrow$ Compter jusqu'à 15625 $\Longrightarrow$ Mettre 49910 = 65535 - 15625 dans TCNT.

TIMSK1 (page 137)

Activer l'interruption d'overflow

http ://github.com/leroilion/avr

TCCR1A – TCCR1B (page 132 – 134)

Mode normal (WGMx = 0), Source d'horloge (CSx = 101), Pas de comparaison (COMx = 0)

TCNT1H – TCNT1L (page 136)

Incrémentation toute les 64μs $\Longrightarrow$ Compter jusqu'à 15625 $\Longrightarrow$ Mettre 49910 = 65535 - 15625 dans TCNT.

TIMSK1 (page 137)

Activer l'interruption d'overflow $\Longrightarrow$ Activer TOIE1

```
1 #include <avr/io.h>
2 #include <avr/interrupt.h>
3 ...
4 TCCR1A = 0bxxxxxx00;
5 TCCR1B = 0bxxx00101;
6 TIMSK1 |= ( 1 << TOIE1 );
7 ...
8 ISR( TIMER1_OVF_vect)
9 { ...
```

Objectif :

Faire un programme qui permet de contrôler la luminosité d'une LED en utilisant le PWM.

Objectif :

Faire un programme qui permet de contrôler la luminosité d'une LED en utilisant le PWM.

Détails techniques :

Objectif :

Faire un programme qui permet de contrôler la luminosité d'une LED en utilisant le PWM.

Détails techniques :

- Utiliser le port B5 pour la sortie de la LED (PORTB7 pour arduino mega).

Objectif :

Faire un programme qui permet de contrôler la luminosité d'une LED en utilisant le PWM.

Détails techniques :

- Utiliser le port B5 pour la sortie de la LED (PORTB7 pour arduino mega).
- Utiliser le TIMER de votre choix en mode PWM

Objectif :

Faire un programme qui permet de contrôler la luminosité d'une LED en utilisant le PWM.

Détails techniques :

- Utiliser le port B5 pour la sortie de la LED (PORTB7 pour arduino mega).
- Utiliser le TIMER de votre choix en mode PWM

Attention

Ne pas oublier l'include <avr/interrupt.h>

Penser à gérer *TOUS* les vecteurs d'interruption

Les pièges

Les pièges

- Économiser la mémoire (problème de la pile)

Les pièges

- Économiser la mémoire (problème de la pile)
- Faire attention avec les *float*, les . et les *double*

Les pièges

- Économiser la mémoire (problème de la pile)
- Faire attention avec les *float*, les . et les *double*
- Économiser la puissance de calcul (calcul en 8 bits)

Les pièges

- Économiser la mémoire (problème de la pile)
- Faire attention avec les *float*, les . et les *double*
- Économiser la puissance de calcul (calcul en 8 bits)
- Faire attention à l'overflow

Les pièges

- Économiser la mémoire (problème de la pile)
- Faire attention avec les *float*, les . et les *double*
- Économiser la puissance de calcul (calcul en 8 bits)
- Faire attention à l'overflow
- Rajouter l'option *volatile* devant les variables

Nous avons vu :

Nous avons vu :

- Les outils de programmation

Nous avons vu :

- Les outils de programmation
- Comment configurer, lire et écrire sur un port

Nous avons vu :

- Les outils de programmation
- Comment configurer, lire et écrire sur un port
- Ce qu'est un registre, et comment le configurer à l'aide du datasheet

Nous avons vu :

- Les outils de programmation
- Comment configurer, lire et écrire sur un port
- Ce qu'est un registre, et comment le configurer à l'aide du datasheet
- L'utilisation des interruptions

Nous avons vu :

- Les outils de programmation
- Comment configurer, lire et écrire sur un port
- Ce qu'est un registre, et comment le configurer à l'aide du datasheet
- L'utilisation des interruptions

Des exemples simple :

Nous avons vu :

- Les outils de programmation
- Comment configurer, lire et écrire sur un port
- Ce qu'est un registre, et comment le configurer à l'aide du datasheet
- L'utilisation des interruptions

Des exemples simple :

- http ://github.com/leroilion/avr

Bibliographie :

Nous avons vu :

- Les outils de programmation
- Comment configurer, lire et écrire sur un port
- Ce qu'est un registre, et comment le configurer à l'aide du datasheet
- L'utilisation des interruptions

Des exemples simple :

- http ://github.com/leroilion/avr

Bibliographie :

- Microcontrôleurs AVR : des ATtiny aux ATmega de Christian Tavernier