Informatique embarquée 5/16

J.-M Friedt

FEMTO-ST/département temps-fréquence

jmfriedt@femto-st.fr

transparents à `jmfriedt.free.fr`

16 février 2017

Masques & décalages

```
static int dio48e_gpio_direction_input(struct gpio_chip *chip, unsigned offset)
{
 struct dio48e_gpio *const dio48egpio = gpiochip_get_data(chip);
 const unsigned io_port = offset / 8;
 const unsigned int control_port = io_port / 3;
 const unsigned control_addr = dio48egpio->base + 3 + control_port*4;
 unsigned long flags;
 unsigned control;

 spin_lock_irqsave(&dio48egpio->lock, flags);

 /* Check if configuring Port C */
 if (io_port == 2 || io_port == 5) {
   /* Port C can be configured by nibble */
   if (offset % 8 > 3) {
     dio48egpio->io_state[io_port] |= 0xF0;
     dio48egpio->control[control_port] |= BIT(3);
   } else {
     dio48egpio->io_state[io_port] |= 0x0F;
     dio48egpio->control[control_port] |= BIT(0);
   }
 } else {
   dio48egpio->io_state[io_port] |= 0xFF;
   if (io_port == 0 || io_port == 3)
     dio48egpio->control[control_port] |= BIT(4);
   else
     dio48egpio->control[control_port] |= BIT(1);
   }
   control = BIT(7) | dio48egpio->control[control_port];
   outb(control, control_addr);
   control &= ~BIT(7);
   outb(control, control_addr);
   spin_unlock_irqrestore(&dio48egpio->lock, flags);
   return 0;
}
```

http://lxr.free-electrons.com/source/drivers/gpio/gpio-104-dio-48e.c

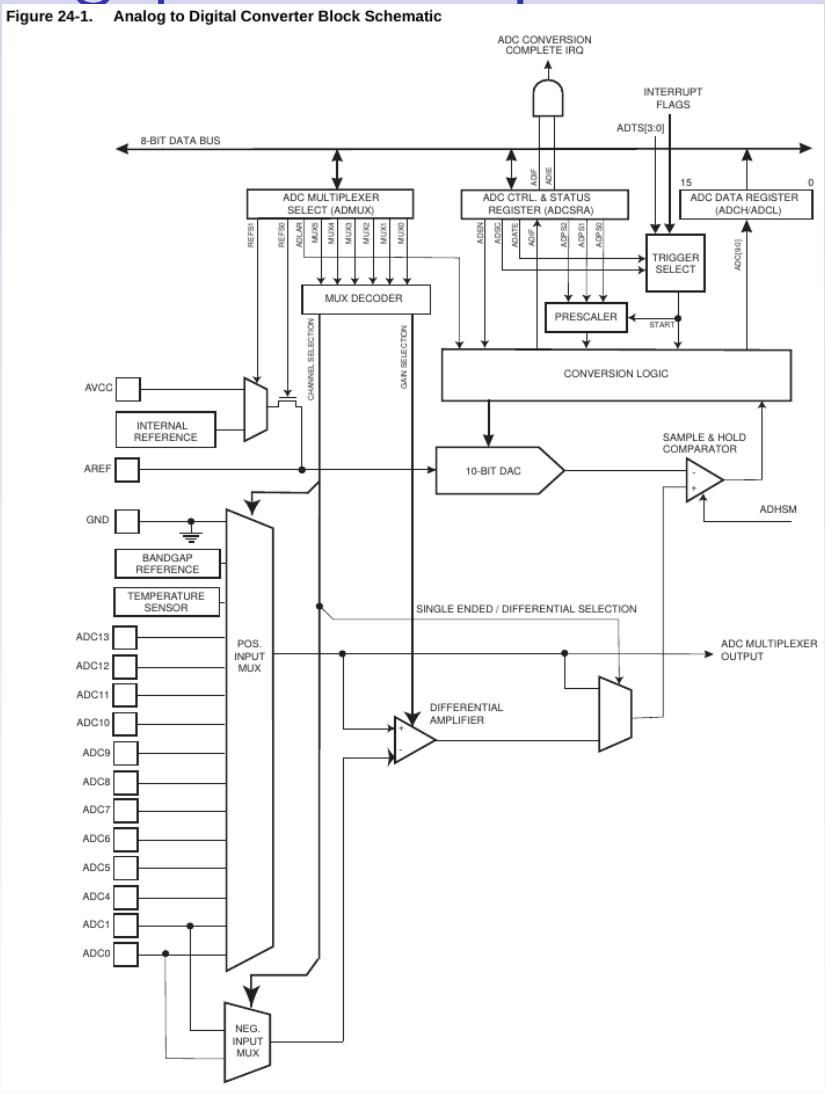
(#define BIT(nr) (1UL << (nr)))

Convertisseur analogique-numérique

1. Les grandeurs physiques sont continues (température, tension, courant, force ...).
2. Un microcontrôleur manipule des grandeurs discrètes (bit, octet, mot).
3. Composant faisant le lien entre les deux mondes : convertisseur analogique-numérique (ADC)
4. 12-bits, approximation successive.
5. Caractéristiques : résolution, exactitude

$$bits = \frac{V_{in}}{V_{ref}} \cdot (2^N - 1)$$

$$V_{in} \in [0 : V_{ref}]$$



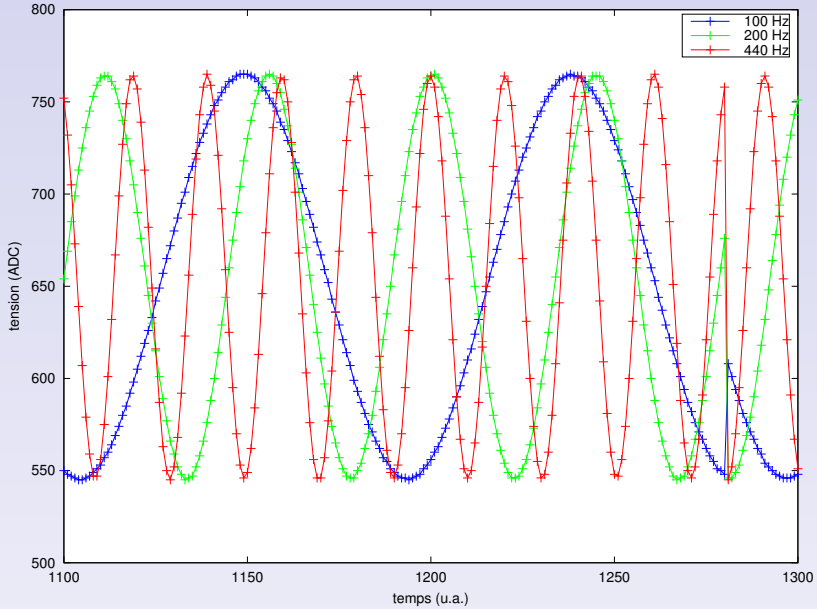
Atmega 32U4

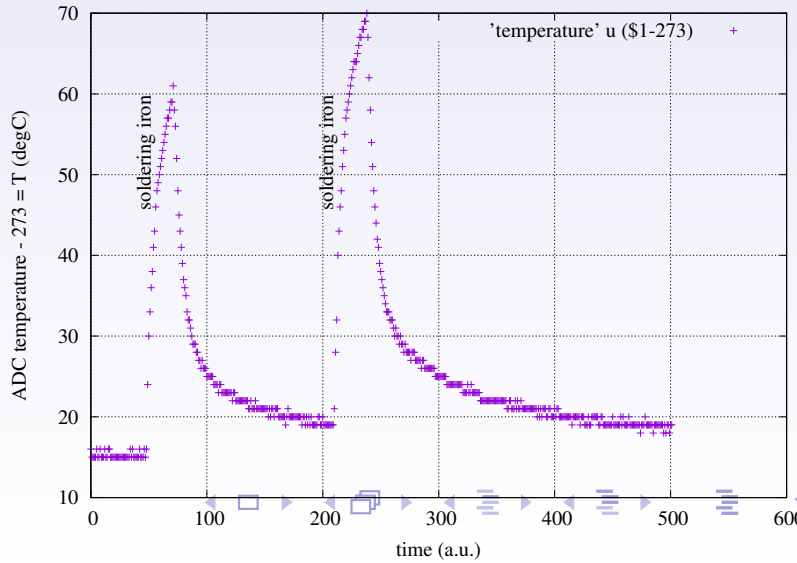
Convertisseur analogique-numérique

1. Les grandeurs physiques sont continues (température, tension, courant, force ...).
2. Un microcontrôleur manipule des grandeurs discrètes (bit, octet, mot).
3. Composant faisant le lien entre les deux mondes : convertisseur analogique-numérique (ADC)
4. 12-bits, approximation successive.
5. Caractéristiques : résolution, exactitude

$$bits = \frac{V_{in}}{V_{ref}} \cdot (2^N - 1)$$

$$V_{in} \in [0 : V_{ref}]$$





Méthodes de déverminage de code embarqué

1. la partie algorithmique du code est portable et se compile sur PC
   $\Rightarrow$ bien séparer les parties algorithmiques et accès au matériel
   $+$ donne accès aux outils de profilage et anayse du PC
2. l'émulateur (plus souple que le matériel, et permet de sonder la machine à états qu'est le processeur)
3. sonde JTAG et gdb (si disponible) : exécution sur le "vrai" matériel avec points d'arrêt et avance pas à pas.

Compilation séparée pour qualifier un logiciel

- tester des conditions de fonctionnement imprévues ou rarement accessibles expérimentalement
- exploiter des outils de déverminage (valgrind) ou de profilage
- test unitaire

Programme principal, bibliothèque

```
int main()
{initialisation();
 while (1) {interroge();traitements();}
}
```

Programme pour microcontrôleur

```
unsigned short interroge (unsigned int freq)
{ GPIO_SetBits(GPIOB,GPIO_Pin_1);
  v = readADC1 (ADC_Channel_8);
  GPIO_ResetBits  (GPIOB,GPIO_Pin_1);
...
  return (v);
}
```

Compilation séparée pour qualifier un logiciel

- tester des conditions de fonctionnement imprévues ou rarement accessibles expérimentalement
- exploiter des outils de déverminage (valgrind) ou de profilage
- test unitaire

Programme principale, bibliothèque

```
int main()
{initialisation();
 while (1) {interroge();traitements();}
}
```

Programme pour PC

```
unsigned short interroge (unsigned int freq)
 reponse =exp(-(((float)freq-f01)/df)*(((float)freq-f01)/df))*3100.;
 reponse+=exp(-(((float)freq-f02)/df)*(((float)freq-f02)/df))*3100.;
 reponse+=(float)((rand()-RAND_MAX/2)/bruit); // ajout du bruit;
 if (reponse<0.) reponse=0.;
 if (reponse>4095.) reponse=4095.;
 usleep(60);
 return((unsigned short)reponse);
```

Cas des interruptions

Version PC

```
void handle_alarm(int xxx)
{ //  printf("RTC : %d %d %d\n",seconde,→
      ↪minute,heure);
  tim0+=10;
  seconde+=10;
  if (seconde>600){minute++;seconde=0;}
  if (minute>60) {heure++;minute=0;}
  alarm(1);
}

void handle_io(int xxx)
{ // io_happened=1;
  global_tab[global_index] = getchar ();
  if (global_index < (NB_CHARS - 1))
     global_index++;
}

int main() {
  signal(SIGALRM, handle_alarm);

  io_happened=0;
  struct termios buf;
  tcgetattr(0, &buf);
  buf.c_lflag &= ~(ECHO | ICANON);
  buf.c_cc[VMIN]=1;
  buf.c_cc[VTIME]=0;
  tcsetattr(0, TCSAFLUSH, &buf);
  signal(SIGIO, handle_io);
    int savedflags=fcntl(0, F_GETFL, 0);
  fcntl(0, F_SETFL, savedflags | O_ASYNC | →
       ↪O_NONBLOCK );
  fcntl(0, F_SETOWN, getpid());
  alarm(1);  // si on veut simuler le timeout
}
```

Version microcontroleur

```
void tim3_isr(void)  // VERSION LIBOPENCM3
{
 if (TIM_GetITStatus(TIM3,TIM_IT_Update)!=→
      ↪RESET)
  {TIM_ClearITPendingBit(TIM3,TIM_IT_Update);
   tim0++;
   seconde++;          // seconde en 1/10 →
        ↪seconde
   if (seconde > 600)
     {seconde = 0;minute++;}
  if (minute > 60)
     {minute = 0;heure++;}
  if (heure > 24)
     {heure = 0;}
  }
}

void usart1_isr(void)
{
  if( USART_GetITStatus(USART1, USART_IT_RXNE→
       ↪))
    {USART_ClearITPendingBit(USART1,→
         ↪USART_IT_RXNE);
     USART_ClearFlag(USART1,USART_IT_RXNE);
     {
      global_tab[global_index]=→
           ↪USART_ReceiveData(USART1);
      if (global_index<(NB_CHARS-1)) →
           ↪global_index++;
     }
    }
}
```

Profiter des outils d'analyse de code

`valgrind` pour étudier les fuites de mémoire

```
int main()
{int i[3],k;
 for (k=0;k<10;k++) {i[k]=k;printf("%d ",i[k]);}
}
```

qui s'exécute pour donner `Segmentation fault` ...

... est analysé par `valgrind -q ./mon_programme` et donne

```
==7410==  Access not within mapped region at address 0x2
==7410==    at 0x8048448: main (demo_valgrind.c:5)
```

(noter la différence de comportement en ajoutant `fflush(stdout)` ou `\n` !)

Profiter des outils d'analyse de code

`gprof` pour étudier le temps d'exécution de chaque fonction

```
void fonction1s() {volatile int k;for (k=0;k<0x1000000;k++) {};}
void fonction2s() {volatile int k;for (k=0;k<0x2000000;k++) {};}
void fonction3s() {volatile int k;for (k=0;k<0x3000000;k++) {};}

int main()
{fonction1s();
 fonction2s();
 fonction3s();
}
```

qui se compile avec l'option `-pg` pour générer à l'exécution `gmon.out` ...

... qui s'analyse par `gprof -bp ./mon_programme` et donne

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	name
55.56	0.10	0.10	1	100.00	100.00	fonction3s
33.33	0.16	0.06	1	60.00	60.00	fonction2s
11.11	0.18	0.02	1	20.00	20.00	fonction1s

Exprimer un problème sous forme d'entiers

- Exemple du filtre :

$$y_n = \sum_{k=0}^{N-1} b_k \cdot x_{n-k}$$

- Soient des coefficients $b_k \in [-1, 1]$
- Comment effectuer ce calcul sur un microcontrôleur ?
- Supposons que les acquisitions x sont sur u bits, le résultat du calcul y sur v bits, comme exprimer b en entiers ?

Exprimer un problème sous forme d'entiers

- Exemple du filtre :

$$y_n = \sum_{k=0}^{N-1} b_k \cdot x_{n-k}$$

- Soient des coefficients $b_k \in [-1, 1]$
- Comment effectuer ce calcul sur un microcontrôleur ?
- Supposons que les acquisitions x sont sur u bits, le résultat du calcul y sur v bits, comme exprimer b en entiers ?
- Si w bits pour b, alors $b_k \cdot x_{n-k}$ s'exprime sur $w + u$ bits

Exprimer un problème sous forme d'entiers

- Exemple du filtre :

$$y_n = \sum_{k=0}^{N-1} b_k \cdot x_{n-k}$$

- Soient des coefficients $b_k \in [-1, 1]$
- Comment effectuer ce calcul sur un microcontrôleur ?
- Supposons que les acquisitions x sont sur u bits, le résultat du calcul y sur v bits, comme exprimer b en entiers ?
- Si w bits pour b, alors $b_k \cdot x_{n-k}$ s'exprime sur $w + u$ bits
- $\sum_{k=0}^{N-1}$ ajoute $\log_2(N)$ bits

Exprimer un problème sous forme d'entiers

- Exemple du filtre :

$$y_n = \sum_{k=0}^{N-1} b_k \cdot x_{n-k}$$

- Soient des coefficients $b_k \in [-1, 1]$
- Comment effectuer ce calcul sur un microcontrôleur ?
- Supposons que les acquisitions x sont sur u bits, le résultat du calcul y sur v bits, comme exprimer b en entiers ?
- Si w bits pour b, alors $b_k \cdot x_{n-k}$ s'exprime sur $w + u$ bits
- $\sum_{k=0}^{N-1}$ ajoute $\log_2(N)$ bits
- donc $v = w + u + \log_2(N) \Leftrightarrow w = v - u - \log_2(N)$

Mise en pratique

Nous quittons l'AVR qui a choisi de développer sa propre libc, pour passer au STM32

- processeur 32 bits capable d'embarquer des bibliothèques (puis environnements exécutifs)
- `summon-arm-toolchain` fournit un script pour générer un ensemble "cohérent" d'outils – explication de sa compilation à `http://jmfriedt.free.fr/summon_arm.pdf`
- le processeur se décline en une multitude de classes : *linker* script approprié.
- Émulateur pour STM32 : `qemu` pour ARM complété de l'émulation de périphériques – `https://github.com/beckus/qemu_stm32`.

Les divers outils de compilation : `Makefile`, `configure`, `cmake`
Les divers outils d'archivage : `svn`, `git`, `hg` (mercurial)