Calibration AT Tiny13

Les familles Atmel de microcontrôleurs Atmel peuvent fonctionner avec une référence de tension interne et avec une horloge interne. La précision de ces références n'est pas excellente, mais suffisante dans un grand nombre de cas. Il est toutefois utile de connaître leur erreur.
Pour étalonner la tension, il suffit de mettre une tension juste inférieure à la tension interne de référence (pour la meilleure précision de mesure), de la mesurer avec un voltmètre un tant soit peu sérieux, et de comparer à la tension que le processeur comprend. La lecture est faite en convertissant la sortie du convertisseur ADC en digits décimaux qu'il suffit de transférer en mode série vers un terminal. Sous Linux le terminal peut être minicom ou gtkterm, sous Windaube c'est le hyperterminal classique (gogoliser pour savoir comment l'installer sous les dernières versions de Windaube). Un convertisseur de niveau_inverseur type MAX232 ou un convertisseur TTL-USB comme le FT232 permettent la liaison physique avec l'ordinateur. Un simple transistor peut dans de nombreux cas, faire l'affaire. La configuration bit doit être réglée à 1200 bauds, 8 bits, pas de parité, 1 ou 2 bits stop (1200-8N1, le montage envoie 2 stop bits, donc lisible en 1 ou 2 stops) qui sont inscrits dans le programme.
La fréquence interne du microcontrôleur est suffisamment précise pour permettre le transfert du signal série sans erreur.

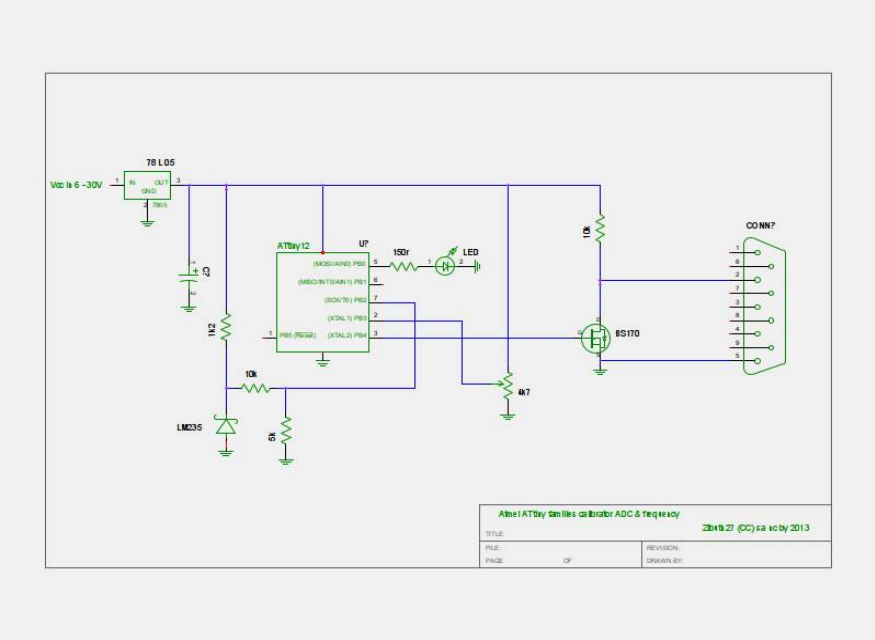
Fonctionnement

Etalonnage de l'horloge interne :

La sortie PB0 (pin5) est programmée pour générer une impulsion de durée 8,53ms toutes les 218,45ms (4,5776Hz), cette même fréquence, divisée par deux, est disponible sur PB1. A mesurer avec un appareil lui-même étalonné (oscilloscope, comparaison avec un étalon, fréquencemètre, etc)
Le temps réel est un peu différent de ce temps théorique, en fonction de la température et de la tension d'alimentation, en tenir compte pour une réalisation à base de ce composant maintenant étalonné.

Etalonnage de la référence de tension interne :

Placer une source de tension étalon de 1V sur PB3 (cas idéal, si on dispose de cette source). Sinon placer un potentiomètre entre à et 5V, mettre un condensateur d'une dizaine de μF entre curseur et masse, curseur relié à PB3.
10ms après l'impulsion de PB0, pour être hors des phases de commutation et de leurs effets sur l'entrée analogique PB3, le convertisseur est lancé pour 10 acquisitions moyennées.
Le résultat de la conversion est converti en 4 chiffres décimaux qui sont envoyés en mode série.
Ce montage avec certaines précautions (moyennage sur dix mesures, acquisition synchrone loin des interruptions, condensateur) permet d'observer une variation de 1 LSB seulement.



Le montage finalement réalisé utilise un simple transistor MOS pour l'adaptation RS232 (un BS170 avec charge de 10kohms vers le +5). Résultat OK sur le port série de mon ordinateur. Officiellement, il faudrait un MAX232 ou un adaptateur USB comme le FT232.

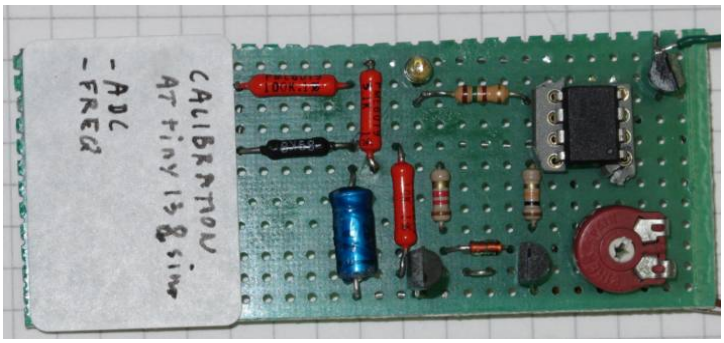
 zibuth27@gmx.fr www.hackerschicken.eu

Calibration AT Tiny13

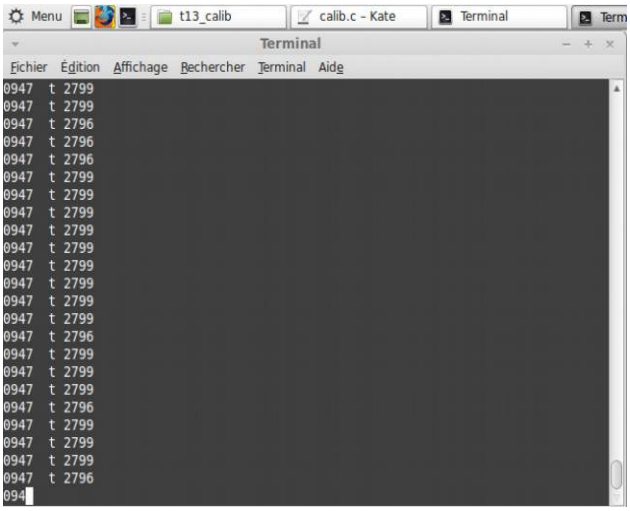
Le but est d'étalonner la référence de tension, la même pour les quatres entrées analogiques du microcontrôleur, il suffit donc d'utiliser une seule des entrées : ADC3 (pin 2). J'ai fait un essai en mettant sur une autre entrée un LM235 (capteur de température analogique) mais lorsque j'ai tenté d'utiliser des variables flottantes pour ajuster la mesure en fonction de la précédente calibration, le compilateur a déclaré forfait : la taille dépassait la capacité de la mémoire pogramme. Les valeurs sont donc lues brutes, sans correction de la référence de tension ni de l'erreur du diviseur de tension (obligatoire car la tension de l'ordre de 3V, dépasse la référence interne).

La tension de référence interne se déduit par une simple règle de trois
Vref= tension lue au voltmètre x 1024 / tension lue (RS232) dans l'exemple ci-dessous la tension voltmètre est de 1,013V ce qui fait une référence interne à 1,095V pour le contrôleur étalonné (pour 1,0 à 1,2 V spécifiés dans la datasheet) soit une erreur de 0,5 %.
Pour retrouver la fréquence d'horloge, il suffit de multiplier la fréquence de PB0 par 262144 (1024 x 256)

Le montage terminé
avec le LM235 monté

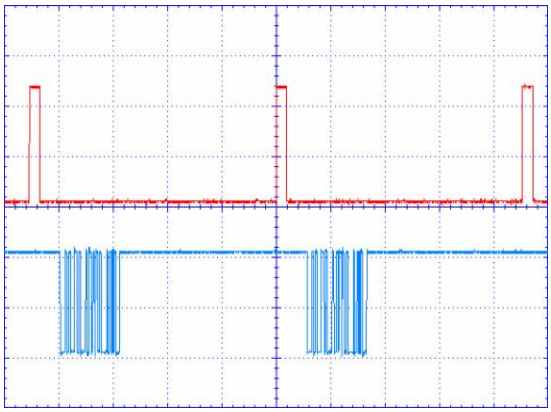


Copie d'écran de la mesure
avec minicom sous LinuxMint
avec transmission de la mesure
du LM235



Signaux (sans température) :

x= 50ms
y rouge= 2V PB0
y bleu = 2V PB4 : RS232 inverse



 zibuth27@gmx.fr www.hackerschicken.eu

Calibration AT Tiny13

Programme

```
/* microcontroller calibration
 *
 * attiny13 & AVR families
 *
 * Zibuth277 24/05/2013
 * (CC) 2013 Creative Commons by,nc,sa
 * status: complete
 * keywords:  ADC 8 bits, frequency, calibration, conversion ADC to 4 dec digits
 */

/* connections
*       PB0 pin 5 PWM out for freq cal
*       PB2 pin 7 LKM235 temperature in
*       PB3 pin 2 to input voltage
*       PB4 pin 3 RS232 inverted out
*       78L05 for powering tiny13 pin 8
*/

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <avr/sfr_defs.h>
#include "avr_compat.h"


void uart_send(unsigned char word)
{       // needs : #include <util/delay.h>
  char i;
 int bit_dur=815;      // 1200 bauds
 // word = ~c;         //complement char

  PORTB &=~_BV(PB4);  // start bit
  _delay_us(bit_dur);
 for(i=0;i<8;i++)
  {
    if((word & 0x01)==1)      PORTB |= _BV(PB4);    // if bit is one
    else                      PORTB &=~_BV(PB4);    // if bit is zero
      word = word/2;
      _delay_us(bit_dur);
  }
  PORTB |=_BV(PB4);
  _delay_us(bit_dur); // 2 stop bits
  _delay_us(bit_dur);
}

int main(void)
{
uint8_t hibyte, lobyte, r1,r2,r3,r4,n=10,i, prescaler=5;
uint16_t valeur;
    DDRB = 0x13;              // output OC1A enable PB0
    PORTB=0x02;                       //
    TCCR0A=0x83;              // mode 3 fast PWM, toggles at OCROA
    TCCR0B=0x05;              // clock speed = clk/1024
    OCR0A=10;
    ADMUX=0x43;                       // ADC3 analog ref = Vcc = internal ref
    ADCSRA=0x86;              // ADC continuous prescaler 64
   TIMSK0 = 0x02;             //(1 << TOIE0);
   sei();

  while (1)           // endless
  {

       loop_until_bit_is_clear(PINB,PB0);   // wait end of PB0 pulse
       _delay_ms(10);                       // wait for settlement
       for(i=0;i<n;i++){
       sbi(ADCSRA,ADSC);             // start conv
       while(ADCSRA& (1<<ADSC));     // wait ADC ready
       lobyte=ADCL;                  // read ADCL first: mandatory
       hibyte=ADCH;
       valeur=valeur+(lobyte+(hibyte*256));
       }
       valeur=valeur/n;
```

```
        r4=valeur/1000;
        valeur=valeur-(1000*r4);
        r3=valeur/100;
        valeur=valeur-(100*r3);
        r2=valeur/10;
        valeur=valeur-(10*r2);
        r1=valeur;

        uart_send(r4+48);
        if(r3>9)r3=r3+65;
        else r3=r3+48;
        uart_send(r3);
        if(r2>9)r2=r2+65;
        else r2=r2+48;
        uart_send(r2);
        if(r1>9)r1=r1+65;
        else r1=r1+48;
        uart_send(r1);
        uart_send(32);
        uart_send(13);

        loop_until_bit_is_set(PINB,PB0);     // wait end of PB0 pulse


//      if bit_is_clear(PINB,PB1) TCCR0B=0;
//      else TCCR0B=prescaler;
/*
insert a level translator MAX232 or FT1302 for USB an run terminal program
1200-8N1 (minicom or hyperterminal)
put a calibrated voltage on ADC, close to the internal reference value, just a bit lower
for maximum accuracy and to avoid confusion with saturation. Compare with the reading of
accurate voltmeter
The voltmeter value, divided by the calibrator reading and multiplied by 1024,
gives the internal ref value.

for clock measurement put PB1 (pin6) to +5V
clock frequency 1.000 000 MHz makes 3.8146 Hz (or 10.24 ms Hi state) on PB0

*/


  }
  return 0;
}

ISR(TIM0_OVF_vect)
{
  PORTB ^= (1 << PB1);
}
```

Mesures :

tiny13 #01 ref= 1,095 V clk=1,1702MHz
tiny13 #02 ref= 1,076 V clk=1,2024MHz

 zibuth27@gmx.fr www.hackerschicken.eu