

MICROCONTROLEURS MICROCONTROLEURS ET SYSTEMES NUMERIQUES TRAVAIL PRATIQUE NO 11

MT GROUPE A	MT Groupe B	EL		
No du Groupe	Premier Etudiant	Second Etudiant	Evaluation	Visa Correcteur

11. INTERFACES ET PÉRIPHÉRIQUES USUELS 2: DALLAS 1-WIRE™ ET CAPTEUR DE TEMPÉRATURE, CLAVIER PC

Ce travail pratique voit l'interfaçage de divers périphériques mettant en oeuvre le convertisseur analogique-numérique interne au microcontrôleur, ainsi que divers protocoles de transmission série. Ce TP se concentre sur les thèmes suivants:

- clavier PC, et
- Dallas 1-Wire™ et capteur de température.

11.1 CLAVIER PC

L'étude du lien entre un PC et son clavier se fait en deux étape. La communication est abordée avant de considérer l'interprétation des informations transmises. Avant toute chose, ATTENTION: ne déconnectez pas les claviers de vos PCs pour cette manipulation, c'est strictement interdit; des claviers sont disponibles dans les armoires.

11.1.1 COMMUNICATION AVEC UN CLAVIER PC AT

La communication entre un clavier et un PC nécessite deux lignes: et . C'est une communication de type s s .

Le signal sur la ligne données est valide au moment du flanc de l'horloge. Un mot transmis consiste en 11 bits consécutifs:

- 1 bit de , suivi de
- 8 bits de , suivis de
- 1 bit de , et finalement se terminant par
- 1 bit de .

Placez le module M4 sur le port D. Téléchargez le programme keyboard1.asm. Placez une sonde connectée au canal 1 de l'oscilloscope sur le point PM4, et une sonde connectée au canal 2 sur le point de mesure PM5. Connectez un clavier qui vous est PS2 au connecteur DIN6 du module. Quel signal observez-vous sur PM4 et PM5, respectivement ?

Configurez l'oscilloscope comme indiqué en Figure 11.1. Placez le signal de manière à ce que les onze coups d'horloge soient pleinement visibles, c'est-à-dire bougez la position du trigger vers la gauche. Reportez les trois signaux data observés correspondant au code des lettres "4," "g" et "F5."

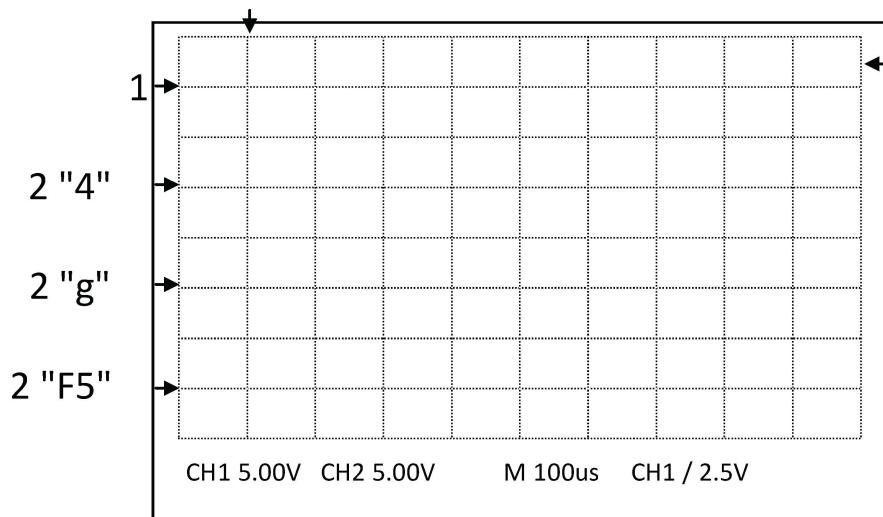


Figure 11.1: Transmission depuis le clavier au microcontrôleur.

Pour observer les signaux suivants, vous pouvez par exemple utiliser l'oscilloscope configuré en SINGLE SEQ avec un trigger à 2.5 V déclenché au flanc descendant de CH1; l'oscilloscope effectuera alors une seule acquisition de trame à l'appui d'un caractère au clavier, et la gardera à l'affichage, jusqu'à ce que vous demanderez une nouvelle acquisition en appuyant sur SINGLE SEQ à nouveau, et qu'un caractère soit pressé au clavier.

Quels sont donc les codes sur 11-bits transmis pour les différentes touches suivantes (il faut enlever les bits dûs au protocole et remettre le flot dans le bon ordre) - n'utilisez pas le pavé numérique du clavier:

- “4:”
 - flot brut observé (11-bits) = ,
 - code caractère = 0b = 0x ;
- “g:”
 - flot brut observé (11-bits) = ,
 - code caractère = 0b = 0x ;
- “F5:”
 - flot brut observé (11-bits) = ,
 - code caractère = 0b = 0x .

Appuyez sur une touche quelconque du clavier, puis appuyez sur la touche SINGLE SEQ de l'oscilloscope afin de réaliser une acquisition de trame, puis relâchez la touche du clavier et observez le changement dans les signaux. Quel est le code qui semble être envoyé au relâchement d'une touche (env. 3ms plus tard) ? 0x .

En réalité, un code unique est émis lorsqu'une touche est activée, c'est le “make.” Puis une autre code unique est émis au moment où la touche est désactivée, c'est le “break.” Le break est constitué de deux bytes. Modifiez votre base de temps sur 1ms et observez les deux bytes.

Quel est le comportement à l'envoi d'un signe en majuscule ? Combien de make et brake sont émis ? Combien de bytes sont émis ? Effectuez la manipulation lentement et observez le signaux.

Ainsi, à qui revient la tâche d'interpréter le type du caractère envoyé ? [] .

L'identification d'un caractère émis par le clavier est basé sur l'échantillonnage du bit de donnée sur le flanc descendant du signal d'horloge. La macro CLK10 (parfois aussi nommée DETECT_10) effectue la détection de flanc descendant. Nous étudions le comportement de cette macro.

CLK10 n'attend pas indéfiniment une transition, mais utilise un timeout. Quel est le timeout de cette boucle ? Une itération de la boucle dure [] cycles. Un timeout a lieu, si après [] itérations, il n'y a pas eu de changement de niveau sur la ligne observée. Le timeout est donc [] microsecondes. Deux sorties de la macro sont possibles, soit :

- l'instruction suivante dans le cas où une transition est détectée, ou
- l'adresse indiqué par l'étiquette timeout si il n'y a pas eu de transition (dans notre cas le branchement se fait vers le main).

CLK10 (DETECT_10) est formé de deux parties. Rajoutez les chemins manquant du diagramme de flux en Figure 11.2.

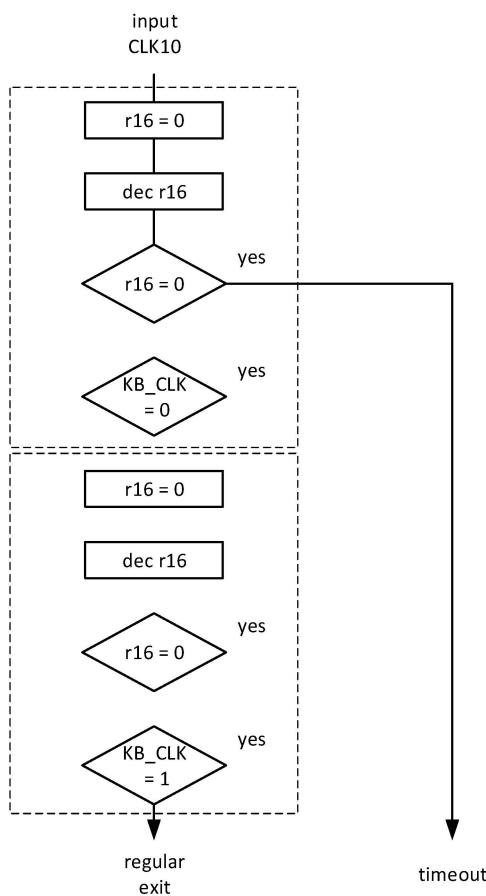


Figure 11.2: Diagramme de flux de la macro CLK10 (DETECT_10).

A quoi sert la première moitié de la macro:

- si la ligne KB_CLK est à ‘0’, [] ;
- si la ligne KB_CLK est à ‘1’, [] ;
- donc, la première moitié de la macro sert à détecter [].

A quoi sert la seconde moitié de la macro:

- si la ligne KB_CLK est à ‘0’, [] ;
- si la ligne KB_CLK est à ‘1’, [] ;
- donc, la deuxième moitié de la macro sert à détecter [].

Ainsi la méthode de détection d'un flanc descendant consiste à [].

11.1.2 DÉCODAGE À L'AIDE D'UNE LOOKUP-TABLE (LUT)

Les codes associés aux touches correspondent à une position physique sur le clavier. D'après le pays et la langue utilisée, différents caractères correspondent à ces touches. Le code n'est pas complet. Appuyez sur les lettres minuscules de l'alphabet ainsi que les chiffres, et complétez la Table 11.1

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00																
10					[]	[]				[]	[]	[]	[]	[]	[]	[]
20	[]	[]	[]	[]	[]	[]				[]	[]	[]	[]	[]	[]	[]
30	[]	[]	[]	[]	[]	[]			,	[]	[]	[]	[]	[]	[]	[]
40	[]	[]	[]	[]	[]	[]			.	[]		[]	[]	[]	[]	[]
50					\					"		"				
60	<									[]		[]	[]			
70	[]	[]	[]	[]	[]	[]				[]	[]	[]	[]	[]		

Table 11.1: Table de transcodage des caractères transmis par le clavier en équivalent ASCII.

Chargez et testez le programme keyboard2.asm. Ce programme utilise une table de transcorrespondance (LUT) afin d'associer à chaque code du clavier le code ASCII de la lettre correspondante. Les entrées “vides” dans le tableau retournent le code ASCII du caractère espace.

Ecrivez la macro LOOKUP en Figure 11.3 qui prend un registre résultat, un registre index, et l'adresse du début de la table. La macro à écrire ne conserve pas le pointeur z.

11.2 DALLAS 1-WIRE™

Le protocole 1-wire de Dallas n'utilise qu'un seul fil pour communiquer et même fournir l'alimentation à des modules périphériques multiples. Placez le module M5 sur le port B, et placez la sonde de l'oscilloscope sur le point de test PM10. Téléchargez le programme wire1_1.asm donné en Figure 11.4. Ce programme initialise l'interface 1-wire; il fait un reset et envoie ensuite un byte sur la ligne.

Configurez l'oscilloscope comme indiqué en Figure 11.5, et reportez-y la trace du signal observé. Indiquez sur la trace les éléments suivants:

```
.macro LOOKUP      ;result,index,tbl
    [ ] zl,@1      ; move index into z_low
    clr [ ]          ; clear z_high
    [ ] zl, low(-2*@2) ; add base address of table
    [ ] zh,high(-2*@2)
    mov [ ],r0       ; load program memory (into r0)
    .endmacro         ; move r0 to result register
```

Figure 11.3: Macro LOOKUP.

```
; file wire1_1.asm target ATmega128L-4MHz-STK300
; purpose Dallas 1-wire(R) interfacing: temperature readout
; module: M5, input port: PORTB
.include "macros.asm"      ; include macro definitions
.include "definitions.asm"

; === initialization (reset) ====
reset:
    LDSP    RAMEND      ; load stack pointer (SP)
    rcall   wire1_init   ; initialize 1-wire(R) interface
    rjmp   main

.include "wire1.asm"        ; Dallas 1-wire(R) routines

main: rcall   wire1_reset   ; reset 1-wire(R) interface
     ldi      a0, 0xf0      ; load the byte to write into reg a0
     rcall   wire1_write   ; call routine to write 1 byte
     WAIT_MS 10
     rjmp   main
```

Figure 11.4: wire1_1.asm.

- le reset pulse,
- le presence pulse,
- les datas qui sont 0b[].

Etudiez la transaction, et complétez les affirmations suivantes:

- la transaction exécutée par wire1_1.asm ne constitue que le début d'une communication, dans lequel le maître vérifie [];
- à l'état de repos la ligne est à []. Toute information s'exprime avec des pulses de polarité [];
- en vous aidant de la fonction CURSOR, mesurez la longueur des différents pulses; le pulse reset fait une remise à l'état initial des modules 1-wire; il est généré par le [] et est le plus long et mesure [] microsecondes;
- le module [] répond avec un pulse de présence plus court, d'une durée égale à [] microsecondes;
- un bit 1 est représenté avec un pulse très court d'environ [] microsecondes;
- un bit 0 est représenté avec un pulse plus long d'environ [] microsecondes.

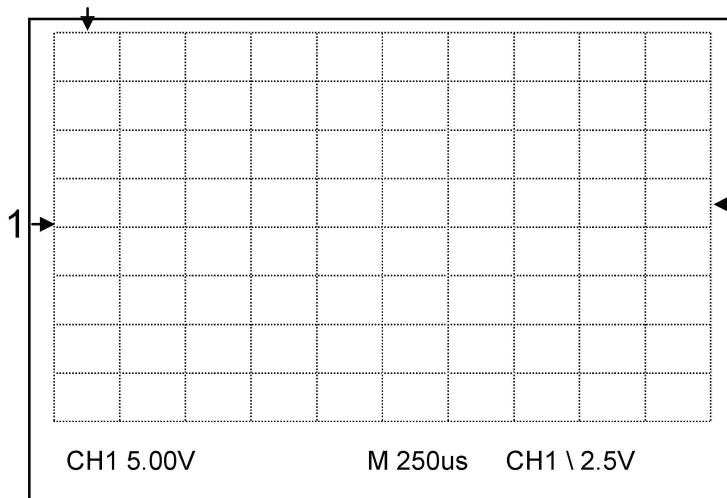


Figure 11.5: Signal observé lors de l'exécution de wire1_1.asm.

Téléchargez le programme wire1_temp2.asm donné en Figure 11.6.

Ce programme lit la température donnée par le capteur de température DS18B20 et l'affiche sur l'écran LCD. Si vous prenez le capteur entre le pouce et l'index, la température affichée augmente.

Etudiez le programme wire1_temp2.asm, puis reportez en Figure 11.7 la séquence sur laquelle est transmise la température.

La séquence de lecture de la température est située

La configuration de l'oscilloscope telle que décrite en Figure 11.7 ne suffit pas. Il faut modifier la position horizontale de la trace afin d'avoir un retard permettant l'observation de la séquence, et utiliser les fonctions RUN/STOP et/ou SINGLE SEQ afin de figer une mesure affichée.

Comment reconnaît-on facilement le début de la séquence de données thermométriques ?

Comment doit-on interpréter le flot de données (bit-stream) transmis ?

Quel est le code température transmis dans votre cas ? 0b et 0b, qui doivent donc être interprétés comme 0x et 0x, qui finalement correspondent à une température de °C.

```

; file      wire1_temp2.asm
; purpose Dallas 1-wire(R) temperature sensor interfacing: temperature
; module: M5, input port: PORTB
.include "macros.asm"          ; include macro definitions
.include "definitions.asm"    ; include register/constant definitions

; === initialization (reset) ===
reset:
    LDSP      RAMEND      ; load stack pointer (SP)
    rcall    wire1_init    ; initialize 1-wire(R) interface
    rcall    lcd_init     ; initialize LCD
    rjmp    main

.include "lcd.asm"           ; include LCD driver routines
.include "printf.asm"        ; include formatted printing routines
.include "wire1.asm"         ; include Dallas 1-wire(R) routines

; === main program ===
main:
    rcall    wire1_reset    ; send a reset pulse
    CA      wire1_write, skipROM ; skip ROM identification
    CA      wire1_write, convertT ; initiate temp conversion
    WAIT_MS 750                ; wait 750 msec

    rcall    lcd_home       ; place cursor to home position
    rcall    wire1_reset    ; send a reset pulse
    CA      wire1_write, skipROM
    CA      wire1_write, readScratchpad
    rcall    wire1_read     ; read temperature LSB
    mov     c0,a0
    rcall    wire1_read     ; read temperature MSB
    mov     a1,a0
    mov     a0,c0

    PRINTFLCD
.db  "temp=", FFRAC2+FSIGN, a, 4, $42, "C ", CR, 0
    rjmp main

```

Figure 11.6: wire1_temp2.asm.

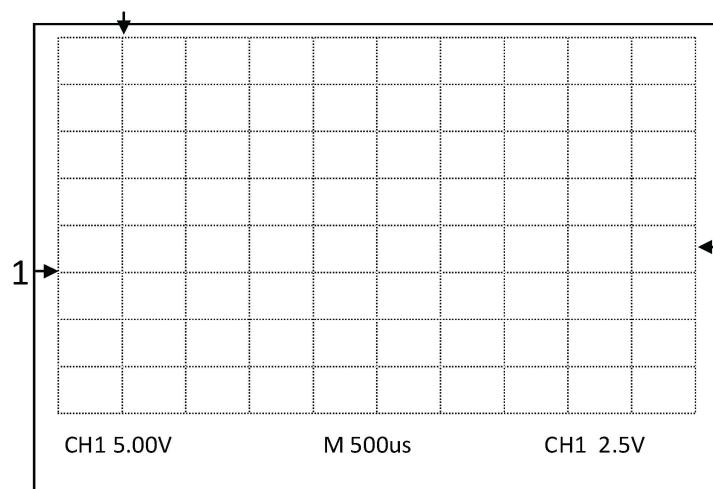


Figure 11.7: Séquence de transmission de la valeur de température.