

Compte-rendu de la tâche du projet sous-traitance

Éléments de la tâche :

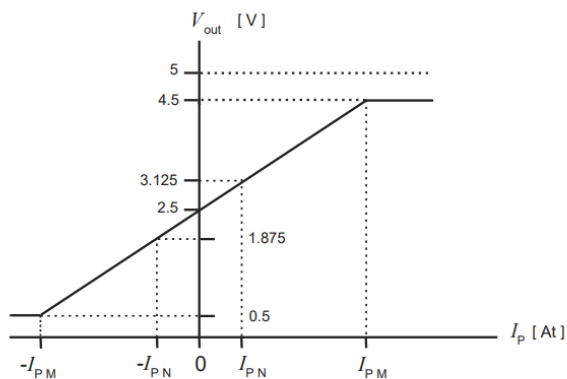
LTS 6-NP : capteur de courant

1602A : écran LCD

S5D5 : carte

Fonctionnement :

Le capteur de courant donne une tension pour image comme ci-dessous



Le capteur a une sensibilité de 104mV par ampère

La tension de sortie du capteur est ensuite récupérée par le CAN, traitée par la carte S5D5 puis la valeur du courant est ensuite envoyé vers le LCD.

Le CAN est codé sur 12 bits avec une plage de 0-3.3V .

Le CAN a un quantum de $q = (3.3 - 0) / 2^{12} = 806 \times 10^{-6}$

Programme :

```
//global variable declaration
ssp_err_t status;
uint32_t val_adc;
uint32_t duree= 500;
float v_out;
float val_courant;
int part_re;
char tab[11];
int n;
int j;

void hal_entry(void)
{
    /* TODO: add your own code here */
    //initialisation de l'adc
    //implicit declaration of function 'init_ports' [-Wimplicit-function-declaration]
    init_ports();
    while(1)
    {
        //recupération de la valeur du capteur
        status = g_adc0.p_api->scanStart(g_adc0.p_ctrl);
        do
        {
            //attente fin de conversion
            status = g_adc0.p_api->scanStatusGet(g_adc0.p_ctrl);
        }
        while(status !=SSP_SUCCESS);
        //lecture de la valeur de l'adc
        //passing argument 3 of 'g_adc0.p_api->read' from incompatible pointer type [-Wincompa]
        status = g_adc0.p_api->read(g_adc0.p_ctrl, ADC_REG_CHANNEL_0, &val_adc);
        //calcul valeur tension en sortie du capteur
        //conversion from 'double' to 'float' may change value [-Wfloat-conversion]

        v_out=0.000806*(float)val_adc;
        //calcul valeur courant capteur en milliampere
        //conversion from 'double' to 'float' may change value [-Wfloat-conversion]
        val_courant=((float)v_out-2.5)/0.10416;
        //implicit declaration of function 'sprintf' [-Wimplicit-function-declaration] | //incompa
        n=sprintf(tab, "%5.4f", val_courant);
        //affichage sur écran lcd
        status = g_ioport.p_api->portWrite(IOPORT_PORT_04, 1, 0xFF); //clear
        status = g_ioport.p_api->portWrite(IOPORT_PORT_04, 2, 0xFF); // curseur au debut
        j=0;

        while(tab[j]!='\0')
        {
            //character constant too long for its type | //unsigned conversion from 'int' to 'iopor
            status = g_ioport.p_api->portWrite(IOPORT_PORT_04, 1280+tab[j], 0xFF); //écriture
            status = g_ioport.p_api->portWrite(IOPORT_PORT_04, 20, 0xFF); //decalage
            j++;
        }
        status = g_ioport.p_api->portWrite(IOPORT_PORT_04, 1389, 0xFF); //écriture
        status = g_ioport.p_api->portWrite(IOPORT_PORT_04, 20, 0xFF); //decalage
        status = g_ioport.p_api->portWrite(IOPORT_PORT_04, 1345, 0xFF); //écriture
    }
}

//conflicting types for 'init_ports' | //no previous declaration for 'init_ports' [-Wmissing-decla]
void init_ports(void)
{
    status = g_adc0.p_api->open(g_adc0.p_ctrl, g_adc0.p_cfg);
    status = g_adc0.p_api->scanCfg(g_adc0.p_ctrl, g_adc0.p_channel_cfg);
}
```

Explication du code :

- 1) Initialisation du CAN (fonction init_port)
- 2) Conversion de la tension du capteur vers le CAN (la boucle Do While permet d'attendre la fin de la conversion)
- 3) Récupération de la valeur traduite par le CAN dans la variable val_adc
- 4) Calcul de la valeur d'entrée du CAN via le pas de quantification. La valeur de tension est stockée dans la variable v_out.
- 5) Calcul de la valeur du capteur de courant via la sensibilité du capteur. Le courant est stocké dans la variable val_courant.
- 6) La fonction sprintf permet de stocker la valeur du courant dans un tableau tab. Chaque caractère est stocké dans le tableau. Le format de stockage du tableau est le format 5.4 c'est-à dire que l'ont a 5 caractères pour la partie réelle et 4 caractères pour la partie décimale.
- 7) Initialisation du LCD, effacement de l'écran et positionnement du curseur au début de l'écran
- 8) Ecriture de la valeur de courant dans le tableau. On écrit tous les caractères du tableau jusqu'à ce que le caractère '\0' n'apparaisse. Puis on écrit les unités ici mA .

Branchement :

Le capteur doit être branché comme ci-joint :

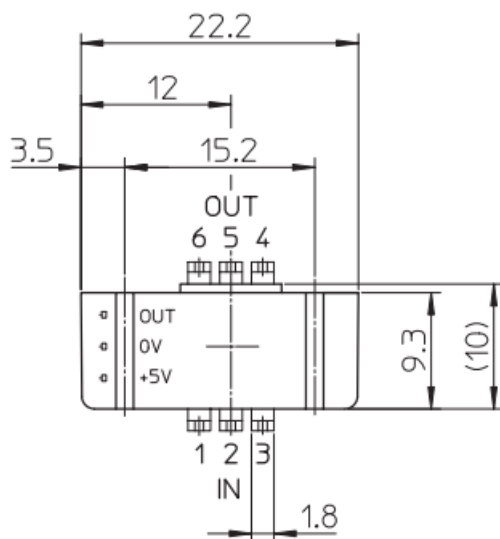
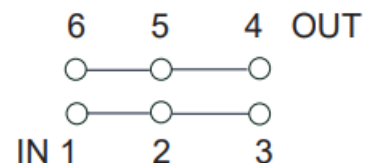


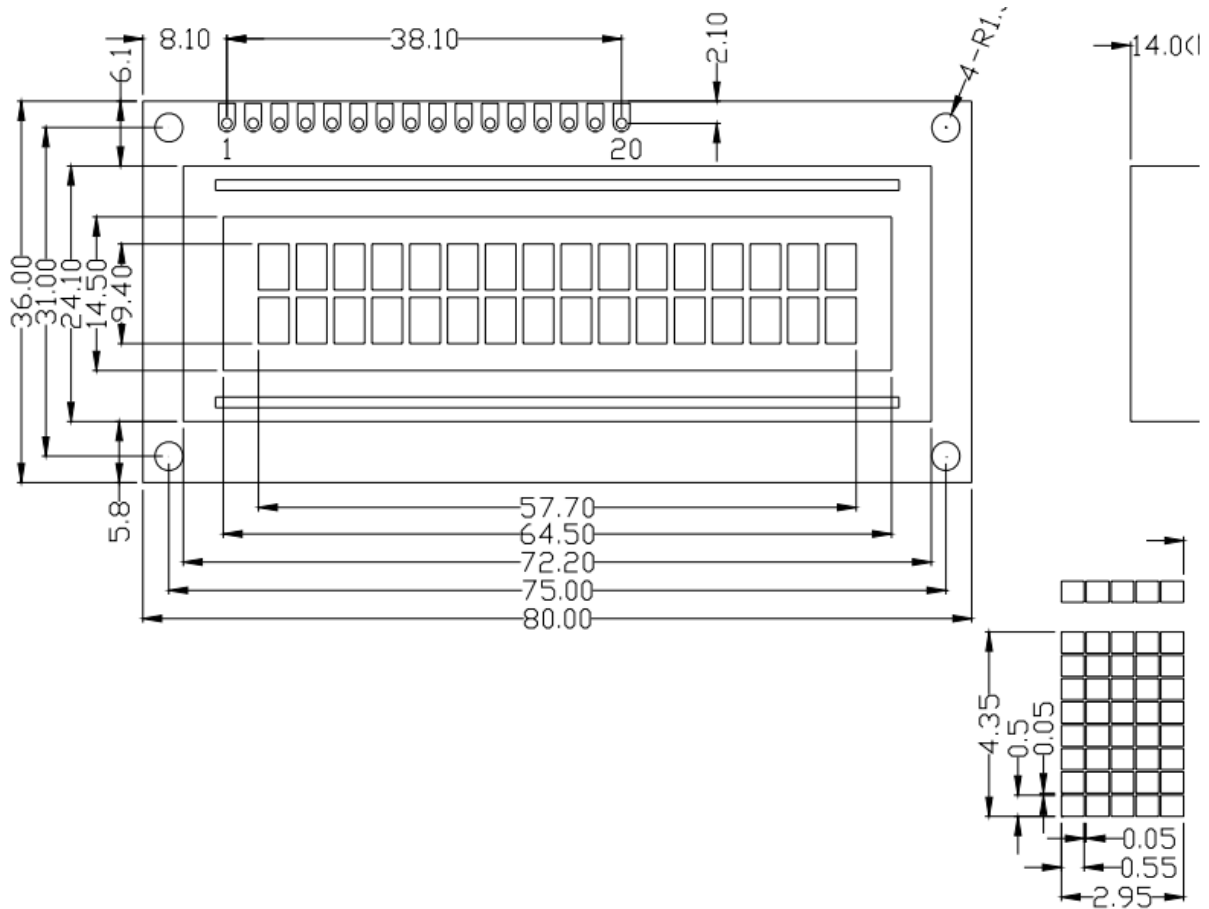
Schéma capteur



câblage des broches

Le V_out du capteur est relié à la broche P000 du S5D5.

Schéma branchement LCD :



| | | | | | | | |
|-----|-----|-----|-----|-----|-----|------|------|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| VSS | VCC | V0 | RS | R/W | E | DB0 | DB1 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| DB2 | DB3 | DB4 | DB5 | DB6 | DB7 | LED+ | LED- |

VCC ET LED+ (A) doivent être alimentés en 5V.

VSS et LED- (K) doivent être à la masse

V0 définit le contraste, il est donc possible de le mettre directement à 5V.

| Placement broches port 4 | | | | | | | | | | |
|--------------------------|-----|---|-----|-----|-----|-----|-----|-----|-----|-----|
| RS | R/W | E | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |

Les broches allant de RS à DB7 sont connectés au port 4

RS correspond à la broche P411 et DB0 correspond à la broche P400.


Récupérer l'archive et la mettre sur le S5D5 :

-Créer un workspace ou en rejoindre un déjà existant

-File->import->general->existing project into workspace

-Cocher "select archive file" -> browse -> chercher l'archive du projet->ouvrir -> finish

-build le projet (icone marteau)

- cliquer sur la flèche  -> debug configuration -> cliquer sur la flèche à coté de renesas GDB hardware debugging puis sur *nom_archive Debug* -> cliquer sur debug = le programme se charge sur la carte.

Attention :

- 1) La plage de valeur d'entrée du CAN est limitée à 3.3V or V_{out} peut aller jusqu'à 4.5V . Il faut donc faire attention à la tension qui arrive au bornes du CAN car cela peut endommager la carte.
- 2) Le LCD ne fonctionnera pas seul, sur la documentation technique du LCD, le LCD doit être alimenté en 5V sinon cela ne marchera pas, or la carte ne délivre que du 3.3V.
De plus la valeur limite pour détecter un '1' logique est de $0.7 \cdot V_{DD} = 3.5V$ pour le LCD
Vu que la carte ne peut délivrer que du 3.3V les '1' logiques envoyé par la carte ne seront pas détectés.
Ci-joint la capture d'écran du mail que j'ai envoyé à Mr James

Bonsoir Enzo,

Vous avez répondu à la question, l'afficheur doit être alimenté en +5V sinon cela ne fonctionnera pas....

Seule option, utiliser un circuit spécialisé ("voltage level translator") pour réhausser la tension sur les broches issues du μC (par contre avez vous sur le système une source de tension +5V ?)

Bonne soirée et bonne vacances.

Salutations.

Mr James.

De: "Enzo LACHEZE" <enzo.lacheze@etu.uca.fr>

À: "Michel JAMES" <Michel.JAMES@uca.fr>

Envoyé: Vendredi 28 Octobre 2022 17:23:05

Objet: probleme projet sous traitance

Bonsoir,

Je vous contacte suite a un questionnement de ma part.

J'ai essayé de tester le programme total avec le LCD. J'ai essayé seulement d'afficher des caractères sur le LCD ce qui n'a pas marché. Sur la doc du LCD demandé (LCD1602A), j'ai remarqué que la valeur minimale pour recevoir un 1 logique est de $0.7 \cdot V_{DD}$ or a 5V pour l'alimenter cela représente une valeur minimale de 3.5V or le S5D5 délivre un 1 logique a 3.3V il me semble.

Faut-il alors relever la tension de sortie des broches pour le LCD ou simplement l'alimenter en 3.3V suffit? J'ai un peu peur que le LCD ne soit pas suffisamment alimenté.

Passez de bonnes vacances.

Cordialement,