Carte *Contrôleur de puissance* & Logiciel associé STM32F103

Introduction

La carte *Contrôleur de puissance*, associée à la *Nucleo Board-F103RB*, permet de générer des commandes PWM 0/3V3. Elle est dédiée à la commande de moteur mais peut être utilisée de manière plus large, pour piloter n'importe quel système utilisant une PWM. Elle est livrée avec un logiciel permettant le contrôle des blocs principaux de la carte : *Toolbox_NRJ.c.*

La carte comprend plusieurs blocs :

- Alimentation 0/9V par bloc transformateur (7 à 9V)
- LED d'affichage des entrées / Sorties à utiliser
- Entrée analogique +/-10V (BNC + bornier à vis ou potentiomètre)
- Entrée analogique ou numérique protégée 0/3V3 (BNC + bornier à vis ou potentiomètre)
- Entrées courant I1, I2 et I3
- Entrées codeur incrémental (deux entrées quadrature A, B, une entrée d'index I)
- Sortie PWM 3 voies
- Sortie PWM auxiliaire 3 voies

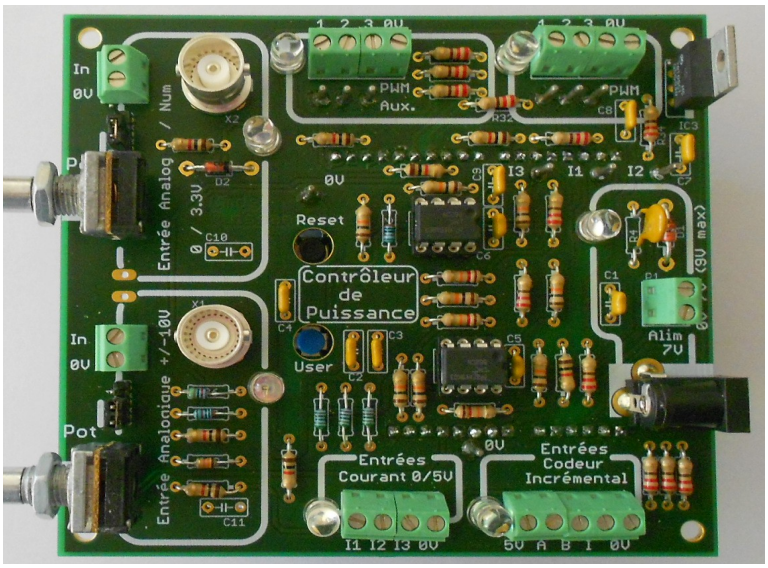
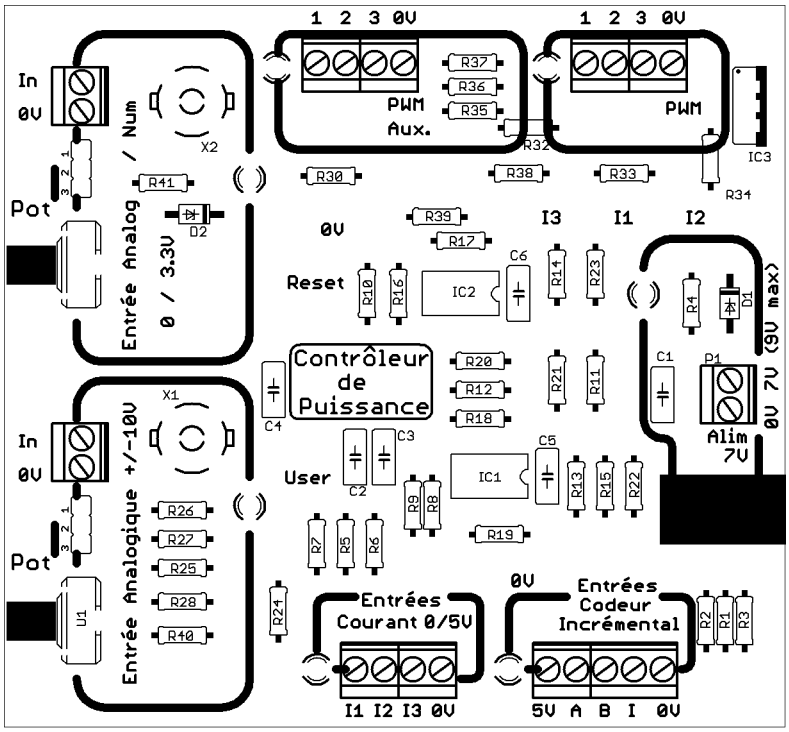


Table des matières



1. La prise en main rapide

L'ensemble doit être alimenté (par l'alimentation transformateur dédiée ou par une alimentation de table), la tension sera de 9VDC max (protection par zéner . **Seulement ensuite,** on connectera un connecteur *micro USB* entre la carte *Nucléo* et le *PC*.

La programmation se fait sous KEIL. Dès lors que le chargement du programme est terminé, un RESET doit être fait (utiliser un tournevis pour atteindre le bouton poussoir sous la carte *Contrôleur de Puissance).*

Le programme sera bâti autour de 2 grands blocs :

- *int main (void)* : Lance toute les configurations, en particulier la fonction

  void Conf_Generale_IO_Carte (void) qui configure toutes les IO de la carte, ainsi que l'horloge du processeur.

  La fonction s'achève classiquement par un while(1).

- Les fonctions d'interruption (voir chapitre associé)

2. LED d'indication

La carte est dotée de LED (verte) qui doivent indiquer quelles sont les entrées et sorties en court d'utilisation. Le programme implanté dans la carte *Nucléo* doit permettre ces indications. Pour cela on utilisera des macros (voir *Toolbox_NRJ.h).*

Les Macros à utiliser :

LED_Courant_On;

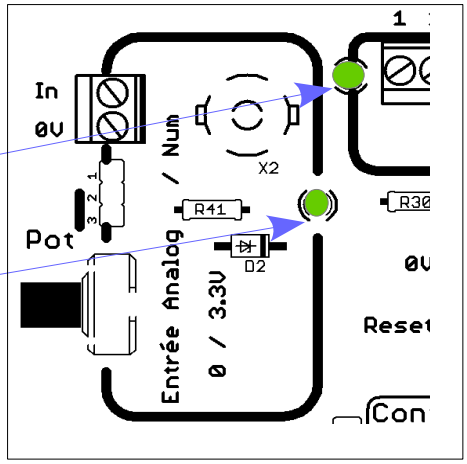
LED_PWM_On;

LED_PWM_Aux_On;

LED_Entree_10V_On;

LED_Entree_3V3_On;

LED_Codeur_On;



3. Entrée analogique + / -10V

Cette entrée est dédiée à recevoir une tension par exemple sinusoïdale d'amplitude maximale 10V. Un circuit résistif permet la correspondance pleine échelle de l'ADC (0-3V3). Il existe 3 manières d'attaquer l'entrée : BNC, bornier à vis, potentiomètre. Un cavalier permet de faire le choix de l'entrée potentiomètre.

Fonctions à utiliser (TOOLBOX_NRJ.h):

Fonction	***Description***	***Paramètres***
void Conf_ADC(void);	Configure l'ADC (à lancer une seule fois).	Aucun
int Entree_10V(void);	Lance la conversion, attend le résultat (1 à 2µs). Renvoie le résultat de conversion	Type int (16 bits non signés) de 0 à 4095 (ADC 12 bits)

4. Entrée analogique / Numérique 0-3V3

Comme l'entrée +/-10V, on y accède par une prise BNC, un bornier ou un potentiomètre. Elle est protégée par une zéner 3V6.
Elle est multi-usage, cependant, un rôle important de cette entrée consiste à recevoir un signal numérique (0/3V3), typiquement la sortie *Sync* d'un GBF*Agilent*, pour activer à chaque front une interruption qui permettra par exemple de faire avancer le pointeur d'une table de cosinus (Application onduleur monophasé ou triphasé). Voir le chapitre 9 sur les interruptions pour plus d'informations.

Dans le cas d'une utilisation analogique, voici les fonctions associées

Fonctions à utiliser (TOOLBOX_NRJ.h):

Fonction	***Description***	***Paramètres***
void Conf_ADC(void);	Configure l'ADC (à lancer une seule fois).	Aucun
int Entree_3V3(void);	Lance la conversion, attend le résultat (1 à 2µs). Renvoie le résultat de conversion	Type int (16 bits non signés) de 0 à 4095 (ADC 12 bits)

5. Les entrées courants

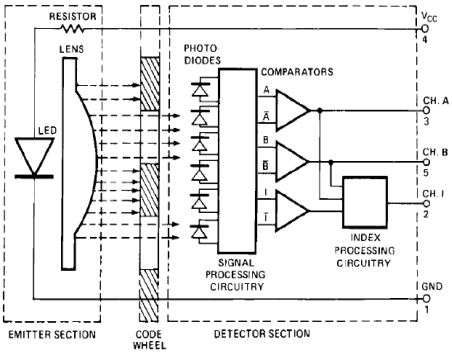
Ces entrées sont prévues pour recevoir un *LTS6-NP* dans la configuration 104mV/A, prévu pour +/-10A. Autrement dit, l'entrée devrait présenter un offset de **2V5** et une amplitude maximale peak-to-peak de **2,04Vpp** pour coïncider avec la pleine échelle des ADC (0-3V3). L'accès se fait par un bornier à vis 4 points (I1, I2, I3 et 0V).

Fonctions à utiliser (TOOLBOX_NRJ.h):

Fonction	***Description***	***Paramètres***
void Conf_ADC(void);	Configure l'ADC (à lancer une seule fois).	Aucun
int I1(void);	Lance la conversion, attend le résultat (1 à 2µs). Renvoie le résultat de conversion	Type int (16 bits non signés) de 0 à 4095 (ADC 12 bits)
int I2(void);	Lance la conversion, attend le résultat (1 à 2µs). Renvoie le résultat de conversion	Type int (16 bits non signés) de 0 à 4095 (ADC 12 bits)
int I3(void);	Lance la conversion, attend le résultat (1 à 2µs). Renvoie le résultat de conversion	Type int (16 bits non signés) de 0 à 4095 (ADC 12 bits)

6. Les entrées codeur incrémental

Cette entrée est prévue pour recevoir un codeur incrémental pour pouvoir par exemple faire un auto-pilotage de machine synchrone. Ces entrées peuvent aussi être utilisées pour recevoir des signaux numériques en provenance de capteur à effet Hall. Normalement prévues pour fonctionner en 0/3V3, ces entrées peuvent accepter du 0/5V grâce à des résistances de protection.



Les entrées A et B sont en quadrature. Sur la référence *AEDB-9140 (Agilent),* Le capteur comporte 360 périodes par tour. Un tour complet provoque donc :

- 720 fronts (montant et descendant sur chacune des voies A ou B)
- 1440 fronts (montant et descendant sur les deux voies cumulées).

La dernière entrée, I, est l'index. Elle permet de transformer le codeur relatif en codeur absolu.

Fonctions à utiliser (TOOLBOX_NRJ.h):

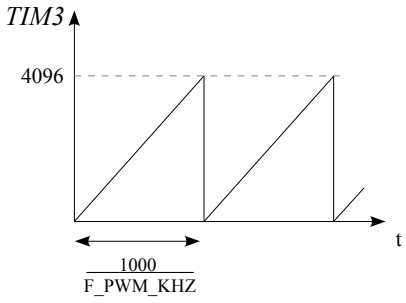
Fonction	***Description***	***Paramètres***
void Conf_Codeur_Inc(char Pas, unsigned int Pleine_Echelle, char IT_Passage_a_O);	Configure le préiphérique TIM2 pour gérer le codeur incrémental. A lancer une seule fois.	- ***Pas*** : Il prendra les valeurs au choix parmi l'ensemble {*Demi_Pas_ChA, Demi_Pas_ChA, Quart_Pas*} - ***Pleine_Echelle :*** Prendra les valeurs 720 ou 1440 selon le choix de *Pas*. Si le codeur est différent du *AEDB-9140,* on mettre la valeur adéquate. - ***IT_Passage_a_0*** : Prendra la valeur *IT_On* ou *IT_Off* selon qu'on souhaitera une mise à 0 automatique du compteur à la détection de l'index.

7. La PWM

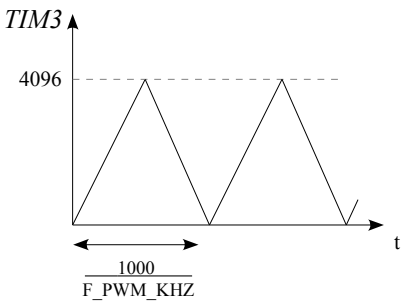
La carte dispose d'une sortie triple PWM (voies 1 à 3). Un seul timer gère l'ensemble des 3 voies : TIM3. Autrement dit, le timer TIM3 fait office de porteuse commune pour les 3 PWM. Il pourra être facilement configuré en évolution dents de scie (*UpRamp*) ou triangle symétrique. Quelque soit la configuration de la porteuse, celle-ci évolue de 0 à 4096.

Les 3 rapports cycliques demandés pourront dont s'étendre de 0 (0%) à 4096 (100%).

Mode UpRamp :



Mode Triangle



Fonctions à utiliser (TOOLBOX_NRJ.h):

Fonction	***Description***	***Paramètres***
void Triangle (float F_PWM_KHZ_Val)	Configure la porteuse en triangle à la fréquence spécifiée (à utiliser qu'une seule fois)	*F_PWM_KHZ_Val* :type float c'est la valeur de la fréquence PWM voulue en kHz.
void UpRamp (float F_PWM_KHZ_Val)	Configure la porteuse en dents de scie à la fréquence spécifiée (à utiliser qu'une seule fois)	*F_PWM_KHZ_Val* :type float c'est la valeur de la fréquence PWM voulue en kHz.
Start_PWM	Macro qui démarre la PWM	Aucun
Stop_PWM	Macro qui stoppe la PWM	Aucun
void Active_Voie_PWM(char Voie)	Active la voie spécifié. (la voie 1 est automatiquement activée lors de la configuration *Triangle* ou *UpRamp*.)	*Voie* : peut prendre la valeur 1,2 ou 3.
void Desactive_Voie_PWM(char Voie)	Désactive la voie spécifié.	*Voie* : peut prendre la valeur 1,2 ou 3.
void Inv_Voie(char Voie)	Inverse la PWM. Utile notamment pour générer des PWM unipolaire sur pont complet.	*Voie* : peut prendre la valeur 1,2 ou 3.
R_Cyc_1(Val)	Macro qui spécifie la rapport cyclique à placer sur la voie 1	*Val* : peut prendre une valeur de 0 (0%) à 4096 (100%)
R_Cyc_2(Val)	Macro qui spécifie la rapport cyclique à placer sur la voie 2	*Val* : peut prendre une valeur de 0 (0%) à 4096 (100%)
R_Cyc_3(Val)	Macro qui spécifie la rapport cyclique à placer sur la voie 3	*Val* : peut prendre une valeur de 0 (0%) à 4096 (100%)

8. La PWM auxiliaire

Elle est construite sur le même principe que la PWM principale. Elle comporte 3 voies. Elle sera utilisée pour :

- commander un moteur LRK et moteur pas à pas (les 3 sorties PWM auxiliaires ne seront pas utilisées comme telles mais comme de simples IO sur les interrupteurs Low Side)
- construire une PWM multi niveaux évoluée

Dans le cas d'une PWM multi niveaux, 3 porteuses déphasées de 120° peuvent être nécessaires. C'est pour cela que la PWM auxiliaire est construite autour de 2 timers différents, TIM1 et TIM4. Ainsi, l'on utilisera par exemple dans le cas d'un onduleur multi niveaux :

- Voie 1 de la PWM auxiliaire (TIM1)
- Voie 2 de la PWM auxiliaire(TIM4)
- Voie 3 de la PWM principale (TIM3)

Fonctions à utiliser (TOOLBOX_NRJ.h):

Fonction	***Description***	***Paramètres***
void Triangle_Aux_Voie_1 (float F_PWM_KHZ_Val)	Configure la porteuse en triangle à la fréquence spécifiée pour la voie 1 (à utiliser qu'une seule fois)	*F_PWM_KHZ_Val* :type float c'est la valeur de la fréquence PWM voulue en kHz.
void Triangle_Aux_Voie_2_3 (float F_PWM_KHZ_Val)	Configure la porteuse en triangle à la fréquence spécifiée pour la voie 2 &3 (à utiliser qu'une seule fois)	
void UpRamp_Aux_Voie_1 (float F_PWM_KHZ_Val)	Configure la porteuse en dents de scie à la fréquence spécifiée sur la voie 1 (à utiliser qu'une seule fois)	*F_PWM_KHZ_Val* :type float c'est la valeur de la fréquence PWM voulue en kHz.
void UpRamp_Aux_Voie_2_3 (float F_PWM_KHZ_Val)	Configure la porteuse en dents de scie à la fréquence spécifiée sur la voie 2&3 (à utiliser qu'une seule fois)	
Start_PWM_Aux_1	Macro qui démarre la PWM auxiliaire voie 1	Aucun
Start_PWM_Aux_2_3	Macro qui démarre la PWM auxiliaire voies 2 et 3	Aucun
Stop_PWM_Aux_1	Macro qui stoppe la PWM auxiliaire voie 1	Aucun
Stop_PWM_Aux_2_3	Macro qui stoppe la PWM auxiliaire voies 2 et 3	Aucun
R_Cyc_Aux_1(Val)	Macro qui spécifie la rapport cyclique à placer sur la voie 1	*Val* : peut prendre une valeur de 0 (0%) à 4096 (100%)
R_Cyc_Aux_2(Val)	Macro qui spécifie la rapport cyclique à placer sur la voie 2	*Val* : peut prendre une valeur de 0 (0%) à 4096 (100%)
R_Cyc_Aux_3(Val)	Macro qui spécifie la rapport cyclique à placer sur la voie 3	*Val* : peut prendre une valeur de 0 (0%) à 4096 (100%)

9. Les interruptions

Le programme à implanter dans la carte Nucléo est prévu pour fonctionner avec deux interruptions différentes :

- L'une est temporelle. Elle permet de créer un séquencement synchrone (utilisation du timer *Systick*) :
  void IT_Principale(void);

- L'autre se déclenche sur front détecté sur l'entrée 0-3V3 (voir chapitre 4)
  void IT_Ext_3V3(void);

Fonctions à utiliser (TOOLBOX_NRJ.h):

Fonction	***Description***	***Paramètres***
*void Conf_IT_Principale_Systick (void (*IT_function) (void),float Periode_IT_us);*	Configure le timer Systick et redirige l'interruption sur la fonction voulue. (à appeler qu'une seule fois)	void (*IT_function) (void) : Nom de la fonction d'interruption à exécuter (dans notre exemple : IT_Principale) *float Periode_IT_us* : Intervalle de temps exprimé en µs entre deux interruptions successives.
*void Conf_IT_Externe_3V3 (void (*IT_function) (void));*	Configure l'interruption externe et redirige l'interruption vers la fonction voulue. (à appeler qu'une seule fois)	void (*IT_function) (void) : Nom de la fonction d'interruption à exécuter (dans notre exemple : *IT_Ext_3V3*)

Les définitions paramétrables par l'utilisateur (accessibles dans *TOOLBOX_NRJ.h*) :

```
/*==========================================================================
                                    USER Define
==========================================================================*/
// Choisir la priorité d'interruption de 0 à 15, 0 le plus prioritaire.
 #define Prio_Systick 2
 #define Prio_IT_3V3 2
 #define Prio_Index_Codeur 0
/*==========================================================================*/
```

Par défaut le codeur optique a la priorité la plus élevée. Les deux autres interruptions (Systick, et interruption externe) sont de même priorité. Il est conseillé de laisser cette configuration. En effet, si les deux interruptions (Systick et IT externe) exploitent l'ADC, il peut se produire un plantage (demande de conversion ADC alors qu'une autre est en cours). L'égalité des niveaux d'interruption empêche cette situation critique.