TP commande numérique :

Gozard Matthieu ; Guillossou Kinann

1. Objectifs

A partir d'un hacheur complet et d'une carte Nucleo-STM32G474RE, vous devez :

1. Réaliser un shell pour commander le hacheur, sur la base d'un code fourni
2. Réaliser la commande des 4 transistors du hacheur en commande complémentaire décalée
3. Faire l'acquisition des différents capteurs
4. Réaliser l'asservissement en temps réel

2. Bonnes pratiques

Pour une meilleure lisibilité du code :

- Toutes les variables sont écrites en minuscule avec une majuscule pour les premières lettre à partir du 2eme mot dans le nom de la variable, par exemple : *uartRxBuffer*
- Les directives (macro avec #define) sont écrites en majuscule, par exemple : *#define UART_TX_SIZE*
- Aucun nombre magique ne peut exister, par exemple : remplacer if(value == 512) par if(value == VAL_MAX) avec #define VAL_MAX 512
- Si vous écrivez deux fois le même code, il faut écrire une fonction pour "factoriser" votre code.
- Il faut tester vos appels de fonction, les fonctions HAL renvoient pour la plupart une valeur pour savoir si son exécution s'est déroulée correctement.

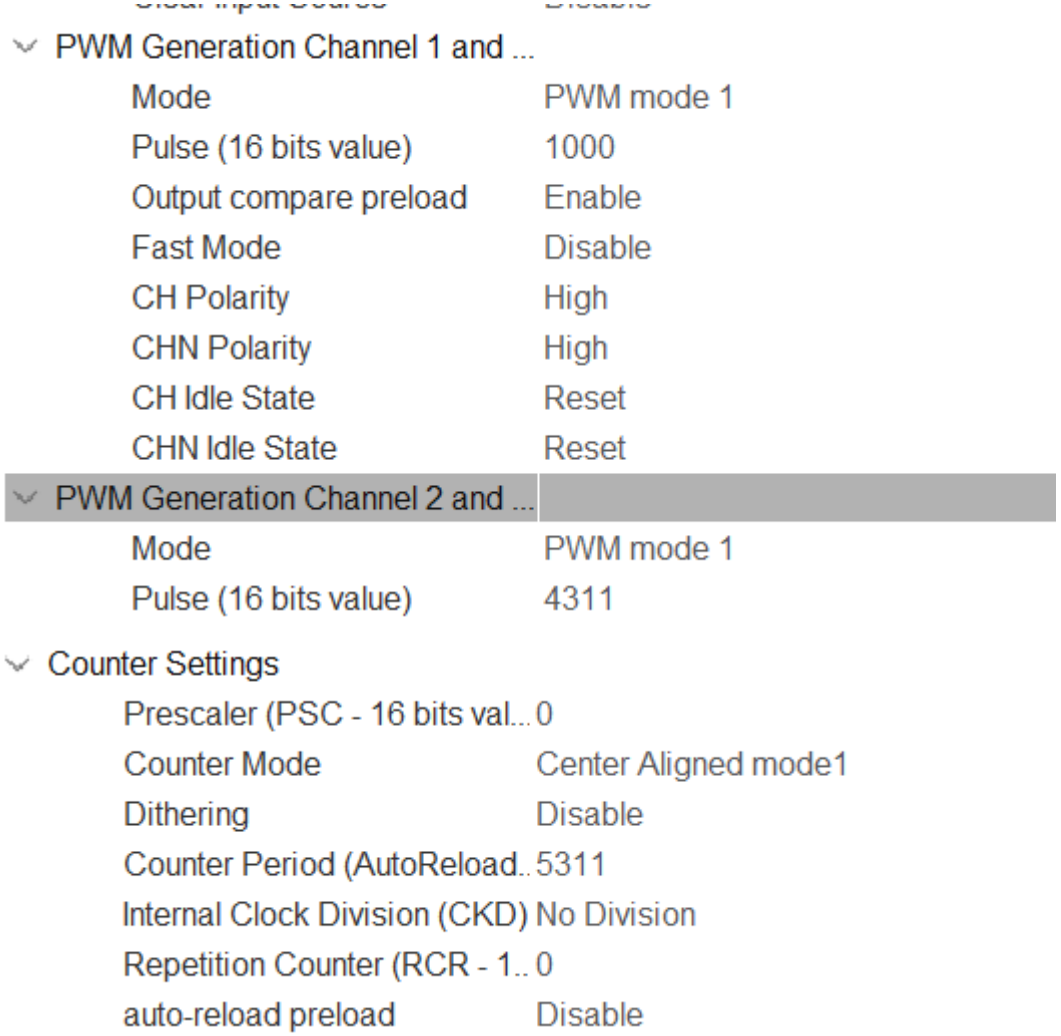
3. Github et Doxygen

Le lien de notre github est le suivant :
https://github.com/Kinann75/Commande_numerique_TP

6. Console UART

7. Commande MCC basique

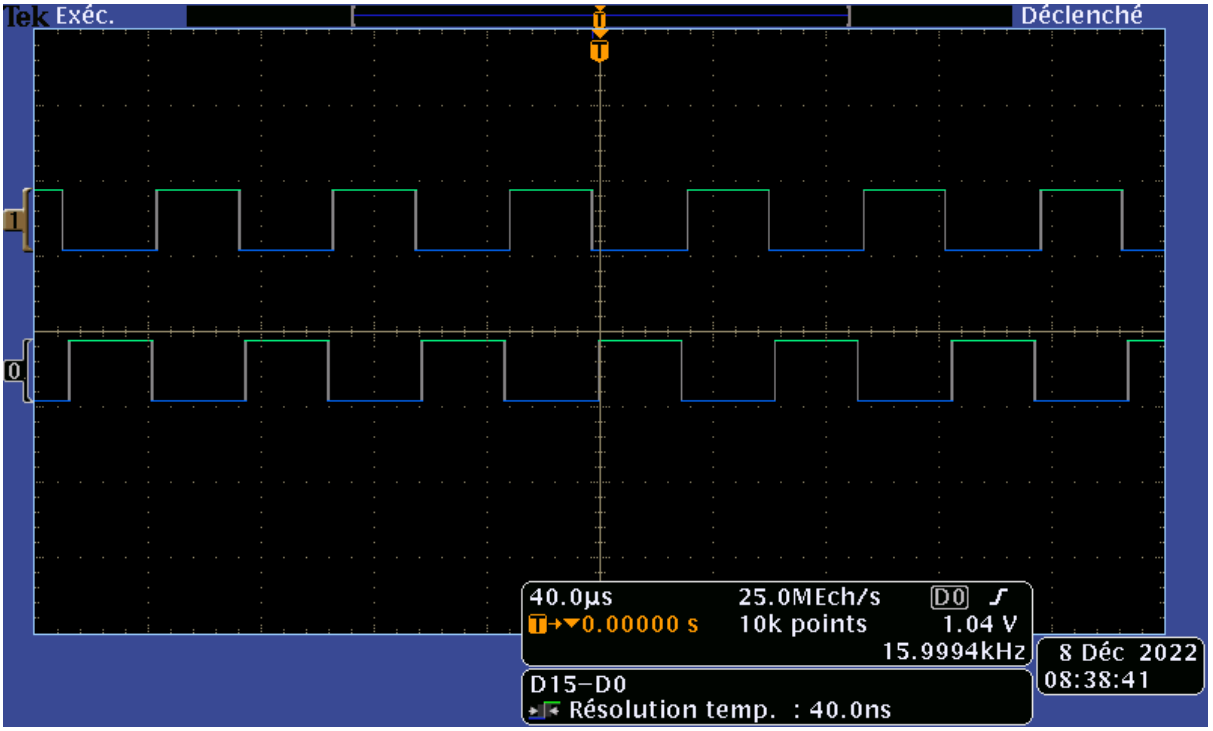
7.1. Génération de 4 PWM



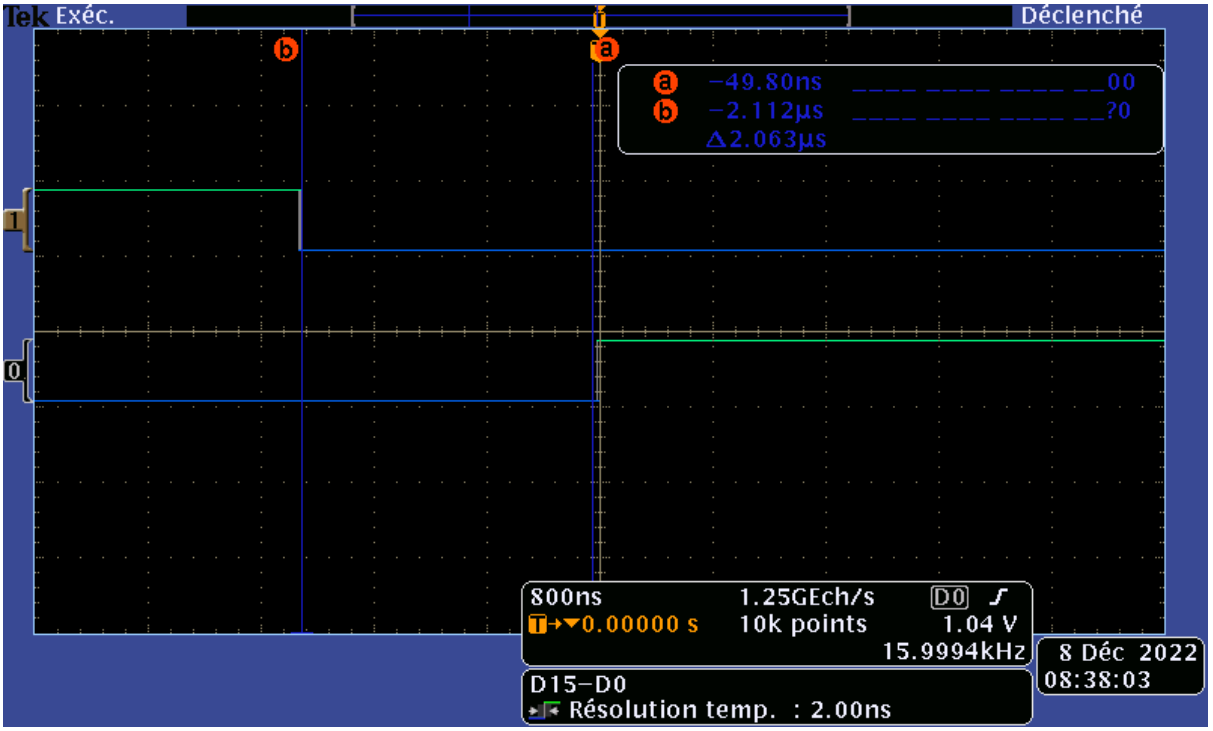
Paramétrage des PWM générés par le timer 1.

Pour choisir la counter période nous utilisons l'expression suivante. fclk/fpwm=1/(1+PSC)(1+CCR) (en prenant PSC=0, ce qui nous permettra d'avoir une plus grande précision sur "Pulse"). On trouve alors CCR=5311.

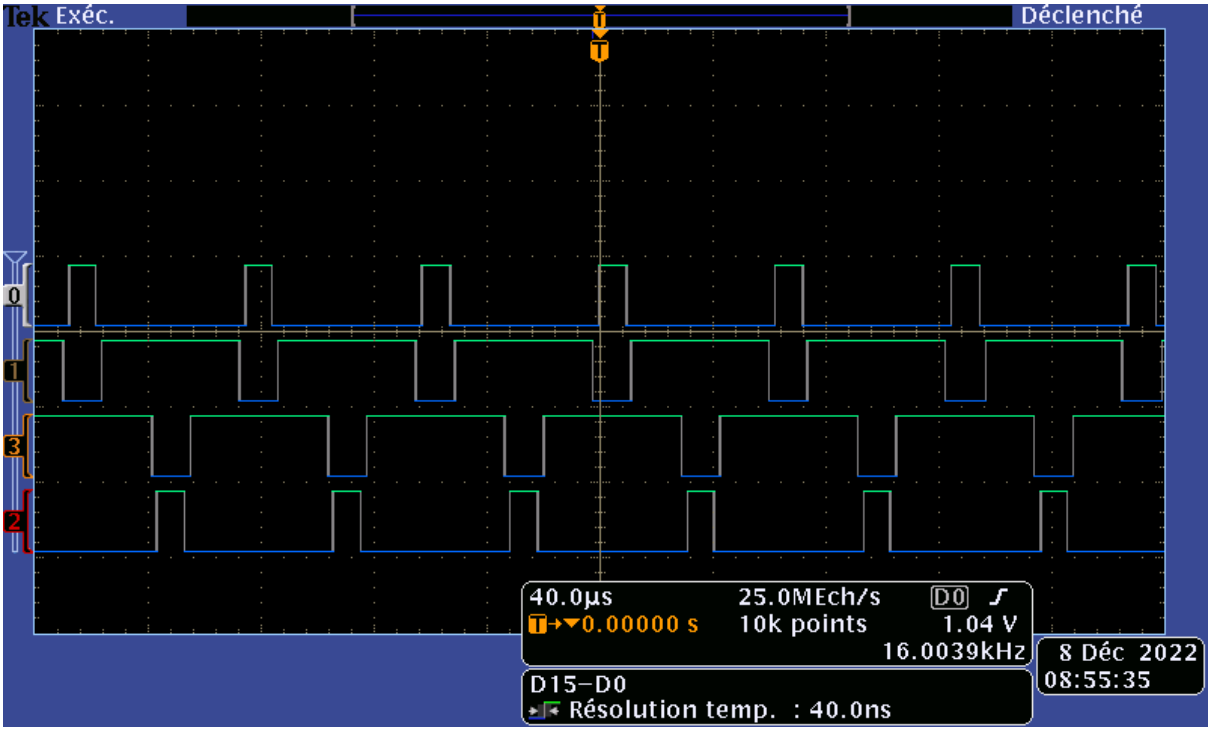
Choisir Dead Time à 205 nous permet d'avoir un temps mort de 2us.



Affichage à l'oscilloscope des signaux PWM1 et PWM1N.



Affichage à l'oscilloscope des signaux PWM1 et PWM1N avec un temps mort de 2us.



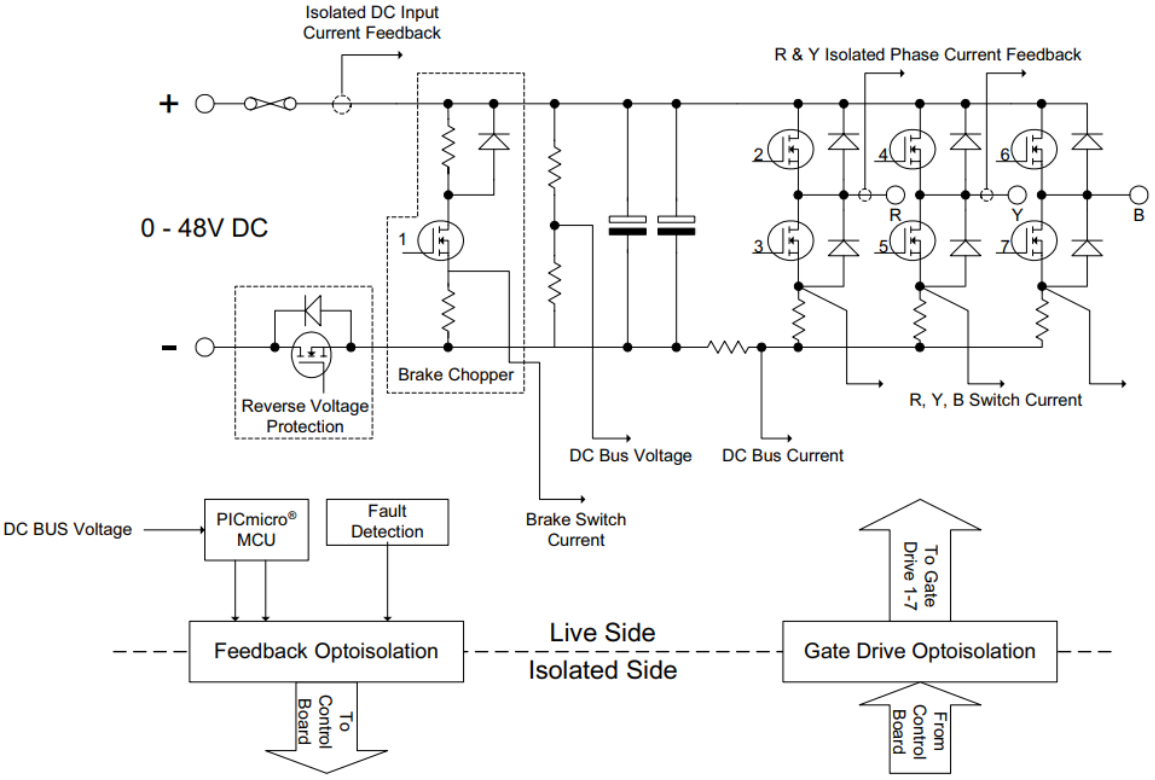
Affichage à l'oscilloscope des signaux PWM1, PWM1N, PWM2 et PWM2N.

7.2. Prise en main du hacheur

Pin	Function	Net Name	Input/Output	Isolated
1	Not Used	—	—	—
2	Yellow Phase Shunt Current Feedback	Y_SHUNT	Output	No if LK20 Fitted
3	DC Bus Shunt Current Feedback	BUS_SHUNT	Output	No If LK22 Fitted
4	Not Used	—	—	—
5	Yellow Phase Voltage Feedback	Y_VPH_SENSE	Output	No If LK25 Fitted
6	Blue Phase Back EMF crossing	B_CROSSING	Output	No if LK27 Fitted
7	Red Phase Back EMF crossing	R_CROSSING	Output	No if LK29 Fitted
8	Rectifier Output Voltage (|VAC|) Feedback	|VAC|_SENSE	Output	No if LK31 Fitted
9	Analog +5V from control PCB (±2%)	ISO_A+5V	Input	Yes
10	PFC Switch Firing Command	CMD_PFC	Input	Yes
11	Blue Phase Top Switch Firing Command	CMD_B_TOP	Input	Yes
12	Yellow Phase Top Switch Firing Command	CMD_Y_TOP	Input	Yes
13	Red Phase Top Switch Firing Command	CMD_R_TOP	Input	Yes
14	Active Low Serial Clock	$\overline{\text{ISO\_SCLK}}$	Output	Yes
15	Active Low Fault	$\overline{\text{FAULT\_ISO}}$	Output	Yes
16	Yellow Phase Hall Current Sensor Feedback	Y_HALL	Output	Yes
17	PFC Hall Current Sensor Feedback	PFC_HALL	Output	Yes
18	Digital GND from control PCB	ISO_GND	Input	Yes
19	Digital +5V from control PCB (±2%)	ISO_+5V	Input	Yes
20	Blue Phase Shunt Current Feedback	B_SHUNT	Output	No if LK19 Fitted
21	Red Phase Shunt Current Feedback	R_SHUNT	Output	No if LK21 Fitted
22	Brake Chopper Switch Shunt Current Feedback	BRAKE_SHUNT	Output	No if LK23 Fitted
23	Blue Phase Voltage Feedback	B_VPH_SENSE	Output	No if LK24 Fitted
24	Red Phase Voltage Feedback	R_VPH_SENSE	Output	No If LK26 Fitted
25	Yellow Phase Back EMF crossing	Y_CROSSING	Output	No if LK28 Fitted
26	DC Bus Voltage Feedback	BUS_SENSE	Output	No if LK30 Fitted
27	Analog GND from control PCB	ISO_AGND	Input	Yes
28	Brake Chopper Switch Firing Command	CMD_BRAKE	Input	Yes
29	Blue Phase Bottom Switch Firing Command	CMD_B_BOT	Input	Yes
30	Yellow Phase Bottom Switch Firing Command	CMD_Y_BOT	Input	Yes
31	Red Phase Bottom Switch Firing Command	CMD_R_BOT	Input	Yes
32	Active Low Serial Data	$\overline{\text{ISO\_DATA}}$	Output	Yes
33	Fault Reset Command	ISO_RESET	Input	Yes
34	Not Used	—	—	—
35	Red Phase Hall Current Sensor Feedback	R_HALL	Output	Yes
36	Digital GND from control PCB	ISO_GND	Input	Yes
37	Digital +5V from control PCB (±2%)	ISO_+5V	Input	Yes

USER SIGNAL CONNECTOR PINOUT (37-PIN, D-TYPE)

En rouge seront tous les pins que nous devons connecter. Ceux-ci comprennent les 4 signaux PWM que nous avons générés précédemment et qui seront connectés sur les pins Red_top, Red_bottom, Yellow_top et Yellow_bottom. Le signal PWM1 sera envoyé sur red_top, PWN1N sur Red_bottom, PW2 sur Yellow_top et PMW2N sur Yellow_bottom. Enfin on générera un signal GPIO pour l'envoyer sur le pin ISO_RESET. Les alimentations 5V et GND sont déjà reliées sur le PCB, nous n'avons donc pas à nous en occuper.



MC1L 3-PHASE LOW VOLTAGE POWER MODULE BLOCK DIAGRAM

7.3. Commande start

Voici ce qu'il va nous falloir générer pour ISO_RESET (à partir d'un GPIO) d'après la datasheet "Power Module 70097A".

- Reset the system by activating the active high ISO_RESET line. The ISO_RESET line is on pin 33 of the 37-pin, D-type (see **Section 1.8 "User Signal Connector Pinout (37-Pin, D-Type)"**). On the dsPICDEM MC1 Motor Control Development Board, this signal is routed to pin 14 of the 30F6010 dsPIC device, which is on Port RE9. The minimum pulse width for the reset is 2 µs. The RESET should be done in coordination with the SPI™ handling routine of the dsPIC device to ensure correct synchronization of the serial interface providing the isolated voltage feedback (see **Section 1.5.7.2 "Isolated Voltage Feedback"**). The system is now ready to use.

Nous devons générer ce signal de deux façons : après avoir appuyer sur le bouton bleu de la carte, ou en écrivant "power on"

```
while (1)
{
    if (flag==1)
    {
        HAL_GPIO_WritePin(ISO_RESET_GPIO_Port, ISO_RESET_Pin,GPIO_PIN_SET);
        for(i=0;i<1000;i++)
            {
            }
        HAL_GPIO_WritePin(ISO_RESET_GPIO_Port, ISO_RESET_Pin,GPIO_PIN_RESET);
        flag=0;
    }
```

```
186 /* USER CODE BEGIN 4 */
187 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
188 {
189     flag=1;
190 }
```

On a finalement mis la boucle for de 0 à 60. Nous avons ainsi généré le signal ISO_RESET comme voulu. Nous avons également ajouté ce code dans la fonction moterPowerOn pour pouvoir générer le signal ISO_RESET depuis le shell.

Il nous faut maintenant pouvoir modifier le rapport cyclique (alpha) des PWM depuis le shell. Voilà ce que nous avons réalisé :

```
53 const uint8_t alpha[]="rapport cyclique\r\n";
```

```
168     else if((strcmp(argv[0],"alpha"==0)))
169     {
170         HAL_UART_Transmit(&huart2, alpha, sizeof(alpha), HAL_MAX_DELAY);
171         set_alpha(atoi(argv[1]));
172     }
```

```
47 void set_alpha(int a)
48 {
49     HAL_UART_Transmit(&huart2, a, sizeof(a), HAL_MAX_DELAY);
50     TIM1->CCR1=a*53;
51     TIM1->CCR2=5311-a*53;
52
53 }
```

7.4. Premiers tests

Nous avons brancher le moteur en respectant les données constructeur du moteur et réalisé les tests suivants :

- Rapport cyclique de 50% : le moteur ne tourne pas, la tension aux bornes du moteur est de 0V.
- Rapport cyclique de 70% : le moteur tourne correctement.
- Rapport cyclique de 100% :
- Rapport cyclique de 0%