

Génération de signaux MLI complémentaires

Dans le cadre de notre projet nous avons dû générer des signaux MLI complémentaires pour commander le pont en H qui permet de mettre en œuvre la chauffe de la panne via le circuit de chauffe LC.

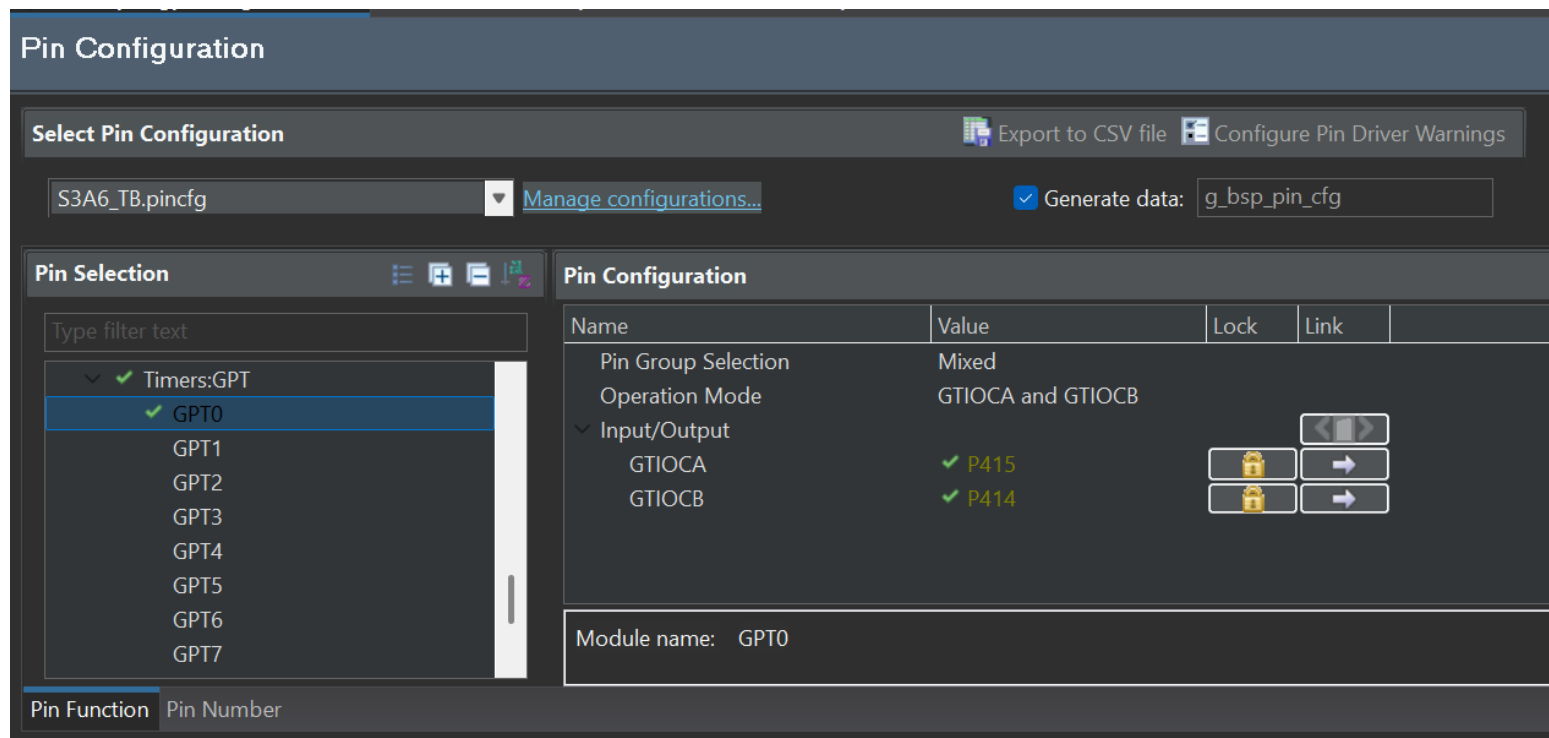
Pour ce faire j'ai utilisé un microcontrôleur S3A6 de chez Renesas pour générer les signaux (mais aussi pour mettre en œuvre la régulation en température, la mesure de température l'écran LCD, etc)

Dans un premier temps j'ai instancié un module timer `r_gpt` paramétré de la façon suivante :

g_timer0 Timer Driver on r_gpt		
Settings	Module g_timer0 Timer Driver on r_gpt	
API Info	Name	g_timer0
	Channel	0
	Mode	PWM
	Duty Cycle Range (only applicable in PWM mode)	Shortest: 2 PCLK, Longest: (Period - 1) PCLK
	Period Value	1000
	Period Unit	Hertz
	Duty Cycle Value	0
	Duty Cycle Unit	Unit Raw Counts
	Auto Start	True
	GTIOCA Output Enabled	True
	GTIOCA Stop Level	Pin Level Low
	GTIOCB Output Enabled	True
	GTIOCB Stop Level	Pin Level High
	Callback	NULL
	Overflow Interrupt Priority	

Ce module sert normalement de timer pour générer des interruptions à intervalles de temps définis mais il présente aussi un mode PWM qui permet de générer un signal MLI sur une sortie GTIOA dont la fréquence peut être modifiée via le paramètre Period Value (ici je règle 1000 Hz par défaut mais je viendrai changer directement la fréquence dans le code pour me mettre à la fréquence de résonance du circuit LC).

Une fois correctement paramétré je viens indiquer quelle broche correspond à la sortie de mon timer r_gpt :



Ici je choisis les broches P415 et P414 pour respectivement les sorties GTIOCA et GTIOCB.

Maintenant je dois initialiser mon module au démarrage du programme :

```
// Fonction principale
void main_thread_entry(void) {
    // Initialisation timer GPT
    status = g_timer0.p_api->open(g_timer0.p_ctrl, g_timer0.p_cfg);
    if(status != SSP SUCCESS) while(1);

    // Paramétrage MLI complémentaire
    R_GPTB0->GTIOR_b.GTIOA = 0x06;
    R_GPTB0->GTIOR_b.GTIOB = 0x19;

    R_GPTB0->GTIOR_b.OADFLT = 0x0;
    R_GPTB0->GTIOR_b.OAHLDT = 0x1;

    R_GPTB0->GTIOR_b.OBDFLT = 0x1;
    R_GPTB0->GTIOR_b.OBHLD = 0x1;
}
```

J'utilise ici la fonction `g_timer.p_api->open` disponible via la librairie SSP fournie par Renesas. Ensuite comme vous pouvez le voir je viens écrire dans les registres de mon module `r_gpt` pour générer l'autre signal MLI complémentaire. En effet il n'est pas possible de générer des sorties complémentaires directement via le menu

configuration du module, le seul moyen étant de modifier directement les registres du timer.

Je vais donc modifier les bits du registre GTIOR (page 420 doc technique registre S3A6) :

22.2.14 General PWM Timer I/O Control Register (GTIOR)

Address(es): GPT32m.GTIOR 4007 8034h + 0100h × m (m = 0 to 1),
GPT16m.GTIOR 4007 8034h + 0100h × m (m = 2 to 7)

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
NFCSB[1:0]	NFBEN	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Value after reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
NFCSA[1:0]	NFAEN	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Value after reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Symbol	Bit name	Description	R/W
b4 to b0	GTIOA[4:0]	GTIOCA Pin Function Select	See Table 22.5 .	R/W
b5	—	Reserved	This bit is read as 0. The write value should be 0.	R/W
b6	OADFLT	GTIOCA Pin Output Value Setting at the Count Stop	0: The GTIOCA pin outputs low when counting stops 1: The GTIOCA pin outputs high when counting stops.	R/W
b7	OAHL D	GTIOCA Pin Output Setting at the Start/Stop Count	0: The GTIOCA pin output level at the start/stop of counting depends on the register setting 1: The GTIOCA pin output level is retained at the start/stop of counting.	R/W
b8	OAE	GTIOCA Pin Output Enable	0: Output is disabled 1: Output is enabled.	R/W
b10, b9	OADF[1:0]	GTIOCA Pin Disable Value Setting	<div> <div>b10 b9</div> <div>0 0: None of the below options are specified</div> <div>0 1: GTIOCA pin is set to Hi-Z in response to control the output negation</div> <div>1 0: GTIOCA pin is set to 0 in response to control the output negation</div> <div>1 1: GTIOCA pin is set to 1 in response to control the output negation.</div> </div>	R/W
b12, b11	—	Reserved	These bits are read as 0. The write value should be 0.	R/W
b13	NFAEN	Noise Filter A Enable	0: The noise filter for the GTIOCA pin disabled 1: The noise filter for the GTIOCA pin enabled.	R/W

Table 22.5 Settings of GTIOA[4:0] and GTIOB[4:0] bits

GTIOA/GTIOB[4:0] bits					Function		
b4	b3	b2	b1	b0	b4	b3, b2*1,*2,*3	b1, b0*2
0	0	0	0	0	Initial output is low	Retain output at cycle end	Output retained at GTCCRA/GTCCRB compare match
0	0	0	0	1			Low output at GTCCRA/GTCCRB compare match
0	0	0	1	0			High output at GTCCRA/GTCCRB compare match
0	0	0	1	1			Output toggled at GTCCRA/GTCCRB compare match
0	0	1	0	0		Low output at cycle end	Output retained at GTCCRA/GTCCRB compare match
0	0	1	0	1			Low output at GTCCRA/GTCCRB compare match
0	0	1	1	0			High output at GTCCRA/GTCCRB compare match
0	0	1	1	1			Output toggled at GTCCRA/GTCCRB compare match
0	1	0	0	0		High output at cycle end	Output retained at GTCCRA/GTCCRB compare match
0	1	0	0	1			Low output at GTCCRA/GTCCRB compare match
0	1	0	1	0			High output at GTCCRA/GTCCRB compare match
0	1	0	1	1			Output toggled at GTCCRA/GTCCRB compare match
0	1	1	0	0		Toggle output at cycle end	Output retained at GTCCRA/GTCCRB compare match
0	1	1	0	1			Low output at GTCCRA/GTCCRB compare match
0	1	1	1	0			High output at GTCCRA/GTCCRB compare match
0	1	1	1	1			Output toggled at GTCCRA/GTCCRB compare match
1	0	0	0	0	Initial output is high	Retain output at cycle end	Output retained at GTCCRA/GTCCRB compare match
1	0	0	0	1			Low output at GTCCRA/GTCCRB compare match
1	0	0	1	0			High output at GTCCRA/GTCCRB compare match
1	0	0	1	1			Output toggled at GTCCRA/GTCCRB compare match
1	0	1	0	0		Low output at cycle end	Output retained at GTCCRA/GTCCRB compare match
1	0	1	0	1			Low output at GTCCRA/GTCCRB compare match
1	0	1	1	0			High output at GTCCRA/GTCCRB compare match
1	0	1	1	1			Output toggled at GTCCRA/GTCCRB compare match
1	1	0	0	0		High output at cycle end	Output retained at GTCCRA/GTCCRB compare match
1	1	0	0	1			Low output at GTCCRA/GTCCRB compare match
1	1	0	1	0			High output at GTCCRA/GTCCRB compare match
1	1	0	1	1			Output toggled at GTCCRA/GTCCRB compare match
1	1	1	0	0		Toggle output at cycle end	Output retained at GTCCRA/GTCCRB compare match
1	1	1	0	1			Low output at GTCCRA/GTCCRB compare match
1	1	1	1	0			High output at GTCCRA/GTCCRB compare match
1	1	1	1	1			Output toggled at GTCCRA/GTCCRB compare match

Je viens dans un premier temps mettre la valeur 6 à GTIOA. Cette valeur permet de définir le comportement des transitions entre front montant et descendant en fonction de la valeur du timer. En somme la valeur 6 permet de paramétrer la MLI pour qu'elle passe de l'état haut à l'état bas en fonction de la valeur du timer et du rapport cyclique.

De même la valeur 19 mise pour GTIOB permet de faire l'exact inverse et ainsi d'avoir un comportement complémentaire du signal A.

Je viens ensuite mettre la valeur du bit OADFLT à 0 pour que la sortie A soit à l'état bas lorsque le timer est arrêté et je mets le bit OAHLD à 1 pour que la valeur de l'état de la sortie reste constante entre le moment où le timer arrête et reprend le comptage.

Je viens faire la même chose pour la sortie B mais cette fois-ci je mets OBDFLT à 1 pour que la sortie soit à l'état haut lorsque le timer est arrêté (pour ne pas griller les transistors du pont en H en les fermant tous si j'arrête le timer).

Enfin je crée une fonction MLI qui permettra de changer les valeurs de fréquences et rapport cyclique à chaque appel de la fonction (les variables f et alpha sont définies comme variables globales et peuvent être modifiées par d'autres fonctions) :

```
// Fonction qui définit la valeur du rapport cyclique et de la fréquence du timer GPT de la PWM
void MLI(void){
    // Changement de format
    uint32_t F_int;
    uint8_t alpha_int;
    F_int = (uint32_t) f;
    alpha_int = (uint8_t)(alpha);
    // Changement valeurs rapport cyclique et periode
    status = g_timer0.p_api->dutyCycleSet(g_timer0.p_ctrl, alpha_int, TIMER_PWM_UNIT_PERCENT, 0); //0 pour GTIOCA
    status = g_timer0.p_api->periodSet(g_timer0.p_ctrl, F_int, TIMER_UNIT_FREQUENCY_HZ);

    status = g_timer0.p_api->dutyCycleSet(g_timer0.p_ctrl, alpha_int, TIMER_PWM_UNIT_PERCENT, 1); // 1 pour GTIOCB
    status = g_timer0.p_api->periodSet(g_timer0.p_ctrl, F_int, TIMER_UNIT_FREQUENCY_HZ);
}
```

De ce fait j'arrive bien à obtenir le résultat suivant, de deux signaux complémentaires en sortie du microcontrôleur pour commander mon pont en H :

