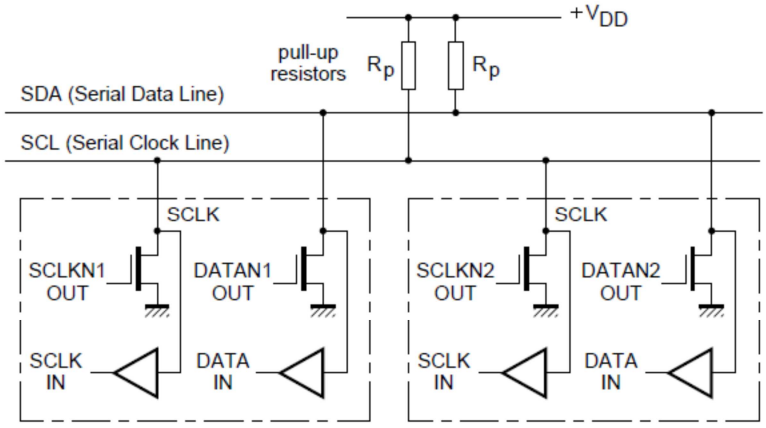
SYNTHESE LOGIQUE

Thermomètre – i2c

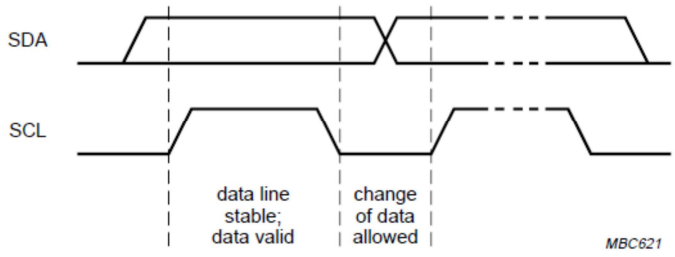
1. Présentation

Le bus **i2c** développé par **Philips Semiconductor** dans les années 80 permet de faire communiquer des circuits intégrés en utilisant trois fils seulement, c'est un bus série. Les sorties sont à collecteur ouvert, avec résistances de rappel à $+V_{CC}$. Pour mettre à 0 une sortie, on la relie à ***0V***, pour la forcer à 1, on la met en haute impédance : ceci permet d'éviter les conflits électriques.

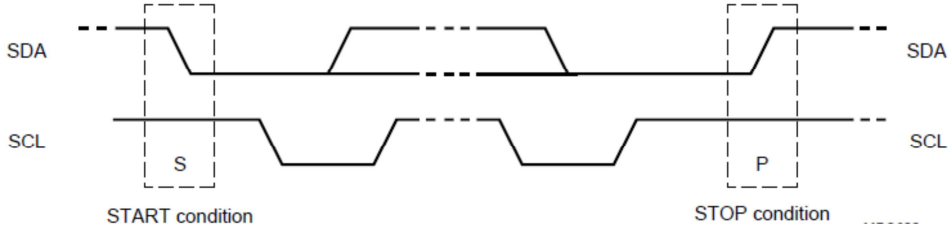
- SCL : horloge (100kHz, 400kHz ou 3,2MHz), c'est un bus synchrone
- SDA : fil de données bidirectionnel
- GND : masse, référence de tous les signaux



L'objectif de ce projet est de faire communiquer un composant FPGA avec un thermomètre-thermostat via le bus **i2c**. La fréquence maximale de l'horloge du circuit vaut 400kHz (datasheet), dans ce projet on adopte la fréquence **100kHz** (qui peut plus peut moins). Le transfert d'un bit se fait à l'état haut du signal d'horloge, la donnée doit être stable pendant toute la durée de l'impulsion.



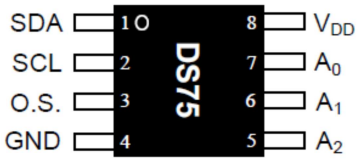
Chaque octet transféré débute par un **start** et se termine par un **stop.** Ceux-ci se distinguent respectivement par un front descendant et un front montant de **SDA** pendant que **SCL** est à l'état haut.



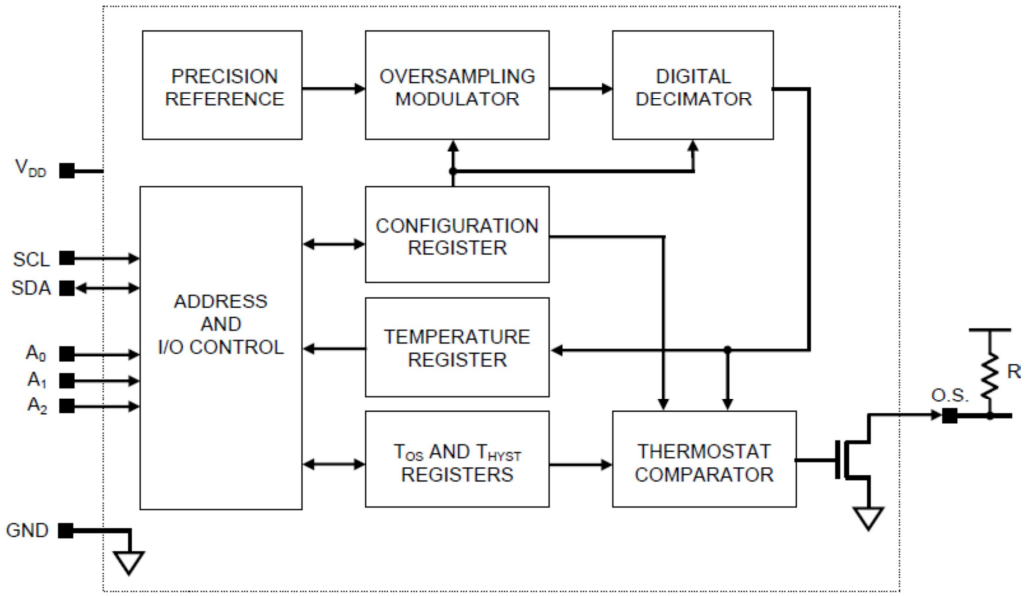
2. Le circuit intégré DS75

Le circuit DS75 du fabricant DALLAS Semiconductor est un thermomètre thermostat digital. Voici ses caractéristiques essentielles :

- Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
- ±2°C Accuracy Over a -25°C to +100°C Range
- Thermometer Resolution is User-Configurable from Nine (Default) to 12 Bits (0.5°C to 0.0625°C Resolution)
- 9-Bit Conversion Time is 150ms (Max)
- Thermostatic Settings are User-Definable
- Data is Read/Written Via 2-Wire Serial (Interface (SDA and SCL Pins)
- Multidrop Capability Simplifies Distributed Temperature-Sensing Applications
- Wide Power-Supply Range (+2.7V to +5.5V).



La figure suivante montre le synoptique du circuit DS75 :



☑ Dans ce TP, seul le mode Thermomètre sera exploité.

☑ La résolution du convertisseur analogique numérique peut être de 9 bits (par défaut), 10 bits, 11 bits ou 12 bits. Le mode 9 bits suffira pour ce TP.

☑ La température mesurée est stockée dans 2 registres 8 bits : $\mathbf{T_H}$ et $\mathbf{T_L}$. $\mathbf{T_H}$ contient la partie entière et $\mathbf{T_L}$ la partie fractionnaire. Le bit de poids fort de $\mathbf{T_H}$ est le bit de signe (codage en complément à 2).

	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
MS Byte	S	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	bit 7	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
LS Byte	2^{-1}	2^{-2}	2^{-3}	2^{-4}	0	0	0	0

☑ Le circuit possède une adresse unique afin de le distinguer des autres circuits sur le bus. L'adresse est sur 7 bits dont 4 sont gravés dans le silicium. Les 3 restants sont imposés par trois broches du composant : ici les broches sont reliées à $+V_{CC}$. Le dernier bit est un bit de lecture écriture : écriture=0, lecture=1.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	0	0	1	A_2	A_1	A_0	$R/\overline{W}$
				1	1	1	

☑ Quatre registres permettent de communiquer avec le circuit. Un pointeur de registre permet de sélectionner le registre avec lequel on veut communiquer.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	P1	P0

REGISTER	P1	P0
Temperature	0	0
Configuration	0	1
T_{HYST}	1	0
T_{OS}	1	1

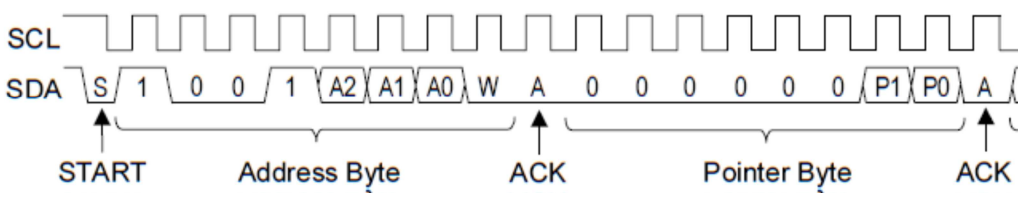
☑ Le registre de configuration permet à l'utilisateur de choisir le mode de fonctionnement, la résolution … Le registre est accessible en lecture et écriture.

MSb	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	LSb
0	R1	R0	F1	F0	POL	TM	SD

☑ Dans le registre de configuration, seuls les bits R_0 et R_1 concernent le mode thermomètre. Ces 2 bits sont à zéro par défaut, ce qui correspond au mode 9 bits adopté ici.

R1	R0	THERMOMETER RESOLUTION	MAX CONVERSION TIME
0	0	9–bit	150 ms
0	1	10–bit	300 ms
1	0	11–bit	600 ms
1	1	12–bit	1200 ms

☑ Au final, seul registre **Température** sera impliqué dans ce projet. La sélection du registre concerné se fait en écrivant la valeur du pointeur.

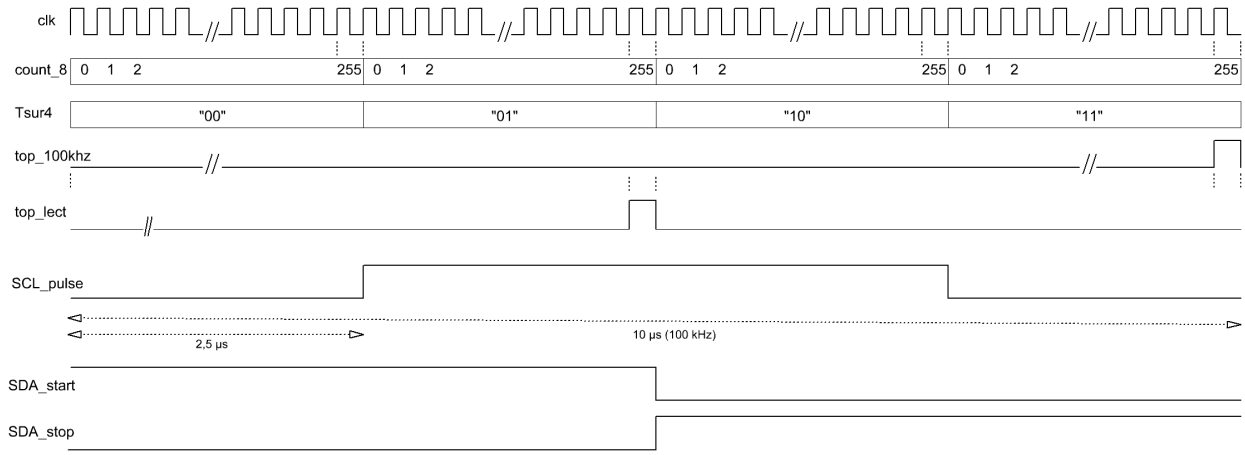


3. Première partie du projet

Cette partie consiste à lire la température sans arrêt, en respectant les temps de conversion tout de même, et afficher le résultat sur 8 LED.
Le mode 9 étant choisi, on ne gardera que les 8 bits de poids fort avec une résolution de un degré. Donc, seul le registre $\mathbf{T_H}$ sera affiché.
Un seul programme **i2c_ctrl** s'occupera de la génération des signaux nécessaires à la communication avec le circuit DS75, la lecture de la température et son affichage sur les LED.

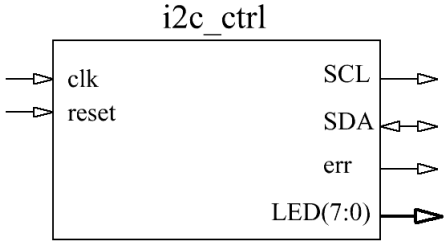
3.1. Génération de l'horloge

Le signal **SCL_pulse** de fréquence 100kHz servira d'horloge pour le bus.
SCL_pulse sera généré à partir du signal **CLK** qui a une fréquence de **100MHz**. Deux compteurs sont utilisés pour le générer, un compteur sur 8 bits (count_8) et un autre sur 2 bits (Tsur4).
Le compteur **count_8** compte sans interruption de 0 à 255 les périodes de **CLK**.
Le compteur **Tsur4** s'incrémente lorsque **count_8** atteint la valeur 255.
Le signal **top_100khz** passe à 1 lorsque **count_8** vaut 255 et **Tsur4** vaut 3. Le nom de ce signal est très parlant puisqu'il possède une fréquence de 100kHz. C'est ce signal qui va cadencer les bits sur le bus.
Le signal **top_lect** servira pour lire les bits en plein milieu afin d'éviter les erreurs.
Les signaux **SDA_start** et **SDA_stop** serviront pour les phases de start et de stop.



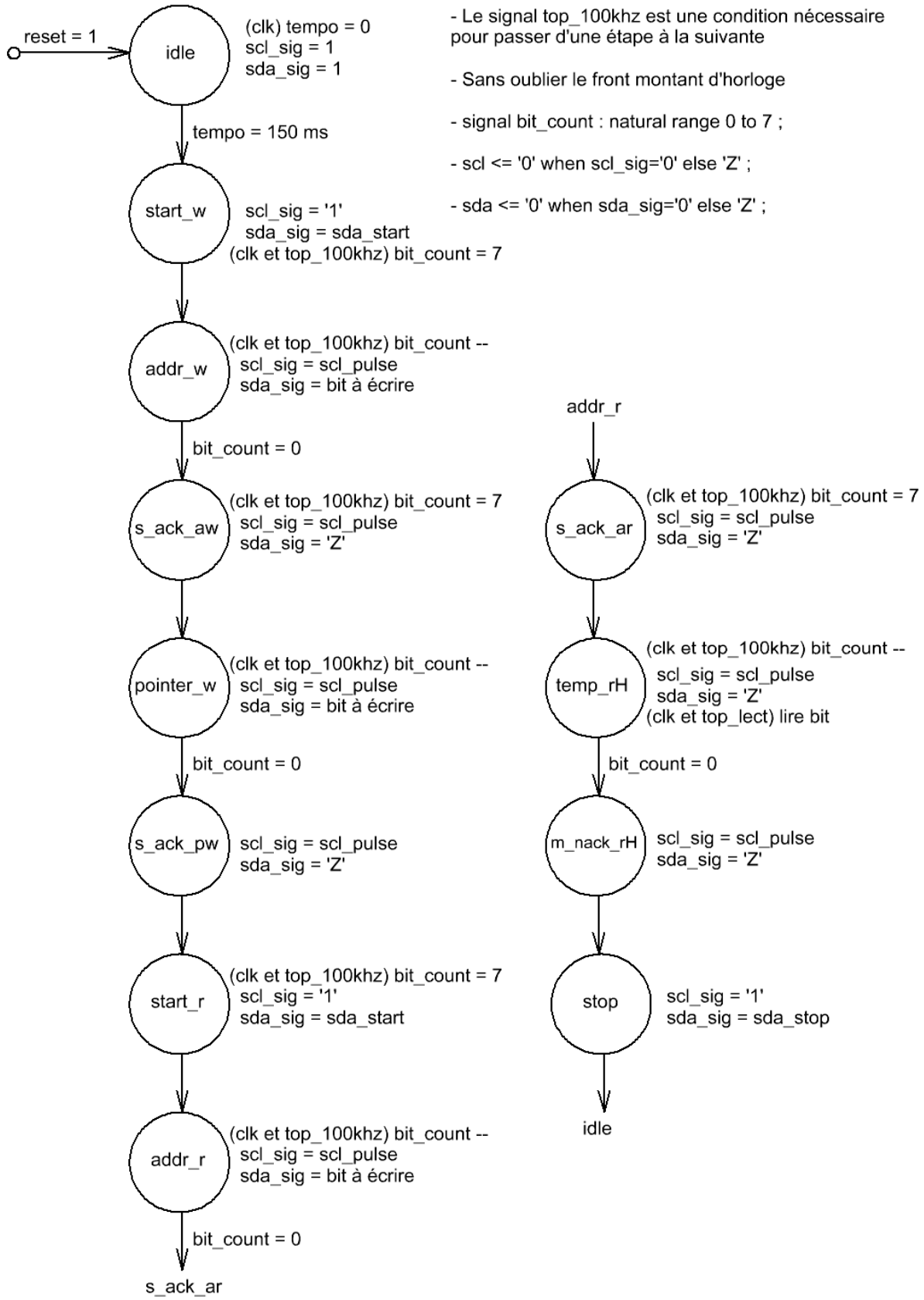
3.2. Réalisation du projet

- ☑ Créer un nouveau projet nommé **thermo_led**
- ☑ Créer un nouveau module VHDL nommé **i2c_ctrl**



- ☑ Ecrire un **process** permettant de concevoir les deux compteurs **count_**8 et **Tsur4** .Générer les signaux **top_100khz**, **top_lect, scl_pulse, sda_start** et **sda_stop** présentés ci-dessus. Simuler le fonctionnement du programme et vérifier les signaux obtenus.

☑ Compléter le programme en y ajoutant le code de la machine d'états suivante :



La temporisation de 150 ms (ou plus) est réalisée avec un compteur.
La condition (**clk**) signifie que l'action se fait sur front montant d'horloge.
La condition (**top_100khz)** signifie **top_100khz = 1**.

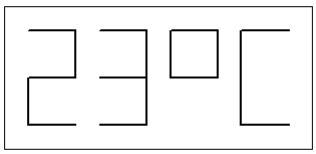
- ☑ Simuler le programme un fois complété et synthétisé.
- ☑ Ecrire le fichier de contraintes.

```
NET "err"    LOC = "P2"   | IOSTANDARD = "LVCMOS33";           #Sch name = LED15
NET "scl"    LOC = "B13"  | IOSTANDARD = "LVCMOS33";           #Sch name = JA1
NET "sda"    LOC = "D17"  | IOSTANDARD = "LVCMOS33" | PULLUP;  #Sch name = JA3
```

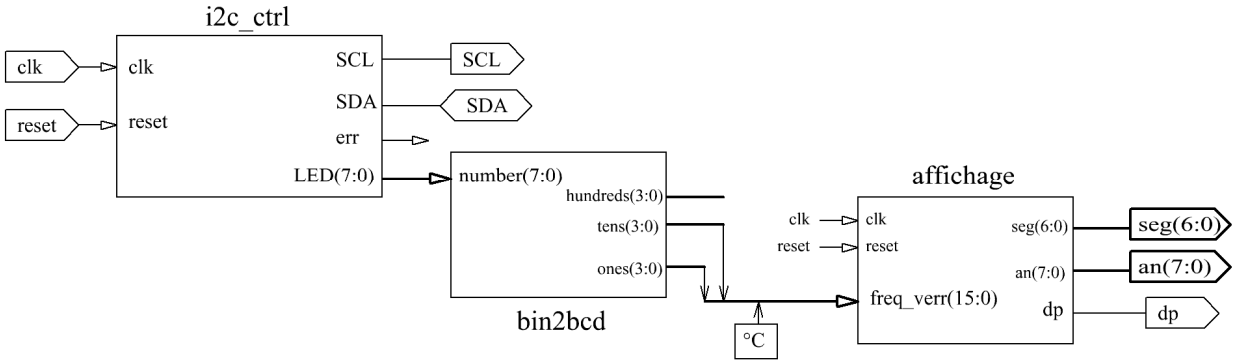
- ☑ Compiler le projet et programmer le FPGA pour tester le fonctionnement.

4. Deuxième partie du projet

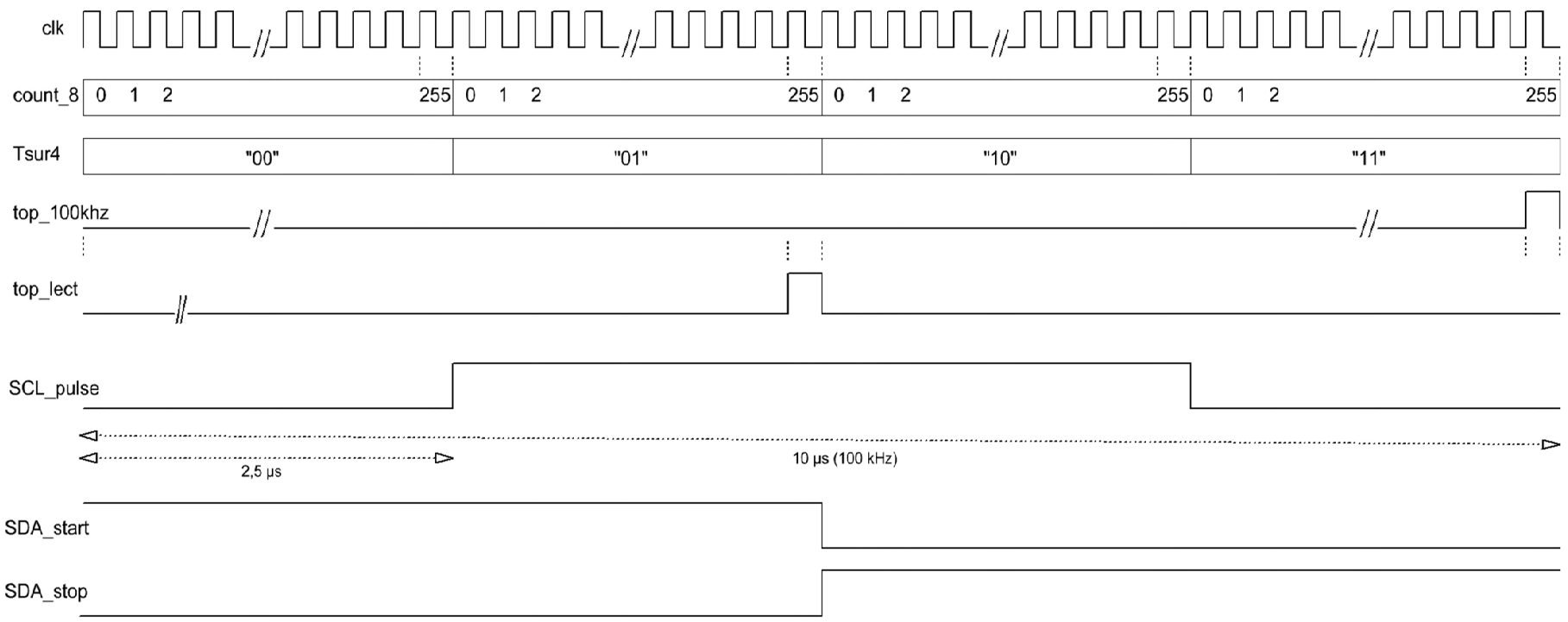
Dans cette partie, la température en binaire sera convertie en BCD avant d'être affichée en clair. Exemple :



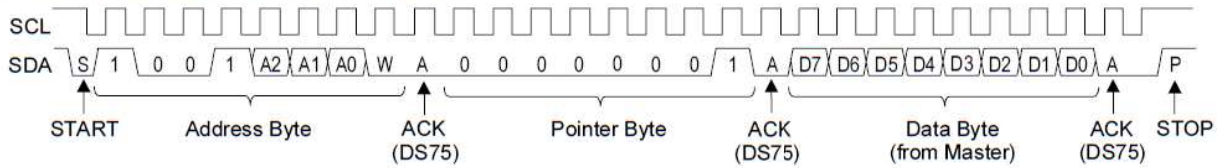
La figure suivante donne le synoptique du projet :



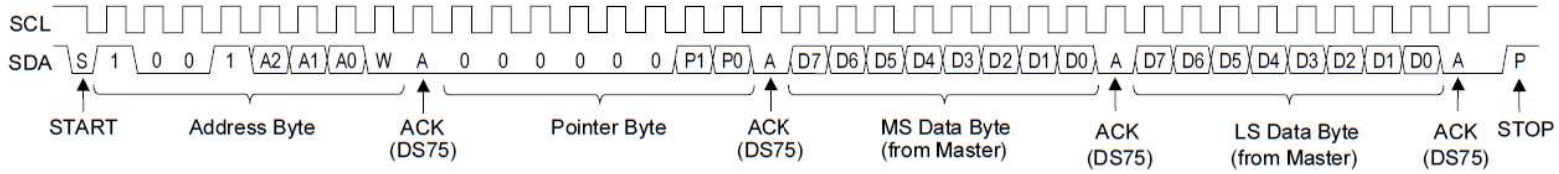
- ☑ Créer un nouveau projet nommé **thermo_affi**
- ☑ Y inclure les modules i2c_ctrl et affichage.
- ☑ Créer le convertisseur bin2bcd (voir le cours), le simuler.
- ☑ Créer un module de niveau 1 nommé thermo_2015, le compléter selon le synoptique précédent.
- ☑ Créer le fichier de contraintes associé.
- ☑ Compiler et tester sur la maquette.


clk
count_8
0 1 2
255
0 1 2
255
0 1 2
255
0 1 2
255
Tsur4
"00"
"01"
"10"
"11"
top_100khz
top_lect
SCL_pulse
2,5 µs
10 µs (100 kHz)
SDA_start
SDA_stop

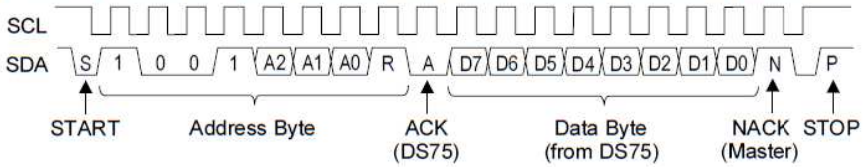
a) Write to the Configuration Register



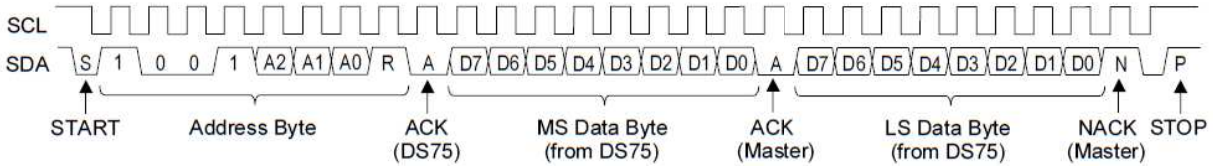
b) Write to the T_{OS} or T_{HYST} Register



c) Read From the Configuration Register (current pointer location)



d) Read 2-Bytes From the Temperature, T_{OS} or T_{HYST} Register (current pointer location)



e) Read Single Byte (new pointer location)