

**Compte-rendu TEST**

(Date : 12/03/24)

Etudiante : Malek SAHLI

Cursus : Master 1 LBD

**Année universitaire 2023/2024**

**Partie A**

**Question 1 :**

* Quelle est l’adresse I2C du capteur (7 bits) ?

-L’adresse de I2C de IS2DW12 sensor est 0x29 (7 bits).

**Question 2 :**

* Quelle grandeur physique ce capteur mesure-t-il ?

- Pression différentielle

- Pression absolue

-Température

**Question 3 :**

* Quelle est la fréquence maximale de récupération des donnees (ODR) ?
* Le débit de données maximal (ODR) du capteur IS2DW12 est de 100 Hz.

**Question 4 :**

* Quelle est le format des données en Provence du capteur ?

- Le capteur IS2DW12 génère des données au format suivant :

• Pression : entier signé 24 bits

• Température : entier signé 16 bits

**Question 5 :**

* Citer des applications possibles pour ce capteur ?

- Voici quelques exemples d’applications possibles pour le capteur IS2DW12 :

• Mesure de la pression barométrique

• Contrôle CVC

• Automatisation industrielle

**Partie B :**

L’analyse de l’application :

**Initialisation des périphériques** : l'initialisation des périphériques STM32 nécessaires, tels que GPIO, UART (USART2) et I2C (I2C1), en appelant les fonctions HAL\_Init(), SystemClock\_Config(), MX\_GPIO\_Init(), MX\_USART2\_UART\_Init() et MX\_I2C1\_Init().

**Initialisation du contexte de l'accéléromètre** : dev\_ctx est initialisé avec des fonctions de lecture , écriture et de temporisation spécifiques à la plate-forme pour l'accéléromètre LIS2DW12. Il est utilisé par les fonctions du pilote LIS2DW12 pour communiquer avec le capteur via I2C.

**Vérification de l'identifiant du dispositif** : La fonction lis2dw12\_device\_id\_get() est appelée pour obtenir l'identifiant du dispositif. Si l'identifiant ne correspond pas à celui attendu (LIS2DW12\_ID), le programme entre dans une boucle infinie, ce qui peut être interprété comme une gestion d'erreur en cas de non-détection du capteur.

**Réinitialisation et configuration de l'accéléromètre :** On réinitialise et configure ensuite l'accéléromètre LIS2DW12 avec différentes options telles que le bloc de mise à jour des données, l'échelle complète, le chemin de filtrage, le mode d'alimentation et le débit de données de sortie

**int** **main**(**void**)

{

/\* USER CODE BEGIN 1 \*/

stmdev\_ctx\_t dev\_ctx;

dev\_ctx.write\_reg = platform\_write;

dev\_ctx.read\_reg = platform\_read;

dev\_ctx.mdelay = HAL\_Delay;

dev\_ctx.handle = &hi2c1;

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_USART2\_UART\_Init();

MX\_I2C1\_Init();

/\* USER CODE BEGIN 2 \*/

/\* Check device ID \*/

lis2dw12\_device\_id\_get(&dev\_ctx, &whoamI);

**if** (whoamI != LIS2DW12\_ID)

**while** (1) {

/\* manage here device not found \*/

}

/\* Restore default configuration \*/

lis2dw12\_reset\_set(&dev\_ctx, PROPERTY\_ENABLE);

**do** {

lis2dw12\_reset\_get(&dev\_ctx, &rst);

} **while** (rst);

/\* Enable Block Data Update \*/

lis2dw12\_block\_data\_update\_set(&dev\_ctx, PROPERTY\_ENABLE);

/\* Set full scale \*/

//lis2dw12\_full\_scale\_set(&dev\_ctx, LIS2DW12\_8g);

lis2dw12\_full\_scale\_set(&dev\_ctx, *LIS2DW12\_2g*);

/\* Configure filtering chain

\* Accelerometer - filter path / bandwidth

\*/

lis2dw12\_filter\_path\_set(&dev\_ctx, *LIS2DW12\_LPF\_ON\_OUT*);

lis2dw12\_filter\_bandwidth\_set(&dev\_ctx, *LIS2DW12\_ODR\_DIV\_4*);

/\* Configure power mode \*/

lis2dw12\_power\_mode\_set(&dev\_ctx, *LIS2DW12\_HIGH\_PERFORMANCE*);

//lis2dw12\_power\_mode\_set(&dev\_ctx, LIS2DW12\_CONT\_LOW\_PWR\_LOW\_NOISE\_12bit);

/\* Set Output Data Rate \*/

lis2dw12\_data\_rate\_set(&dev\_ctx, *LIS2DW12\_XL\_ODR\_25Hz*);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

uint8\_t reg;

/\* Read output only if new value is available \*/

lis2dw12\_flag\_data\_ready\_get(&dev\_ctx, &reg);

**if** (reg) {

/\* Read acceleration data \*/

**memset**(data\_raw\_acceleration, 0x00, 3 \* **sizeof**(int16\_t));

lis2dw12\_acceleration\_raw\_get(&dev\_ctx, data\_raw\_acceleration);

//acceleration\_mg[0] = lis2dw12\_from\_fs8\_lp1\_to\_mg(data\_raw\_acceleration[0]);

//acceleration\_mg[1] = lis2dw12\_from\_fs8\_lp1\_to\_mg(data\_raw\_acceleration[1]);

//acceleration\_mg[2] = lis2dw12\_from\_fs8\_lp1\_to\_mg(data\_raw\_acceleration[2]);

acceleration\_mg[0] = lis2dw12\_from\_fs2\_to\_mg(

data\_raw\_acceleration[0]);

acceleration\_mg[1] = lis2dw12\_from\_fs2\_to\_mg(

data\_raw\_acceleration[1]);

acceleration\_mg[2] = lis2dw12\_from\_fs2\_to\_mg(

data\_raw\_acceleration[2]);

**sprintf**((**char** \*)tx\_buffer,

"Acceleration [mg]:%4.2f\t%4.2f\t%4.2f\r\n",

acceleration\_mg[0], acceleration\_mg[1], acceleration\_mg[2]);

//HAL\_UART\_Transmit(&huart2, tx\_buffer, strlen((char const \*)tx\_buffer), 1000);

**printf**((**char** \*)tx\_buffer);

}

}

/\* USER CODE END 3 \*/