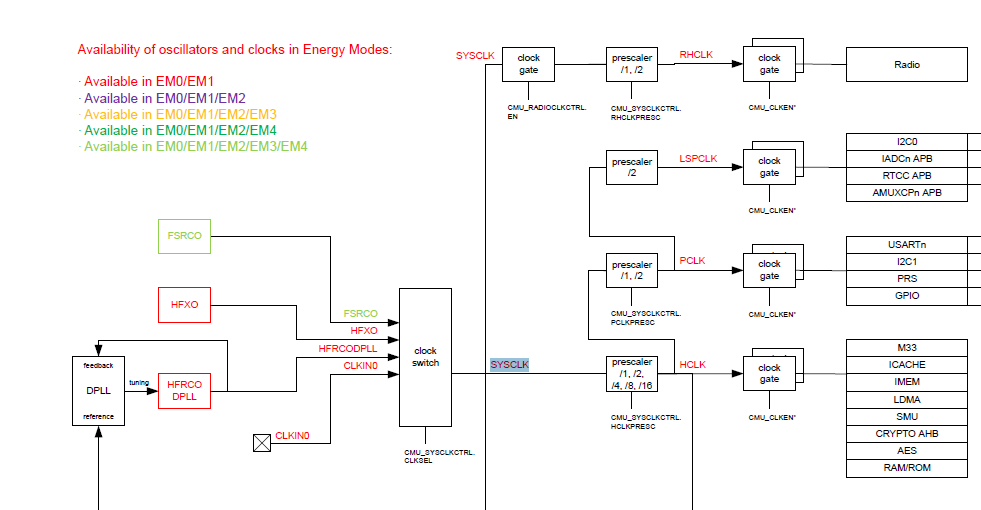
SiliconLabs

Driver i2cSpm.

Le framework Silabs est fait de tel façon que les changements de SYSCLK impliquent un re-calcul de toutes les fréquences/prescaler dans les drivers. Cela permet de conserver autant que possible les fréquences de sortie des différents modules (i2C, SPI,….) sans que le programmeur n’ait à changer les settings des modules.

Cela implique qu’une augmentation de la fréquence SYSCLK va augmenter la vitesse de calcul de la CPU et ne modifier la fréquence des sorties (i2C,SPI,.. ) que si il est impossible de calculer un prescaler qui permette de conserver la fréquence originale définie.

Les 2 modules I2C n’utilisent pas la même clock source. Le module I2c1 peut monter plus haut en fréquence.

Le driver défini des ratios de clock en fonction de la fréquence choisie :

* 100kbits -> (4 :4)
* 400 kbits -> (6 :3)
* 1Mbits -> (11 :6)

Cela veut dire que pour 100kbits le rapport clock high et clock low est de 50%. Pour 400Kbits 60% low et 30% high etc…. C’est pour respecter la norme ( ?), car il semble que les montées du signal SDA sont lentes et qu’il faut attendre avant de faire le sampling.

Ces ratios impliquent que les fréquences demandées dans la config ne sont pas forcément respectées en réalité (ex 400kbits demandé, 350kbits obtenu).

Pour résoudre ces problèmes dans le cas uniquement d’un système master/slave, il faut :

1. Initialiser à la main les configs d’instances de l’I2C en forçant le ratio 4 :4 (i2cClockHLRStandard) pour toutes les vitesses :

**#if** SL\_I2CSPM\_SENSOR\_SPEED\_MODE == 0

**#define** SL\_I2CSPM\_SENSOR\_HLR i2cClockHLRStandard

**#define** SL\_I2CSPM\_SENSOR\_MAX\_FREQ I2C\_FREQ\_STANDARD\_MAX

**#elif** SL\_I2CSPM\_SENSOR\_SPEED\_MODE == 1

**#define** SL\_I2CSPM\_SENSOR\_HLR i2cClockHLRStandard

**#define** SL\_I2CSPM\_SENSOR\_MAX\_FREQ I2C\_FREQ\_FAST\_MAX

**#elif** SL\_I2CSPM\_SENSOR\_SPEED\_MODE == 2

**#define** SL\_I2CSPM\_SENSOR\_HLR i2cClockHLRStandard

**#define** SL\_I2CSPM\_SENSOR\_MAX\_FREQ I2C\_FREQ\_FASTPLUS\_MAX

**#endif**

I2CSPM\_Init\_TypeDef init\_sensor = {

.port = SL\_I2CSPM\_SENSOR\_PERIPHERAL,

.sclPort = SL\_I2CSPM\_SENSOR\_SCL\_PORT,

.sclPin = SL\_I2CSPM\_SENSOR\_SCL\_PIN,

.sdaPort = SL\_I2CSPM\_SENSOR\_SDA\_PORT,

.sdaPin = SL\_I2CSPM\_SENSOR\_SDA\_PIN,

.i2cRefFreq = 0,

.i2cMaxFreq = SL\_I2CSPM\_SENSOR\_MAX\_FREQ,

.i2cClhr = SL\_I2CSPM\_SENSOR\_HLR

};

Fichier : sl\_i2cspm\_instance.h

1. Initialiser Modifier la méthode I2CSPM\_Init pour forcer la sortie clock en pushpull :

**void** **I2CSPM\_Init**(I2CSPM\_Init\_TypeDef \*init)

**{**

/\* Output value must be set to 1 to not drive lines low. Set

SCL first, to ensure it is high before changing SDA. \*/

GPIO\_PinModeSet(init->sclPort, init->sclPin, *gpioModePushPull*, 1);

GPIO\_PinModeSet(init->sdaPort, init->sdaPin, *gpioModeWiredAndPullUp*, 1);

**}**

1. Il faudrait également ré-ecrire la méthode I2C\_BusFreqSet(..) pour changer les calculs du registre CLKDIV pour obtenir les fréquences que l’on souhaite sans être dépendant du ratio des clocks.

