

Modulation/Démodulation PSK sur Simulink

Creation du model:	2
Test et validation du model:	8
Le test MIL (model in loop) :	8
Le test SIL (Software in loop) :	10

Creation du model:

On a essayé de mettre en pratique le filtre Chebyshev passe bas dans une application réelle qui répond à la problématique de transmission d'information récupérer par un système de mesure.

On utilise la modulation numérique de phase PSK (phase shift keying) pour moduler notre signal afin de le transmettre dans un espace bruité, le model utilise pour modéliser l'espace est 'AWGN' (Additive White Gaussian Noise) avec une source de bruit (Gaussian noise).

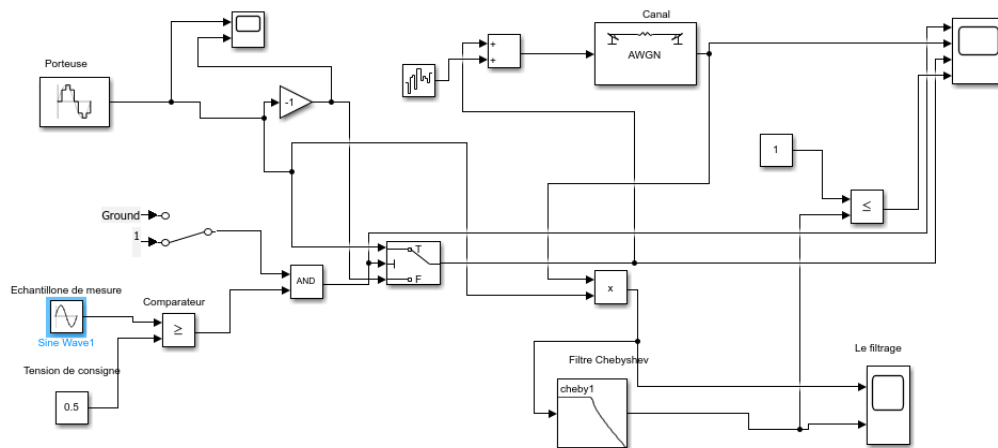


Figure 1 : Application complète

On parle ici de la modélisation d'une boucle de contrôle basé sur une application réelle, avec un bloc d'instrumentation et d'acquisition de données (échantillonnées de mesure), un comparateur de consigne et un bouton poussoir pour contrôler l'application.

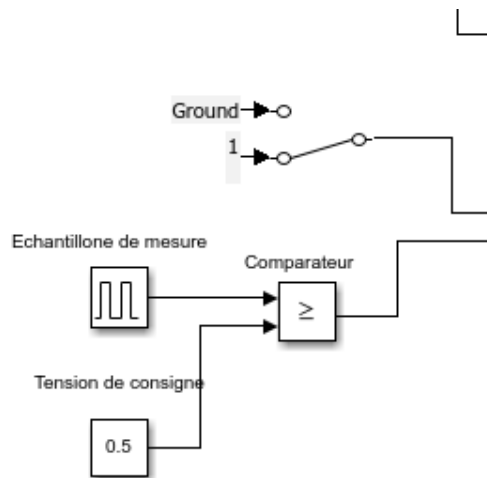


Figure 2 : Bloc d'instrumentation

On passe par la suite au bloc transmission de données, le bloc principal dans le système car il nous permet de transmettre l'information de commande dans un milieu bruite.

Alors ce bloc est reparti en deux bloc majeurs : la modulation et la démodulation.

La modulation peut être définie comme le processus par lequel le signal est transformé de sa forme originale en une forme adaptée au canal de transmission, l'information transporté dans l'application est un signal numérique, on utilise alors la modulation numérique et spécialement PSK pour coder le signal sous une onde porteuse de haute fréquence et a une amplitude capable de résisté l'erreur générer par le canal de transmission.

La porteuse est caractérisée par une fréquence de 4Hz avec une amplitude de 4V pour résister au canal de bruit.

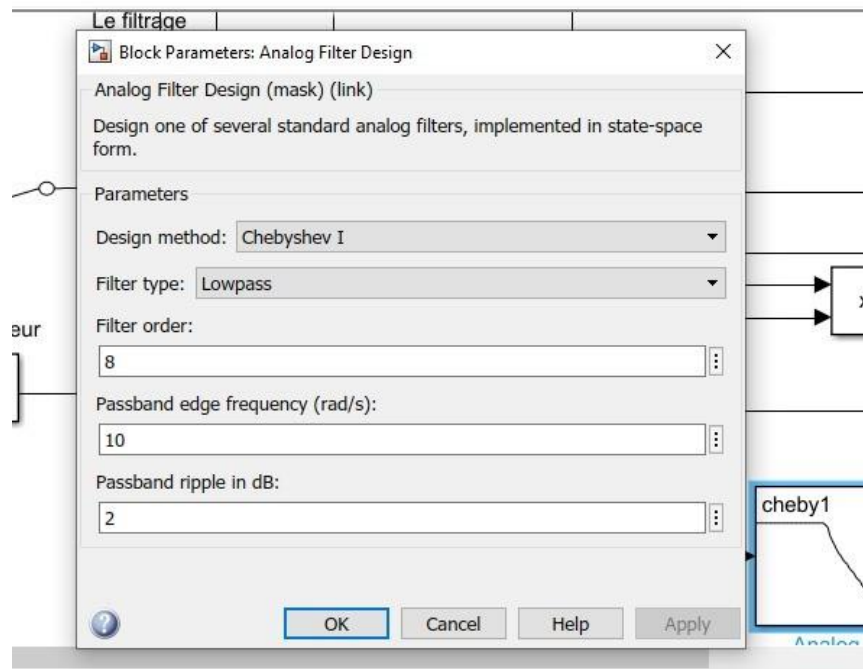


Figure 3: La porteuse

La modulation de l'information est faite au niveau de la phase du signal porteur, le déphasage en les deux signaux est 180 degrés.

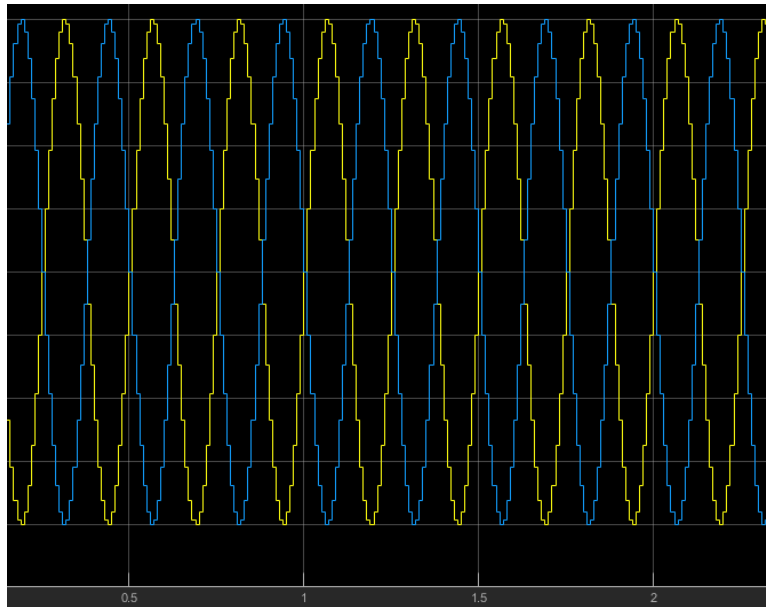


Figure 4: Le déphasage

La modulation numérique PSK consiste à attribuer au signal modulant les valeurs de la porteuse et de son signal déphasé, pour pouvoir par la suite récupérer le signal et l'utiliser.

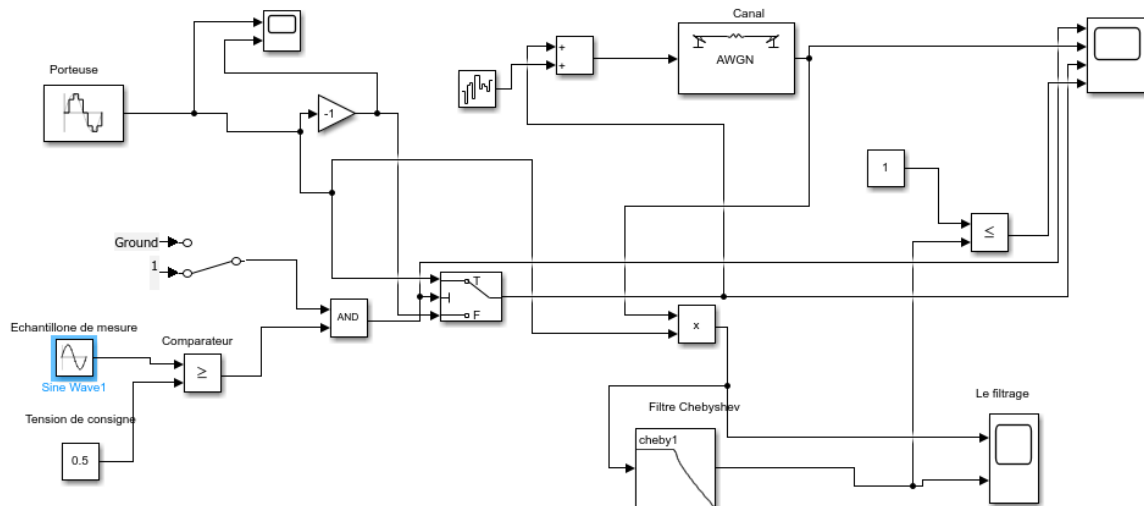


Figure 5 : Modulation numérique

On passe par la suite au canal de transmission, c'est la phase de transmission, le canal de transmission peut être soit un câble, soit une onde électromagnétique par un

Système d'antenne, le modèle utilisé pour modéliser l'espace est 'AWGN' (Additive White Gaussian Noise) avec une source de bruit (Gaussian noise) avec un rapport signal bruit $SNR=10\text{dB}$.

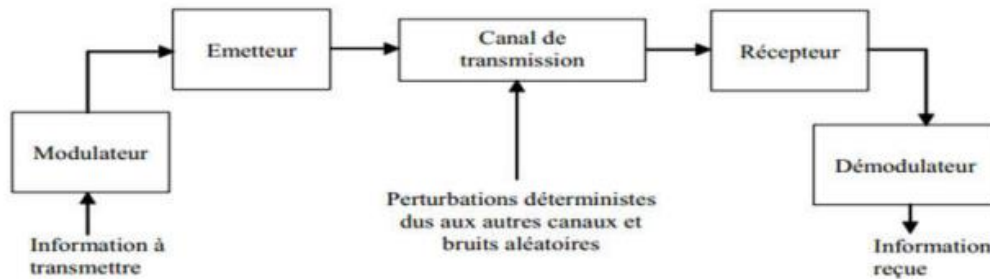


Figure 6 : Chaîne de transmission

La démodulation consiste à effectuer l'opération inverse, c'est-à-dire repasser dans un domaine spectral basse fréquence pour récupérer uniquement le signal BF. Elle se fait de la même manière avec un mélangeur et un oscillateur appelé oscillateur local, ce bloc peut être repartit en 3 étapes, la multiplication du signal à la sortie du canal par la porteuse, ensuite filtrer le signal en utilisons

le filtre Chebyshev, pour une fréquence porteuse de 4Hz, on a choisi les paramètres d'un filtre passe bas à ordre 20 pour éliminer les hautes fréquences avec un déphasage presque nulle.

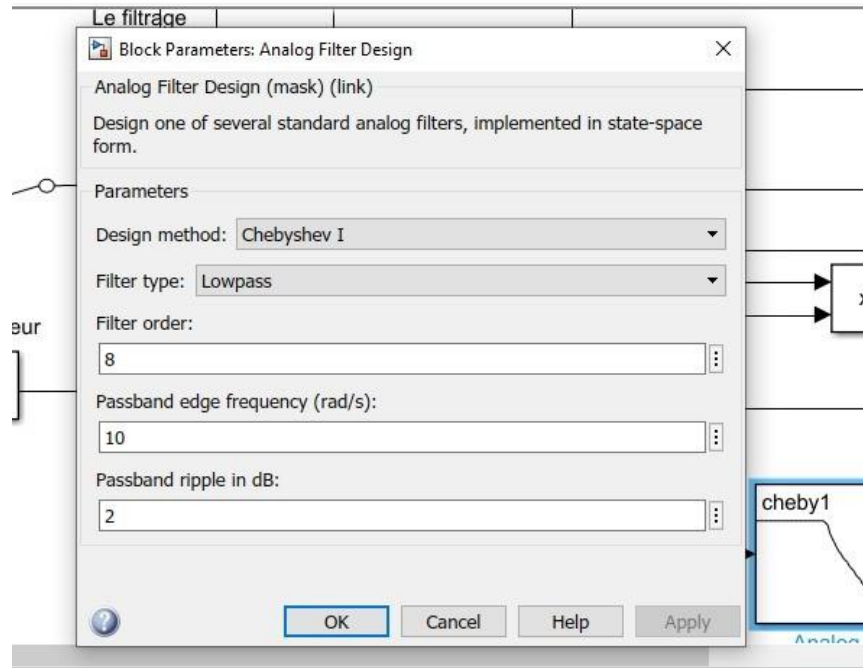


Figure 7: Caractéristique du filtre

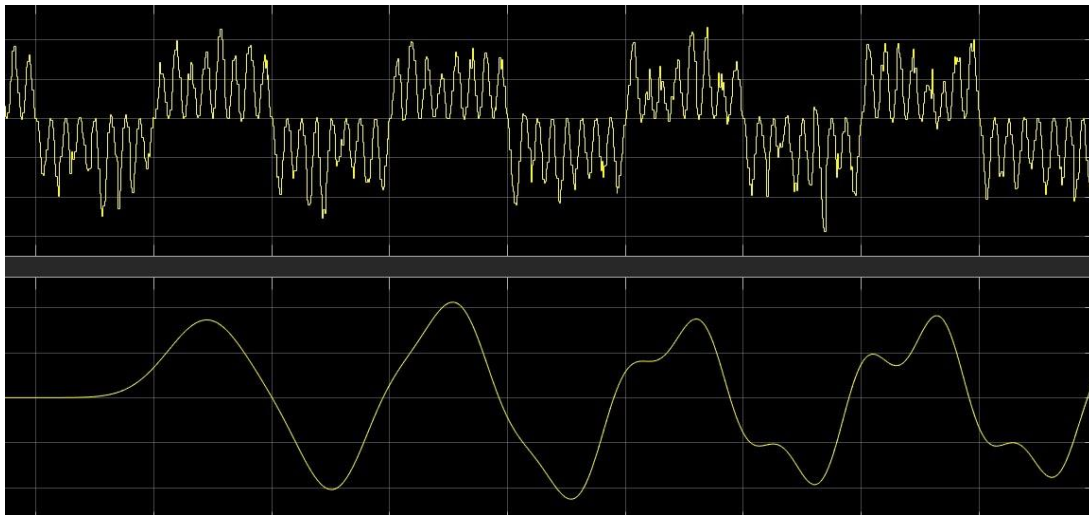


Figure 8: L'effet du filtre

Le résultat est le suivant, on a répondu au besoin de la conservation de l'information.

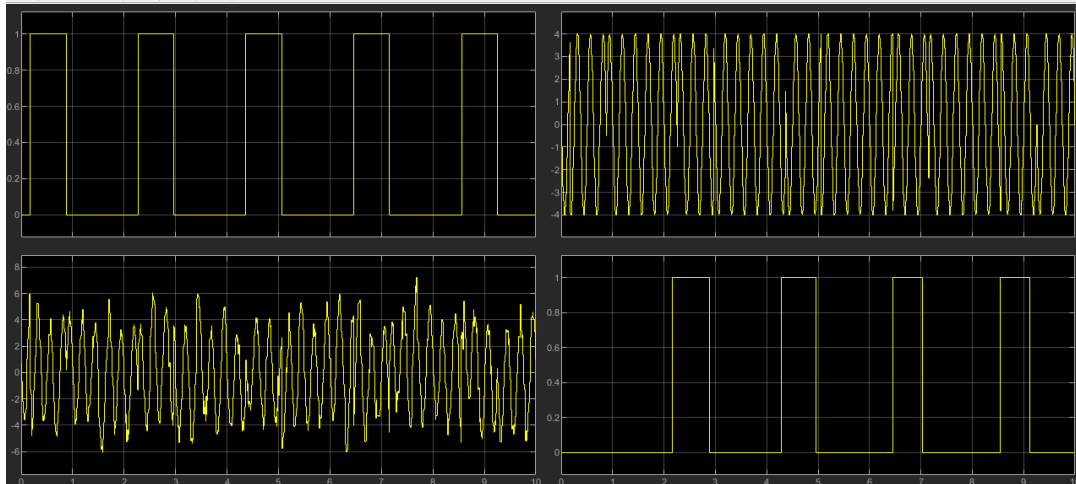


Figure 9: Resultat finale

On visualise premièrement le signal module qui représenté l'information utile (sortie du comparateur), dans le deuxième graphe on visualise le signal modulé ou on peut visualiser le changement de la phase à chaque changement d'état, ensuite le signal après la transmission.

Et finalement on récupéré le signal comme c'est présentée dans l'application.

Test et validation du model:

Le test MIL (model in loop) :

Le Test MIL (Model-in-the-Loop) comme est dit précédemment est une étape cruciale du processus de développement de systèmes embarqués, où le modèle du système est intégré dans un environnement de simulation. Cela permet de vérifier le comportement du système avant même que du code ne soit écrit. n valide par la suite les résultats de l'application :

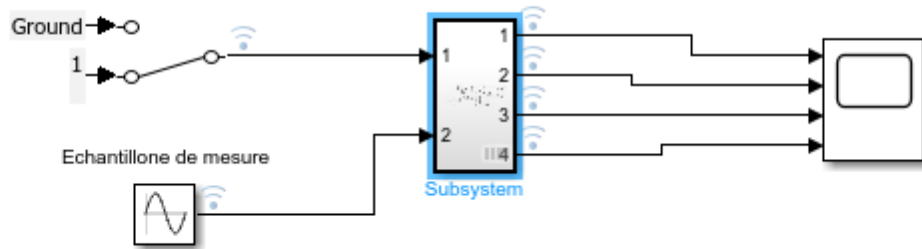


Figure 10: Creation du subsystem

Il faut s'assurer que les entrées et les sorties de votre sous-système sont correctement connectées aux autres parties de votre modèle à l'extérieur de la boucle.

Ensuite on configure les paramètres de simulation dans Simulink en fonction des besoins de votre test MIL, comme expliqué dans les étapes précédentes.

On lance enfin la simulation pour exécuter le test MIL avec la boucle que vous avez créée.

La vérification que le comportement du système correspond à ce qui est attendu.

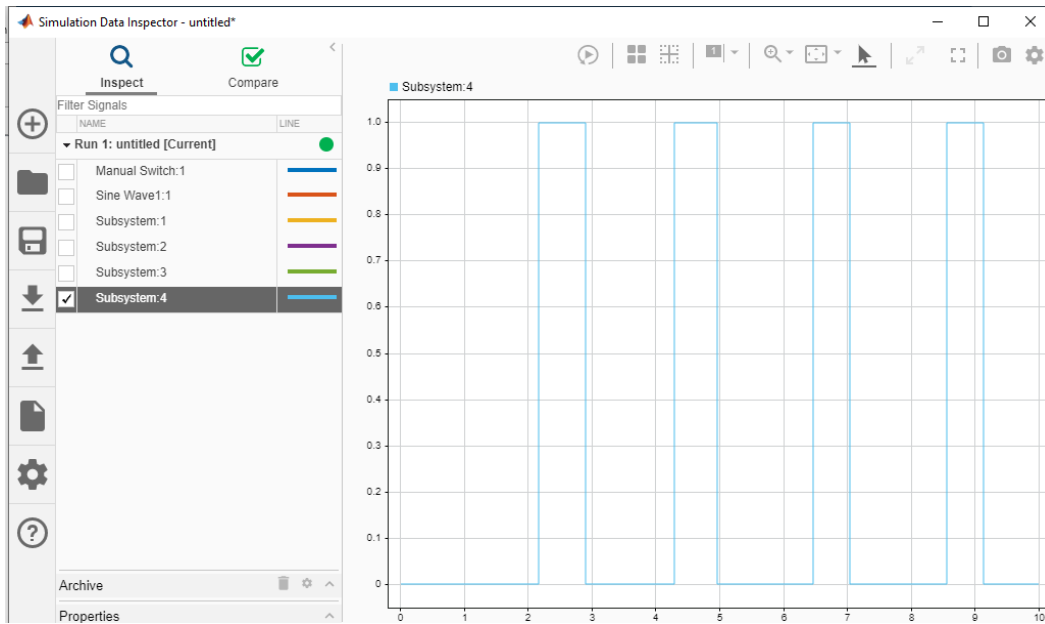


Figure 11: Test MIL

Le test SIL (Software in loop) :

On valide les résultats et on génère le code C qui réfère au comportement de l'application :

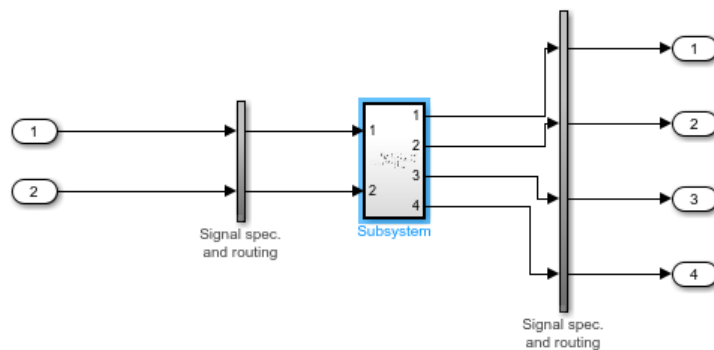
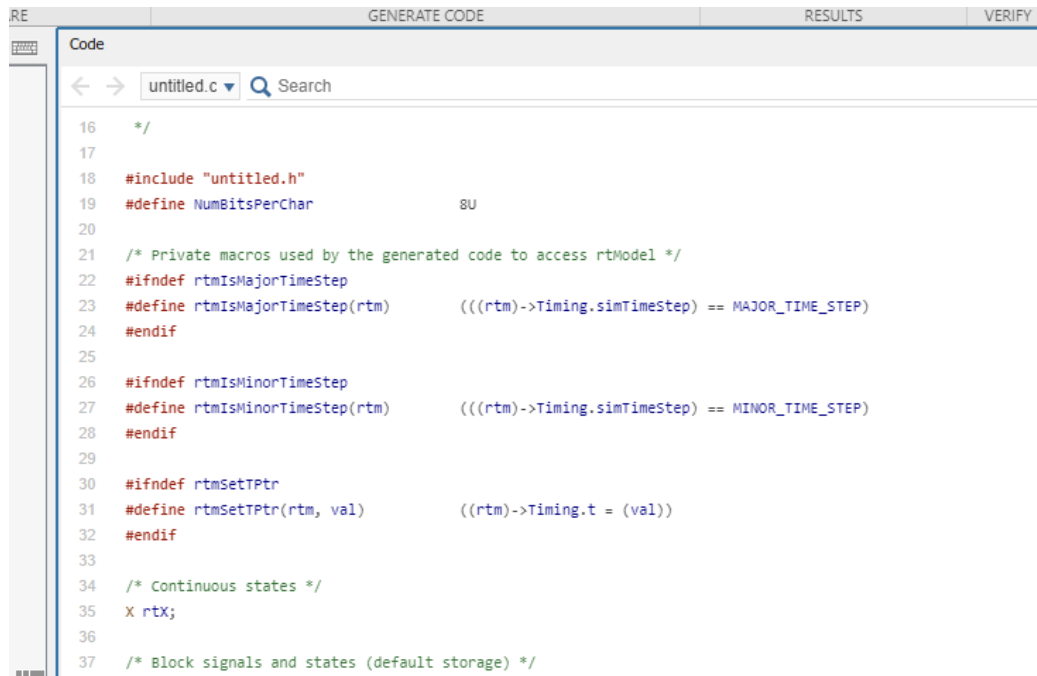


Figure 12: Harness création

La compilation et la génération du code est fait par **Code génération**.



```
16  */
17
18  #include "untitled.h"
19  #define NumBitsPerChar      8U
20
21  /* Private macros used by the generated code to access rtModel */
22  #ifndef rtmIsMajorTimeStep
23  #define rtmIsMajorTimeStep(rtm)      (((rtm)->Timing.simTimeStep) == MAJOR_TIME_STEP)
24  #endif
25
26  #ifndef rtmIsMinorTimeStep
27  #define rtmIsMinorTimeStep(rtm)      (((rtm)->Timing.simTimeStep) == MINOR_TIME_STEP)
28  #endif
29
30  #ifndef rtmSetTPtr
31  #define rtmSetTPtr(rtm, val)         ((rtm)->Timing.t = (val))
32  #endif
33
34  /* Continuous states */
35  X rtX;
36
37  /* Block signals and states (default storage) */
```

Figure 13: Génération du code

Le code généré est en langage C embarqué, qui est compatible avec les architectures ARM de la famille **cortex M**.