情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2010-SLDM-144 No.51
Vol.2010-EMB-16 No.51
Vol.2010-MBL-53 No.51
Vol.2010-UBI-25 No.51
2010/3/27

SILSを対象としたMCUペリフェラルプログラムの自動コード生成

松村郁生[†1]　小松秀昭[†1]

リアルタイムかつ高速な処理においては，ポーリングに比べて高速な応答が可能な，ペリフェラルと割込みを機能的に用いたプログラミングが必要とされる．既存のSILS技術ではポーリング処理の対象となるようなロジック部のコード生成については扱っているが，ペリフェラルと割込みを機能的に用いるような処理については扱っていない．本稿ではペリフェラルと割込みを機能的に用いたプログラムをサポートするSiLモデル仕様とコード生成の枠組みについて提案する．また，SiLモデル仕様についてエンジン制御モデルを対象にしたケーススタディについて述べる．

Autocoding of MCU Peripheral Program for SILS

Ikuo Matsumura[†1] and Hideaki Komatsu[†1]

In real-time and high-speed systems, programs need to leverage peripherals and interrupts in functional manner because it enables faster response than polling. Existing SILS technology covers only code generation of such programs on polling. In this paper, we propose a SiL model specification and a framework of code generation to enable generation of programs that leverage peripherals and interrupts in functional manner. And we also describe a case study of the SiL model specification on engine control model.

1. はじめに

制御系システムのコントローラをMCU(マイクロコントローラ）で実装する際には，演算語長による誤差や量子化誤差，演算遅延などが発生する．これに対しTargetLink®[⋆1]など既存のSILS(Software-in-the-Loop Simulation)技術では，信号のデータ型の指定や固定小数点数の倍率の設計を行うことで，MCUでの実装で生じる誤差の影響を検証することができる．これらのSILSツールは，シミュレーションと実装のためにSimulinkモデルからC言語などMCUソフトウェアの実装に用いるプログラミング言語の自動コード生成(Autocoding)を行う．

一方，リアルタイムシステムでは処理の開始や終了の時間に関する制約を満たすことが重要となる．リアルタイムかつ高速な応答が求められる処理では，ポーリングよりも高速な，ペリフェラルと割込みを機能的に用いたプログラミングが必要とされる．リアルタイムかつ高速な処理には例えば自動車のエンジン制御があり，今後もESC(Electronic Stability Control)など自動車のアクティブセーフティ技術や，悪条件下でのロボットの姿勢制御などにおいて重要性が増すと予想される．ペリフェラルと割込みの機能的なプログラミングは，ポーリング処理におけるインターバルタイマ割込みやネットワークデータ受付けを行う割込みのような単純な処理とは異なり，外界や種々のペリフェラルからの多数の割込みが様々なタイミングで発生する．このため，人の手でこのような処理を時間制約が満たされるようプログラミングし，不具合の発見と修正を行うことは容易でなくコストがかかる．従来のSILS技術はポーリング処理の対象となるようなロジック部のコード生成については扱っているが，ペリフェラルと割込みを機能的に用いるような処理については扱っていない．本稿ではリアルタイムかつ高速な処理を対象に，ペリフェラルと割込みを機能的に用いたプログラムをサポートするSiLモデル仕様と，コード生成の枠組みについて提案する．以下では単純な割込みハンドリングを含むペリフェラルの操作一般をペリフェラル・プログラムと呼ぶ．

以下，2節でSiLの位置づけとSiLモデルの仕様について述べ，3節でペリフェラル・プログラムのコード生成の枠組みについて述べる．4節で生成する割込み処理について述べる．5節でエンジン制御モデルを題材にしたケーススタディについて述べる．6節で関連研究について述べ，7節で結論を述べる．

2. SiLモデル仕様

この節ではまずSiLモデルの位置づけについて述べ，SiLモデルの仕様とペリフェラル・プログラムのコード生成の枠組みについて述べる．

2.1 SiLの位置づけ

一般にMiL(Model-in-the Loop)モデルは，コントローラの実装に関する制約のない理想

†1 日本アイ・ビー・エム（株） 東京基礎研究所
IBM Research - Tokyo

⋆1 http://www.dspaceinc.com/ww/en/inc/home/products/sw/pcgs/targetli.cfm

Vol.2010-SLDM-144 No.51
Vol.2010-EMB-16 No.51
Vol.2010-MBL-53 No.51
Vol.2010-UBI-25 No.51
2010/3/27

的な制御系モデルである．ここでコントローラとは，プラントの出力を所望の値に維持・変化させるためのプラントへの入力（操作量）を求める機能である．また SiL(Software-in-the Loop) モデルは，コントローラの一部が MCU ソフトウェアの実装言語レベルまで詳細化された段階の制御系モデルを指す．PiL(Processor-in-the Loop) モデルはさらに，コントローラの一部が実際の MCU によって実現された制御系モデルである．一方，1 つの MiL モデルに対して性能やコストの異なる複数の MCU 実装（PiL モデル）が存在しうる．SiL モデルは，この MCU 実装が満たすべき制約の仕様として捉えることが可能である．そこで本稿では SiL を，実装に対する制約が MiL に対して妥当かつ MCU 実装として実現可能性であるかを検証するものであると位置づける．

通常 MCU 実装を含む PiL モデルが MiL モデルと全く同じ信号出力を実現することは不可能である．このためある信号線に対し，MiL モデルでの信号出力からどれだけ離れているかを表す量「MiL との乖離」を考える．MCU 実装を含んでいる PiL モデルは，MiL との乖離ができるだけ小さくなる様に実装しなければならない．この「MiL との乖離」を生じさせる要因には，MCU 構成要素における誤差と遅延がある．ここで誤差とは値の乖離を指し，量子化誤差，演算語長による誤差（丸め誤差，打切り誤差，情報落ち，桁落ち）を含む．遅延は時間の乖離を指し，演算遅れ，センサやアクチュエータの遅れ，ペリフェラルの遅れを含む．SiL モデルには，MiL との乖離をどれだけ許容するかを MCU 実装に対する仕様として記述する．

前述のように本稿では SiL を，実装に対する制約が MiL に対して妥当かつ MCU 実装として実現可能性であるかを検証するものであると位置づける．つまり SiL モデルに対する操作には次の 2 つがある．

(1) 妥当性の検証
SiL モデルとして記述された仕様が MiL に対して妥当であることを検証する．シミュレータが SiL モデルに記述された MiL との乖離を発生させ，制御系全体に対する影響を調べる．

(2) 実現可能性の検証
具体的な MCU が与えられたとき，SiL モデルに記述された乖離の制約が実現可能であることを検証する．自動的または半自動的に，乖離の発生要因である遅延と誤差の制約を MCU の構成要素に割り当てる．実現可能性の検証は，MCU のターゲットプロセッサ用のコード生成も含む．

本稿では実現可能性の検証についてとりあげ，ペリフェラル・プログラムのコード生成について述べる．

設定項目	設定対象	Simulink モデル表現	周期的な制御機能	非周期的な制御機能
許容乖離度	制御機能の入出力信号線	max-difference 注釈	-	-
実行間隔	ブロック	tsample 注釈	指定必須	指定不可
最大遅延	ブロック	max-delay 注釈	指定可能	指定必須

表 1 機能レベルの SiL モデルの仕様
Table 1 Specification of Function-Level SiL Model

2.2 SiL モデル仕様

この節では，SiL モデルに記載すべき要素と，SiL モデルに基づくコード生成の枠組みについて述べる．

2.2.1 機能レベルの SiL モデル

SiL モデルにはまず信号線に対して，許容する MiL からの乖離（以下，許容乖離度と呼ぶ）を指定する必要がある．全ての信号線に許容乖離度を指定すると手間がかかる上，実装の際に計算方法が変わることで意味をなさなくなる信号線もある．このため，制御を実現する機能の単位（以下，制御機能と呼ぶ）ごとに，その入出力に対して許容乖離度を指定する．また，制御機能には周期的なものと非周期的なものがあり，時間制約を検証するためにはそれぞれ実行間隔と最大遅延が重要な意味を持つ．非周期的な制御機能は 1 つ以上のトリガイベント（制御機能を実行するトリガとなるイベント）を持つ．最大遅延は，トリガイベントのいずれかが生じてから出力にその影響が反映されるまでの最大の時間である．また，最大遅延は周期的な制御機能についても意味を持つ．このように制御機能の単位を明らかにして許容乖離度と実行間隔および最大遅延を指定したモデルを「機能レベルの SiL モデル」と呼ぶ．機能レベルの SiL モデルの仕様を**表 1** に示す．ただし許容乖離度は乖離度の尺度とともに入出力信号線に対して指定するものとする．

2.2.2 MCU 依存の SiL モデル

遅延と誤差の制約を割り当てる対象として，コントローラの構成要素を SiL モデルに表現する必要がある．コントローラを構成する要素は次の 3 つに分類することができる．

(1) ソフトウェア：メモリに格納された命令として CPU により実行される機能
(2) ペリフェラル：ソフトウェアから駆動されるハードウェア機能
(3) センサとアクチュエータ：光や磁気などの信号と電気的な信号とを変換するハードウェア機能

(1) ソフトウェアの機能単位（以下，タスクと呼ぶ）も制御機能と同様に，周期的なもの

Vol.2010-SLDM-144 No.51
Vol.2010-EMB-16 No.51
Vol.2010-MBL-53 No.51
Vol.2010-UBI-25 No.51
2010/3/27

入出力/設定項目	Simulink モデル表現	周期的実行の場合	非周期的実行の場合
トリガ入力	Trigger Port	指定不可	指定必須
実行完了出力	done 出力ポート	指定可能	指定可能
実行間隔	tsample 注釈	指定必須	指定不可
最大遅延	max-delay 注釈	指定可能	指定必須

表 2 タスクの入出力と設定項目
Table 2 Input/Output and Parameters of Tasks

入出力/設定項目	Simulink モデル表現
アナログ入力	analog 入力ポート
デジタル出力	digital 出力ポート
量子化ビット数	resolution 注釈

表 3 A/D 変換器 (ADC) の入出力と設定項目
Table 3 Input/Output and Parameters of A/D Converter

入出力/設定項目	Simulink モデル表現
デジタル入力	digital 入力ポート
アナログ出力	analog 出力ポート
量子化ビット数	resolution 注釈

表 4 D/A 変換器 (DAC) の入出力と設定項目
Table 4 Input/Output and Parameters of D/A Converter

入出力/設定項目	Simulink モデル表現	備考
アラーム時間入力	max 入力ポート	-1 を指定するとフリーラニング動作
スタートパルス入力	start 入力ポート	パルスを入力するとカウント開始
停止パルス入力	reset 入力ポート	
アラームパルス出力	alarm 出力ポート	max までカウントするとパルスを出力
カウント値出力	current 出力ポート	
カウント間隔	intarval 注釈	カウントアップする際の時間間隔

表 5 タイマ (Timer) の入出力と設定項目
Table 5 Input/Output and Parameters of Timers

付加機能	識別子
ポート出力	⟨port-output⟩
デイジーチェーン	⟨daisy-chain⟩
インプットキャプチャ	⟨input-capture⟩

表 6 タイマの付加機能の識別子
Table 6 Identifiers of Additional Function of Timers

と非周期的なものがある．このため周期的なタスクには実行間隔と最大遅延を，非周期的なタスクにはトリガイベントと最大遅延を指定する．MAAB による制御アルゴリズムのスタイルガイド[1] では，周期的なタスクはその実行間隔を tsample の値として，非周期的なタスクのトリガイベントをトリガポートとして指定する．さらにタスクは計算機上で実行されるため，離散ブロックのみ利用できる．本稿でもタスクの実行間隔とトリガイベント，離散ブロックの利用に関して 1) のスタイルガイドを踏襲する．また，あるタスクが別の処理を駆動することがあるため，タスクの実行完了イベントをモデルに表現できる必要がある．このため実行完了のイベントを done という出力ポートとして表現する．以上を整理するとタスクの入出力と設定項目は**表 2** となる． また，システムの初期化時に何らかの処理を実行したり，初期化処理の終了をトリガとしてタスクやペリフェラルを起動することが必要な場合がある．このためシステムの初期化後に実行されるタスク（初期タスク）を SiL モデル仕様として用意する．初期タスクはトリガ入力を持たない非周期的タスクであり，Simulink モデル表現では ⟨initial⟩ という識別子によって他のタスクと区別する．

(2) ペリフェラルには A/D 変換器，タイマなどがあり，その機能に対応したブロックを用いて記述する．ペリフェラルの入出力と設定項目の仕様は MCU の仕様で与えられるものとする．ペリフェラルには基本的な機能に付加的な機能が付け加わったバリエーションを持つものがある．例えばタイマには指定時刻にアラームを発生させる基本的なタイマの他に，ポート出力機能（外部ポートにアラームが発生するまで出力を行う機能）やデイジーチェーン機能（アラーム時に割込み処理を介在させずに別のタイマをスタートさせる機能[⋆1] ），インプットキャプチャ機能（外部ポート信号イベントの発生を検出しその時のタイマ値をレジスタに記録する機能）などを付加したバリエーションがある．タイマにはこれらの付加機能を複数持つものもあり，全ての機能の組合せに対してペリフェラルの仕様を定めるのは効率的でない．このためペリフェラルの入出力と設定項目の仕様は，必要に応じて基本機能と付加機能の形で定義する．組込みシステムでよく利用される A/D 変換器 (ADC)，D/A 変換器 (DAC)，タイマ (Timer) の入出力と設定項目をそれぞれ**表 3**，**表 4**，**表 5** に示す．また上で述べたタイマの付加機能の識別子を**表 6** に，付加機能の入出力と設定項目の仕様を**表 7** に示す．ADC，DAC はタスクと同様に周期的または非周期的な実行タイミングが考えられる．このため ADC，DAC の入出力と設定項目の仕様は表 2 を含む．

(3) センサとアクチュエータには，ロータリーエンコーダやソレノイドなどがあり，その機能に対応したブロックを用いて記述する．センサとアクチュエータのパラメータと入出力は外部の仕様で与えられるものとする．

以上のようにコントローラの構成要素が明らかになることで，各要素に遅延と誤差の制約

⋆1 ルネサステクノロジ社のマイクロコンピュータにおけるオフセット付きワンショットパルス機能[2]

Vol.2010-SLDM-144 No.51
Vol.2010-EMB-16 No.51
Vol.2010-MBL-53 No.51
Vol.2010-UBI-25 No.51
2010/3/27

指定必須となる付加機能	入出力/設定項目	Simulink モデル表現
⟨port-output⟩	ポート出力	port-output 出力ポート
⟨port-output⟩	出力を High から Low にするか Low から High にするか	output 注釈 （“high to low” or “low to high”）

表 7 タイマの付加機能の入出力と設定項目の仕様
Table 7 Input/Output and Parameters of Additional Function of Timers

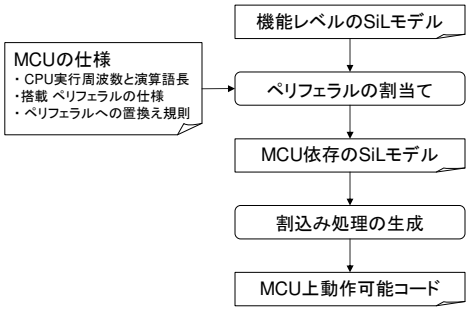

図 1 ペリフェラル・プログラムのコード生成の枠組み
Fig. 1 A Code Generation Framework of Peripheral Programs

を割当てることができる．

3. コード生成の枠組み

図 1 に 2 節で述べた機能レベルの SiL モデル，MCU 依存の SiL モデルを用いたコード生成の枠組みを示す． MCU の仕様は MCU が搭載するハードウェアの仕様であり，CPU の実行周波数と演算語長，搭載ペリフェラルの仕様を含む．また MCU の仕様は，機能レベル SiL における特定のブロックと信号線とのパターンを，MCU のペリフェラルを用いた特定のパターンで置き換える規則を含む．コード生成器は機能レベルの SiL モデルと MCU の仕様をもとに，MCU における処理に対してペリフェラルを割当てる．ペリフェラルを割当てることでソフトウェアで処理する部分（タスク）が確定する．この段階で演算語長が確定するので，タスク内のブロックが用いるデータの型を決定する．

利用するペリフェラルが定まると全ての割込み処理の内容が確定する．このため MCU 依存の SiL モデルをもとに割込み処理のコード生成を行い，MCU 上で動作可能なコードを生成することができる．次の節では生成すべき割込み処理について述べる．

4. 割込み処理コードの生成

この節では，ソフトウェアタスクを割込み処理コードとして実現する方法について述べる．一般的な MCU における割込みコントローラは，割込みシグナルごとに優先度とハンドラのアドレスをソフトウェアから設定することで動作する．

まず割込みにおいて注意すべき点と，本稿におけるアプローチについて述べる．その後，コード生成の出力となる割込み処理について述べる．

4.1 割込み処理における課題とアプローチ

ソフトウェアタスクを一般的な割込みコントローラで実現する際には，次の 2 点を考慮する必要がある．

(1) 割込みイベントの消失を防ぐ
(2) 時間制約を満たしながら排他制御を行う

割込みイベントの消失は，実際には同一の割込みが連続して生じたにも関わらず CPU には割込みイベントが 1 つしか通知されない現象である．割込みイベントの消失を防ぐためには，発生した割込みをすぐにソフトウェア側で受け付けて割込みシグナルをリセットした後，残りの処理を実行する必要がある．このような割込みハンドリングは OS のデバイスドライバにおいて行われており，例えば Linux では割込みを受付ける処理と残りの処理はそれぞれ Top Half, Bottom Half と呼ばれている．割込みイベントの消失を防ぐため生成コードでは，割込みを一旦受付けて割込みシグナルをリセットし，短時間のうちに終了する処理が必要である．

割込み処理で用いられる代表的な排他制御には，クリティカルセクションにおける割込みの無効化，ロックの取得と解放，Lock-free/Wait-free 同期[3)4)] がある．割り込みの無効化はオーバーヘッドが少ないが，クリティカルセクションが長くなると割込みを受付けられない時間も長くなる．この場合，タスクの時間制約を満たせない危険性や割込みイベントの消失の危険性がある[5)]．ロックの取得と解放はマルチプロセッサにおいても利用できる排他制御方式であるが，割込みの無効化に比べてオーバーヘッドが大きい．ロックに伴うタスクのブロッキングは，時間制約を満たす上で障害となる．一方 Lock-free/Wait-free 同期はブロッキングを行わないため時間制約を満たす上で有利であるが，CAS(Compare and Swap) のような命令をサポートした CPU が必要である．これらの排他制御の方式のうちどれがよいかは，利用するプロセッサの数や利用できる CPU 命令などによって異なる．

以下では，CAS のような命令をサポートしない CPU のユニプロセッサ環境を想定し，割

Vol.2010-SLDM-144 No.51
Vol.2010-EMB-16 No.51
Vol.2010-MBL-53 No.51
Vol.2010-UBI-25 No.51
2010/3/27

込み処理の排他制御をロックと Lock-free/Wait-free 同期を用いずに行う場合を考える．前述のように，割込みの無効化には割込みイベントの消失の危険性がある．このため本稿では，割込みの無効化の代わりに，優先度上限プロトコル[6](PCP, priority ceiling protocol) の考え方を導入する．PCP はロックを用いた排他制御における優先順位の逆転 (priority inversion) を防ぐための手法である．RCP ではセマフォ毎に，それをロックする可能性のあるタスクの優先度の最大値（シーリング）を求めておく．あるタスクがセマフォをロックしようとする際，他のタスクによってロックされている全てのセマフォより優先度が高い場合にのみロックが許される．そうでない場合タスクはブロックされ，その際最も高いシーリングのセマフォをロックしているタスクの優先度が，ブロックされたタスクの優先度まで引き上げられる．PCP におけるシーリングをセマフォのかわりにリソースに対して考え，セマフォのロックのかわりにリソースアクセスを伴うタスクの実行を考えると，ロックを行わない場合にも PCP を考えることができる．

続く節では，割込みの受付と，ロックを行わない PCP に基づく割込み処理について述べる．

4.2 生成する割込み処理

生成コードを次の 3 つのルーチンで構成する．

割込み受付ルーチン： 論理タスクごとに生成される ISR (割込みサービスルーチン)．割り込みの発生を受付けてタスクを生成し，ディスパッチルーチンにタスクの発生を通知する．割込みの発生をすぐ知るために，割込み受付ルーチンの優先度は全て最高レベルに設定する．

ディスパッチルーチン： MCU に対して 1 つ生成される ISR．実行可能なタスクのなかから，最も優先度の高いもを選んで実行する．実行可能なタスクを管理するキュー（タスクキュー）を 1 つ持つ．プリエンプションを実現するために，サービスルーチンより高い優先度を設定する．

サービスルーチン： 論理タスクごとに生成されるルーチン．論理タスク本体の機能を実行する．

また，共有リソース R に対して R のシーリング $\mathrm{ceil}(R)$ を，R にアクセスする論理タスクの優先度の最大値に設定する．

各ルーチンの処理は次のとおりである．ただし，$\mathrm{pri}(T)$ はタスク T の優先度を指す．また，タスク T_1 と T_2 が排他的であるとは，T_1 に対応する論理タスクと T_2 に対応する論理タスクがリソースを共有していることを指す．

割込み受付ルーチン

(1) 割込みイベントの消失を防ぐため，受付けた割込みのシグナルをリセットする
(2) 受付けた割込みに対応するタスク T_1 を生成し，タスクキューに追加する
(3) タスクキューに T_1 と排他的なタスク T_2 がある場合は，T_2 の実行中に T_1 が実行されるのを防ぐため，$\mathrm{pri}(T_2)$ を $\max\{\mathrm{pri}(T_1), \mathrm{pri}(T_2)\}$ にセットする
(4) タスクの到着を通知するため，ディスパッチルーチンの割込みシグナルをセットして終了する

ディスパッチルーチン

(1) タスクキューから最も優先度の高いタスクの中で先着のもの T_1 を取り出す
(2) 実行中のタスク T_2 に対して下の条件が成り立つ場合，T_2 の実行コンテキストをスタックに退避して T_1 を実行する
- $\mathrm{pri}(T_1) > \mathrm{pri}(T_2)$
- $\mathrm{pri}(T_1)$ がタスクキュー内のタスクがアクセスする全てのリソースのシーリングよりも大きい

サービスルーチン

(1) 論理タスク本体の機能を実行する

5. ケーススタディ

この節ではリアルタイムかつ高速な応答が要求される処理として，自動車のエンジン制御を対象にケーススタディーを行う．考察の対象とするエンジン制御系モデルの概要を**図 2** に示す． 運転者 Driver はプラント Plant の速度 Velocity が目標速度 60km/h となるように，目標との偏差 Deviation を入力としてアクセルペダルの開度 Accel Pedal を調節する．プラントの速度はコントローラから与えられるスロットルバルブの開度 Throttle，燃料噴射 Injection および点火 Ignition から定まる．コントローラはプラントで生じた燃焼の空燃比 A/F とクランク角 Crank Angle（クランクシャフトの角度），カム角 Cam Angle（カムシャフトの角度）を読み取る．コントローラはアクセルペダルの開度に基づいてスロットルバルブの開度を調整する．コントローラはさらに空燃比から燃料噴射の量を，クランク角とカム角から燃料噴射と点火のタイミングを決定する．SiL モデルを作成する前には MiLS によってこのエンジン制御モデルが期待通り動作することが確認されており，その際の信号出力のデータが得られている．

Vol.2010-SLDM-144 No.51
Vol.2010-EMB-16 No.51
Vol.2010-MBL-53 No.51
Vol.2010-UBI-25 No.51
2010/3/27

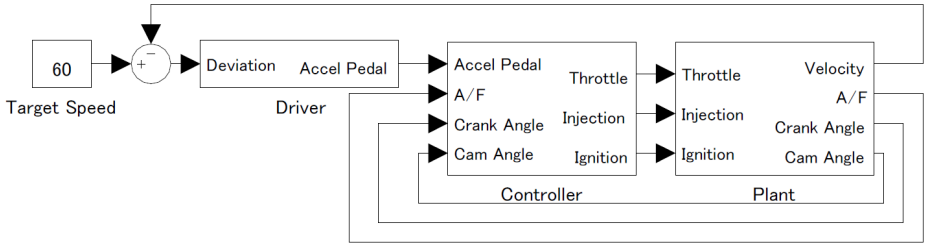

図 2 エンジン制御系モデルの概要
Fig. 2 An Overview of Engine Control System Model

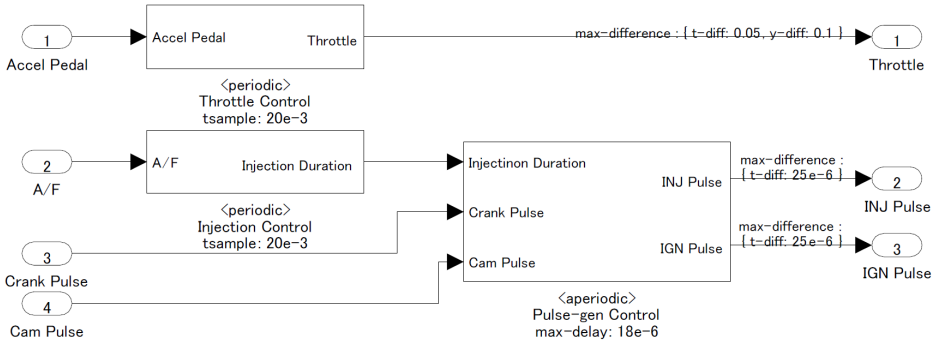

図 3 機能レベルの SiL モデルにおける MCU
Fig. 3 The MCU Part of Function-Level SiL Model

5.1 機能レベルの SiL モデル

この節では，図 2 のモデルに対応した機能レベルの SiL モデルについて考える．まず，コントローラ部分が信号の性質に基づいてセンサ/アクチュエータ部分と MCU 部分に分離されたモデルを考える．Simulink における信号は時間とともに変化する値[7] であり，センサ/アクチュエータは光や磁気などの信号と電気的な信号とを変換する装置[8] である．このため信号が電気的か否かを基準にすることで，コントローラをセンサ/アクチュエータと MCU に分離することが可能である．また，クランク角とカム角を知るためのセンサとして，ここではロータリーエンコーダを用いるものとする．つまり，クランクシャフトとカムシャフトが一定角度回転するごとにセンサはパルスを出力する．このようなセンサ/アクチュエータを利用すると，機能レベルの SiL モデルにおける MCU は**図 3** のようになる． モデルは 3 つの制御機能ごとにサブシステム化されており，信号線と制御機能は表 2 に示した入出力と注釈を持つ．スロットル開度の制御 Throttle Control と燃料噴射期間の制御 Injection Control は周期的な制御機能であり，tsample によってどちらも 20 ms の実行間隔を持つことが指定されている．

パルス生成制御 Pulse-gen Control はクランク角センサとカム角センサからのパルス Crank Pulse，Cam Pulse と Injection Control が計算した燃料噴射期間 Injection Duration の 3 つの信号を入力として受け取る．受け取った 3 つの入力をもとに，燃料噴射と点火を行うアクチュエータへの信号 INJ Pulse，IGN Pulse を出力する．この制御機能には max-delay によって最大遅延が 18 μsec であることが指定されている．Pulse-gen Control のトリガイベントを生じさせる入力は Crank Pulse，Cam Pulse の 2 つであり，出力は INJ Pulse，IGN Pulse の 2 つである．従ってこの入出力の 4 つの組合せ全てに対し，入力によるトリガイベントが生じてからそのイベントに基づく計算結果が出力に反映されるまでの時間が 18 μsec 以下でなければならない．

Throttle Control の出力信号線には max-difference により最大乖離度が指定されている．ここでは「MiL の信号出力に比べて時間方向と値方向に ±t-diff，±y-diff の領域に収まっていなければならない」という最大乖離度の尺度が定義されているとする．この例では MiL の信号出力に比べて時間方向に ±0.05，値方向に ±0.1 だけ離れた領域に信号出力が収まらなければならない．Pulse-gen Control の 2 つのパルス出力 INJ Pulse，IGN Pulse も同様に，MiL の信号出力に比べて時間方向に ±25 μs だけ離れた領域に信号出力が収まらなければならない．

5.2 MCU 依存の SiL モデル

この節では MCU の仕様が与えられたときの，図 3 に対応する MCU 依存の SiL モデルについて考える．仕様が与えれた MCU は ADC, DAC, Timer, Timer/port-output（ポート出力機能付きタイマ）が十分な数搭載されており，Timer/daisy-chain（デイジーチェーン機能付きタイマ）と Timer/input-capture（インプットキャプチャ機能付きタイマ）は搭載されていないものとする．このような MCU 仕様が与えられた場合の MCU 依存の SiL モデルの MCU を**図 4** に示す．

図 3 における Throttle Control には ADC adc1 と DAC dac1 の 2 つのペリフェラルが割当てられている．Injection Control には ADC adc2 が割当てられている．adc1, adc2, dac1

Vol.2010-SLDM-144 No.51
Vol.2010-EMB-16 No.51
Vol.2010-MBL-53 No.51
Vol.2010-UBI-25 No.51
2010/3/27

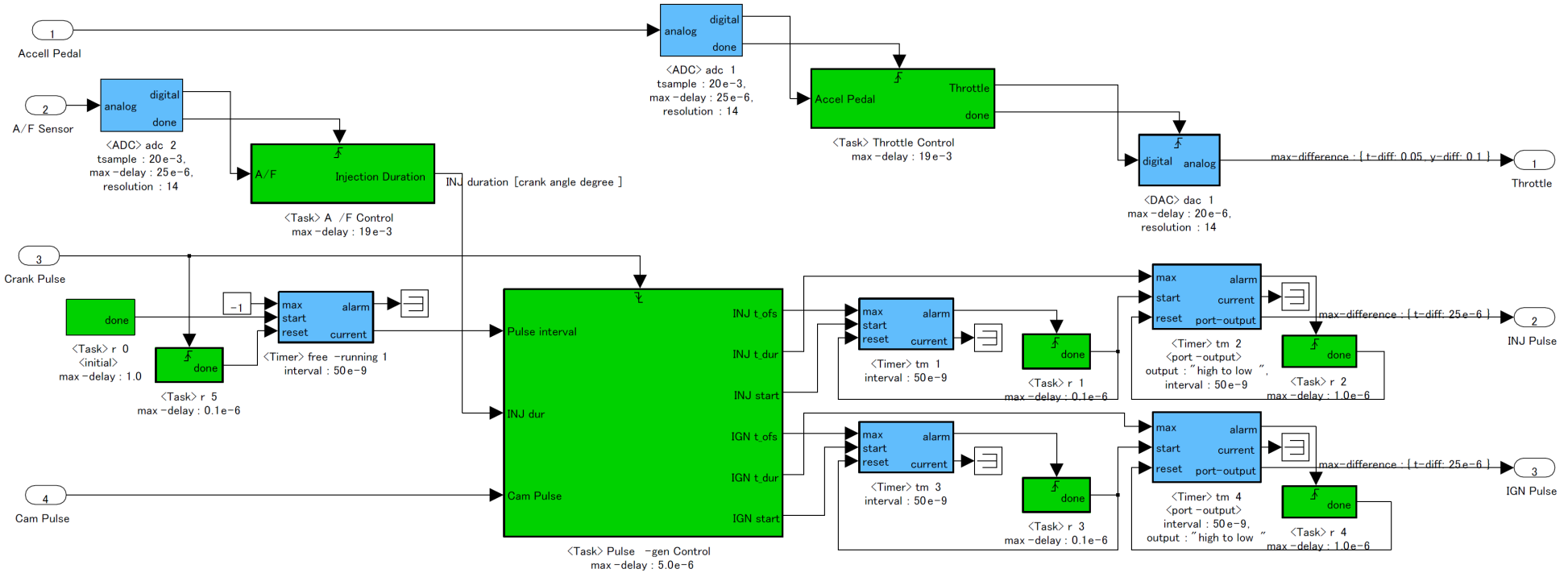

図 4 MCU 依存の SiL モデルの MCU
Fig. 4 The MCU Part of MCU-Specific SiL Model

には，MCU 仕様によって定められた最大遅延と解像度がそれぞれ max-delay, resolution によって 25 μs，14 bit に指定されている．この 2 つの制御機能は周期的であるため adc1, adc2 はいずれも tsample によって実行周期が 20 ms と指定されている．この値は図 3 における Throttle Control，Injection Control の tsample の値と一致する．

図 3 における Pulse-gen Control には 5 つのタイマ free-running1, tm1, tm2, tm3, tm4 が割当てられている．Pulse-gen Control タスクは燃料噴射量やパルス出力のタイミングを計算し，タイマを操作する．全てのタイマには，MCU 仕様で定められた実行間隔が interval によって 50 ns と指定されている．free-running1 には max に “-1” をセットすることでフリーラニング動作をさせている．free-running1 はクランクパルスの到着間隔を知るために用いられる．tm1 と tm2 はそれぞれ燃料噴射パルスを出力するまでのオフセット時間（INJ t_ofs）と，出力開始から出力終了までのホールド時間（INJ t_dur）をカウントするために用いられる．tm3 と tm4 も同様に，点火パルスのオフセット時間とホールド時間をカウントするために用いられる．アクチュエータにパルス出力を行うために tm2, tm4 には ⟨port-output⟩ によってポート出力機能をもつタイマが割当てられている．さらに tm2, tm4 には output によって，出力変化の方式が “high to low” に指定されている．5 つのタイマの開始とリセットを行うために，6 つのタスク r0〜r5 が配置されている．このうち r0 はフリーラニングタイマの動作をスタートさせるタスクであり，⟨initial⟩ によって初期タスクとして指定されている．Pulse-gen Control とあわせたこれらの 7 つのタスクには max-delay によって最大遅

Vol.2010-SLDM-144 No.51
Vol.2010-EMB-16 No.51
Vol.2010-MBL-53 No.51
Vol.2010-UBI-25 No.51
2010/3/27

延が指定されている．このように，MCU を構成するペリフェラルとタスクが確定することで誤差と遅延を評価することができる．また，よくデザインされた「ペリフェラルへの置換え規則」を用いることでこのようなペリフェラルの割当てとタスクの配置を機械的に行うことができると考えられる．

6. 関連研究

ペリフェラルを Simulink の上でモデル化する手法として 9),10) がある．これらの手法では OSEK OS で定義されているタイマなどのペリフェラルに対応するブロックを用いてモデルを記述する．この手法では，記述されたモデルをもとに OSEK OS 上で動作するコード生成を行うことができる．しかし，ペリフェラルや割込みを機能的に用いるプログラムのコード生成は対象としていない．

11) はセンサの障害時などで意図しない多数の割込みイベントが発生した際に，プロセッサの過負荷を防ぐような割込みの処理方式である．この手法では，割込み発生頻度から異常を判断することができる．しかしこの手法では，共有リソースにおける排他制御については対象としていない．

7. 結　　論

本稿ではリアルタイムかつ高速な処理を対象に，ペリフェラルと割込みを機能的に用いたプログラムをサポートする SiL モデル仕様と，コード生成の枠組みについて提案した．リアルタイムかつ高速な処理としてエンジン制御をとりあげ，SiL モデル仕様についてのケーススタディを行った．今後の課題としてペリフェラルの仕様，置換え規則，割当てアルゴリズムの詳細な検討が必要である．また，生成する割込み処理についても検証が必要である．

参 考 文 献

1) MathWorks Automotive Advisory Board (MAAB): CONTROL ALGORITHM MODELING GUIDELINES USING MATLAB, Simulink, and Stateflow Version 2.0 (2007).
2) Renesas Technology Corp.: SH-2E SH7058F-ZTAT ハードウェアマニュアル (2004).
3) Ramamurthy, S.: A Lock-Free Approach to Object Sharing in Real-Time Systems, PhD Thesis, The University of North Carolina at Chapel Hill (1997). Adviser-Anderson, James.
4) Hohmuth, M. and Härtig, H.: Pragmatic Nonblocking Synchronization for Real-Time Systems, *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, Berkeley, CA, USA, USENIX Association, pp.217–230 (2001).
5) Davari, S. and Sha, L.: Sources of Unbounded Priority Inversions in Real-time Systems and a Comparative Study of Possible Solutions, *SIGOPS Oper. Syst. Rev.*, Vol.26, No.2, pp.110–120 (1992).
6) Goodenough, J.B. and Sha, L.: The Priority Ceiling Protocol: A Method for Minimizing the Blocking of High Priority Ada Tasks, *IRTAW '88: Proceedings of the second international workshop on Real-time Ada issues*, New York, NY, USA, ACM, pp.20–31 (1988).
7) The MathWorks, inc.: *Simulink 7 User's Guide* (2009).
8) Lion, K.S.: Transducers: Problems and Prospects, *IEEE Transactions on Industrial Electronics and Control Instrumentation*, Vol.16, No.1, pp.2–5 (1969).
9) Thomsen, T., Köster, L. and Stracke, R.: Connecting Simulink to OSEK: Automatic Code Generation for Real-time Operating Systems with TargeLlink (2001).
10) Sunwoo, M.: Development of SILS and RCP for OSEK-OS Based ECU, *Proceedings of the 17th IFAC World Congress, 2008* (2008).
11) Regehr, J. and Duongsaa, U.: Preventing Interrupt Overload, *Proceedings of the 2005 ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems*, Vol.40, No.7, ACM New York, NY, USA, pp.50–58 (2005).