開発プロセス間における
抽象度の異なる制御モデルの
接続策定ガイドライン

(ver.1.0)

改訂履歴

Rev.	日付	内容	会社名	承認者
1.0	2023/02/27	初版	ルネサス エレクトロニクス	船橋

目次

1. 背景

現代の自動車開発では、「自動運転」、「コネクティッド」、「コックピット連携」等の「走る・曲がる・止まる」以外の開発が必要である。機能の複雑化によって検証工数が肥大化する等の課題が生じており、解決のためには開発プロセスを跨いだ連携が求められる。対応のために経済産業省にて「自動車産業におけるモデルのあり方に関する研究会」が発足した。これまでに開発プロセス間の 1D シミュレーションモデル（以下、モデルと記す）連携とモデルベース開発（以下、MBD と記す）の今後の活用を考慮した、モデル流通のための『自動車開発におけるプラントモデル I/F ガイドライン[(1)]』や『ガイドライン準拠モデル[(2)]』が公開されて いる。2021 年 9 月には MBD 推進センター(JAMBE)が発足した。前述のガイドラインと準拠モデルは JAMBE に 移管され、展開や拡張の活動が継続されているが、制御ソフトウェア(以下、SW と記す)開発を対象としたモデル流通は議論されていない。

また、各開発フェーズにて OEM やサプライヤにより、検討や検証に用いる制御モデルの抽象度が異なることが考えられ、モデル流通により開発を促進するためには、抽象度の異なる制御モデルを矛盾なく接続するためのガイドラインが必要である。

具体的には図 1-1 に示した通り、制御の機能・ソフトウェア・ハードウェア全体を『制御装置』と定義し、開発の上流工程から自動車の制御装置がどのように連携するのかを想定する。その上で、『それぞれの開発段階で抽象度が異なる制御モデル同士を接続するために、どのようなインターフェース(以下、I/F と記す) を設定すれば連携できるか』を示すガイドライン（以下、本ガイドラインと記す）が求められる。

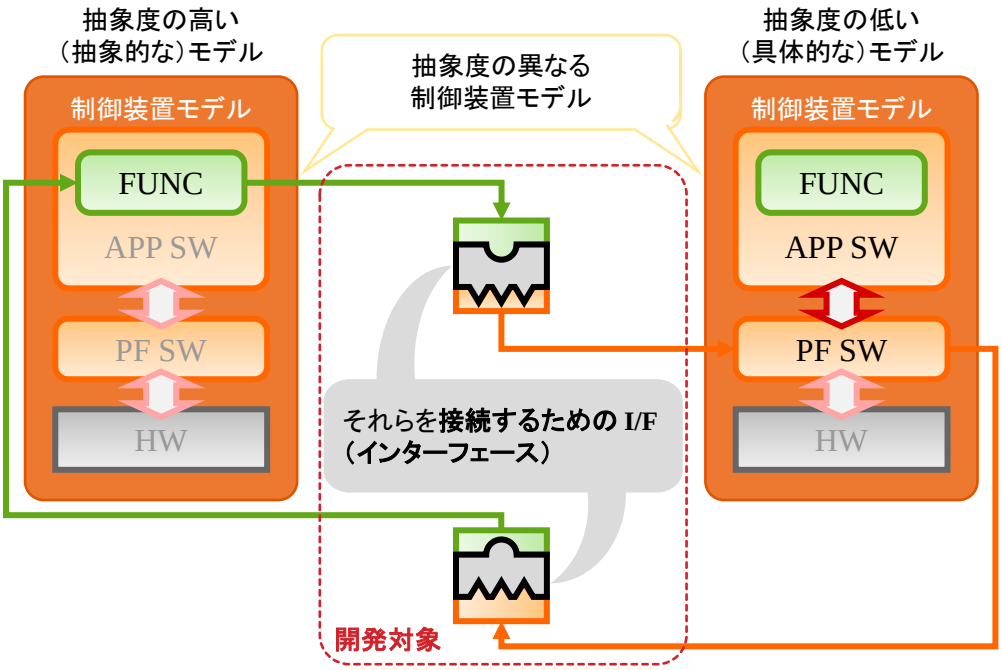


図 1-1. 制御装置の定義と抽象度の異なる制御モデルの接続概要

2. 目的

自動車産業における ECU 制御開発において、企業内・企業間の制御モデル流通を促進することを目的とし、下記 3 点の課題を解消するために本ガイドラインを設定する。

① 対象とする制御モデルの抽象度をガイドラインの種別より選択

② 抽象度の異なる制御モデル接続における注意点の理解

③ 抽象度の異なる制御モデルの接続の効率化

制御モデルの抽象度の異なる接続については、本ガイドラインに記載した事例を基に、様々なユースケースでの議論が進むことを期待する。

2.1. 前提条件

本ガイドラインはガイドライン準拠モデル[(2)]（MILS 環境）と開発におけるユースケースを想定したアプケーション SW と MBD 開発環境（SPILS 環境）の接続に対する有効性の検証を範囲とする。

2.2. 略語および略称の説明

略語/略称	英語名	日本語名
I/F	Interface	インターフェース
MBD	Model Based Development	モデルベース開発
OEM	Original Equipment Manufacturer	自動車メーカ
CoGW	Communication Gate Way	コミュニケーションゲートウェイ
MILS	Model In the Loop Simulation	モデルインザループシミュレーション
SILS	Software In the Loop Simulation	ソフトウェアインザループシミュレーション
PILS	Processor In the Loop Simulation	プロセッサインザループシミュレーション
SPILS	Simulated Processor In the Loop Simulation	仮想プロセッサインザループシミュレーション
SW	Software	ソフトウェア
PF	Platform	プラットフォーム
HW	Hardware	ハードウェア
FUNC	Function	機能
APP	Application	アプリケーション
ECU	Electrical Control Unit	電子制御コンピュータ
CAN	Control Area Network	コントロールエリアネットワーク
LSB	Least Significant Bit	分解能
SOME/IP	Scalable service-Oriented MiddlewarE over IP	サービス志向通信の上位レイヤープロトコル
TCP	Transmission Control Protocol	コネクション型通信のプロトコル
UDP	User Datagram Protocol	コネクションレス型通信のプロトコル

3. ガイドラインの構成・対象

3.1. 本ガイドラインの構成

本ガイドラインは以下及び図 3-1 に示した構成とする。制御開発プロセスの各設計フェーズ間において抽象度の異なる制御モデル同士の接続に関するプロセス範囲の設定、モデル種類の定義、モデル接続方法とその方法を用いた事例の提示を行う。

本ガイドラインの目的にしたがって、以下の構成で MILS⇔SILS⇔SPILS を通じたモデル接続に関して解説を行う。

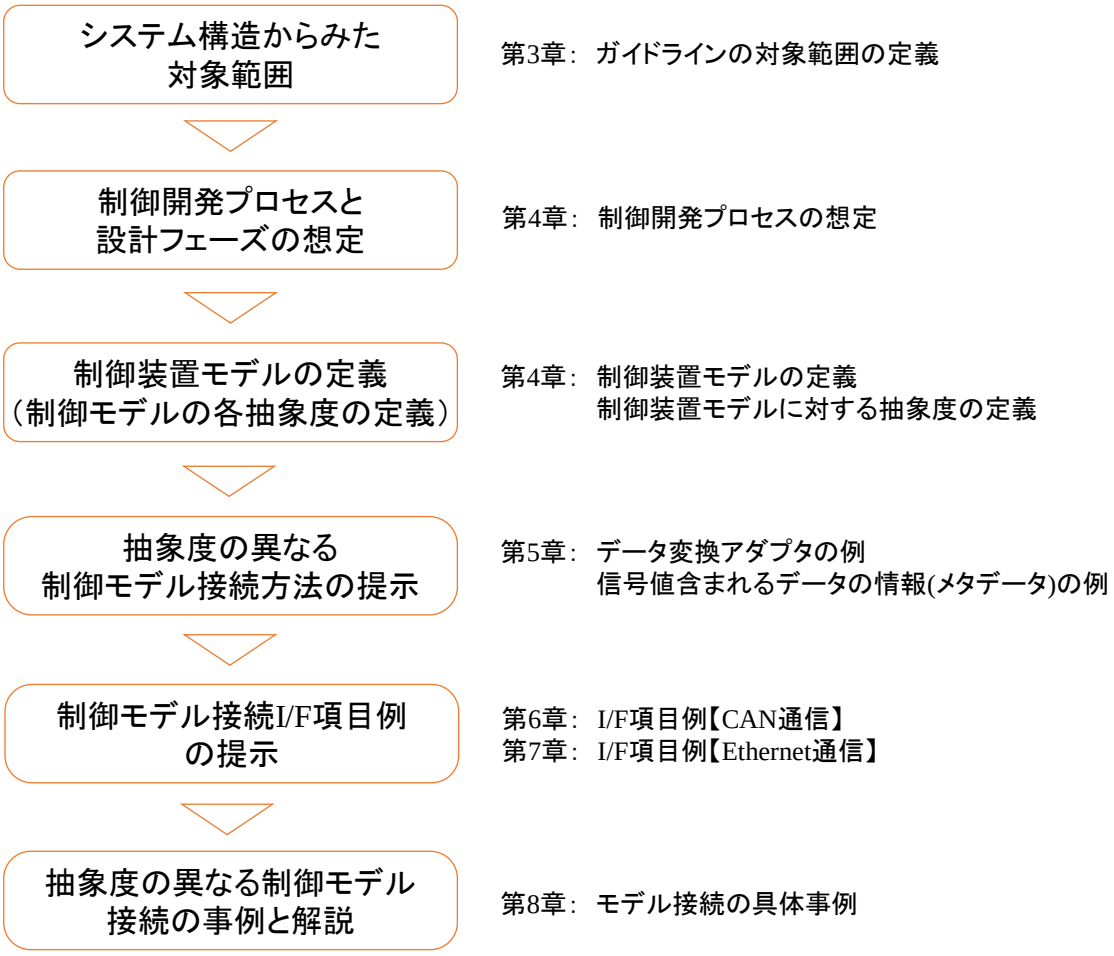


図 3-1. 本ガイドラインの構成

3.2. システムの構造からみた対象範囲

本ガイドラインでは MILS⇔SILS⇔SPILS を通じたモデル接続と図 3-2 に示した対象を範囲とする。制御開発において、開発対象システムのモデルは制御モデルとプラントモデルから成り立つ。本ガイドラインでは、異なる抽象度の制御モデル間ならびにプラントモデルにおける、入出力の接続を対象とする。

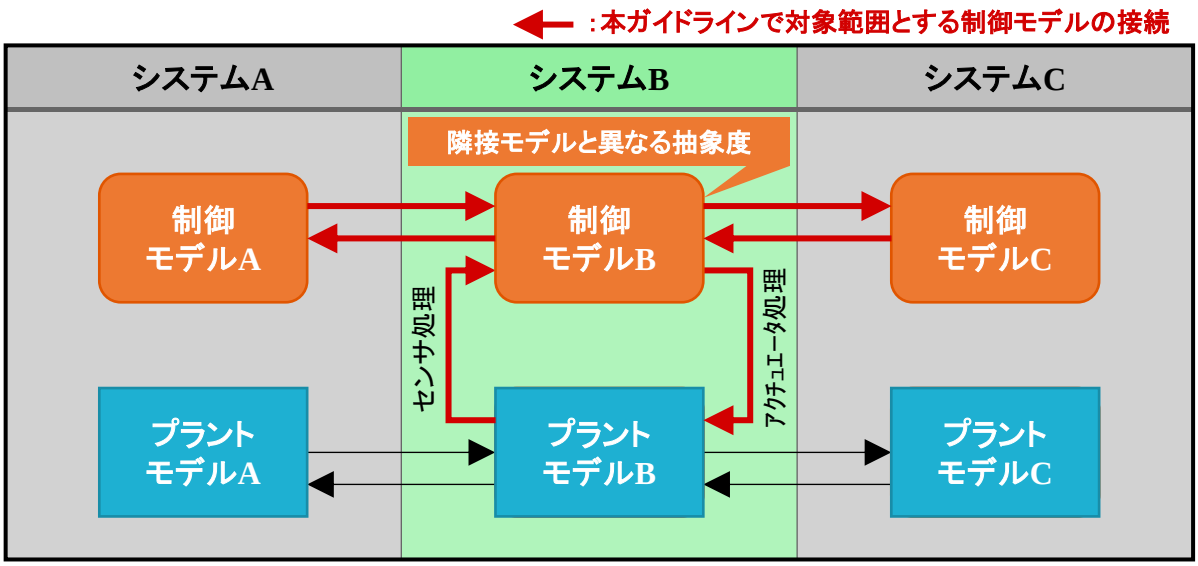


図 3-2. システムの構造からみた本ガイドラインの対象範囲

（補足）自動車システムの例

本ガイドラインでは図 3-3 に示した自動車を構成する最上位の部品をシステムの例とする。

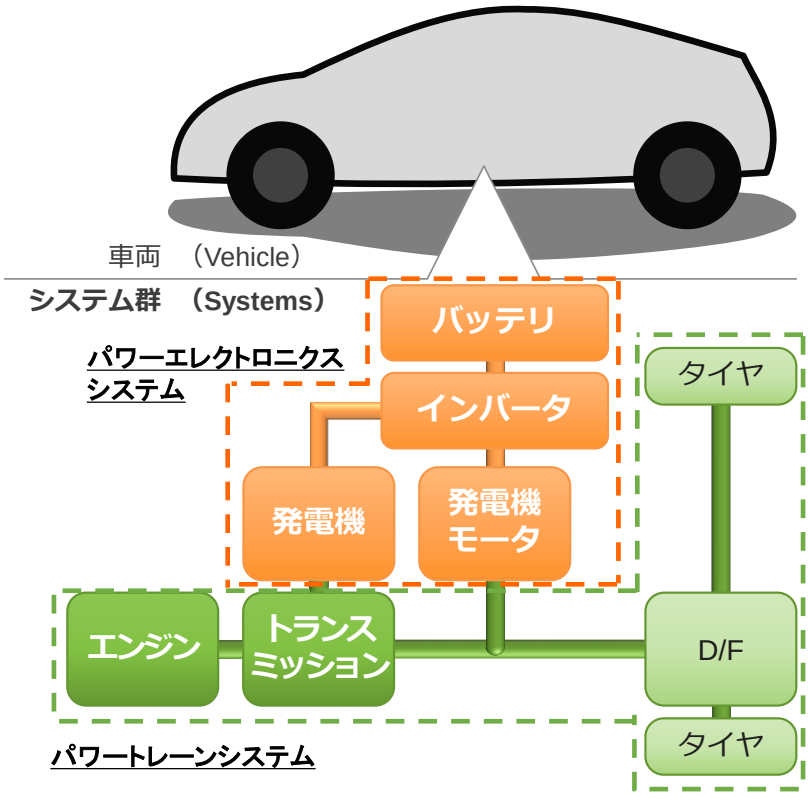
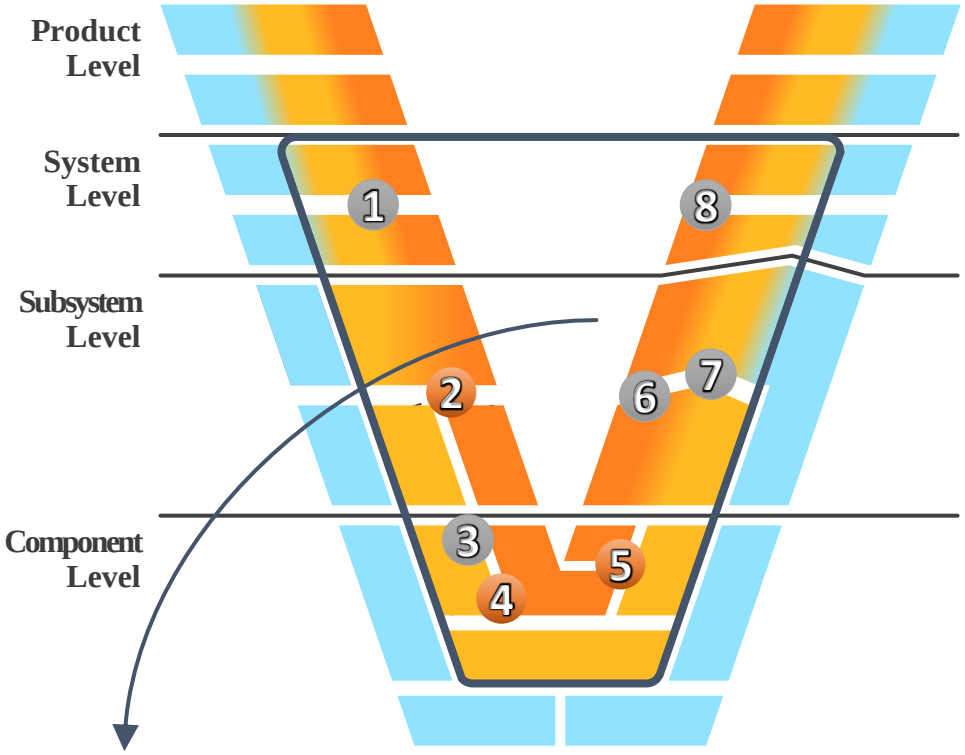


図 3-3. 自動車のシステム構成例

4. ガイドラインで扱うモデルの定義

4.1. 制御ソフトウェア開発プロセスの想定

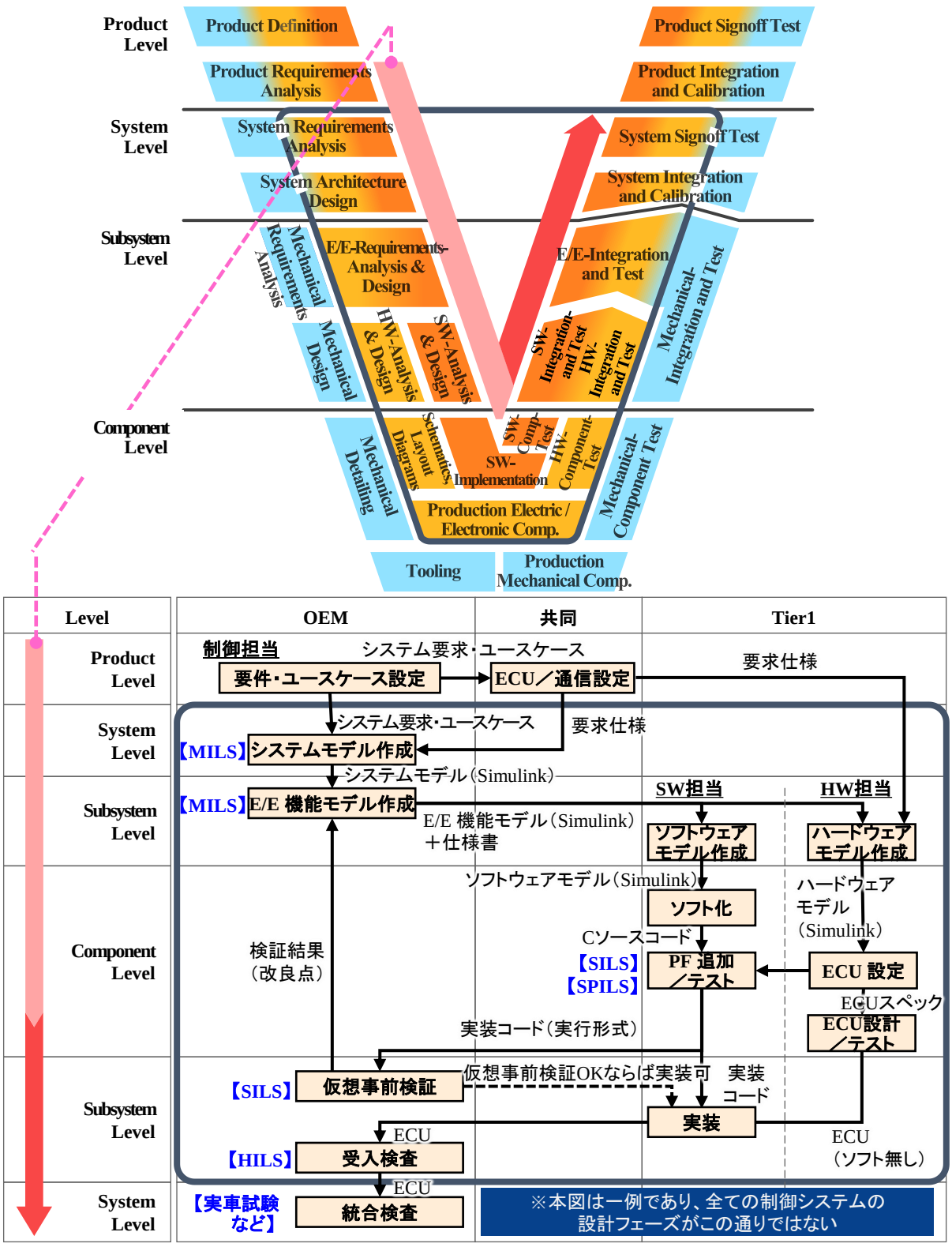
本ガイドラインでは、図 4-1 に示した制御ソフトウェア開発プロセス（①⇒⑧）[3]のうち、**②制御 機能開発、④SW 設計とコード生成、⑤SILS/SPILS 検証**で扱われる制御モデルの接続を扱う。



ソフトウェア開発	モデル抽象度が異なる場合の事例
① System Modeling & Analysis	EVの電力消費量のような車両全体の機能・性能について、複数の設計案を定量的に分析して検討する。
② Function / Controller Design & Simulation	車両全体の機能・性能要件の実現に向け、モータやバッテリなどのシステムの詳細機能とSW・HW等への割当を検討し、仕様を確定する。
③ Functional Evaluation with RCP	エンジンなどの既存システムについて、制御機能の追加・変更の際、制御モデルを実時間駆動させて、実物に対する動作の検討をする。
④ Software-Design & Code Generation	制御ECU上で動作するSWを開発するために、要件定義となる上位モデルからSWを自動生成する。また、HWに依存する部分や詳細化の必要な部分については、追加のコーディングをする。
⑤ SILS / SPILS (PILS)	モータやバッテリなどの各制御ユニット向けのオブジェクトファイルを、実装前に仮想検証して、仕様に合った振る舞いであるか検討する。
⑥ Virtual “non-realtime” ECU Validation	機能レベルの動作が確保されていることをシミュレータ及び実機で確認する。
⑦ Virtual “realtime” ECU Validation	機能に加えてタイミングが確保されていることをシミュレータ及び実機で確認する。
⑧ HILS	試作された複数の制御ユニットについて、例えばハイブリッド制御のような連携制御の検証を、実車両の完成を待たずに実施する。

図 4-1. 制御ソフトウェア開発プロセスの概要

制御ソフトウェア開発における各設計フェーズの例[3]を図 4-2 に示す。制御装置設計の詳細化が進むにつれ、検討に使用される制御モデルの抽象度も変化する。



- （補足）ガイドライン対象者：枠内業務に携わる制御系開発者やソフトウェア開発者

図 4-2. 制御ソフトウェア開発における各設計フェーズの例

4.2. 制御装置モデルの定義

本ガイドラインでは、制御ソフトウェア開発プロセスの各設計フェーズ間で抽象度の異なる制御モデルの接続を考慮するため、図 4-3 に示した通り、『制御モデル』を『制御装置モデル』に再定義する。

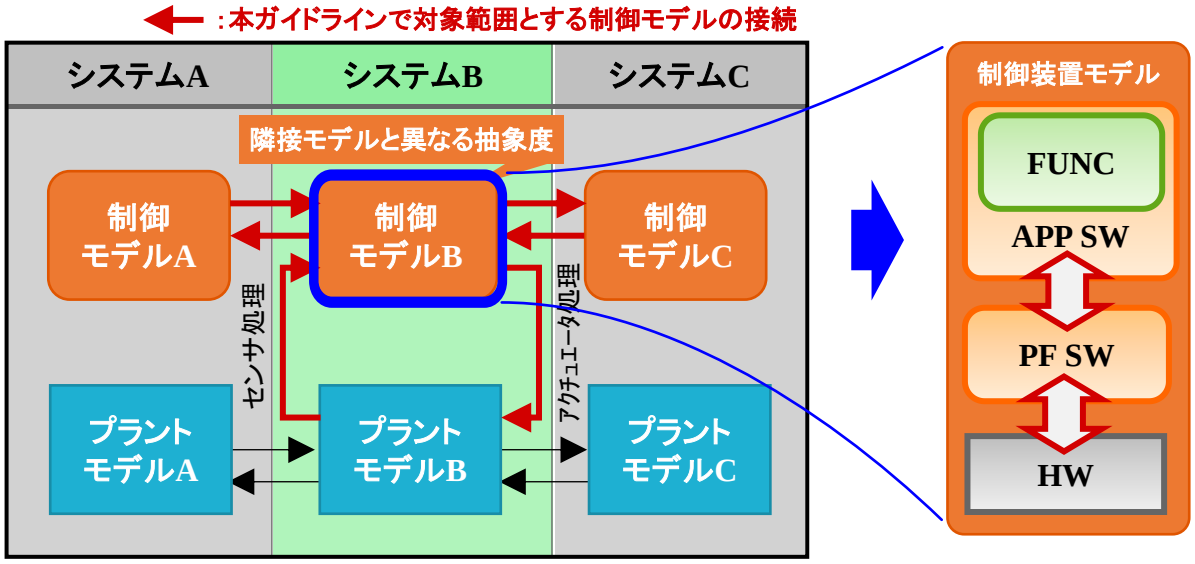


図 4-3.『制御モデル』から『制御装置モデル』への再定義

図 4-4 に示した通り、制御装置モデルでは、その内部を『FUNC』、『APP SW』、『PF SW』『HW』に分割する。機能設計において、最も抽象度の高いモデルは制御装置モデル内の『FUNC』に相当する。

制御装置モデルは実際の制御装置（以下、ECU と記す）の構造を参照してモデル構造を定義し、他 ECU との通信やセンサーアクチュエータ、コネクティッドなどの外部装置と電気的に接続されている。内部ではそれぞれのソフトウェアの処理によって制御機能を実現している。

ここで、制御装置の構成を電気的に処理する HW 部（以下、HW と記す）、ソフトウェア的に処理するアプリケーション層（以下、APP SW と記す）、及びプラットフォーム層（以下、PF SW と記す）として定義する。

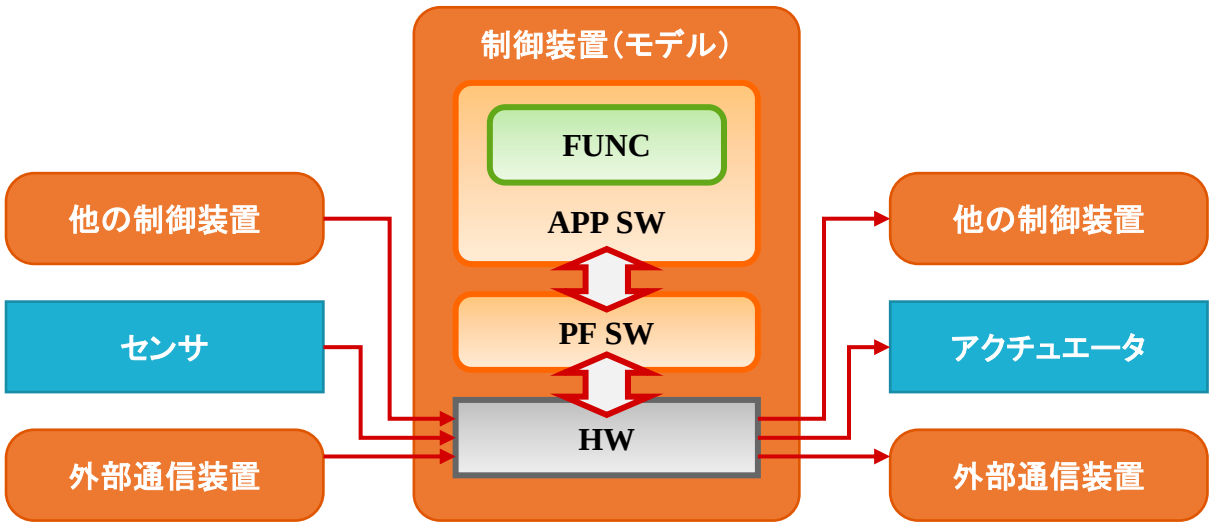


図 4-4. 制御装置モデルの定義

4.3. 制御装置モデルに対する抽象度の定義

制御装置モデルに対して、用途に応じた異なる抽象度を定義する。制御装置モデルを下記 a～d の 4 つの抽象度として定義する。図 4-5 に各モデル抽象度の構成を示す。

a. 制御機能モデル

b. 制御 SW モデル

c. 制御 PF モデル

d. 制御 ECU モデル

本ガイドラインで想定する開発プロセス範囲では、a. 制御機能モデル ～ c. 制御 PF モデルを対象とする。

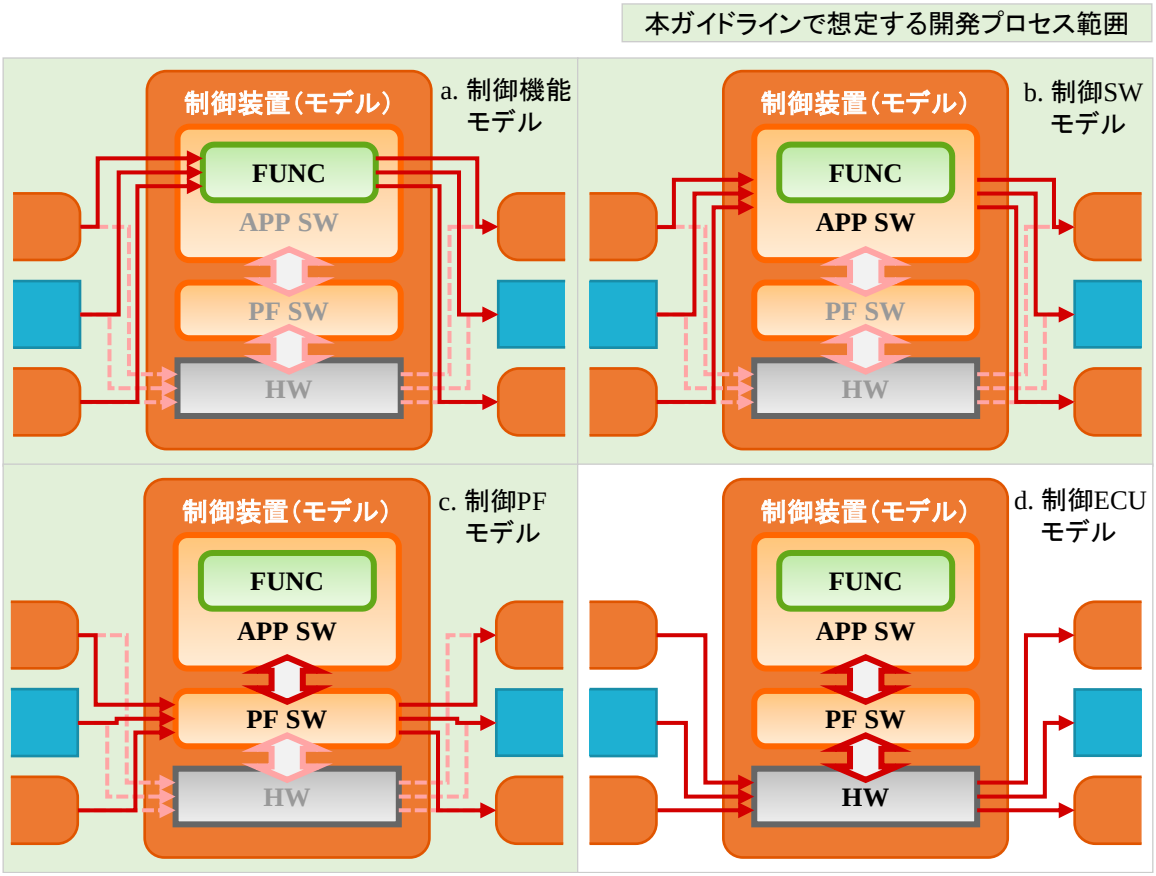


図 4-5. 各モデル抽象度の構成

4.4. 対象ターゲットの設定：対象レイヤーとそれぞれの制御設計フェーズでの検証

ソフトウェア開発の設計フェーズを、下記①～④及び、図 4-6 に示した通り、4 つのレイヤーに 分けて設定する。制御(装置)の開発プロセスは①機能設計を基として、②ソフトウェア及び③プラットフォームの設計と④ハードウェア(ECU)の設計に分岐する。各設計の検証に MBD を用いているが従来開発プロセスでは開発フェーズによって異なる抽象度のモデルを使用しており、モデル流用が行いにくい。

① 機能設計：MILS を用いた検証

② ソフトウェア設計：SILS を用いた検証

③ プラットフォーム設計：SILS や SPILS を用いた検証

④ ECU 設計：SPILS を用いた検証

本ガイドラインでは、①機能設計から③プラットフォーム設計までを対象とする。

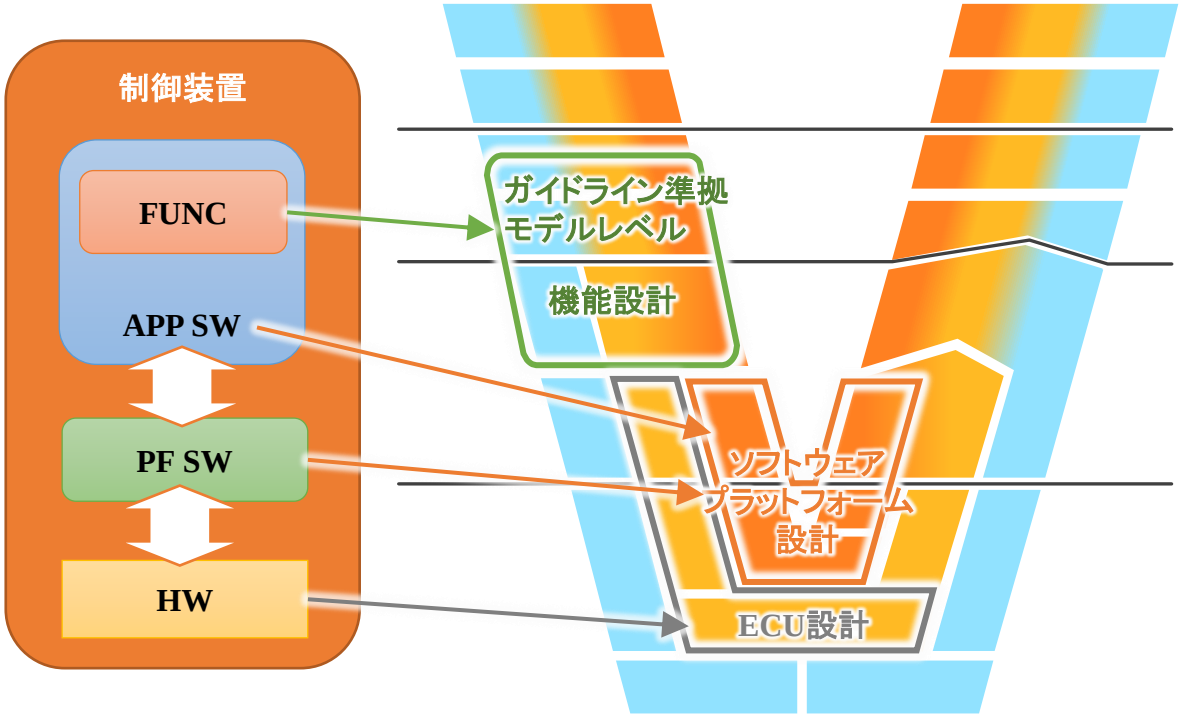


図 4-6. 制御開発プロセスの対象レイヤー

5. 異なる抽象度のモデル接続に関する課題

5.1. 制御装置モデルの I/F 項目について

各制御開発プロセス間で扱われる異なる抽象度のモデルを接続する場合、信号値の受け渡しには変換や付加が必要である。各抽象度のモデルが扱う信号のメタデータの設定が必要である。モデルの I/F は、モデルの入出力信号である。

図 5-1 に示した通り、その信号値が何を意味するか定義された情報であるメタデータがなければ信号値の理解できず、異なる抽象度のモデルを接続する際に信号値の変換が不可能となる。モデルの I/F を理解し、モデルを接続させるためには、それぞれのモデルの抽象度に合致したメタデータ項目を設定する必要がある。

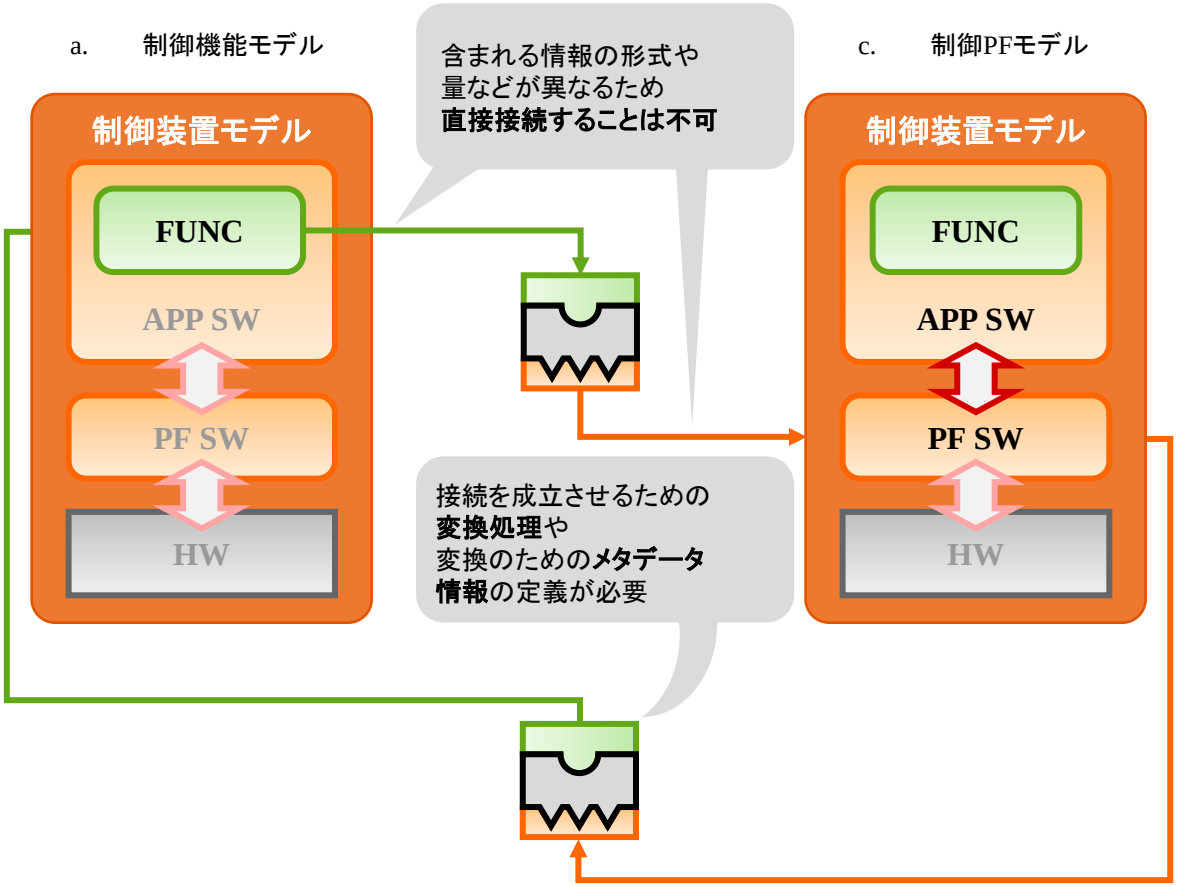


図 5-1. メタデータ項目設定の必要性(制御機能モデルと制御 PF モデル間の接続例)

5.2. 異なる抽象度の制御モデル接続のための変換アダプタについて(1)

異なる抽象度の制御モデルを接続する場合の具体的な方法としては、図 5-2 に示したデータ変換アダプタの考慮と制御モデルの抽象度に応じた選択が必要となる。

制御モデル接続のための入出力信号の変換について、本ガイドラインでは、抽象度が異なる制御モデル同士の間に挿入して作用する、データ変換アダプタを提案する。抽象度の違いは図 5-2 に示した制御モデル同士の差分で表現できる。差分解消のための入出力変換は、事前に接続する各制御モデルの抽象度を比較し、接続が成立する変換アダプタを選択する。図 5-2 では、例としてプラットフォーム設計の際に考慮される通信仕様に合わせ、赤点線の順に変換アダプタを選択した。

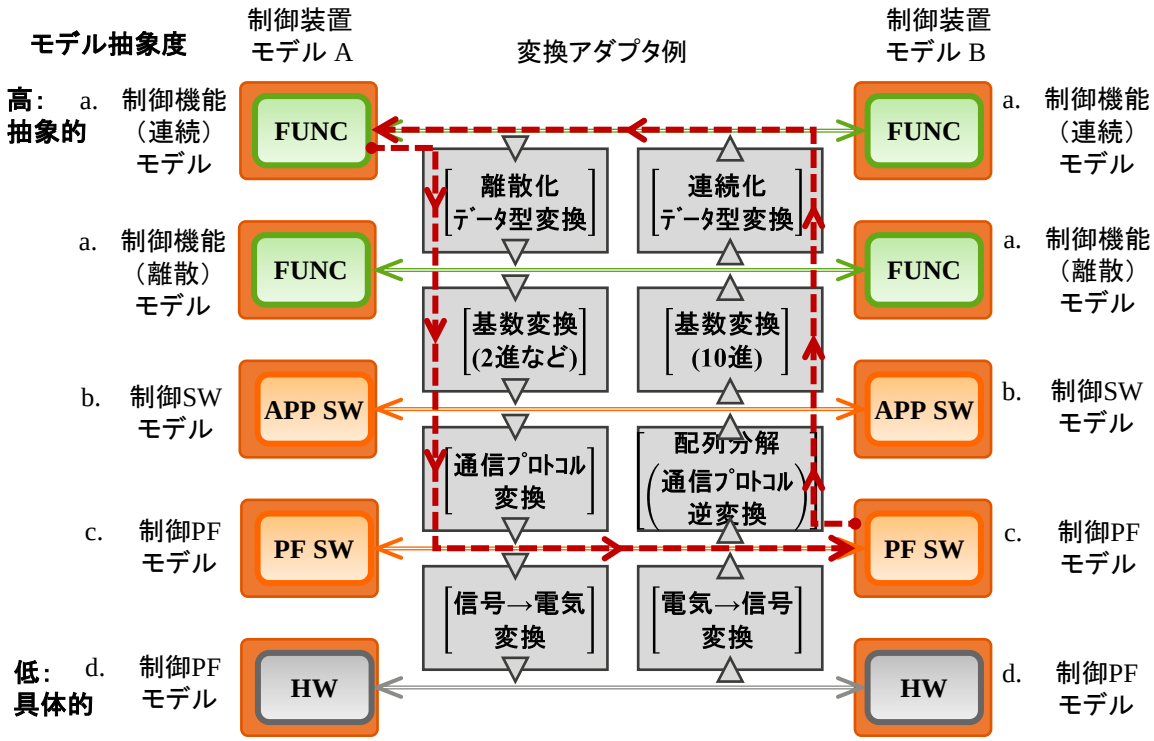


図 5-2. 異なる抽象度の制御モデル接続のための変換アダプタの例
[制御機能(連続)モデルと制御 PF モデル間の接続例]

5.3. モデル接続のための変換アダプタについて(2)

異なる抽象度のモデル間に設置する変換ブロックを構築し、データ変換アダプタを実現する。

入出力変換用のブロックを接続することで抽象度の差分を解決する方法を、実現手段の例として図 5-3 に示す。図 5-3 で示した通り、前節で選択したデータ変換、基数変換、通信プロトコル変換への対応のための変換アダプタをブロック形式で構築して挿入した。各変換アダプタの入出力と内部機能を検討する際にはメタデータ項目を参照する。

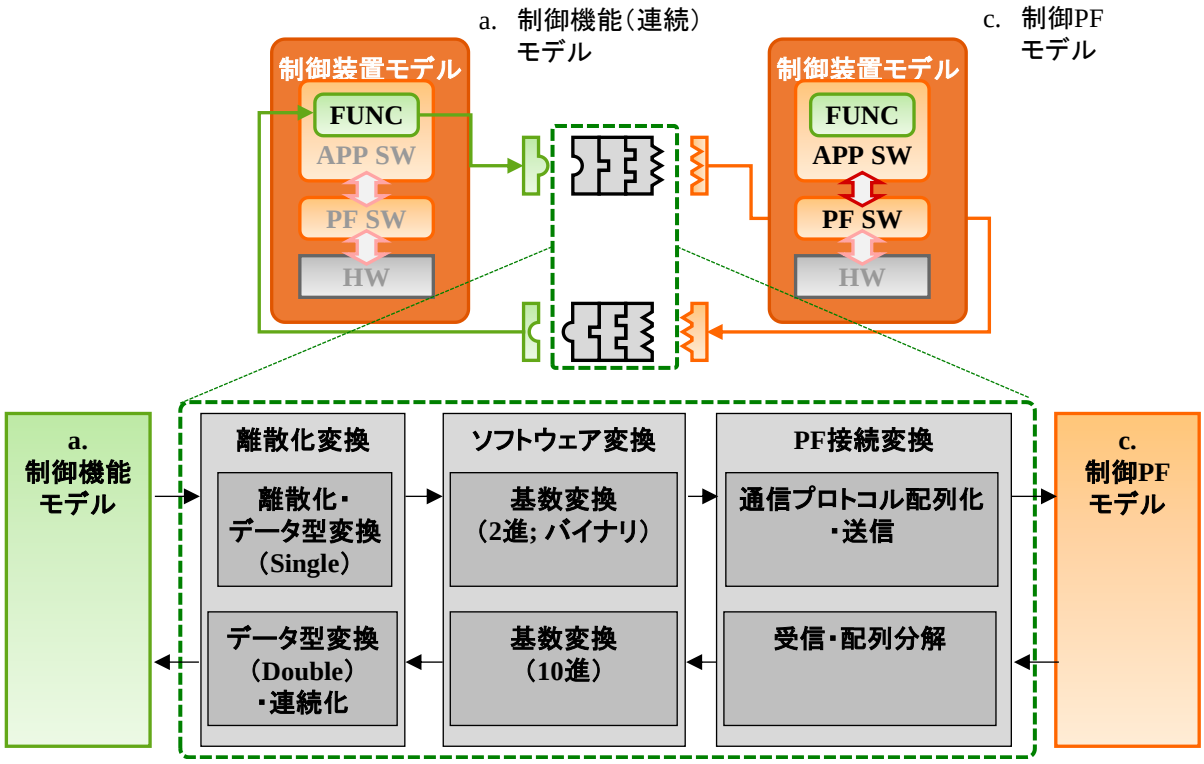


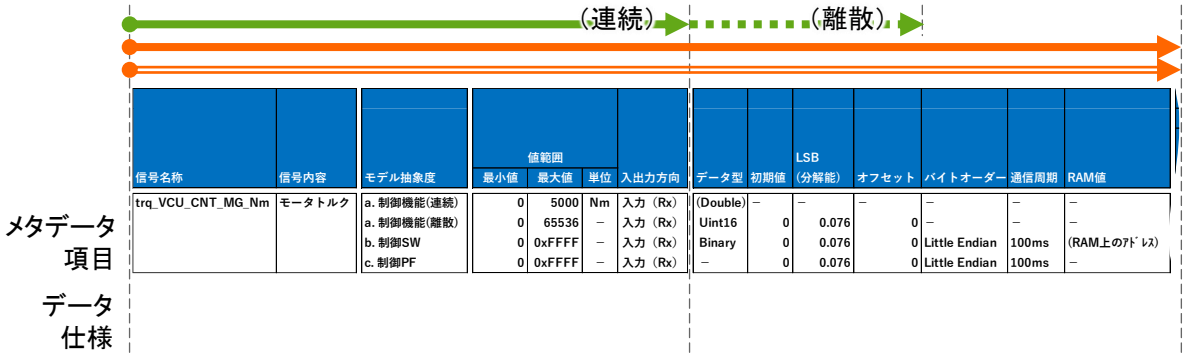
図 5-3. 変換ブロックで構築したデータ変換アダプタの例

5.4. 信号値に含まれるデータの情報(＝メタデータ)について

データ変換アダプタを準備する際には、各信号に含まれるデータの内容を参照して意図した出力が接続されるように I/F を設定する。

制御開発プロセスの異なる設計フェーズ間でモデルを接続する際は、図 5-4 に示した通り、入出力情報の内容をあらかじめ設定しておくことでモデル間の接続が容易となり、データ変換アダプタは設定されたデータ定義に基づいて事前に準備する。

入出力情報(メタデータ)例



信号名称	信号内容	モデル抽象度	値範囲 最小値	値範囲 最大値	値範囲 単位	入出力方向	データ型	初期値	LSB (分解能)	オフセット	バイトオーダー	通信周期	RAM値
trq_VCU_CNT_MG_Nm	モータトルク	a. 制御機能(連続)	0	5000	Nm	入力（Rx）	(Double)	–	–	–	–	–	–
		a. 制御機能(離散)	0	65536	–	入力（Rx）	Uint16	0	0.076	0	–	–	–
		b. 制御SW	0	0xFFFF	–	入力（Rx）	Binary	0	0.076	0	Little Endian	100ms	(RAM上のｱﾄﾞﾚｽ)
		c. 制御PF	0	0xFFFF	–	入力（Rx）	–	0	0.076	0	Little Endian	100ms	–

入出力情報に基づいたデータ変換アダプタの例

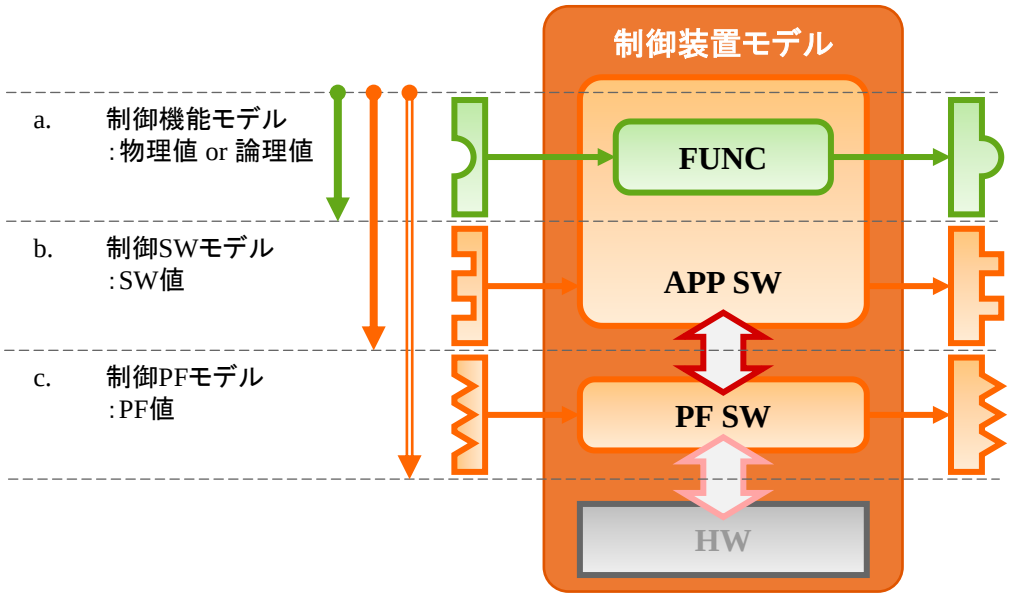


図 5-4. 入出力情報(メタデータ)とデータ変換アダプタの例

5.5. メタデータの例

各信号に含まれるメタデータの例を表 5-1. に記す。

表 5-1. 各信号に含まれるメタデータの例

信号値	メタデータ	備考
物理値・論理値	信号名称	-
	値範囲	-
	単位	-
	入出力方向	-
	説明	-
SW 値	信号名称	-
	値範囲	-
	LSB(分解能)	車載ソフトウェア用固定小数点数
	オフセット	車載ソフトウェア用固定小数点数
	データ型	-
	初期値	-
	ポート	-
	バイトオーダー	2Byte 以上のデータで使用
	周期	-
	RAM 名	-
PF 値	信号名称	-
	値範囲	-
	CAN ID	CAN:初期化処理、送受信処理
	CAN ボーレート	CAN:通信速度設定処理
	CAN DLC	CAN:送受信処理
	CAN データ位置	CAN:送受信処理

6. 各抽象度における制御モデルの I/F 項目例【CAN 通信】

6.1. 制御機能モデル

制御機能モデルの構成を図 6-1. 制御機能モデルの構成に示し、I/F の信号値とメタデータの例を表 6-1 及び下記に記す。

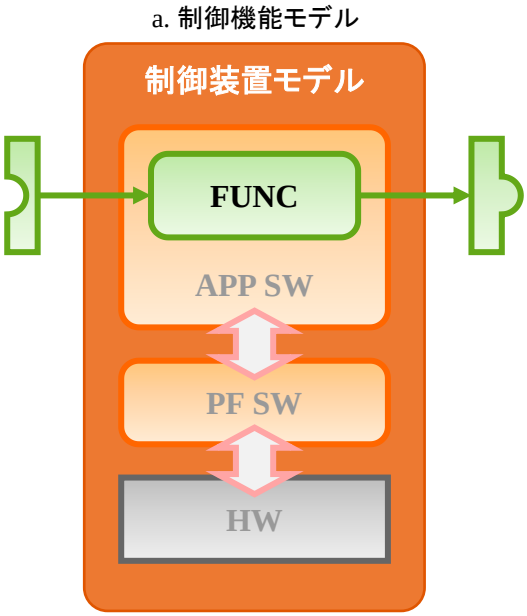


図 6-1. 制御機能モデルの構成

信号値

- 物理値または理論値
  （→ 対象データの意味を理解できる形式の数値）

メタデータ

- 信号名称
- 信号内容
- 値範囲（最小値｜最大値｜単位）
- 入出力方向
- ＝＝＝
- データ型
- 初期値
- LSB（分解能）
- オフセット

表 6-1. 制御機能モデルにおける I/F の信号値とメタデータの例

信号名称	信号内容	モデル抽象度	値範囲			入出力方向	データ型	初期値	LSB（分解能）	オフセット
			最小値	最大値	単位					
trq_VCU_CNT_MG_Nm	モータトルク	a. 制御機能(連続)	0	5000	Nm	入力（Rx）	(Double)	–	–	–
		a. 制御機能(離散)	0	65536	–	入力（Rx）	Uint16	0	0.076	0

6.2. 制御 SW モデル

制御 SW モデルの構成を図 6-2. 制御 SW モデルの構成に示し、I/F の信号値とメタデータの例を表 6-2 及び下記に記す。

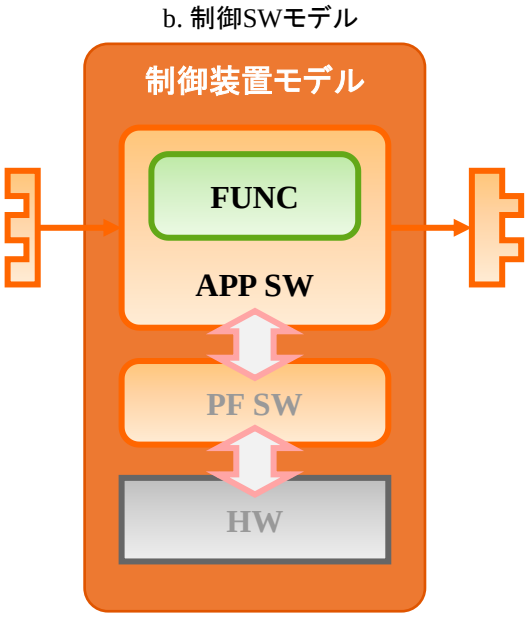


図 6-2. 制御 SW モデルの構成

信号値

・ SW 値

（→ 16 進数 または 2 進数に変換された数値）

メタデータ

・ メタデータ（a.制御機能モデル）

＋

・ 通信周期
・ データ型
・ 初期値
・ LSB（分解能）
・ オフセット
・ バイトオーダー

表 6-2. 制御 SW モデルにおける I/F の信号値とメタデータの例

信号名称	信号内容	モデル抽象度	値範囲			入出力方向
			最小値	最大値	単位	
trq_VCU_CNT_MG_Nm	モータトルク	b. 制御SW	0	0xFFFF	－	入力（Rx）

＋

データ型	初期値	LSB（分解能）	オフセット	バイトオーダー	通信周期	RAM値
Binary	0	0.076	0	Little Endian	100ms	（RAM上のｱﾄﾞﾚｽ）

6.3. 制御 PF モデル

制御 PF モデルの構成を図 6-3. 制御 PF モデルの構成に示し、I/F の信号値とメタデータの例を表 6-3 及び下記に記す。

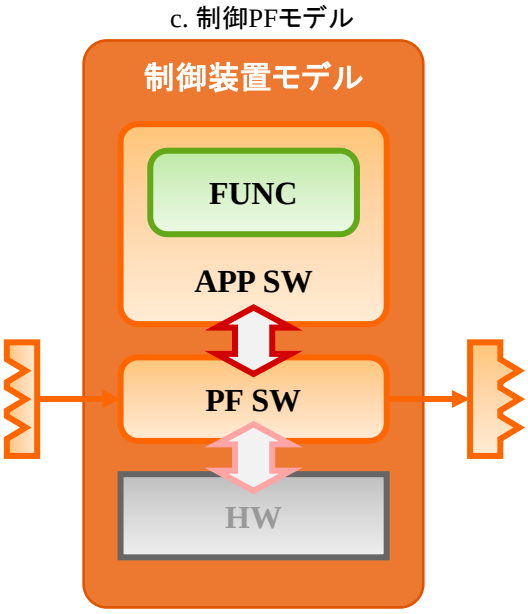


図 6-3. 制御 PF モデルの構成

信号値

PF 値

（→ 対象マイコンでの通信形式に合わせた数値）

メタデータ

・ メタデータ(a.制御機能モデル ＋ b.制御 SW モデル)

＋

・ CAN 通信規格に合わせたメタデータ
・ フレーム名
・ ID
・ DLC
・ データ位置
・ ビット長
・ 送信元
・ 送信先
・ ボードレート

表 6-3. 制御 PF モデルにおける I/F の信号値とメタデータの例

信号名称	信号内容	モデル抽象度	値範囲			入出力方向
			最小値	最大値	単位	
trq_VCU_CNT_MG_Nm	モータトルク	c. 制御PF	0	0xFFFF	–	入力（Rx）

＋

データ型	初期値	LSB（分解能）	オフセット	バイトオーダー	通信周期	RAM値
–	0	0.076	0	Little Endian	100ms	–

＋

CAN 通信							
フレーム名	ID (hex)	DLC (Byte)	データ位置 (Bit)	ビット長 (Bit)	送信元	送信先	ボーレート
MG2Torque	0x60	8	8	16	MG2	HV	500kbps

7. 各抽象度における制御モデルの I/F 項目例【Ethernet 通信】

7.1. 制御機能モデル

制御機能モデルの構成を図 7-1. 制御機能モデルの構成に示し、I/F の信号値とメタデータの例を表 7-1 及び下記に記す。

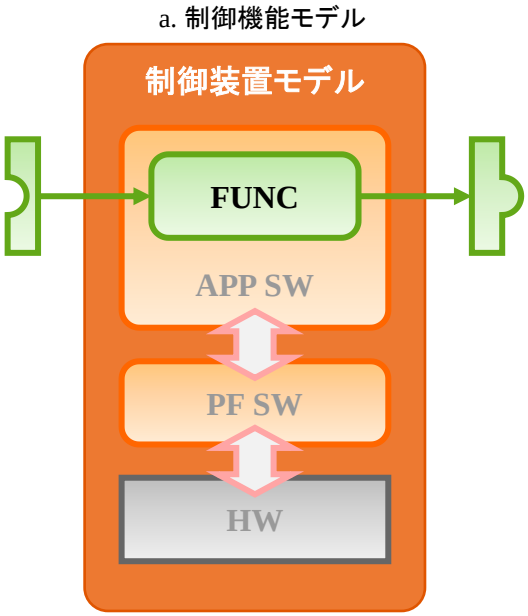


図 7-1. 制御機能モデルの構成

信号値
- 物理値または理論値

（→ 対象データの意味を理解できる形式の数値）

メタデータ
- 信号名称
- 信号内容
- 値範囲（最小値｜最大値｜単位）
- 入出力方向

＝＝＝
- データ型
- 初期値
- LSB（分解能）
- オフセット

表 7-1. 制御機能モデルにおける I/F の信号値とメタデータの例

信号名称	信号内容	モデル抽象度	値範囲			入出力方向	データ型	初期値	LSB（分解能）	オフセット
			最小値	最大値	単位					
trq_VCU_CNT_MG_Nm	モータトルク	a. 制御機能(連続)	0	5000	Nm	入力（Rx）	(Double)	–	–	–
		a. 制御機能(離散)	0	65536	–	入力（Rx）	Uint16	0	0.076	0

7.2. 制御 SW モデル

制御 SW モデルの構成を図 7-2. 制御 SW モデルの構成に示し、I/F の信号値とメタデータの例を 表 7-2 及び下記に記す。

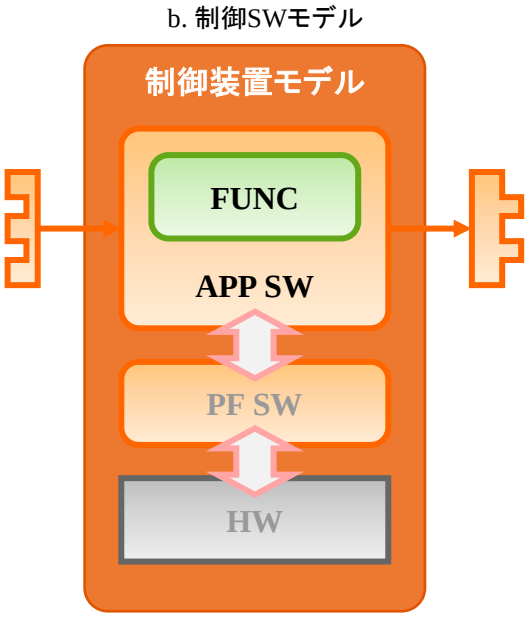


図 7-2. 制御 SW モデルの構成

信号値

- SW 値
- （→ 16 進数 または 2 進数に変換された数値）

メタデータ

- メタデータ（a.制御機能モデル）

＋

- 通信周期
- データ型
- 初期値
- LSB（分解能）
- オフセット
- バイトオーダー
- RAM 値

表 7-2. 制御 SW モデルにおける I/F の信号値とメタデータの例

信号名称	信号内容	モデル抽象度	値範囲			入出力方向
			最小値	最大値	単位	
trq_VCU_CNT_MG_Nm	モータトルク	b. 制御SW	0	0xFFFF	－	入力（Rx）

＋

データ型	初期値	LSB（分解能）	オフセット	バイトオーダー	通信周期	RAM値
Binary	0	0.076	0	Little Endian	100ms	(RAM上のｱﾄﾞﾚｽ)

7.3. 制御 PF モデル

制御 PF モデルの構成を図 7-3. 制御 PF モデルの構成に示す。

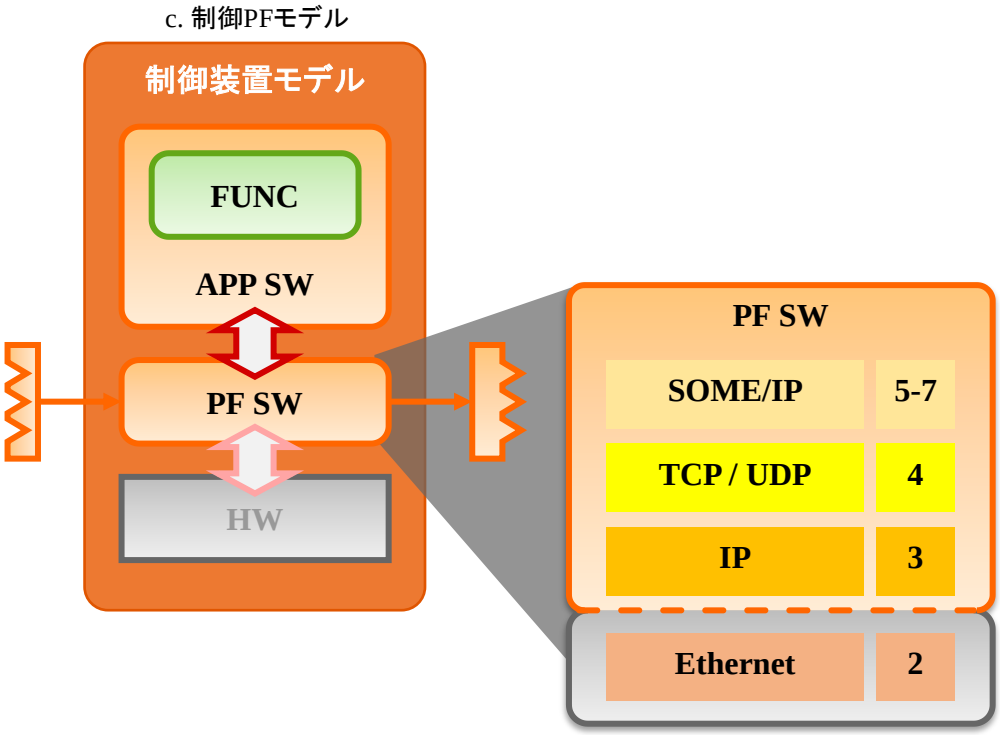


図 7-3. 制御 PF モデルの構成

Ethernet 通信では、通信プロトコルが OSI 参照モデルの各層で定義される。本ガイドライン では、アプリケーション層からデータリンク層までを PF 値とした。OSI 参照モデルによる通信 プロトコルの定義の例を図 7-4 に示す。

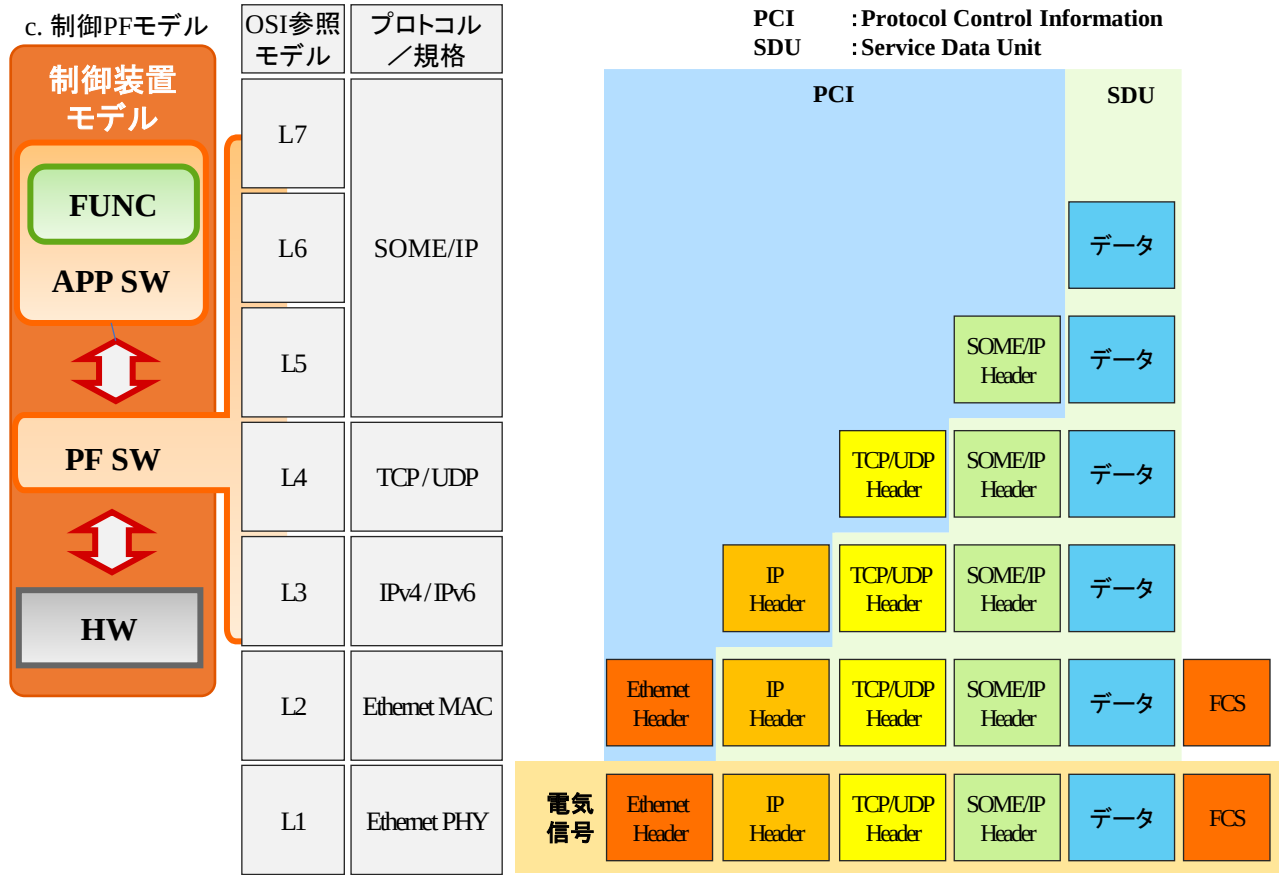


図 7-4. OSI 参照モデルによる通信プロトコルの定義の例

信号値

- PW 値
  （→ 対象マイコンでの通信形式に合わせた数値）

メタデータ

- メタデータ（a.制御機能モデル＋b.制御 SW モデル）

＋

- 車載 Ethernet 通信規格に合わせたメタデータ
- SOME/IP （L7：アプリケーション層 ～ L5：セッション層）
- TCP / UDP （L4：トランスポート層）
- IP （L3：ネットワーク層）
- イーサネット（L2：データリンク層）

I/F の信号値とメタデータの例を表 7-3 及び下記に記す。

表 7-3. 制御 PF モデルにおける I/F の信号値とメタデータの例

信号名称	信号内容	モデル抽象度	値範囲			入出力方向
			最小値	最大値	単位	
trq_VCU_CNT_MG_Nm	モータトルク	c. 制御PF	0	0xFFFF	–	入力（Rx）

＋

データ型	初期値	LSB（分解能）	オフセット	バイトオーダー	通信周期	RAM値
–	0	0.076	0	Little Endian	100ms	–

＋

Ethernet 通信			
Ethernet	IP	選択	
		TCP	UDP	SOME/IP

制御 PF モデルの各 PF SW のメタデータ項目例を以下に示す。図 7-5 と表 7-4 に、PF SW の SOME/IP 部について示す。

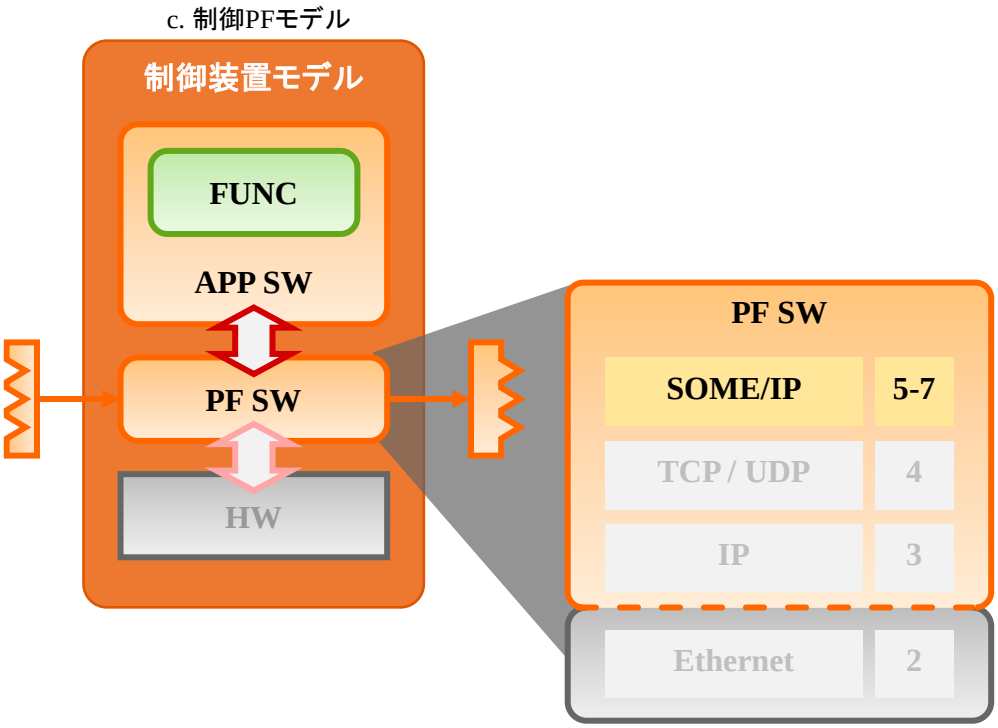


図 7-5. PF SW の SOME/IP 部

表 7-4. SOME/IP のメタデータ項目例

メタデータ（SOME/IP）	説明
Message ID	イベントを識別するための識別子
Length	メッセージの長さ
Request ID	クライアント ID とセッション ID からなる一意の識別子
Protocol Version	使用される Some/IP ヘッダー形式の識別
Interface Version	Some/IP サービスの 現在のバージョン説明
Message Type	メッセージのタイプ
Return Code	要求が正常に処理されたかを通知

図 7-6 と表 7-5 に、PF SW の TCP 部について示す。

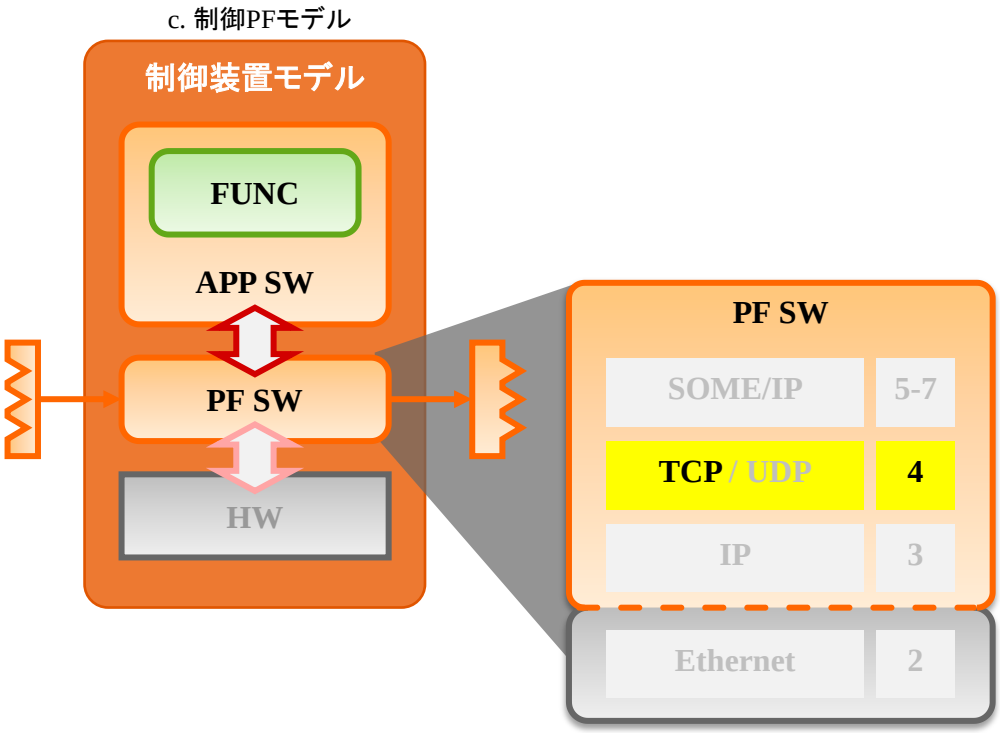


図 7-6. PF SW の TCP 部

表 7-5. TCP のメタデータ項目例

メタデータ（TCP）	説明
送信元ポート番号	送信元のポート番号の値
宛先ポート番号	宛先のポート番号の値
シーケンス番号	パケットに付けられる通し番号 受信した相手の確認応答番号を使用
確認応答番号	確認応答番号 受信した相手のシーケンス番号＋データサイズ
データオフセット	TCP ヘッダの長さ
予約	全ビット「0」、将来の拡張用
コントロールフラグ	URG, ACK, PSH, RST, SYN, FIN の 6bit
ウィンドウサイズ	受信側が一度に受信可能なデータ量
チェックサム	TCP ヘッダとデータ部のエラーチェック用
緊急ポインタ	URG が「1」の場合に使用されるフィールド
オプション	TCP 通信の性能向上（拡張）用
パディング	「0」で TCP ヘッダ長を 32bit 整数に調整

図 7-7 と表 7-6 に、PF SW の UDP 部について示す。

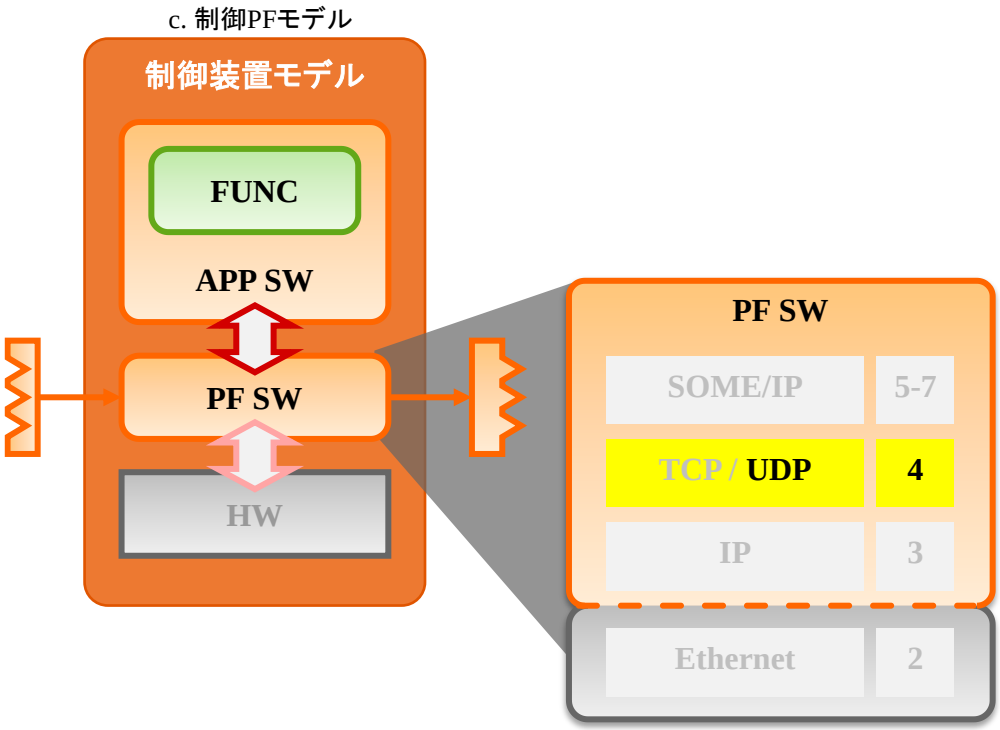


図 7-7. PF SW の UDP 部

表 7-6. UDP のメタデータ項目例

メタデータ（UDP）	説明
送信元ポート番号	送信元のポート番号の値
宛先ポート番号	宛先のポート番号の値
パケット長	UDP ヘッダと UDP データの 長さを合計したサイズの値
チェックサム	UDP ヘッダとデータ部のエラーチェック用

図 7-8 と表 7-7 に、PF SW の IP について示す。

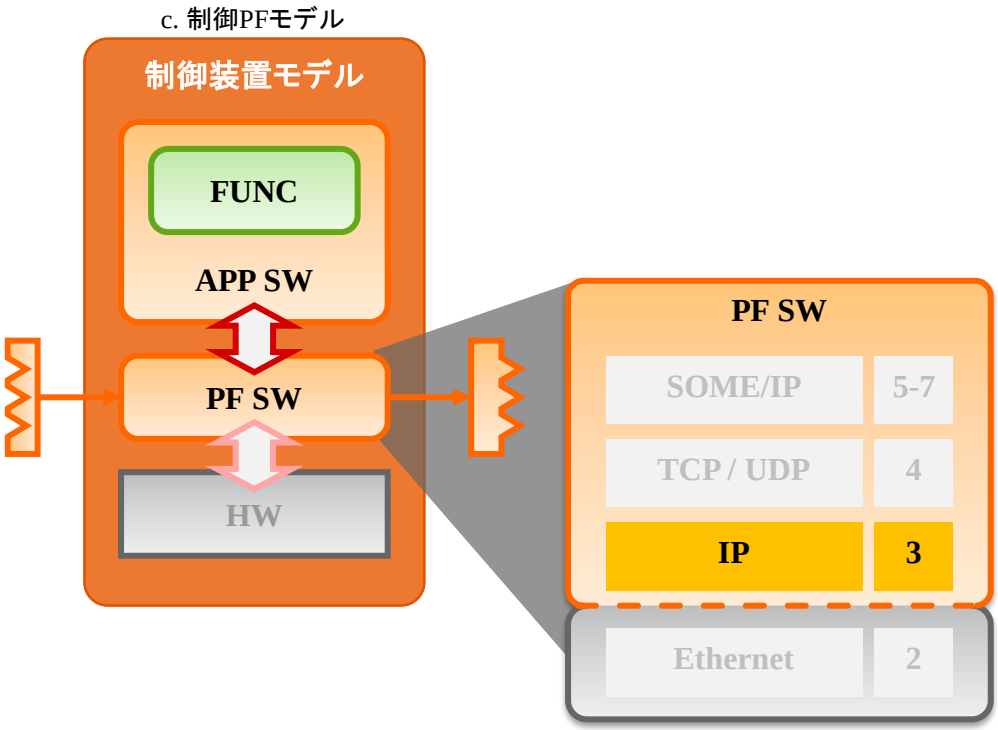


図 7-8. PF SW の IP 部

表 7-7. IP のメタデータ項目例

メタデータ（IP）	説明
バージョン	IP バージョン
ヘッダ長	IP ヘッダの長さ
サービスタイプ	パケットの優先順位を指定する情報
パケット長	IP ヘッダを含むパケット全体のサイズ
識別番号	分割パケットを分割前パケットに 戻すための識別番号
フラグ	パケット分割（フラグメント化）に関する制御情報
フラグメントオフセット	分割パケットの分割前パケットでの位置情報
生存時間	ネットワークをルーティングできる数
プロトコル	上位のトランスポート層で使用しているプロトコル
ヘッダチェックサム	IP ヘッダの誤り検出のためのチェックサム情報
送信元 IP アドレス	送信元の IP アドレスの情報
宛先 IP アドレス	宛先の IP アドレスの情報
オプション	IP パケット通信の拡張情報
パディング	IP ヘッダ長を 32bit 単位になるよう0埋め調整

図 7-9 と表 7-8 に、PF SW の Ethernet 部について示す。

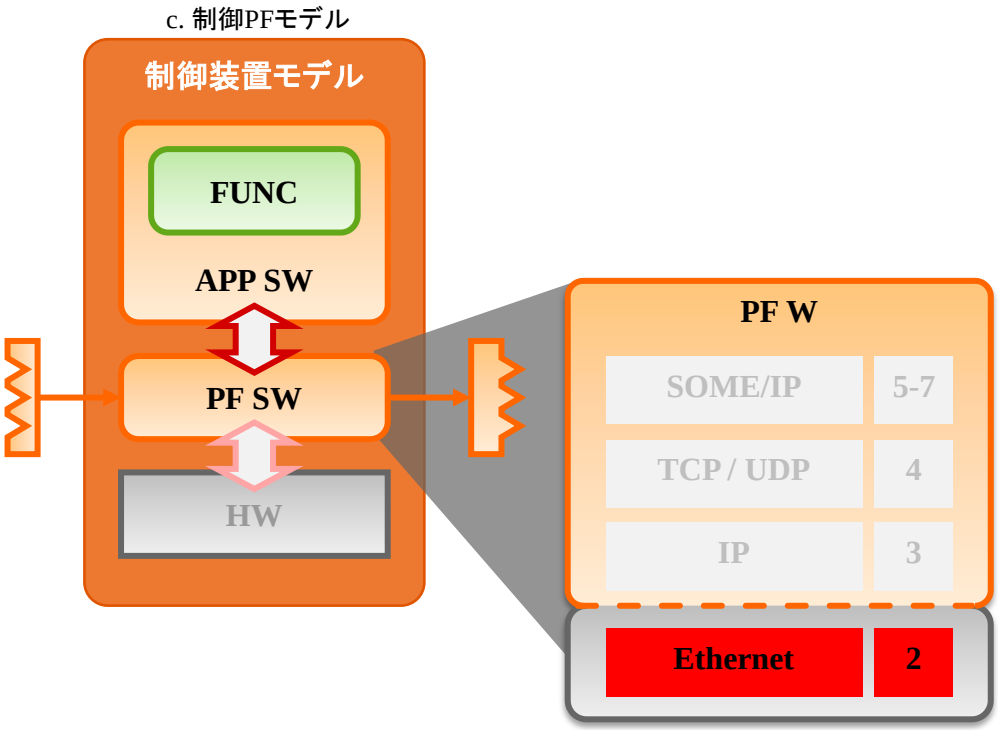


図 7-9. PF SW の Ethernet 部

表 7-8. Ethernet のメタデータ項目例

メタデータ(Ethernet)	説明
プリアンブル	受信側と送信側で同期を行う際に使用される 先頭 7(byte):10101010 末尾 1(byte):10101011
宛先 MAC アドレス	宛先の MAC アドレスの値
送信元 MAC アドレス	送信元の MAC アドレスの値
長さ	データの長さの値
タイプ	上位層のプロトコル種類を示す値 例)IPv4:0x0800 など
FCS	フレーム全体の CRC チェック情報 エラー検出のため、送信、受信双方で同じチェックを実施

8. モデル接続の具体事例

8.1. 想定するユースケース

本ガイドラインでは、表 8-1、表 8-2 に示すユースケースを想定する。

表 8-1. ユースケース 1

ケース名	Tier1 サプライヤの社内検証
主担当者	Tier1 サプライヤ SW 開発担当者
内容	Tier1 サプライヤは OEM からの要求仕様書をもとにモータユニットを開発していた。主担当者は MCU の SW と別の担当者が設計した HW を統合した際のスペックの妥当性検証について OEM から提供いただいたプラントモデルを用いた SPILS 検証により判断することとした。 ECU の I/O は CAN 形式で行うこととし、使用する命令セットシミュレータも Simulink との連成シミュレーションが可能であるものを用いる。 ただし、主担当者は上記連成シミュレーションならびに CAN 通信データを Simulink モデルで取り扱うためのデータ変換が今回初めてのトライである。

表 8-2. ユースケース 2

ケース名	OEM の開発制御モデルと既存 ECU_SW の I/O の統合
主担当者	OEM 制御開発者
内容	主担当者は車両全体 ECU の SW を開発しており、Simulink の制御モデルを開発車両全体モデルによる MILS にて検証し見通しを出すフェーズまで進めた。 次のフェーズとして他 ECU と連成させたときのエラー有無をより実機に近い環境で確認するためモデル中の既存 SW モデルを全て実行ファイルに置き換えたいと考えた。 同時期にモータユニット開発する Tier1 サプライヤから開発中の MCU が SPILS 検証にて要求を満足したと連絡があったことから SW を提供いただき OEM 内で SILS 検証に用いることとした。 作成した制御モデルの I/O は物理量であり今回使用する実行ファイルの I/O はデジタル量であるためデータ変換が必要であるが主担当者は上記のようなデータ変換が今回初の取り組みである。

ユースケース 1、2 を Tier1 サプライヤの MCU 開発の視点で見た開発フェーズに当てはめると図 8-1 のようにユースケース 1 は Component Level の PF 追加/テストのフェーズに該当し、ユースケース 2 は Subsystem レベルの仮説事前検証フェーズに該当する。

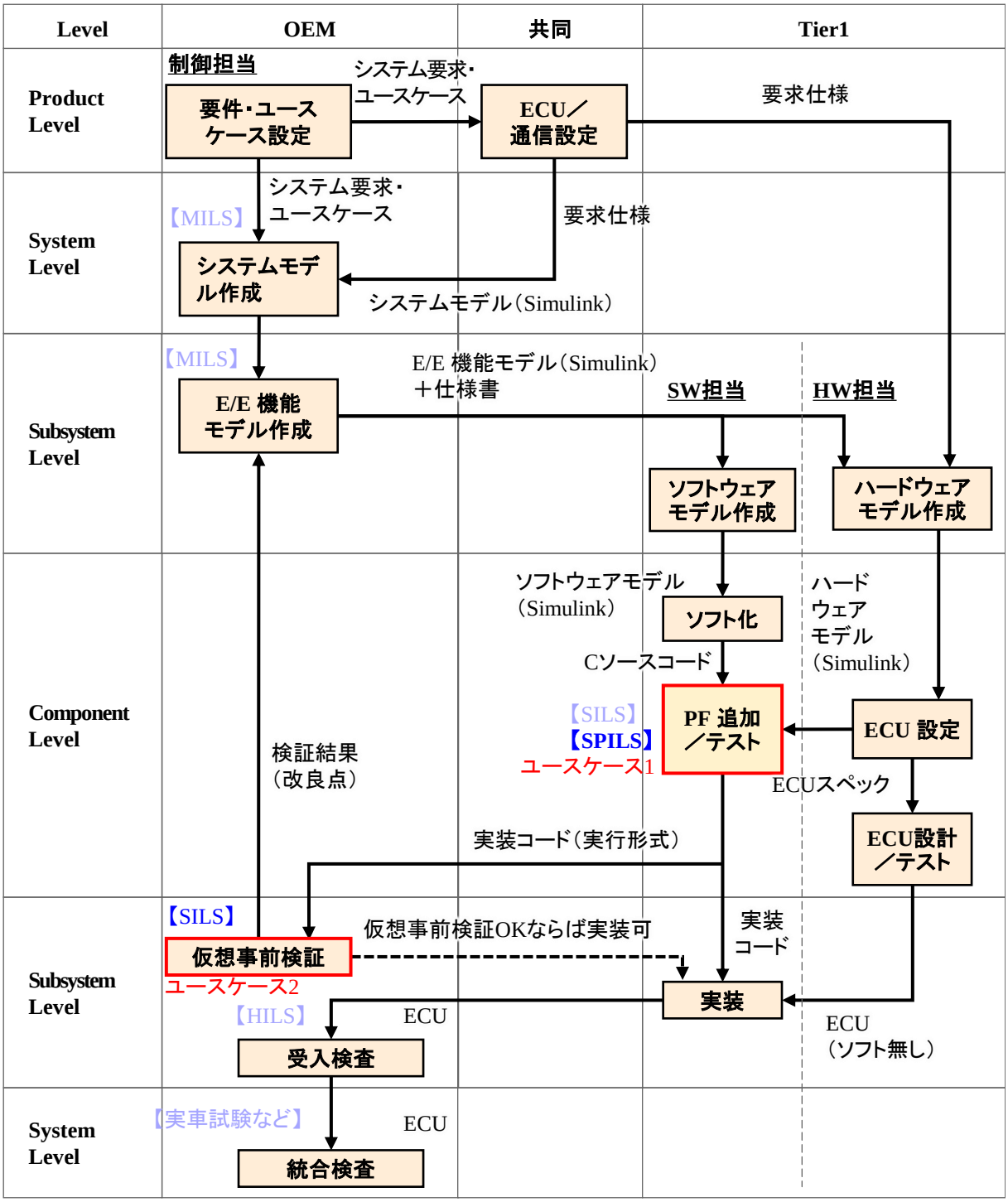
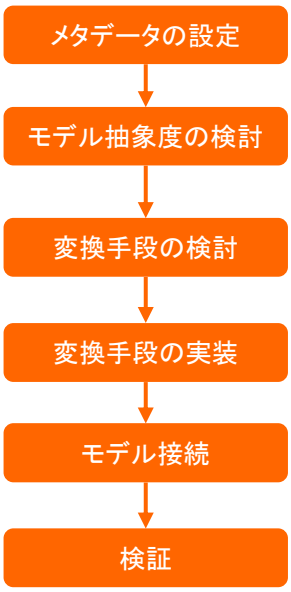


図 8-1. MCU 開発フェーズ

8.2. モデル接続実現までの手順

図 8-2 に異なる抽象度のモデルを接続するまでの手順を示す。手順の概要は接続するモデルについて抽象度の差異をメタデータ抽出により明らかにし、必要な変換内容と変換手段を検討する。そして変換手段の実装ならびにモデル間を接続しモデルの挙動を検証する。



項目	概略
メタデータの設定	抽象度に差異が生じるモデル同士の入出力I/Fにおける信号値の内容ならびに使用されている情報（メタデータ）を抽出し、抽象度の差異を把握する。
モデル抽象度の検討	モデルごとの信号値の内容やメタデータからそれぞれのモデルの抽象度を定義し、接続に必要な変換処理の内容を把握する。
変換手段の検討	変換処理を割り当てるシミュレーション環境や実装手段を決定する。
変換手段の実装	検討内容に従って変換手段を実装する。
モデル接続	実装した変換手段を用いてモデル接続する。
検証	作成したモデルを用いて検証する。

図 8-2. モデル接続の手順

8.3. 説明時のサンプル

手順の説明に用いるモデルは「次世代自動車等の開発加速化に係るシミュレーション基盤構築事業費補助金」に係る「車両性能シミュレーションモデル」にて公開されている「シリーズハイブリッド自動車用燃費モデル[(2)]」（以下、ガイドライン準拠モデル）とし解説する。ユースケース 1、2 どちらにおいても共通する MCU モデルを実行形式ファイルへの置き換えに着目し、モデルでは図 8-3 に示すように MG2 モデルを実行形式ファイルに置き換えることとする。

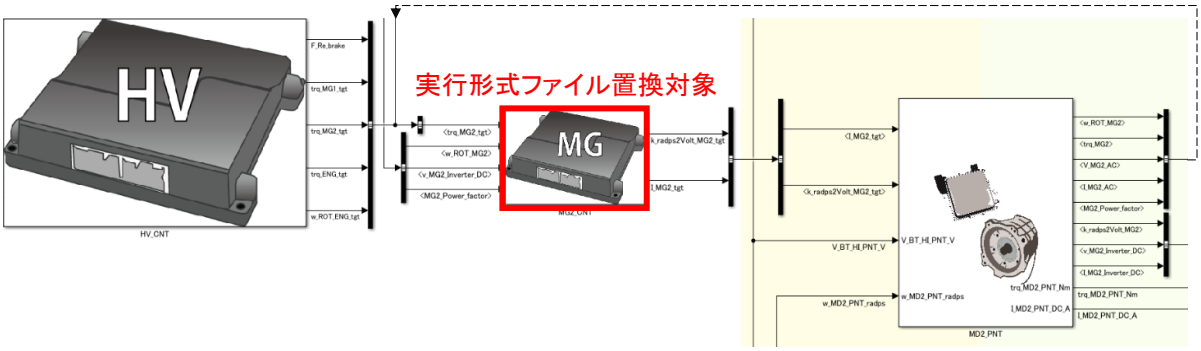


図 8-3. 置き換え対象

8.4. 手順 1（メタデータの設定）について

本手順では異なる抽象度のモデル同士が接続される境界の入出力データに着目して、それぞれの抽象度のモデルが必要とするデータの形式などを明示するために、メタデータの設定を行う。今回のサンプルにおけるメタデータ設定対象は図 8-4 の赤枠部分になる。

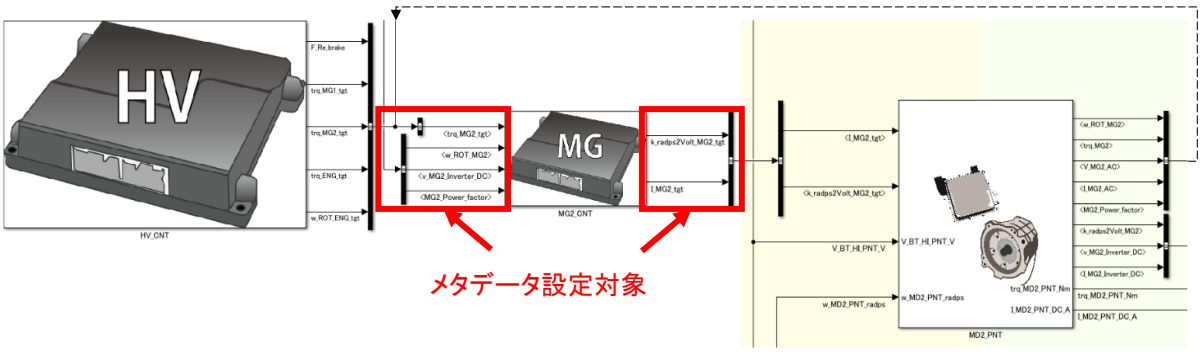


図 8-4. メタデータ設定対象

サンプルを用いたユースケース 1、2 における事例を後述する。

【ユースケース 1】

Tier1 サプライヤ開発者は MCU モデルの入出力 I/F についてメタデータを抽出すべく OEM が作成した Simulink モデルと Tier1 サプライヤが作成した SW 仕様書を確認した。メタデータの相違で大きな点としてはモデルでは連続物理量での取り扱いであるが SW ではデジタル量での取り扱いとなっている。さらに通信を CAN 形式で行うという条件があるため、Tier1 サプライヤの開発者は MCU に関する通信仕様書を確認し CAN 通信に関するメタデータを抽出した。メタデータ抽出結果を表 8-3 に示す。

【ユースケース 2】

OEM 開発者の場合、ユースケース 1 と同様にメタデータを抽出するが通信形式はスコープの範囲外であるため抽出されるメタデータは表 8-3 内のモデルと SW のみである。

表 8-3. メタデータ抽出結果

	モデル	SW	CAN通信
信号値	物理値 論理値	バイナリ値	CANプロトコルデータ
メタデータ	信号名称 値範囲 単位 入出力方向 説明	信号名称 値範囲 LSB（分解能） オフセット データ型 初期値 ポート バイトオーダー 周期 RAM名	信号名称 値範囲 CAN ID CAN ボーレート CAN DLC CAN データ位置

8.5. 手順 2（モデル抽象度の検討）について

本手順では手順 1 で抽出した接続するモデルそれぞれのメタデータから前項にて定義したモデル抽象度をもとに該当する抽象度を選択し、モデル間の接続に必要な変換処理を明らかにする。サンプルを用いたユースケース 1、2 における事例を後述する。

【ユースケース 1】

Tier1 サプライヤ開発者は表 8-4 に示す抽象度定義を参考にモデルと SW の抽象度を確認した。その結果、Simulink モデルは制御機能（連続）モデル、SW は制御 PF モデルに該当すると判断した。図 8-5 に示す抽象度変換参考図をもとに制御機能モデルと制御 PF モデルの変換は「離散化」、「基数変換（2 進数）」に加えて「通信プロトコル変換」を実施することで接続が可能であることを確認した。

【ユースケース 2】

OEM 開発者はユースケース 1 と同様に表 8-4 を参考にモデルと SW の抽象度を確認した。その結果、Simulink モデルは制御機能（連続）モデル、SW は制御 SW モデルに該当すると判断した。また、必要な変換についても図 8-5 をもとに確認し、制御機能モデルと制御 SW モデルの変換は「離散化」と「基数変換（2 進数）」の 2 種を実施することで接続が可能であることを確認した。

表 8-4. モデル抽象度定義

	制御機能（連続）モデル	制御機能（離散）モデル	制御SWモデル	制御PFモデル
信号値	物理値（連続） 論理値	物理値（離散） 論理値	バイナリ値	CANプロトコル データ
メタデータ	信号名称 値範囲 単位 入出力方向 説明	信号名称 値範囲 単位 入出力方向 説明	信号名称 値範囲 LSB（分解能） オフセット データ型 初期値 ポート バイトオーダー 周期 RAM名	信号名称 値範囲 CAN ID CAN ボーレート CAN DLC CAN データ位置

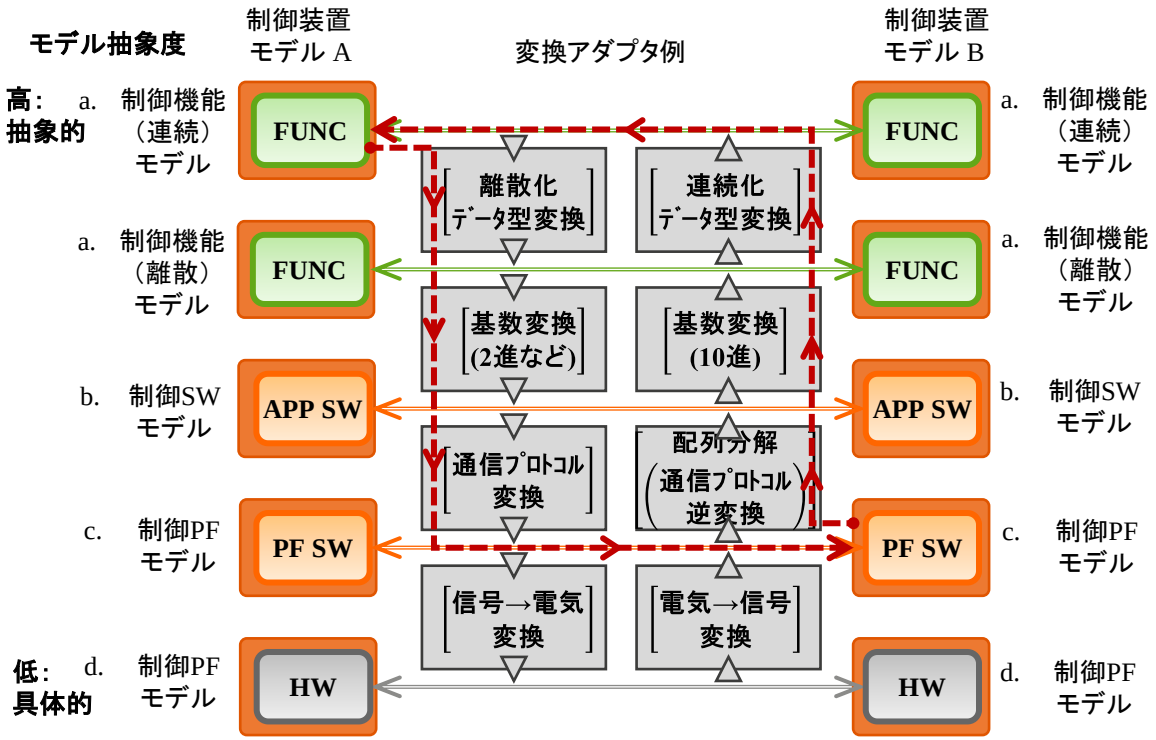


図 8-5. 抽象度変換参考図

8.6. 手順 3（変換手段の検討）について

本手順では手順 2 で明らかにしたモデル接続に必要な変換処理を割り当てるシミュレーション環境を設定する。今回のサンプルでは Simulink モデルに変換処理を割り当てているが、ユースケースに応じて他のシミュレーション環境に割り当てても問題ない。サンプルを用いたユースケース 1、2 における事例を後述する。

【ユースケース 1】

Tier1 サプライヤ開発者は SPILS 検証にて Simulink モデルを実行する MATLAB/Simulink に加えて SW を実行する命令セットシミュレータを検証に使用する。今回の検証では抽象度変換を Simulink モデルに実装することとした。

【ユースケース 2】

OEM 開発者は SILS 検証にて MATLAB/Simulink のみを用いることから抽象度変換も Simulink モデルに実装する。

8.7. 手順 4（変換手段の実装）について

本手順では手順 2、3 で定めた変換処理、割当先の開発環境に沿って機能を実装する。例として今回作成した変換処理の Simulink モデルを 8.7.1.から 8.7.3 に示す。ユースケース 1 では離散化データ型 / 連続化データ型変換アダプタ、基数（2 進 / 10 進）変換アダプタ、通信プロトコル変換アダプタを用い、ユースケース 2 では離散化データ型/連続化データ型変換アダプタ、基数（2 進 / 10 進）変換アダプタを用いることとした。

8.7.1. 離散化データ型 / 連続化データ型変換アダプタ

離散化変換アダプタは、現実を想定した物理量（連続量）で検討される制御機能と、コンピュータで扱われる離散量で検討される制御機能とを接続するための変換アダプタである。離散化データ型変換アダプタの詳細を図 8-6 に連続化データ型変換アダプタの詳細を図 8-7 に示す。

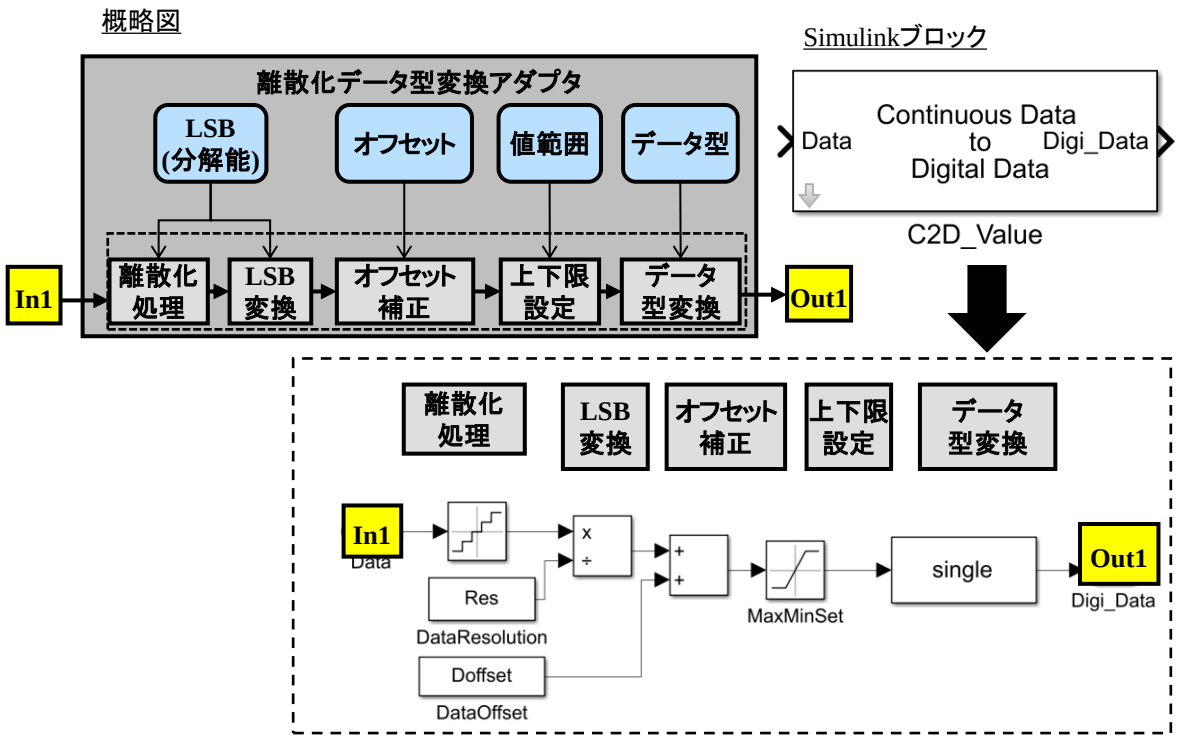


図 8-6. 離散化変換アダプタ詳細

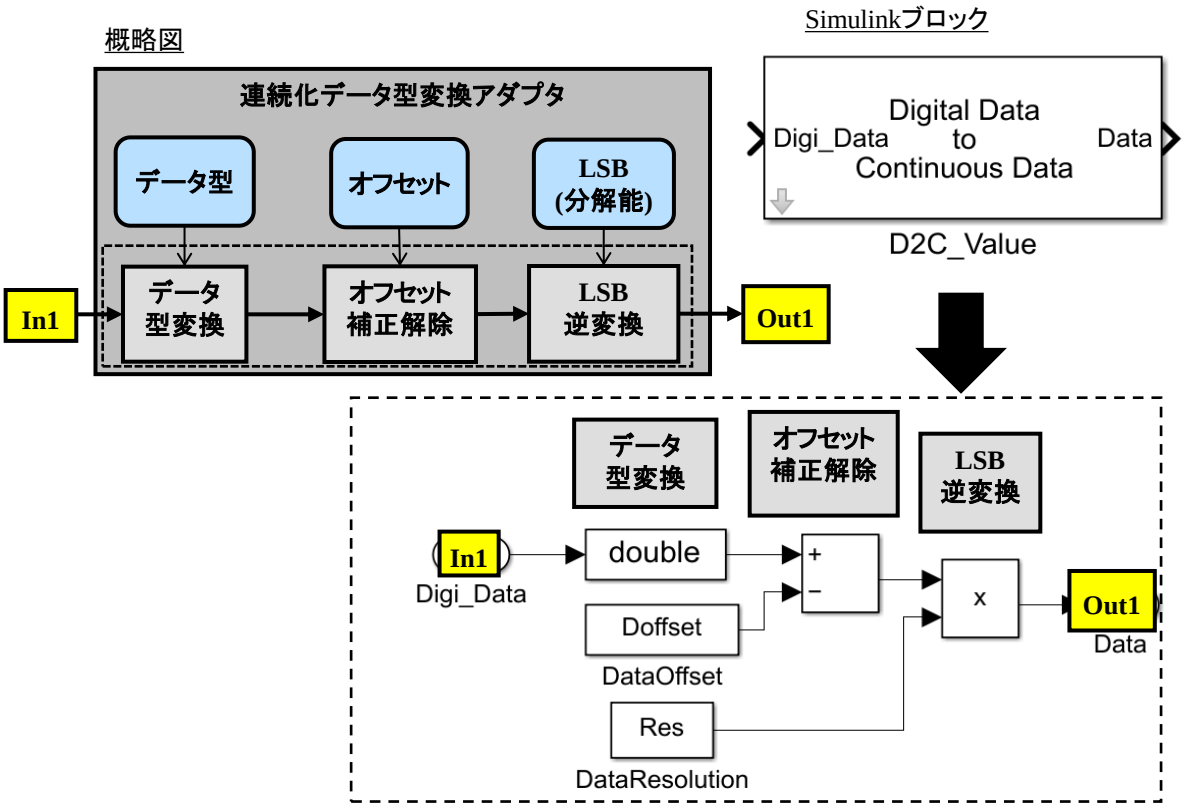


図 8-7. 連続化変換アダプタ詳細

8.7.2. 基数変換アダプタ

SW 変換アダプタは、コンピュータの記憶域やデジタル通信を想定し、離散化したデータを 2 進数（16 進数）へ基数変換する、または逆に 10 進数へ戻すための変換アダプタである。基数変換（2 進）アダプタの詳細を図 8-8 に基数変換（10 進）アダプタの詳細を図 8-9 に示す。

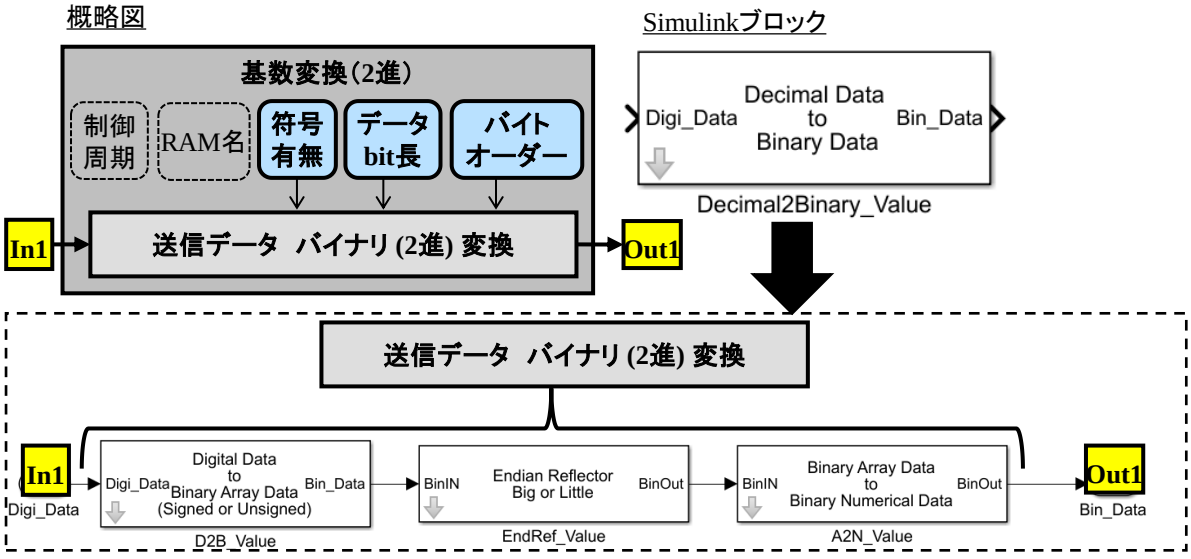


図 8-8. 基数変換（2 進）アダプタ詳細

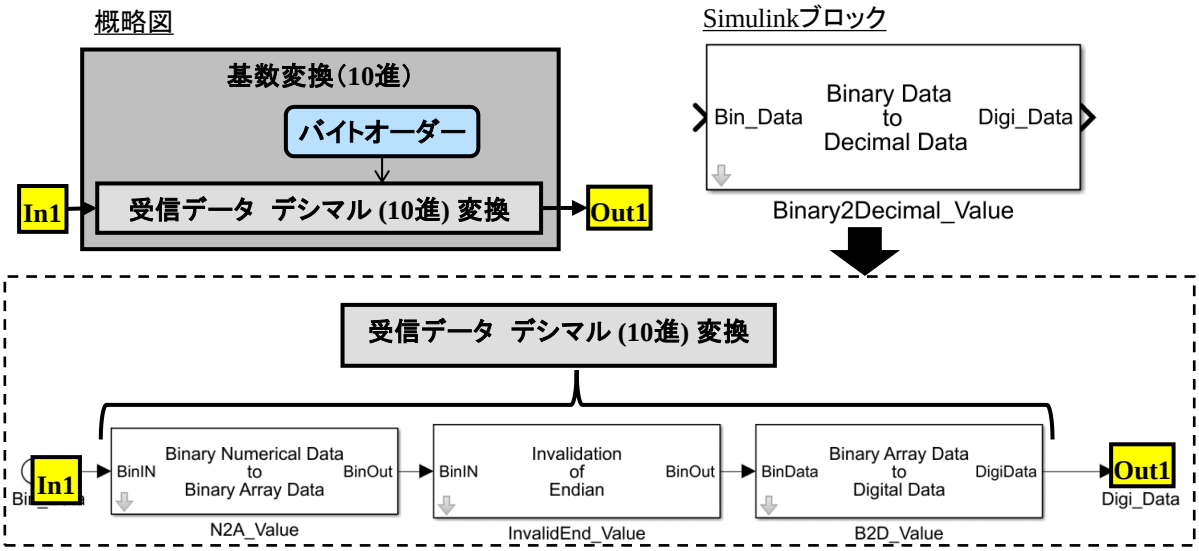
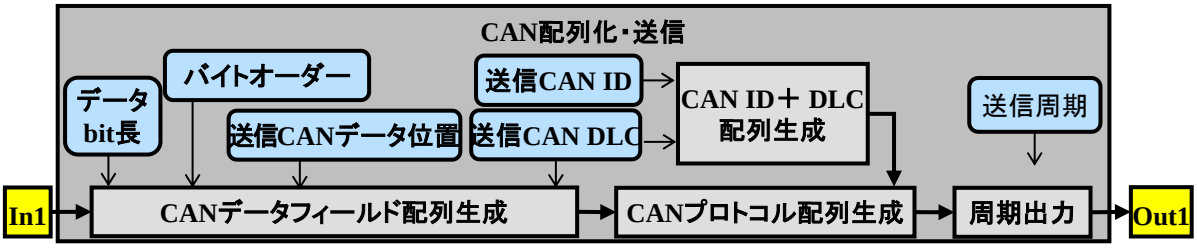


図 8-9. 基数変換(10 進)アダプタ詳細

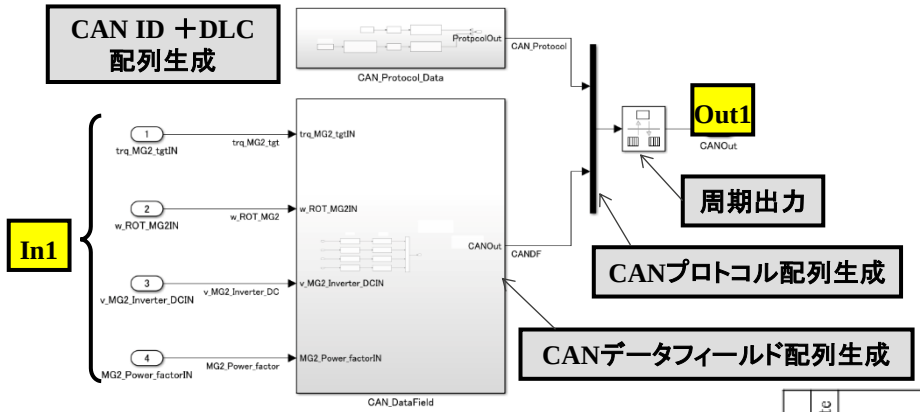
8.7.3. 通信プロトコル変換アダプタ

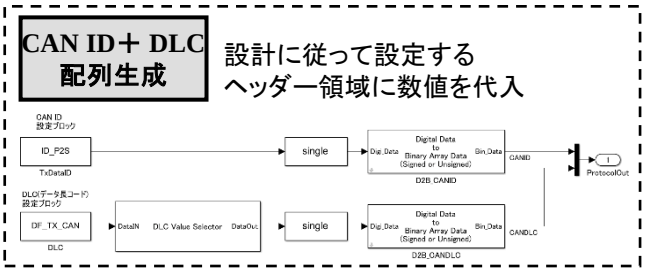
PF 接続変換アダプタは使用する通信プロトコルに応じたデータフォーマットにデータを変換するアダプタである。今回の Simulink のサンプルでは CAN 通信を用いることとし、そのフォーマットにあったアダプタを作成している。通信プロトコル変換アダプタの詳細を図 8-10 に通信プロトコル逆変換アダプタの詳細を図 8-11 に示す。

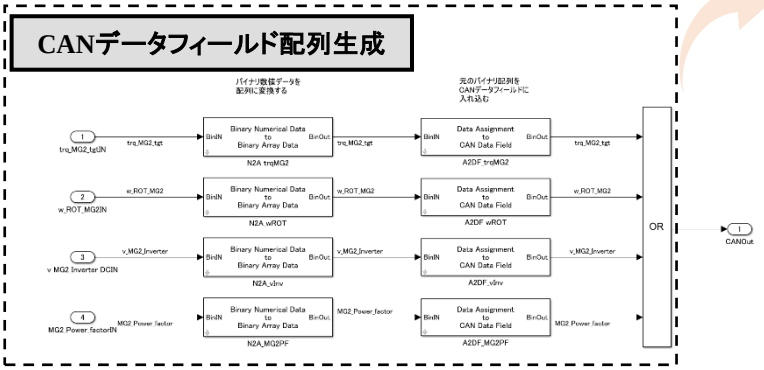
概略図

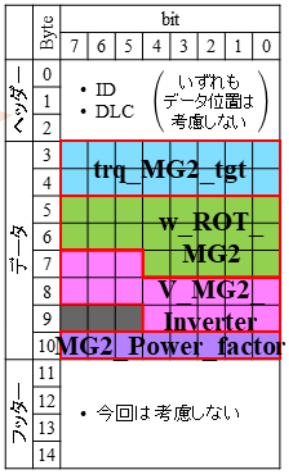


Simulinkブロック









DLCで設定したデータフィールドのデータ長に合わせてデータの配列を生成

図 8-10. 通信プロトコル変換アダプタ詳細

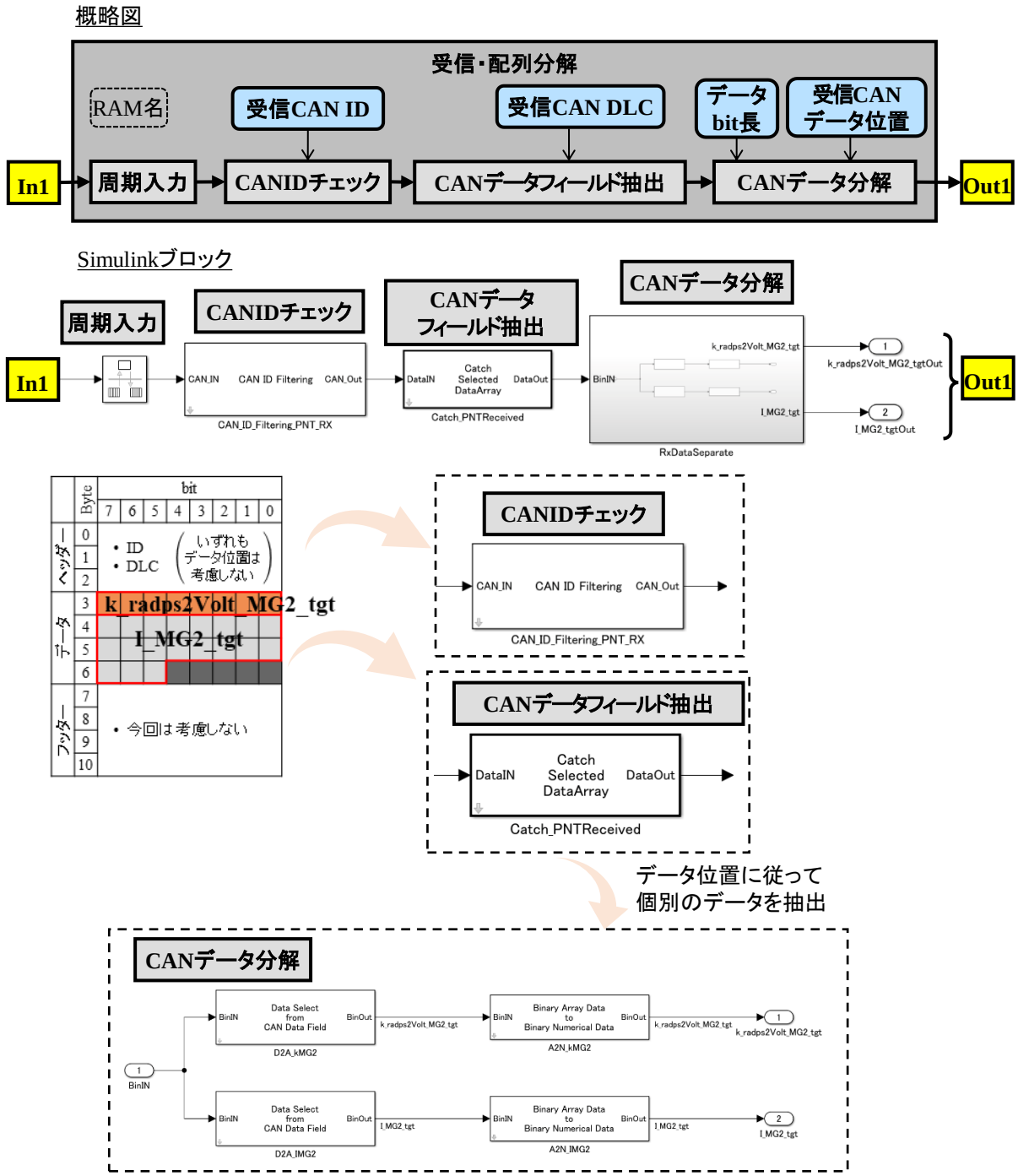


図 8-11. 通信プロトコル逆変換アダプタ詳細

8.8. 手順 5（モデル接続）について

本手順では手順 4 で作成した変換処理機能を用いて抽象度の異なるモデル同士を接続する。サンプルを用いたユースケース 1、2 における事例を後述する。ユースケース 1 に示すようにモデル抽象度だけでなく使用するソフトウェアによってはその固有の要因によりさらに変換要素が必要となる場合も考えられる。サンプルを用いたユースケース 1、2 における事例を後述する。

【ユースケース 1】

Tier1 サプライヤ開発者は連続物理量と CAN プロトコルデータの相互変換ができるように変換ブロックを用意したが、命令セットシミュレータのデータ送受信の要件から CAN データを ID、DLC、データフィールドを 1byte ごと、といった単位でデータを分割して送受信する必要があることが判明した。最終的には図 8-12 のような構成で連成シミュレーションが実現できた。

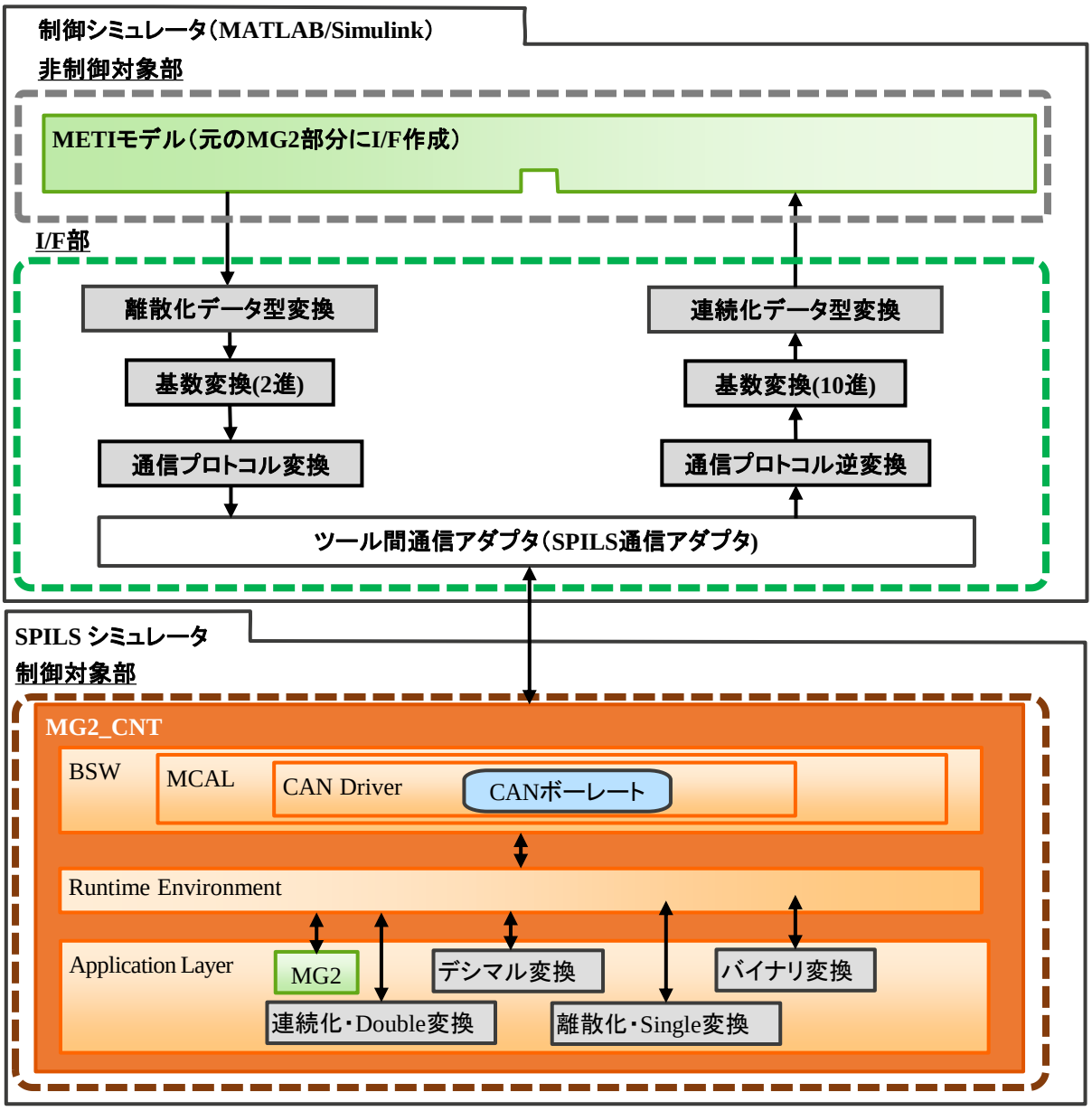


図 8-12. SPILS シミュレーション構成

【ユースケース 2】

OEM 開発者は連続物理量とデジタル量の相互変換ができるように変換ブロックを用意し、図 8-13 のような構成で Simulink モデルを作成しシミュレーションが実現できた。

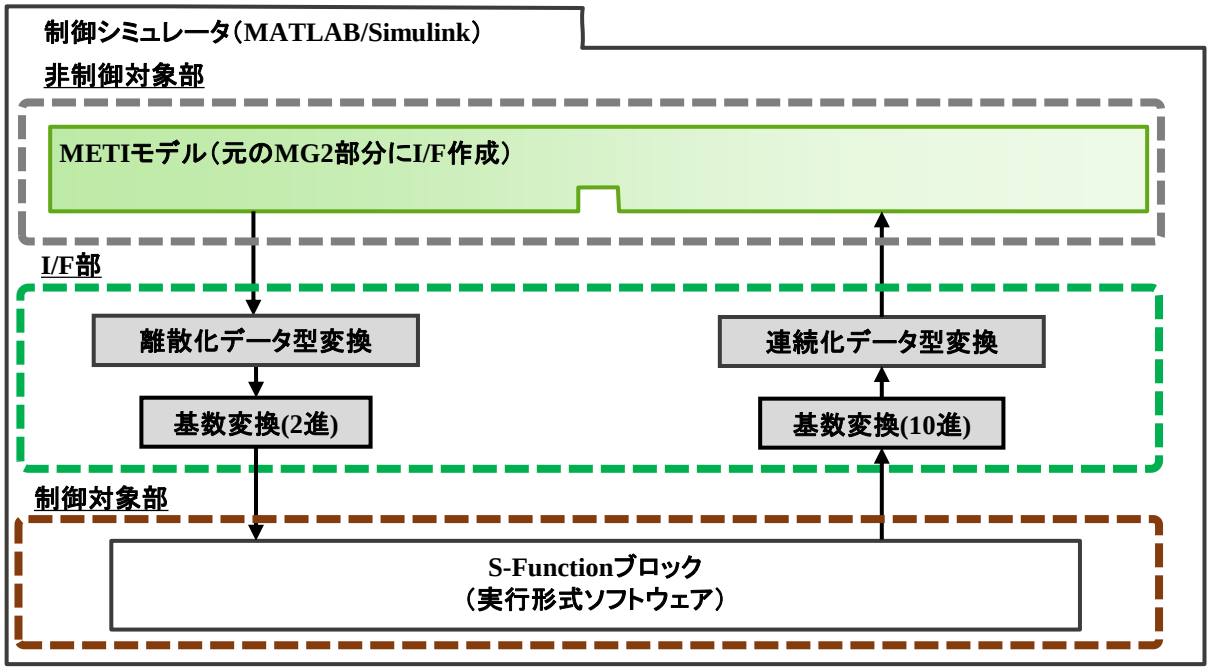


図 8-13. SILS シミュレーション構成

8.9. 手順 6（検証）について

本手順では手順 5 で作成したモデルをもとにユースケースに沿った検証を行い目的に対する妥当性を評価する。ユースケース 1、2 の事例を後述する。

【ユースケース 1】

作成したモデルによる SPILS 検証を実行し、SW と HW を統合した状態で OEM 要求のスペックを満足する結果が得られた。

【ユースケース 2】

OEM 開発の制御モデルを他ソフトウェアが実行形式ファイルに置き換わった状態で検証し、以前検証した他ソフトウェアがモデルである状態と一致した傾向が得られ、開発している制御モデルが妥当であると判断できた。

8.10. ユースケース 1 における手順 6（検証）の実証

ユースケース 1 の SPILS 検証について前述した流れに沿って正しくシミュレーションが可能かガイドライン準拠モデルを使用して実際に検証した。図 8-14 に初回トライ時の SPILS の結果をガイドライン準拠モデルのままシミュレーションした MILS 結果を図 8-15 に示す。結果から SPILS と MILS で傾向としては近似しているが SPILS 結果が一部のシミュレーション時間において振動する傾向が見られた。

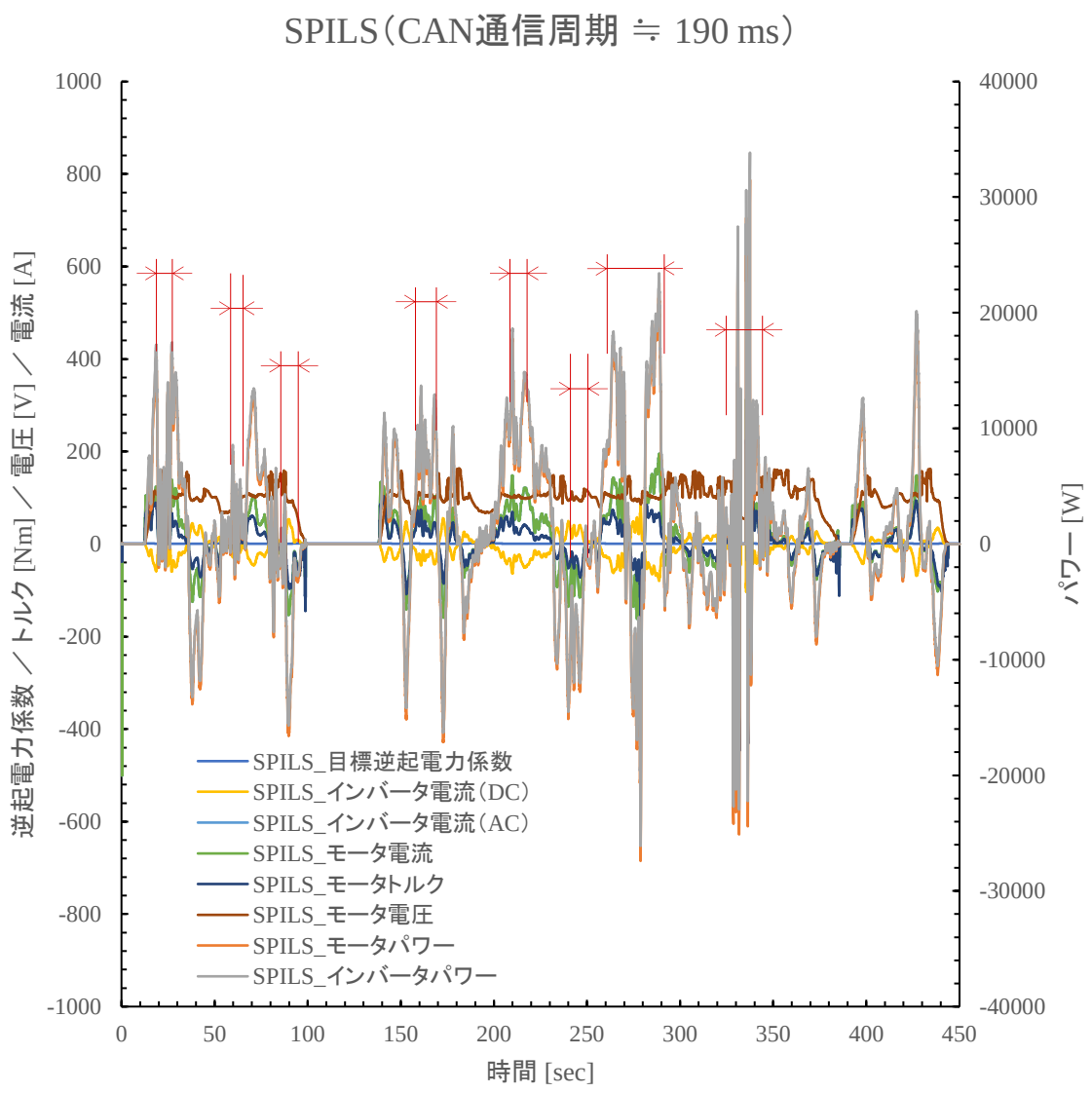


図 8-14. SPILS 結果（初回トライ）

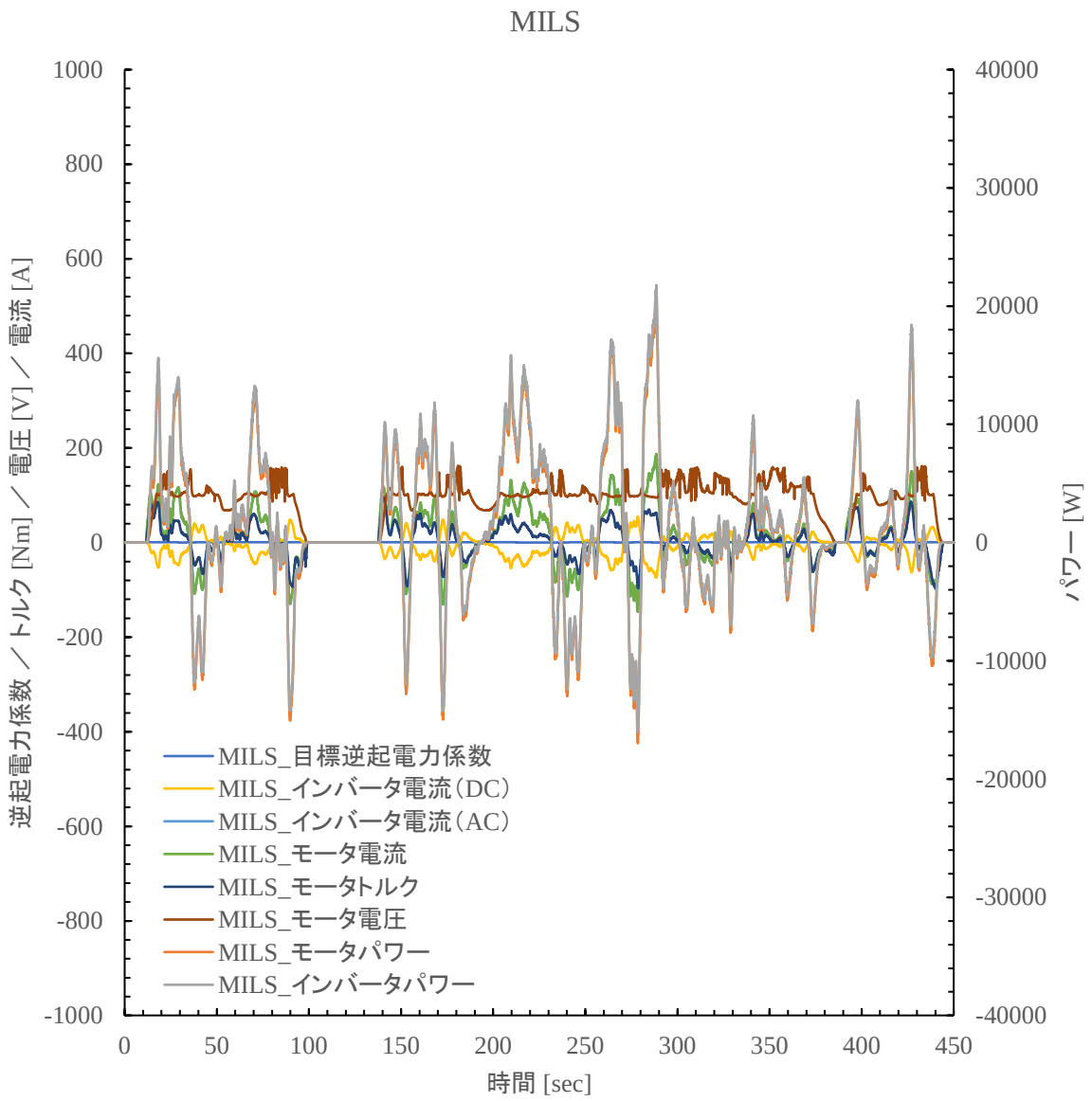


図 8-15 MILS 結果

振動の原因を確認するため、初回トライと同一の環境、かつ MCAL を使用しないシミュレーションを実行して、SPILS と MILS の結果を比較した。SPILS の結果を図 8-16 に示す。結果から SPILS の初回トライ（図 8-14）で見られた値の振動は確認できず、MCAL の設定に要因があることが明らかとなった。MCAL 設定を見直し、通信周期が 190ms と遅いことが原因と推定し、周期を変更してトライする。

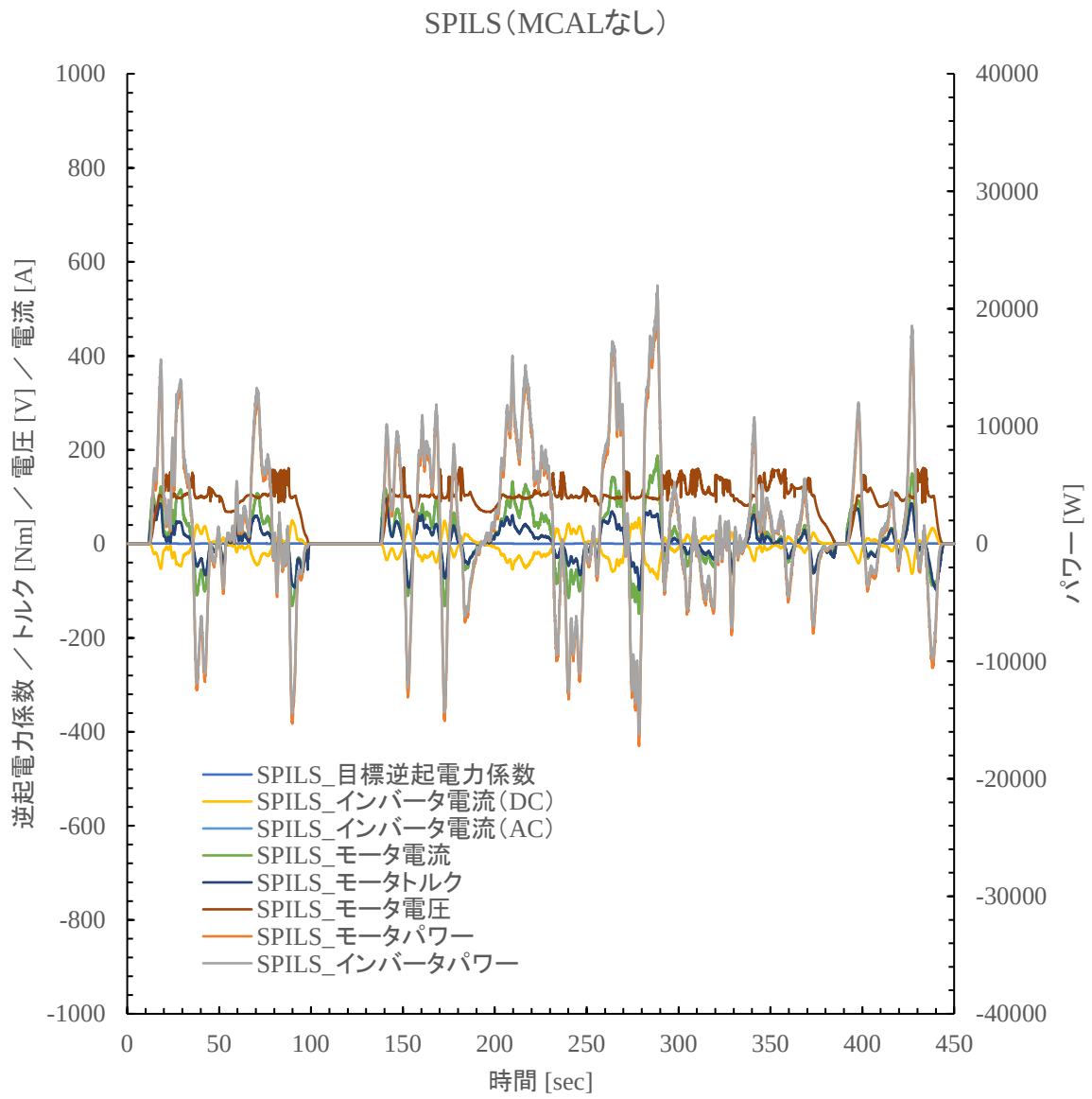


図 8-16. SPILS 結果（MCAL なし）

振動の原因と推定した SPILS モデル側の CAN 通信周期を修正して、再度シミュレーションを実行した。結果を図 8-17 に示す。結果より初回トライ（図 8-14）で見られた値の振動は解消されており、CAN 通信周期が振動の主要因であったことが明らかとなった。

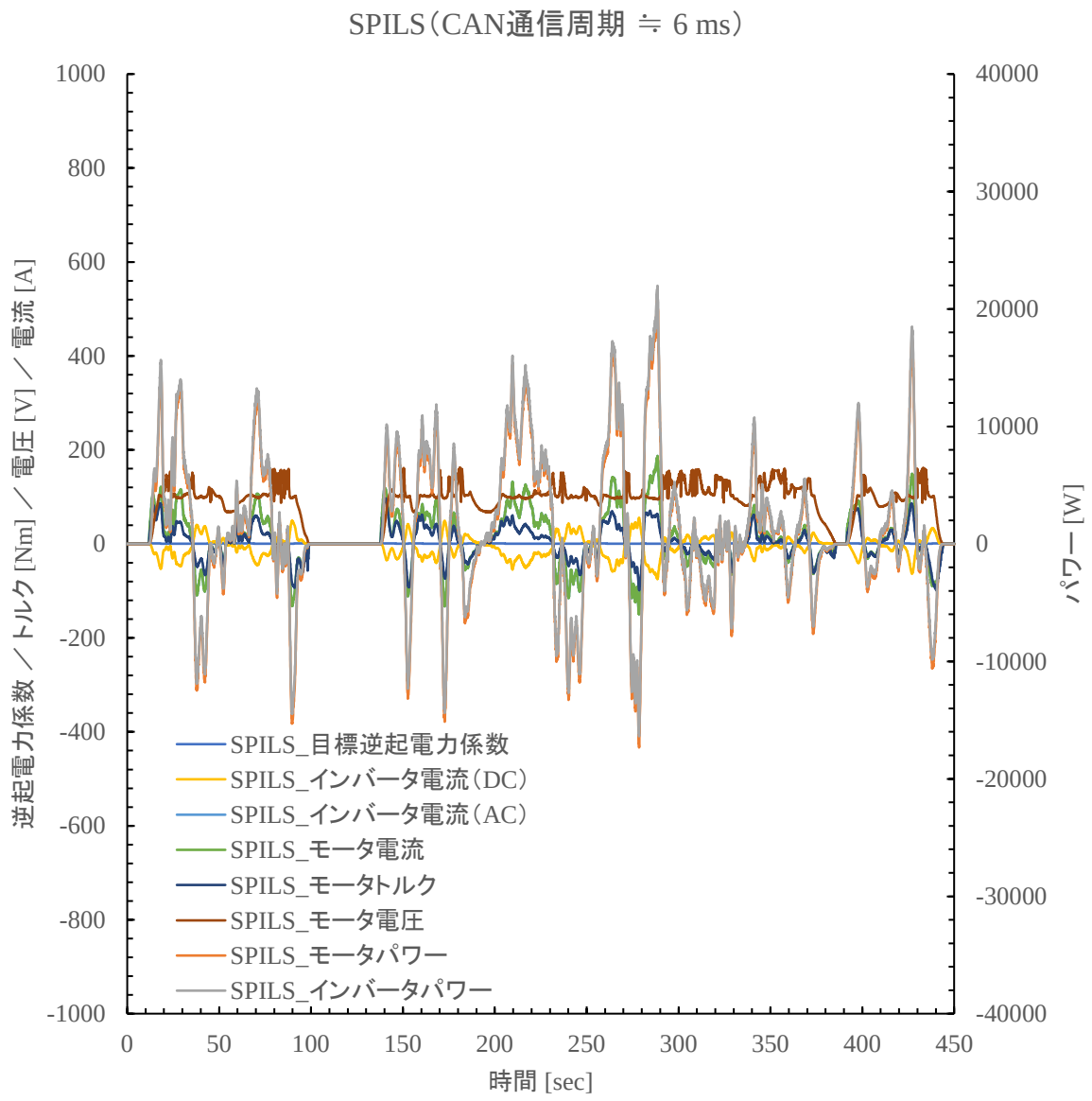


図 8-17. SPILS 結果（通信周期 6ms）

また、CAN 通信周期改善後の SPILS 結果とガイドライン準拠モデルの MILS 結果を比較した。モータパワー（P_MG2_ALL）の値を比較した結果を図 8-18 に示す。結果より SPILS、MILS 結果両者の相互相関係数は 0.9998 が得られており一致した傾向となっていることが明らかとなった。

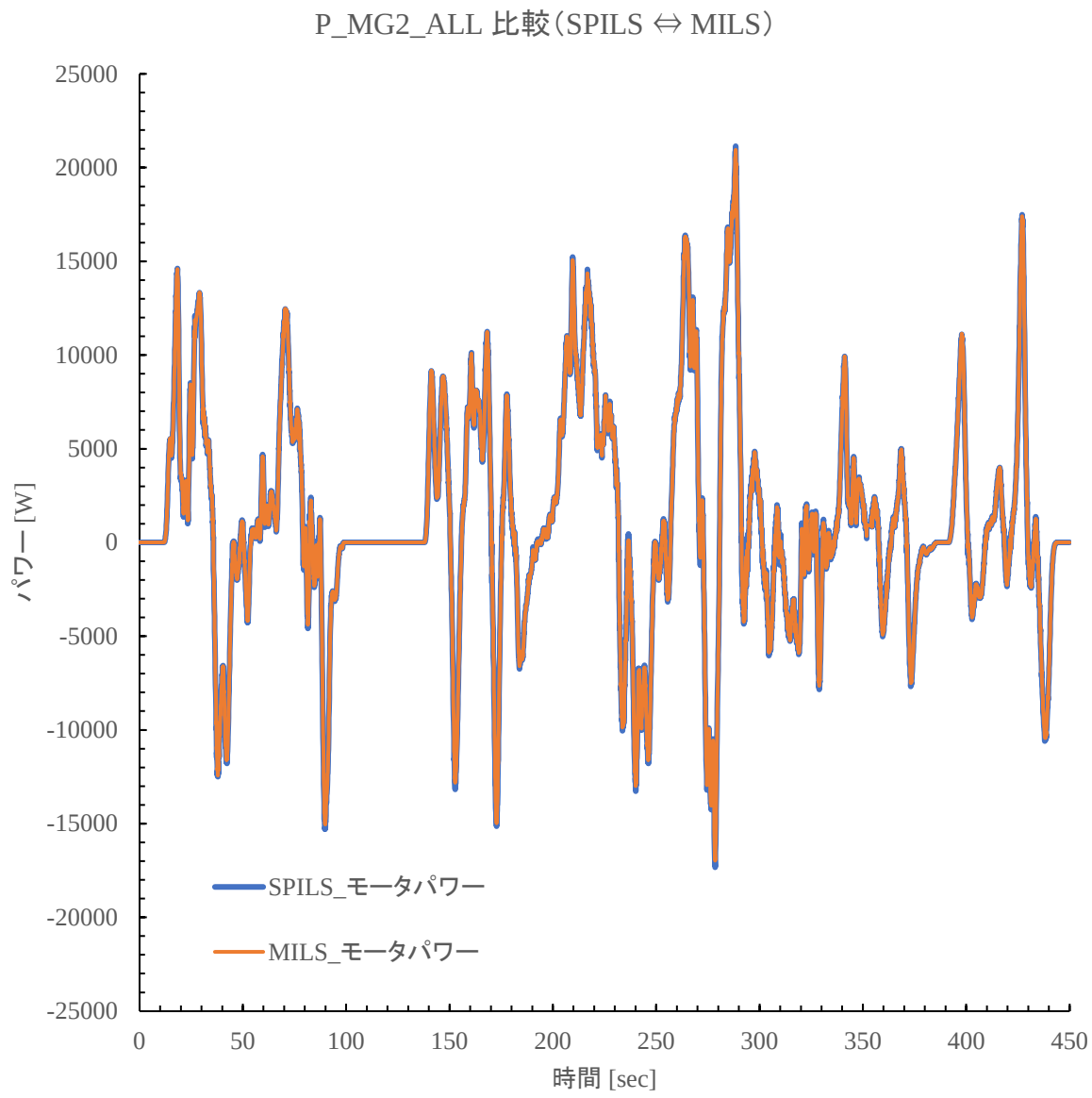


図 8-18. SPILS と MILS 結果の比較（P_MG2_ALL）

8.10.1. TIPS（モデル接続実施事例における注意点）

今回のモデル接続事例を遂行するにあたり、実際に注意を要した内容を表 8-5 にまとめた。

表 8-5. モデル接続注意内容一覧

番号	発生したトラブル	発生状況			発生箇所	対応	備考
		いつ	どのような方法で	何をしようとした			
	接続 I/F に起因したトラブル						
1	データ型不一致による実行エラー	既存MILSモデルとSILSモデルを接続してシミュレーションを実行したとき	設定を注意せずそのまま使用した	Simulinkでのシミュレーション実行	MILSモデル⇒SILSモデルへの入力部	MILSモデル側の出力データをベクトル型から配列へ変更した	今回は配列を使用したが、扱うデータサイズによって、他の方法にも可能性あり
	その他のトラブル（ガイドラインのスコープ外であるが課題となる可能性の大きい内容）						
2	設定不一致エラー	既存MILSモデルとSILSモデルを接続してシミュレーションを実行したとき	設定を注意せずそのまま使用した	Simulinkでのシミュレーション実行	コンフィギュレーションパラメータ設定	コード生成用モデルの設定をMILSモデルに合致させた	マニュアルの作成を推奨⇒トラブルの性質上、ガイドラインには不向き
3	参照先不明エラー	既存MILSモデルとSILSモデルを接続してシミュレーションを実行したとき	設定を注意せずそのまま使用した	Simulinkでのシミュレーション実行	MILSモデルのモニタサブシステム内	モデル内で対象となる‘From’ブロックを削除した	同上
4	プロセッサ不一致エラー	SPILSからSILSに立ち戻ってシミュレーションしたとき	SPILS用に用意した制御装置モデルをそのまま使用した	コード生成	コード生成時の選択項目	RELマイコン⇒Intel（正しい指定）に設定変更した	I/F起因ではないが、関係者間をまたぐときに発生し得るトラブル
5	ビルドエラー	SPILS用の実行ファイルを生成したとき	設定を注意せず生成したコード類をそのまま使用した	開発対象マイコン用の実行形式ファイルのビルド	生成コードのデータ型とビット長の設定	開発対象マイコンの定義に合わせてモデル設定を修正し、再コード生成した	抽象度違いモデルの接続に、入出力データのメタデータ以外情報の必要性も示唆

表 8-5 の内容について番号ごとに詳細を示す。

【項目 1　データ型不一致による実行エラーについて】

発生したエラーと今回の対応を図 8-19 に示す。内容は Simulink モデルと SIL モデルとの接続の際に使用した配列データの次元数の差異によるエラーであり、今回は Simulink モデル側の配列次元数を SIL モデルに合わせて一次元とする対応とした。他の対策として 2 進数データを整数に再変換して送信する方法も考えられるが、今回は桁落ちが発生したため不採用とした。データ送信方法は複数の方法が考えられ、定義差異によるエラーを防ぐ手段については検討を継続する。

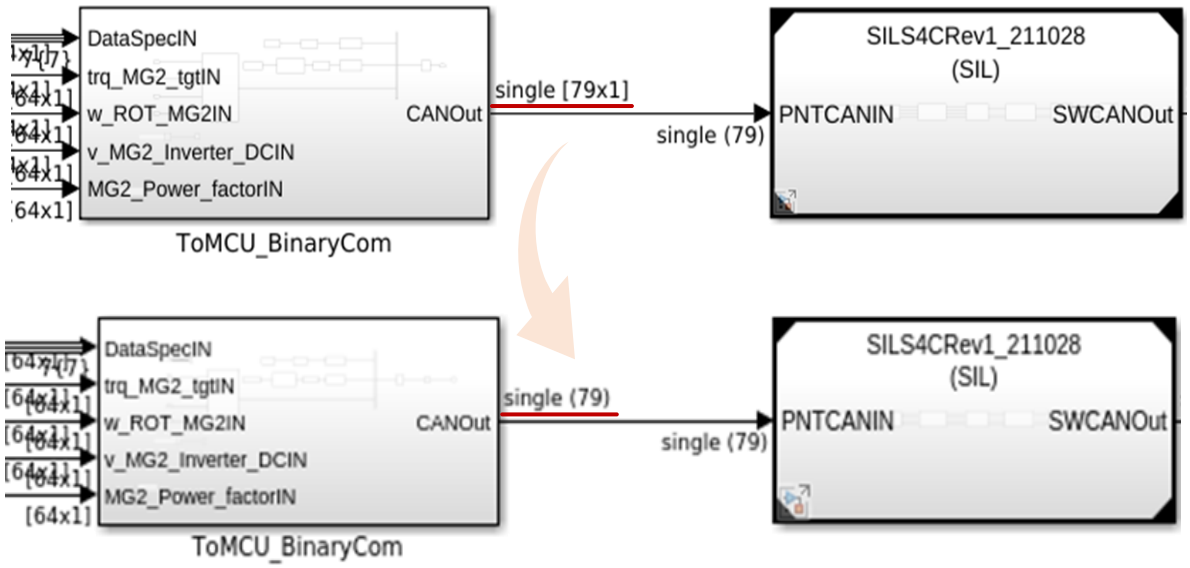


図 8-19. 配列の次元数差異によるエラー

【項目 2　設定不一致エラー】

発生したエラー内容を図 8-20 に示す。SIL モデルの参照先モデルの設定と MIL モデルの設定が不一致していたために生じており、設定を見直すことで解消した。開発する環境の設定については関係者間ですり合わせをするか全社統一などの対応によりエラーを回避可能と考える。

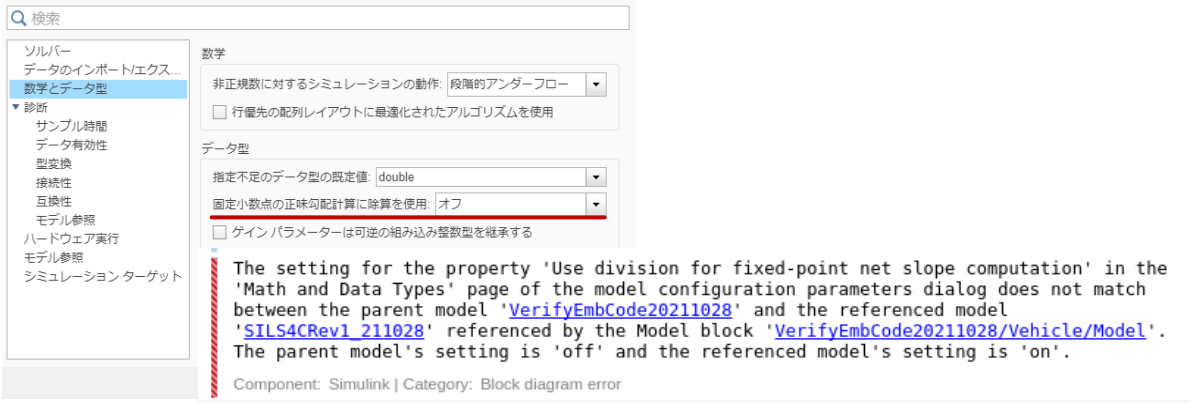


図 8-20. ハードウェア設定と実環境の不一致によるエラー

【項目 3　参照先不明エラー】

発生したエラー内容を図 8-21 に示す。MILS モデル内に存在していた Goto ブロックが SIL モデル置き換え時に消失してしまったことによる参照先不明エラーである。今回は対応する From がデータ表示のブロック（Scope）のみに接続されており、検証上不要な内容であったため From ブロックの削除で対応した。機能上必要な値で上記のようなエラーが発生する場合は必要な値を出力するポートを新規に作成する対応が必要である。

図 8-21. Goto/From 参照先不明によるエラー

【項目 4　プロセッサ不一致エラー】

発生したエラー内容を図 8-22 に示す。SPILS 検証用に Simulink モデルから自動生成したソースコードを SILS 検証に使用した際に発生している。Simulink モデルからソースコードを自動生成する際にターゲットとするプロセッサの種類が選択でき、動作させるプロセッサとターゲットが不一致であるためエラーが発生した。開発フェーズや開発製品などの差異により使用するプロセッサは異なる場合が考えられるため（例えば SILS 時は PC、SPILS 時はマイコンなど）開発時においても発生しうるエラーと考えられる。

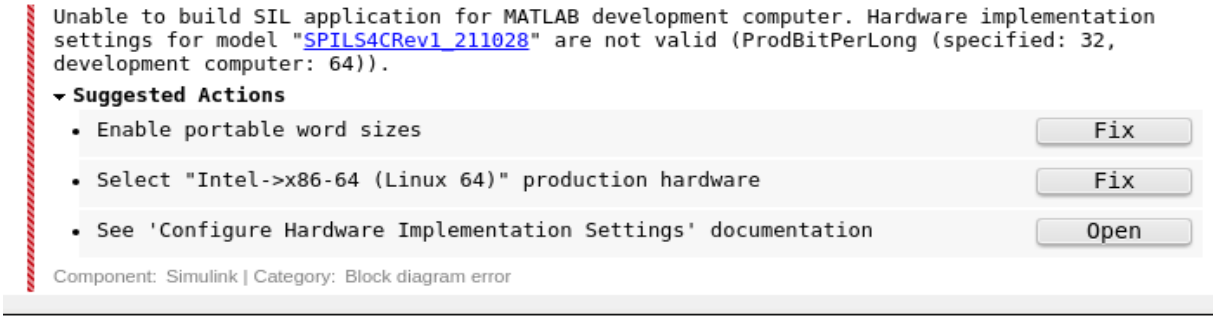


図 8-22. プロセッサ不一致によるエラー

【項目 5　ビルドエラー】

命令セットシミュレータ上でのソースコードのビルド時にビルドエラーが発生した。原因は使用するデータ型の名称とビット長が不一致しているためであった。Simulink モデルからソースコードを生成する際にデータ型名称とビット長を定義する項目が存在する。今回はその設定を見直すことでエラーを解消した（図 8-23）。

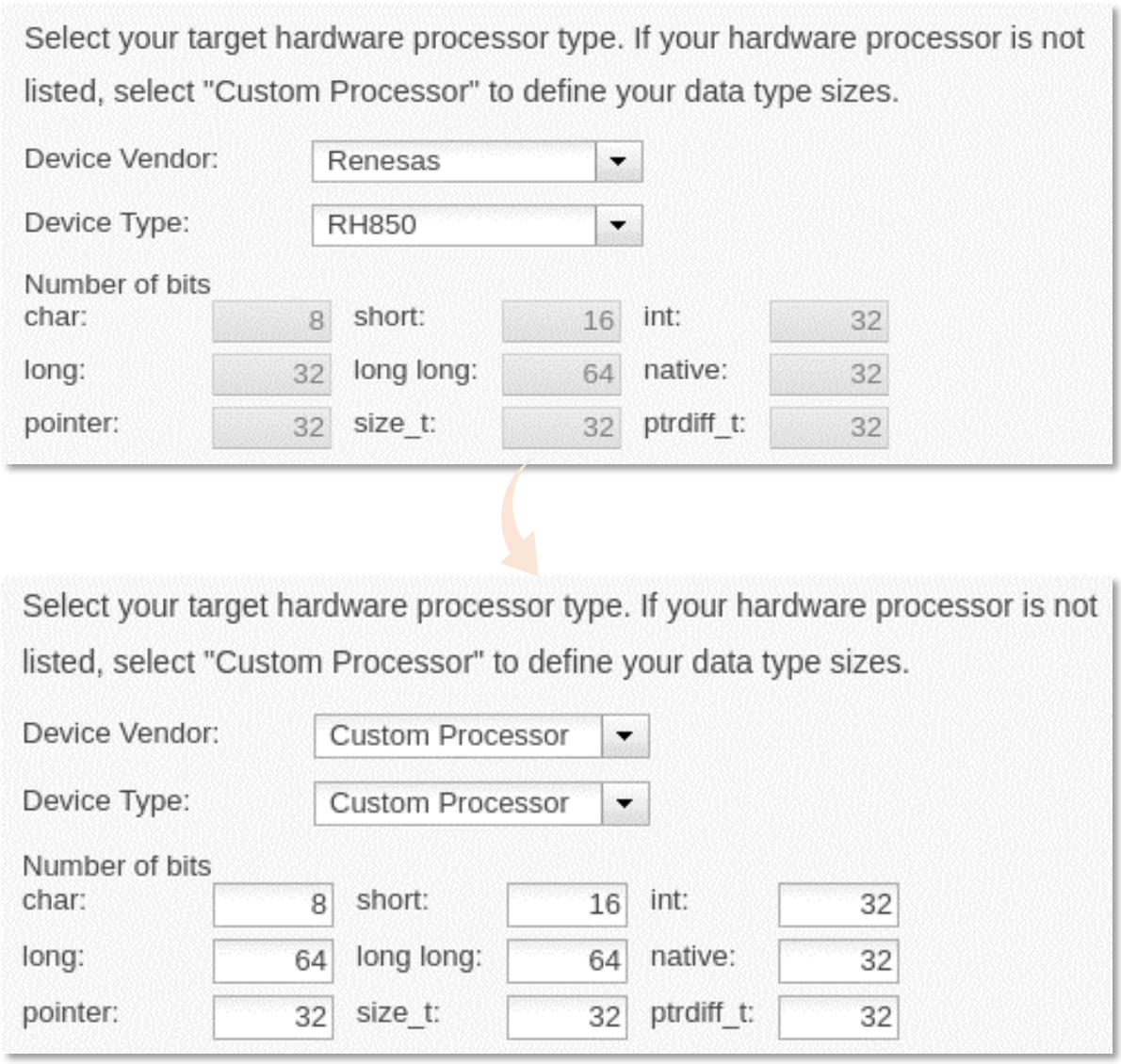


図 8-23. データ型とビット長不一致によるビルドエラー

9. APPENDIX 1 ： モデル接続の具体事例:補足

章 8.10.に記載した SPILS 実証にて MATLAB/Simulink と命令セットシミュレータ間を接続するために実施した内容を補足する。構成を図 9-1 に示す。使用したシミュレーション環境は MATLAB/Simulnk ならびに Synopsys 社が販売する命令セットシミュレータである Virtualizer である。今回の SPILS では Virtualizer 中の仮想マイコンの MCAL 内に存在する CAN Driver と Simulink モデル間で通信を実現している。図 9-1 中緑枠で示す部分がツール依存する通信機能となっており、Simulink モデル内には通信相手である Virtualizer より提供される専用の Simulink ブロックを使用している。また、Virtualizer 側は指定のメモリに入出力データを紐づけるため Python にて通信用のスクリプトを準備した。

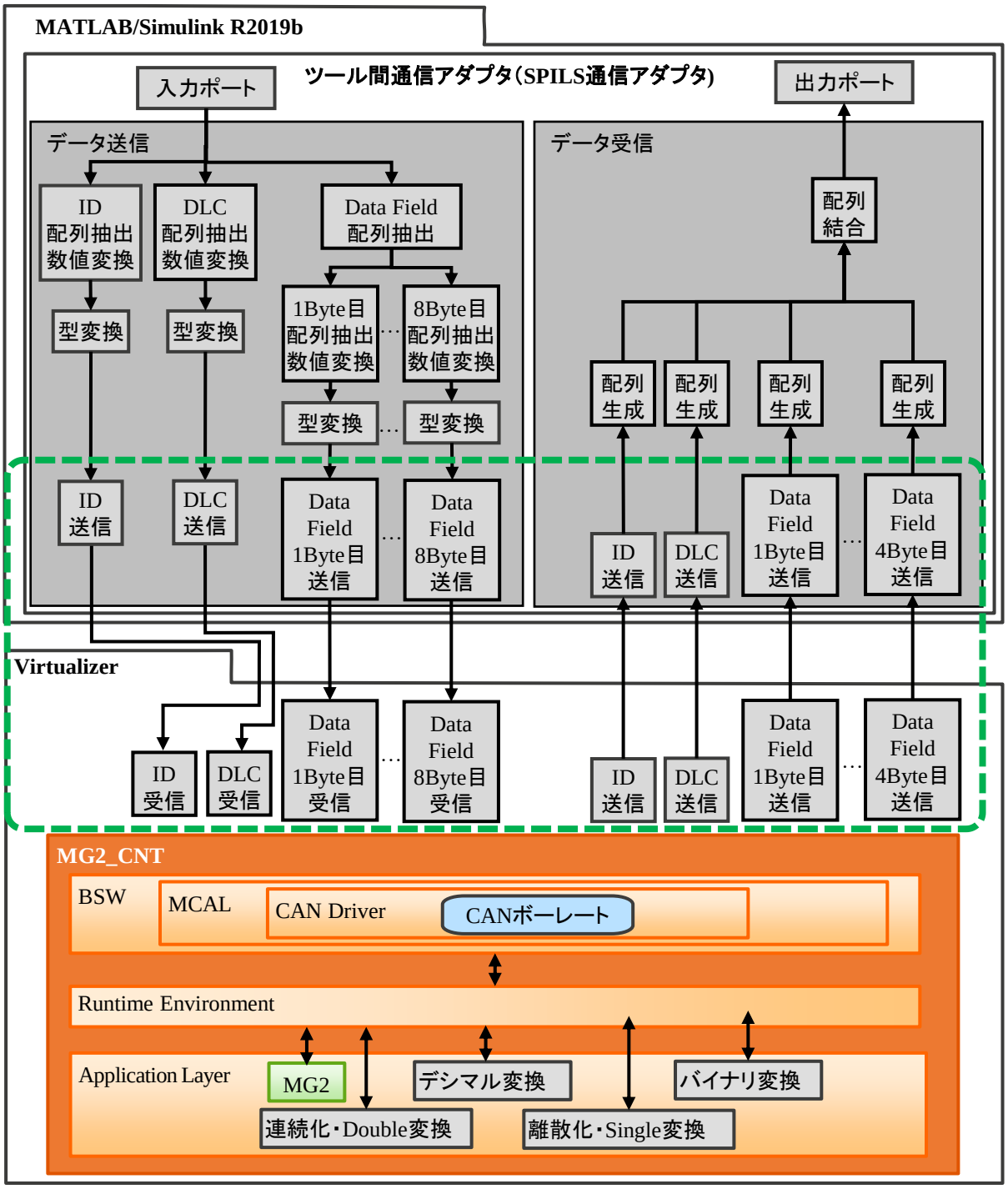


図 9-1. 連成シミュレーション構成図

今回はツール間のデータ通信仕様に基づきバイト毎の送受信を実行するために一旦生成した CAN のデータフィールドを送信可能なサイズに分解している(図 9-2)。本事例のように接続に追加の考慮が必要な場合は使用するシミュレーション環境の通信仕様を参照する必要がある。

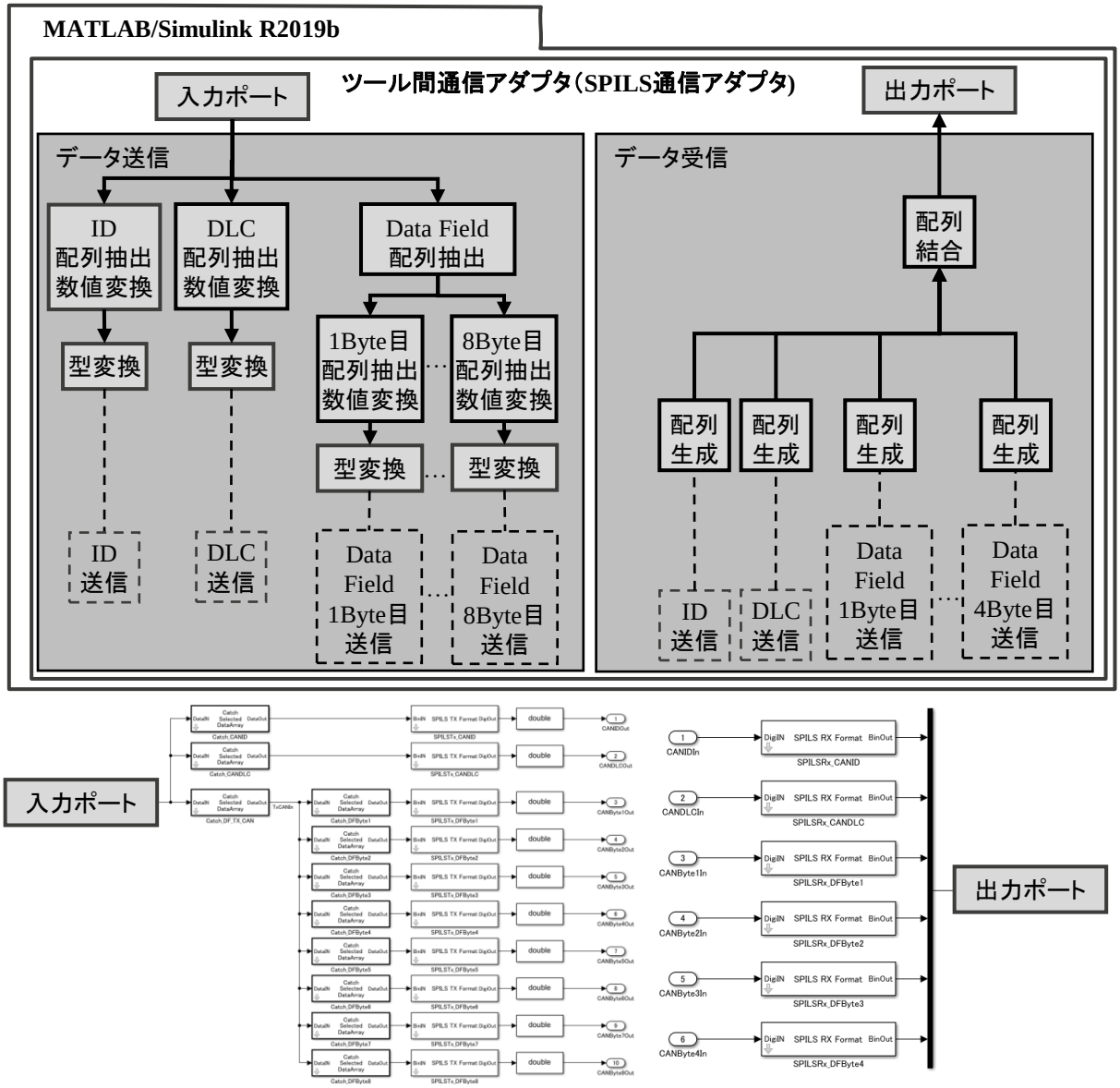


図 9-2. 連成シミュレーション実装内容

10. APPENDIX 2 : 検証結果グラフ

章 8.10 で割愛した他のデータに関する SPILS と MILS の検証結果比較について下記の図 10-1 から図 10-8 に示す。

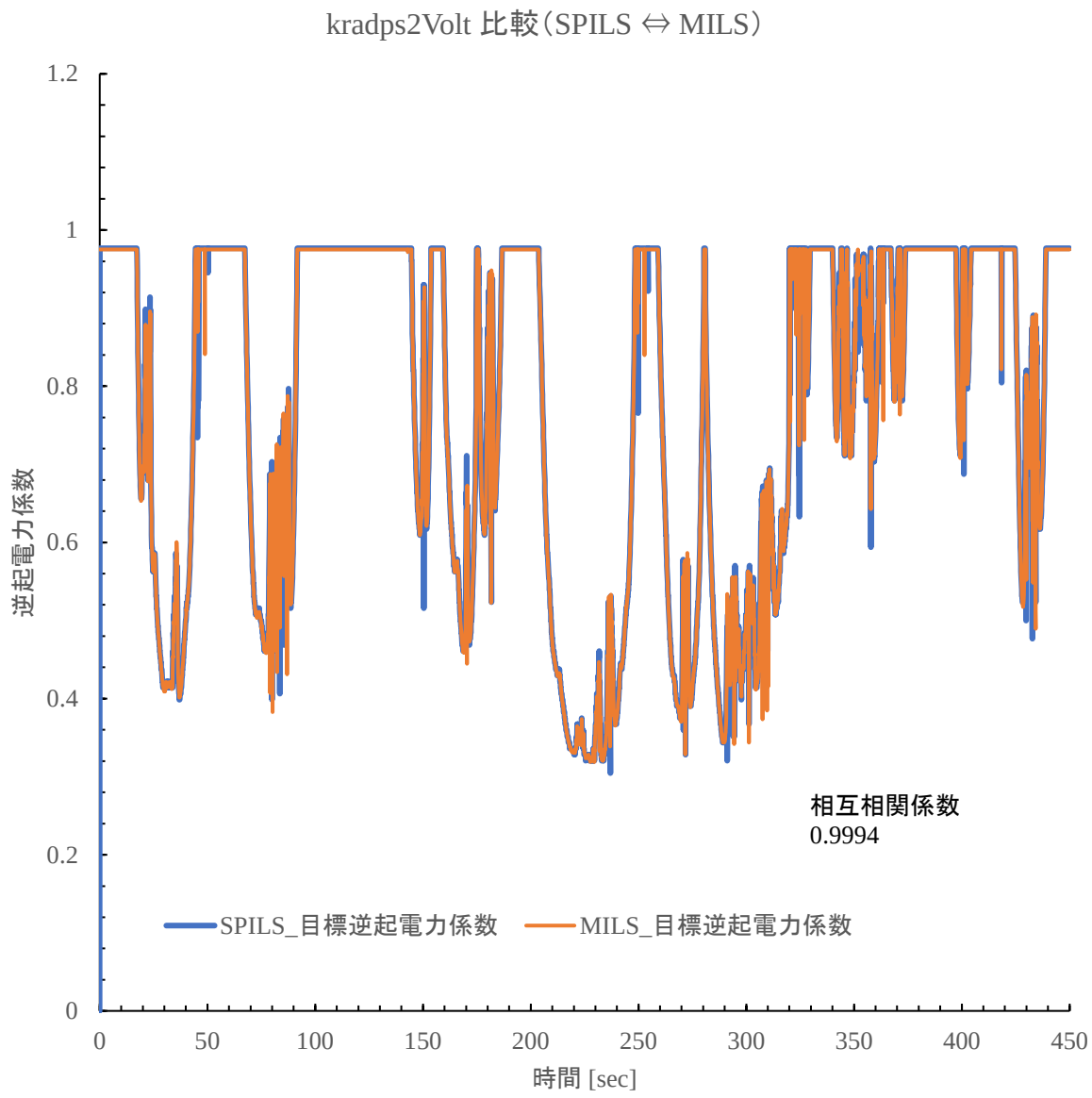


図 10-1. SPILS と MILS 結果の比較（kradps2Volt）

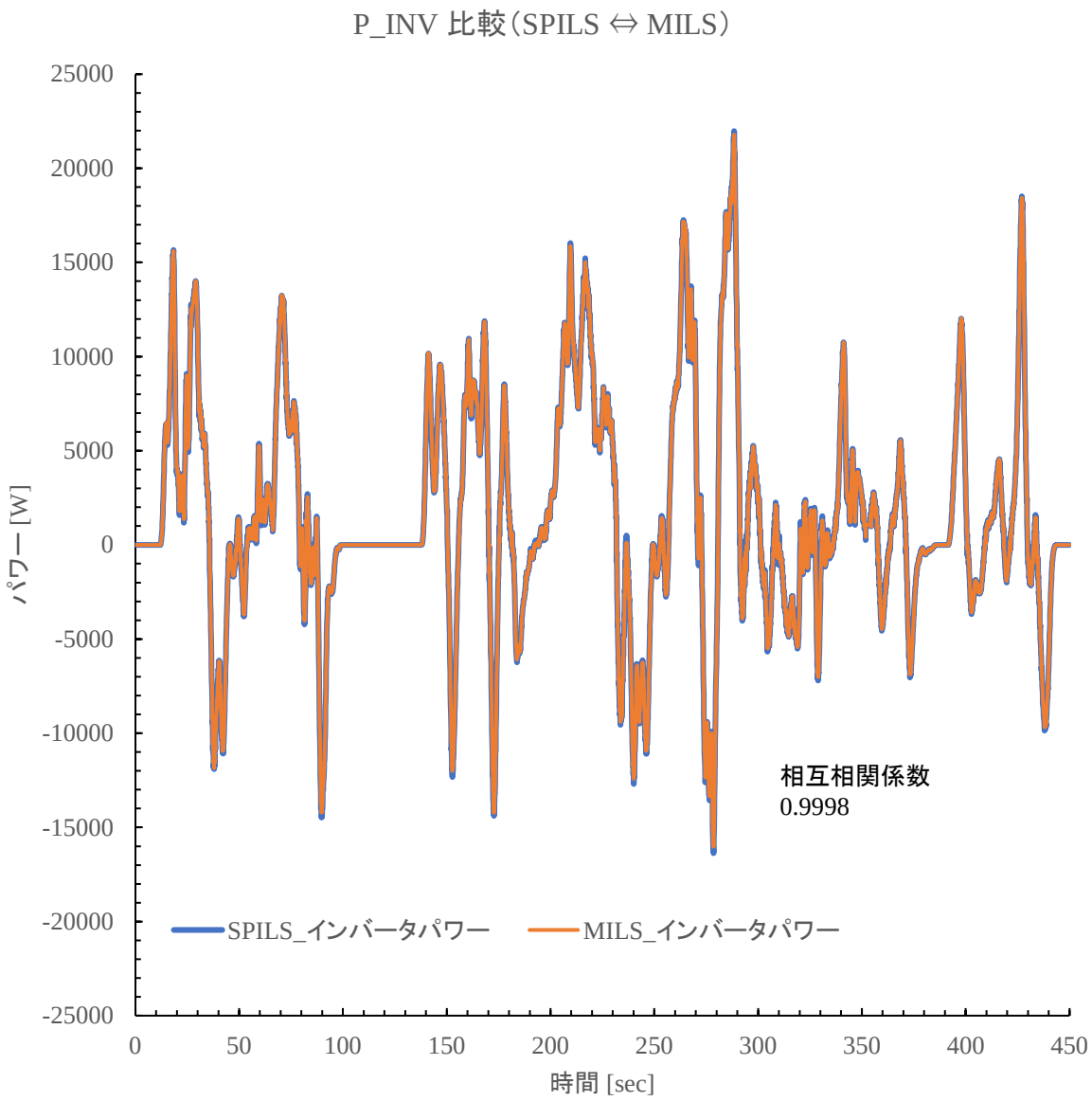


図 10-2. SPILS と MILS 結果の比較（P_INV）

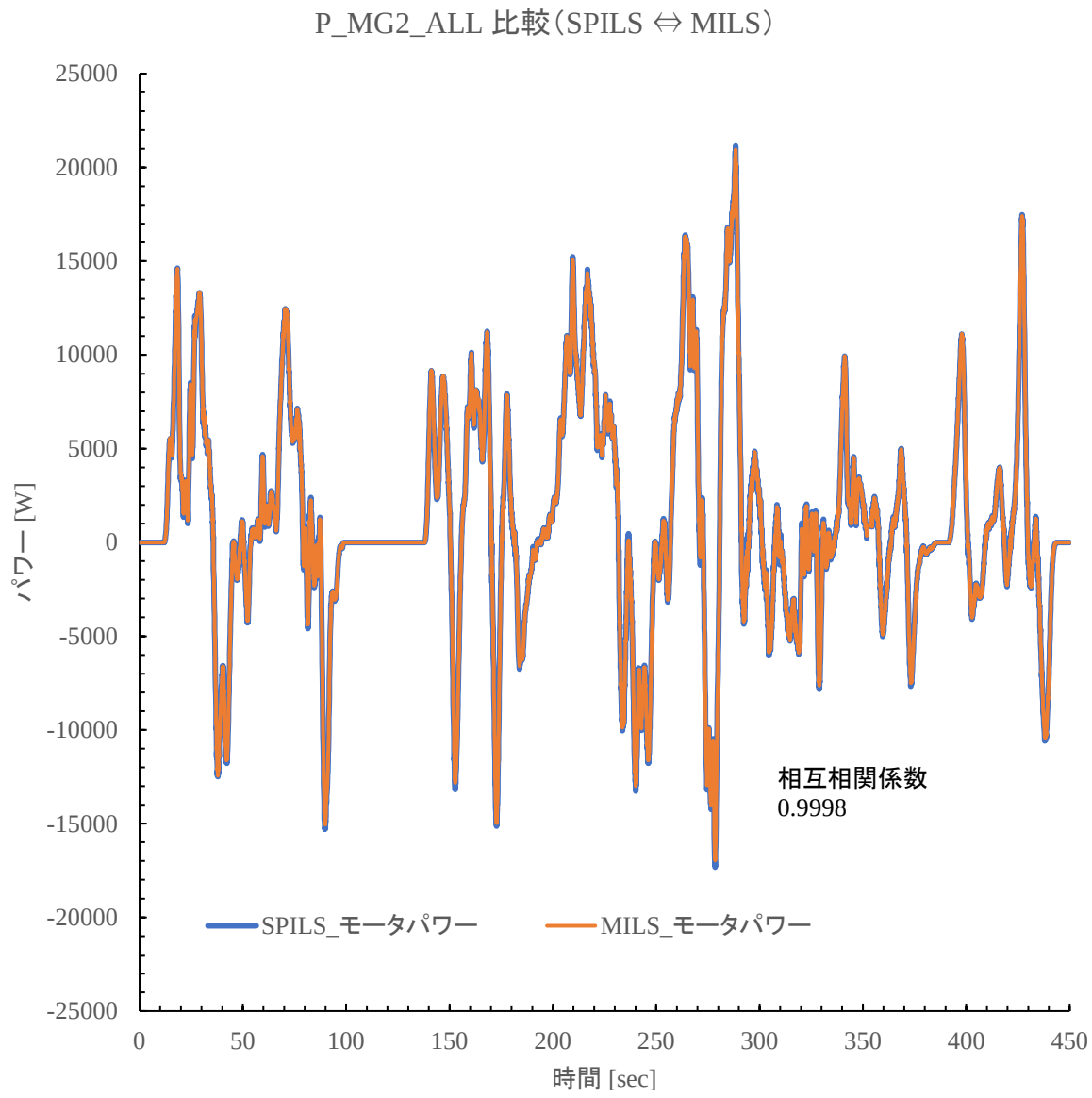


図 10-3. SPILS と MILS 結果の比較（P_MG2_ALL）

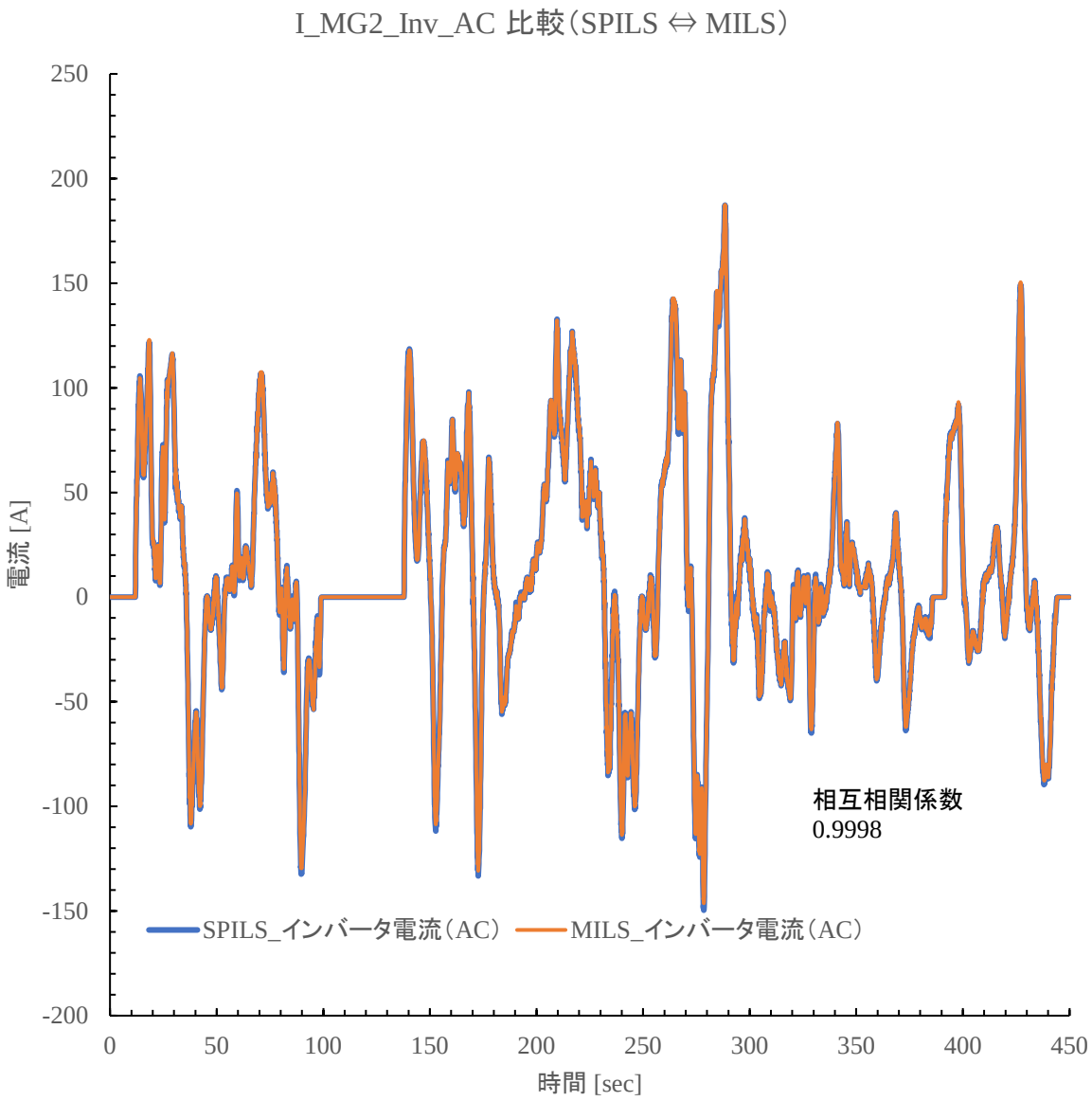


図 10-4. SPILS と MILS 結果の比較（I_MG2_Inv_AC）

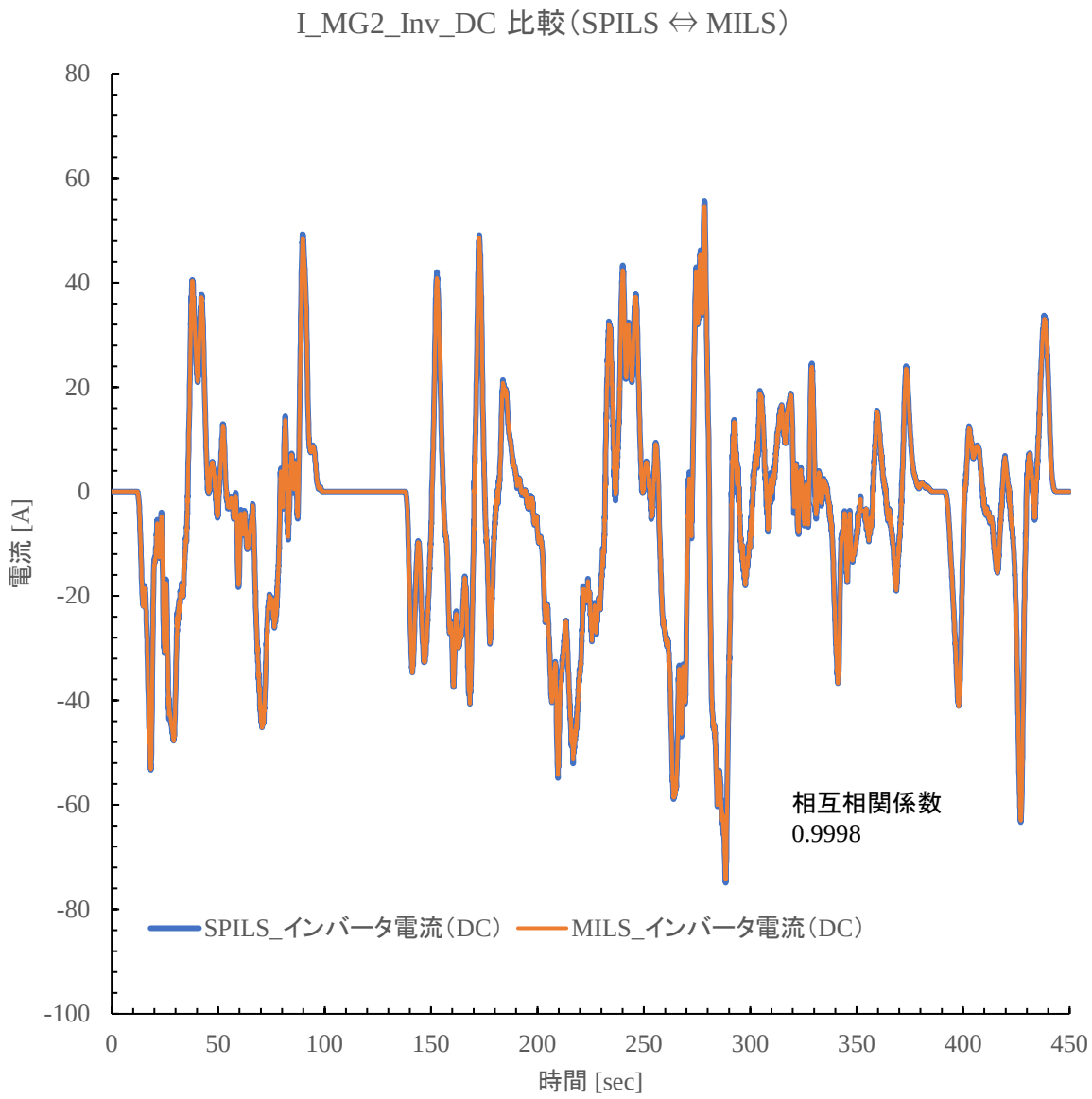


図 10-5. SPILS と MILS 結果の比較（I_MG2_Inv_DC）

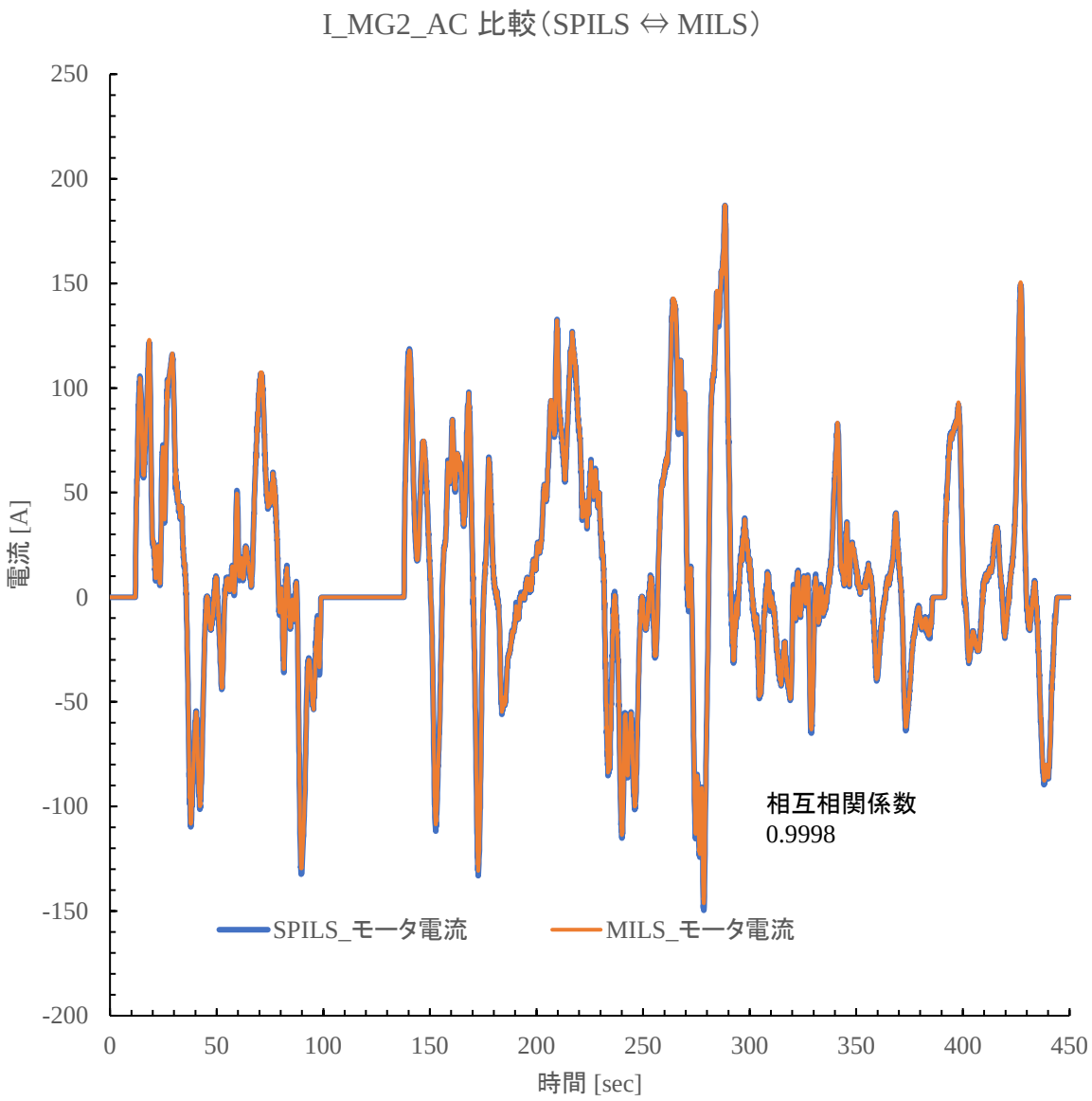


図 10-6. SPILS と MILS 結果の比較（I_MG2_AC）

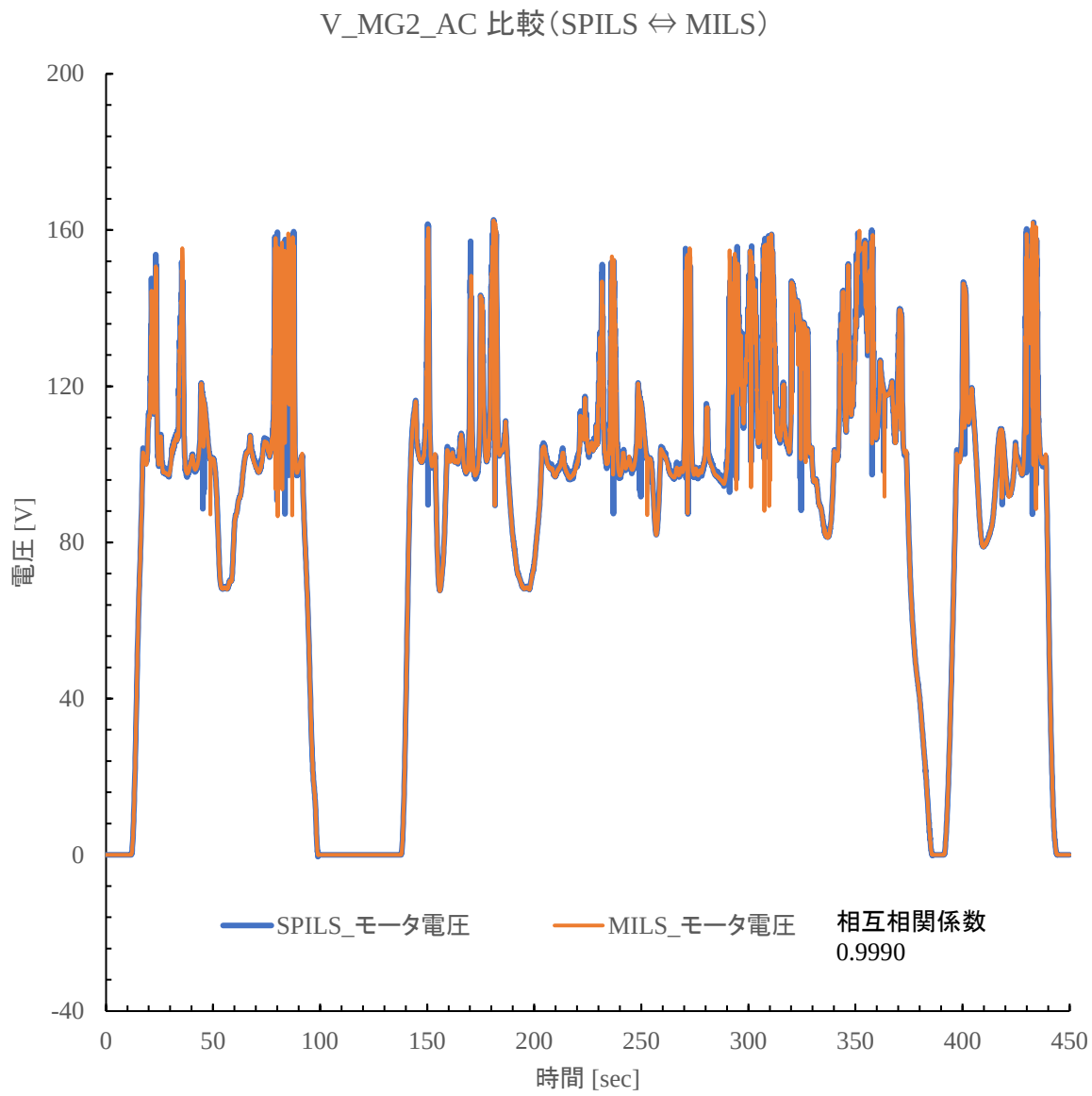


図 10-7. SPILS と MILS 結果の比較（V_MG2_AC）

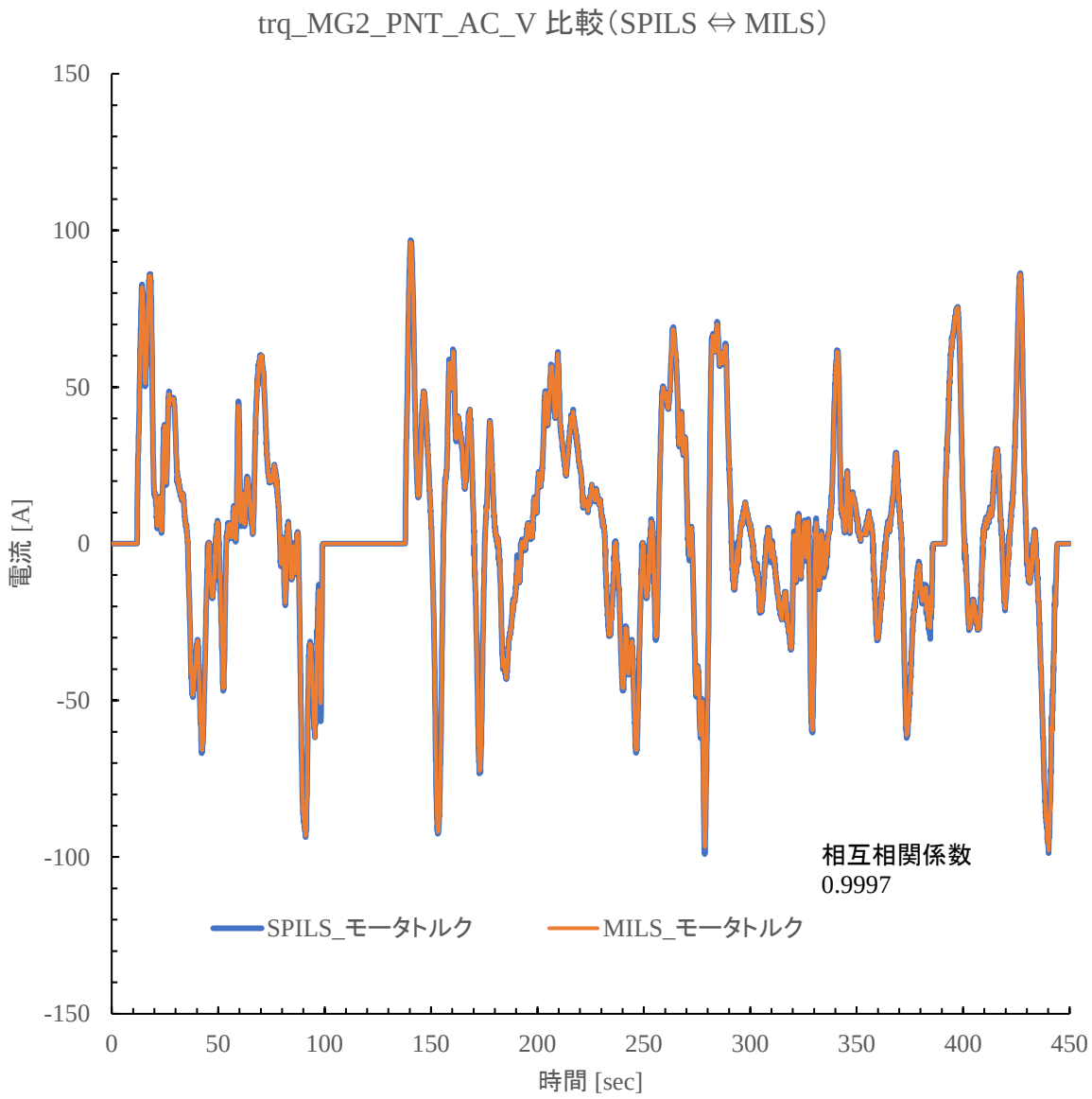


図 10-8. SPILS と MILS 結果の比較（trq_MG2_PNT_AC_V）

11. APPENDIX 3 ： ガイドライン付随 Simulink ライブラリの解説

本ガイドラインに付随する抽象度変換 Simulink ライブラリの全体を図 11-1 に示す。本ライブラリに関する詳細を後述する。

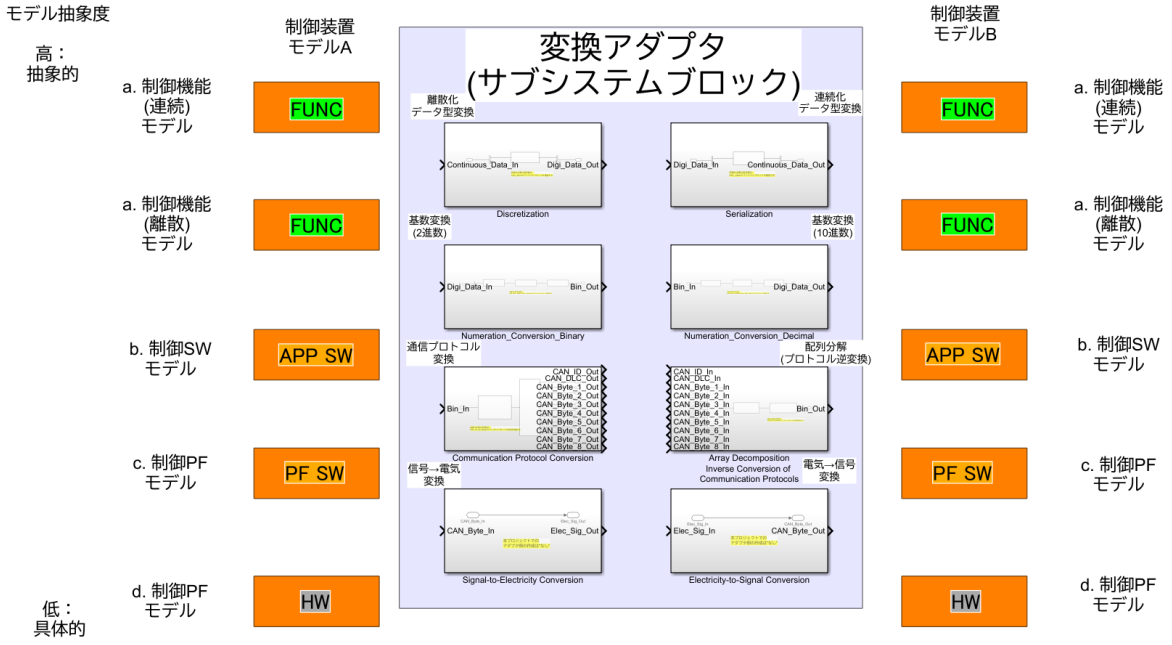


図 11-1. 抽象度変換 Simulink ライブラリ全体図

記載されているアダプタについての説明を図 11-2 に示す。図中赤枠で示すのが高い抽象度のモデルの出力を低い抽象度のモデルに入力する際に使用するアダプタであり、青枠で示すのが低い抽象度のモデルの出力を高い抽象度のモデルに入力する際に使用するアダプタである。なお灰枠で示すアダプタは本ガイドラインの議論対象外であるためアダプタとして用意しているが機能は存在しない。①～⑥の機能は表 11-1 の通りである。

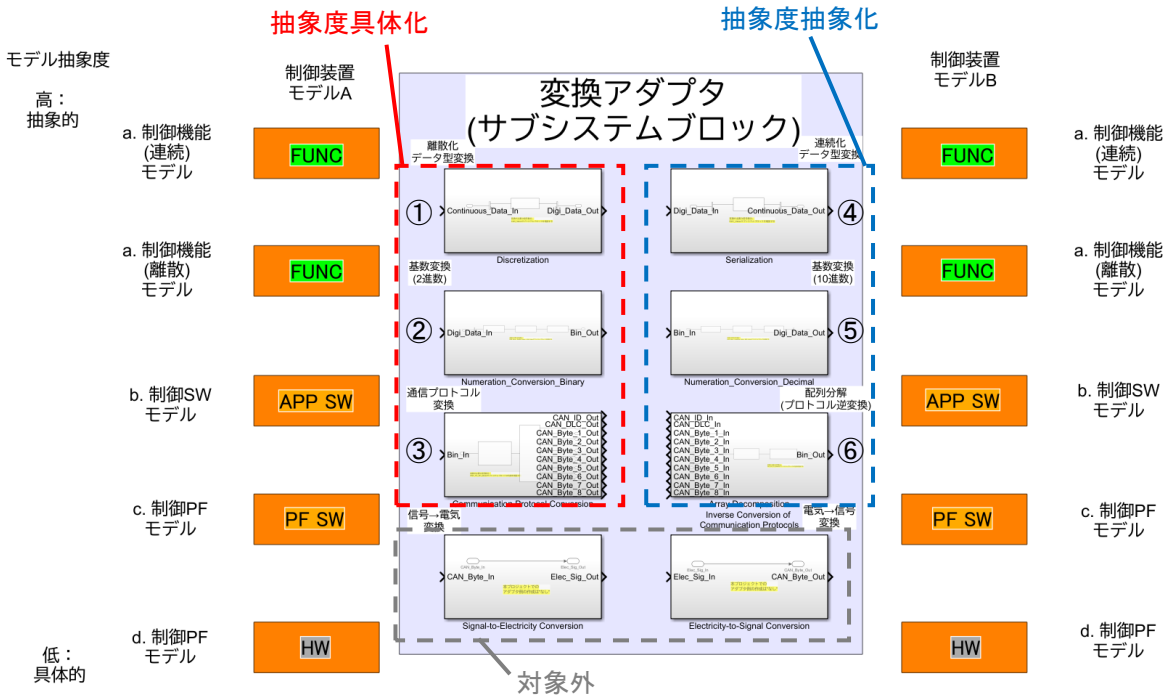


図 11-2. ライブラリのアダプタに関する説明

表 11-1. アダプタ機能の概略

図中番号	名称	機能
①	離散化データ型変換アダプタ	連続データの離散データ変換
②	基数変換(2 進数)アダプタ	デシマルデータのバイナリ変換
③	通信プロトコル変換アダプタ	バイナリデータの CAN 通信 プロトコル生成
④	連続化データ型変換アダプタ	離散データの連続データ変換
⑤	基数変換(10 進数)アダプタ	バイナリデータのデシマル変換
⑥	配列分解(プロトコル逆変換)アダプタ	CAN 通信プロトコルデータから バイナリデータの抽出

図 11-3 は抽象度の高いモデル A の出力を抽象度の低いモデル B の入力に接続する場合を想定して抽象度変換アダプタの使用方法を説明する。制御機能(連続)モデルから制御機能(離散)モデルに抽象度を下げる場合に必要な抽象度変換のアダプタは連続→離散変換のみであるが、制御 SW モデルとの接続の場合は前述した変換に加えてデシマル→バイナリ変換が、制御 PF モデルの場合はさらに CAN プロトコル生成変換が必要になる。

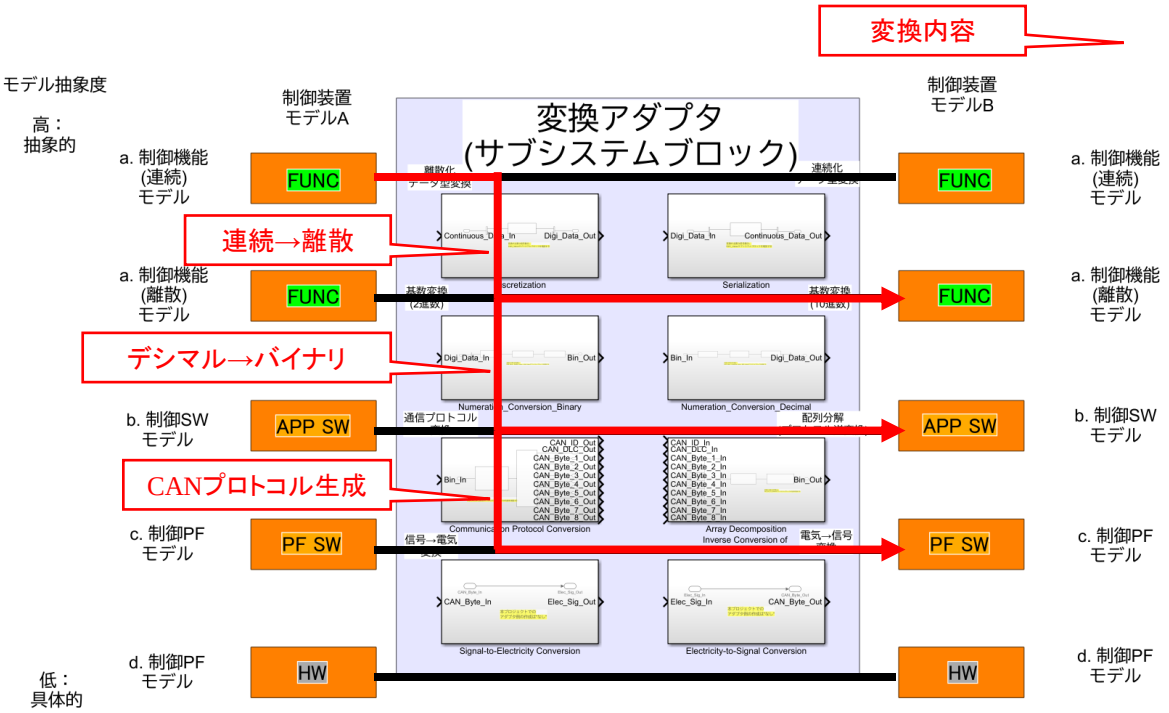


図 11-3. アダプタを用いた抽象度の具体化

図 11-4 は抽象度の低いモデル B の出力を抽象度の高いモデル A の入力に接続する場合を想定して抽象度変換アダプタの使用方法を説明する。制御 PF モデルから制御 SW モデルに抽象度を上げる場合に必要な抽象度変換のアダプタはプロトコルからバイナリデータ抽出の変換のみであるが、制御機能（離散）モデルとの接続の場合は前述した変換に加えてバイナリ→デシマル変換が、制御機能（連続）モデルの場合はさらに離散→連続変換が必要になる。

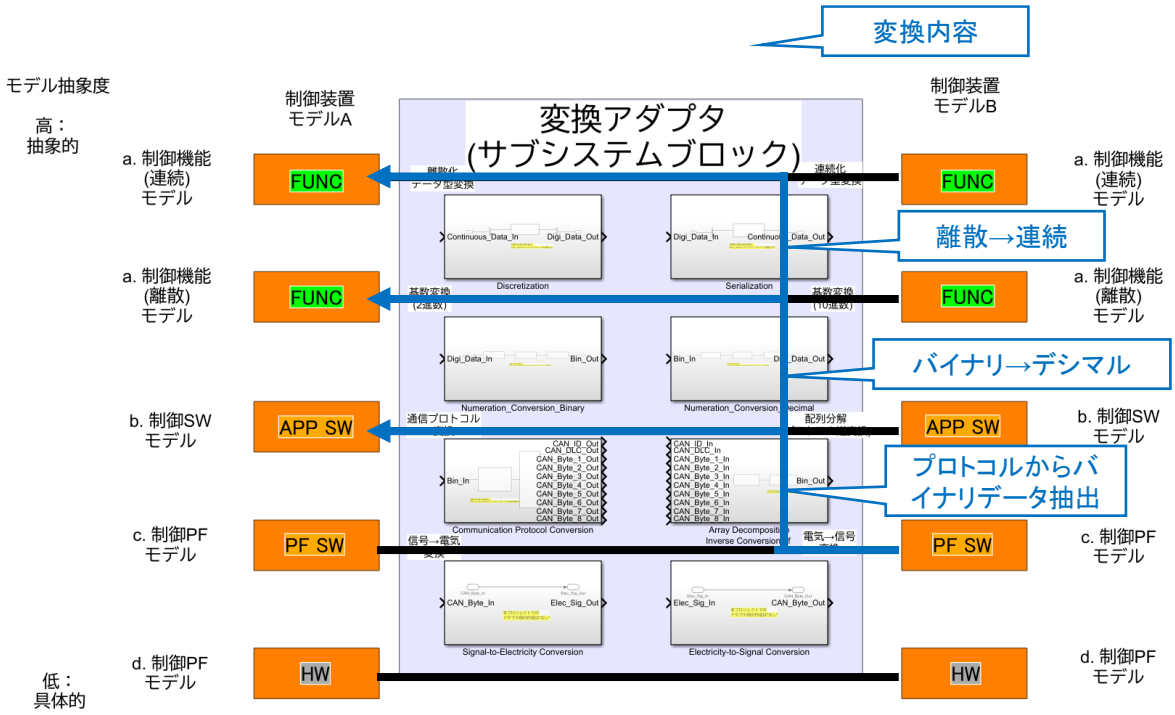


図 11-4. アダプタを用いた抽象度の抽象化

このように異なる抽象度をもつモデル間接続はそれぞれのモデルに当てはまる抽象度の差異によって必要な変換アダプタが決定される。次にそれぞれの変換アダプタの内容や使用方法について説明する。

11.1. 離散化/連続化データ型変換アダプタの使用方法

離散化データ型変換アダプタの内部を図 11-5 に示す。連続データの離散変換は内部に存在する C2D_Value サブシステムが担っており、使用する場合は必要な信号の数だけ入出力バスのポートを作成し、C2D_Value サブシステムを増設する。

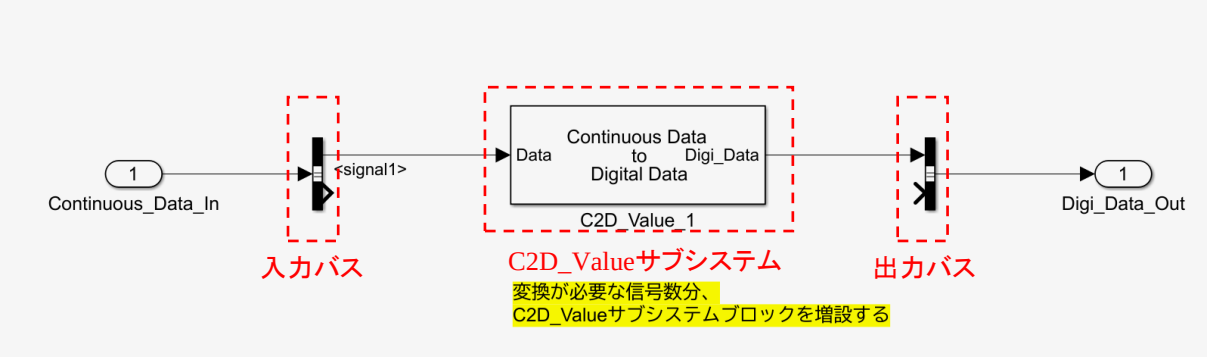


図 11-5. 離散化データ型変換アダプタ内部

C2D_Value サブシステムのパラメータを図 11-6 に示す。各パラメータの意味を表 11-2 に示す。

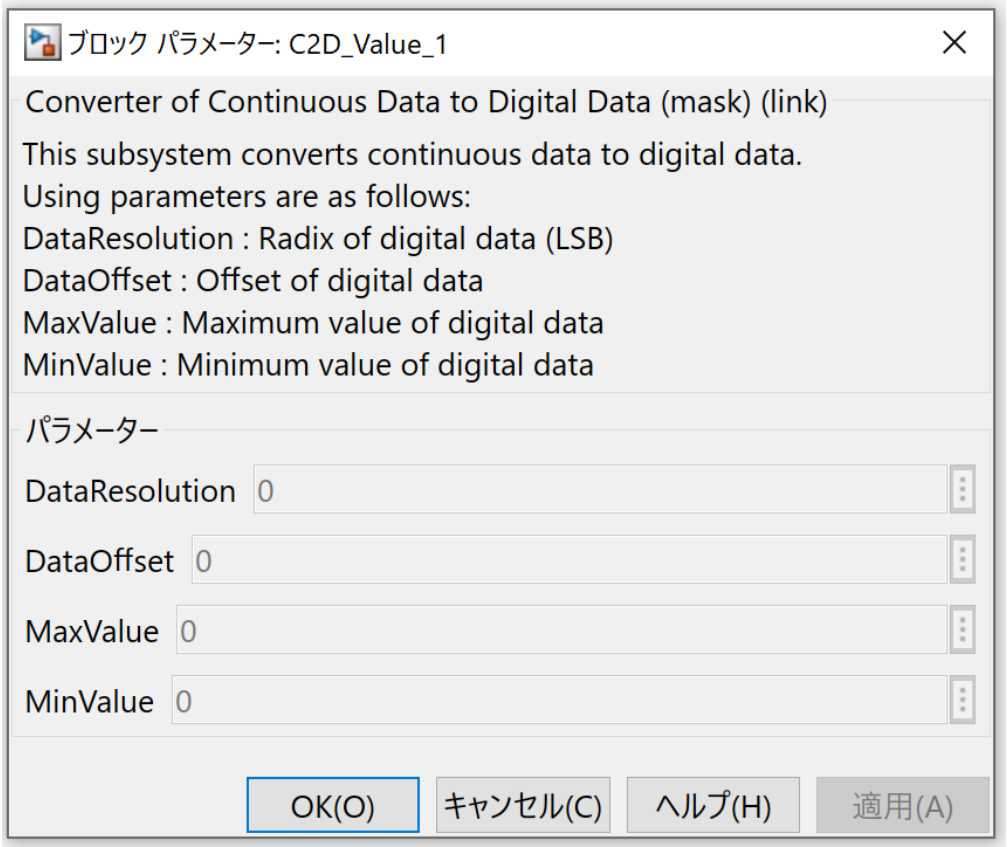


図 11-6. C2D_Value パラメータ

表 11-2. C2D_Value パラメータの意味

パラメータ名	意味
DataResolution	離散化データの分解能（LSB）
DataOffset	離散化データのオフセット量
MaxValue	離散化データの最大値
MinValue	離散化データの最小値

連続化データ型変換アダプタの内部を図 11-7 に示す。連続データの離散変換は内部に存在する D2C_Value サブシステムが担っており、使用する場合は必要な信号の数だけ入出力バスのポートを作成し、D2C_Value サブシステムを増設する。

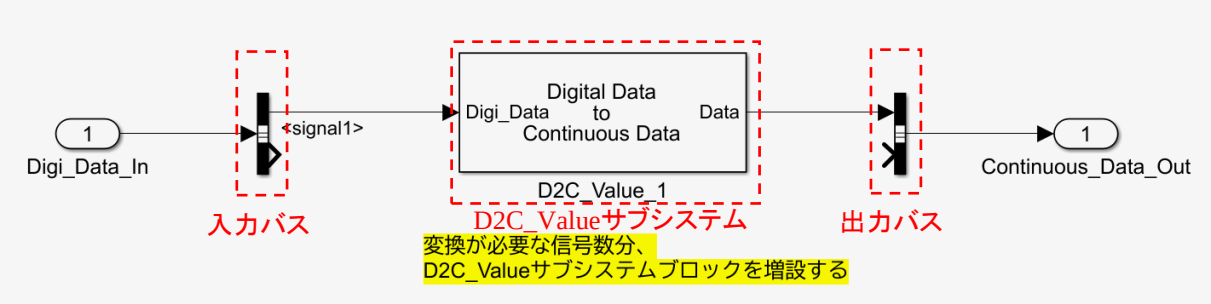


図 11-7. 連続化データ型変換アダプタ内部

C2D_Value サブシステムのパラメータを図 11-8 に示す。各パラメータの意味を表 11-2 に示す。

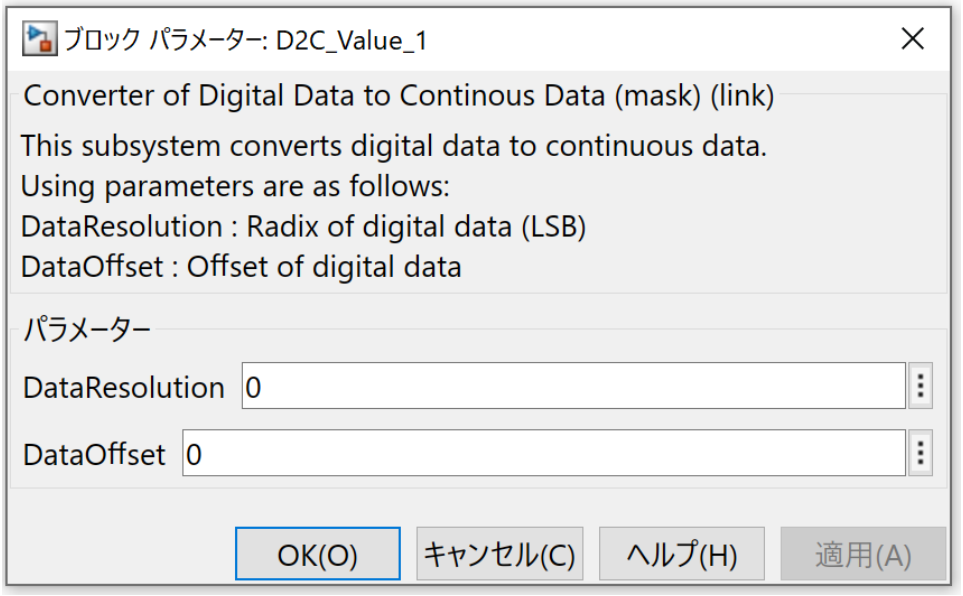


図 11-8. C2D_Value パラメータ

表 11-3. C2D_Value パラメータの意味

パラメータ名	意味
DataResolution	離散化データの分解能（LSB）
DataOffset	離散化データのオフセット量

11.2. 基数変換（2 進数 / 10 進数）アダプタの使用方法

基数変換（2 進数）アダプタの内部を図 11-9 に示す。デシマルからバイナリへの基数変換は内部に存在する D2B_Value サブシステム、EndRef_Value サブシステム、A2N_Value サブシステムが担っており、使用する場合は必要な信号の数だけ入力バスのポートを作成し、上記 3 つのサブシステムと出力ポートを増設する。

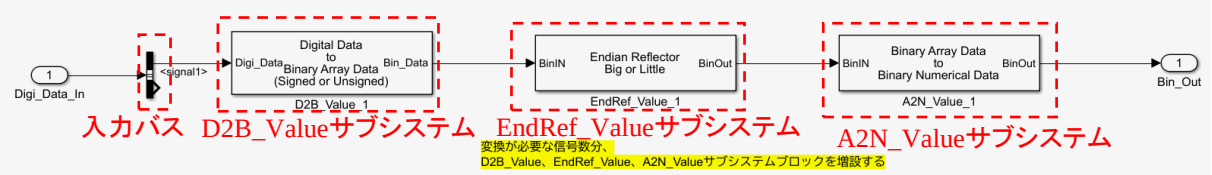


図 11-9. 基数変換（2 進数）アダプタ内部

3 つのサブシステムそれぞれの機能を表 11-4 に示す。

表 11-4. 基数変換(2 進)アダプタ内サブシステム機能

サブシステム名	機能
D2B_Value	入力デシマルデータをバイナリ配列に変換する。
EndRef_Value	入力バイナリ配列を指定したエンディアンに再配列する。
A2N_Value	バイナリ配列データをバイナリ数値データに変換する。

D2B_Value サブシステムのパラメータを図 11-10 に示す。各パラメータの意味を表 11-3 に示す。

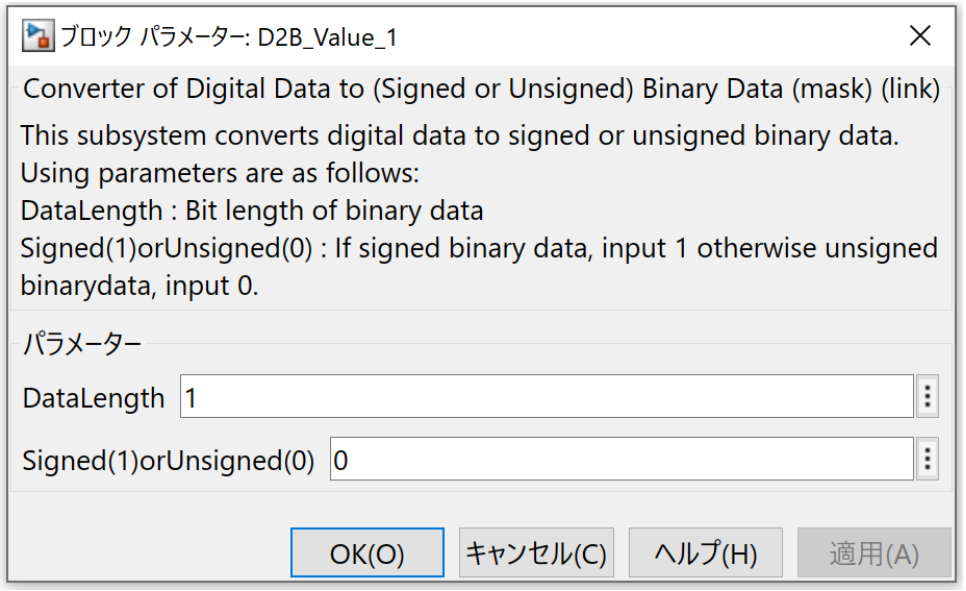


図 11-10. D2B_Value パラメータ

表 11-5. D2B_Value パラメータの意味

パラメータ名	意味
DataLength	データの bit 長[bit]
Signed(1)orUnsigned(0)	データの符号有無(有:1、無:0)

EndRef_Value サブシステムのパラメータを図 11-11 に示す。各パラメータの意味を表 11-6 に示す。

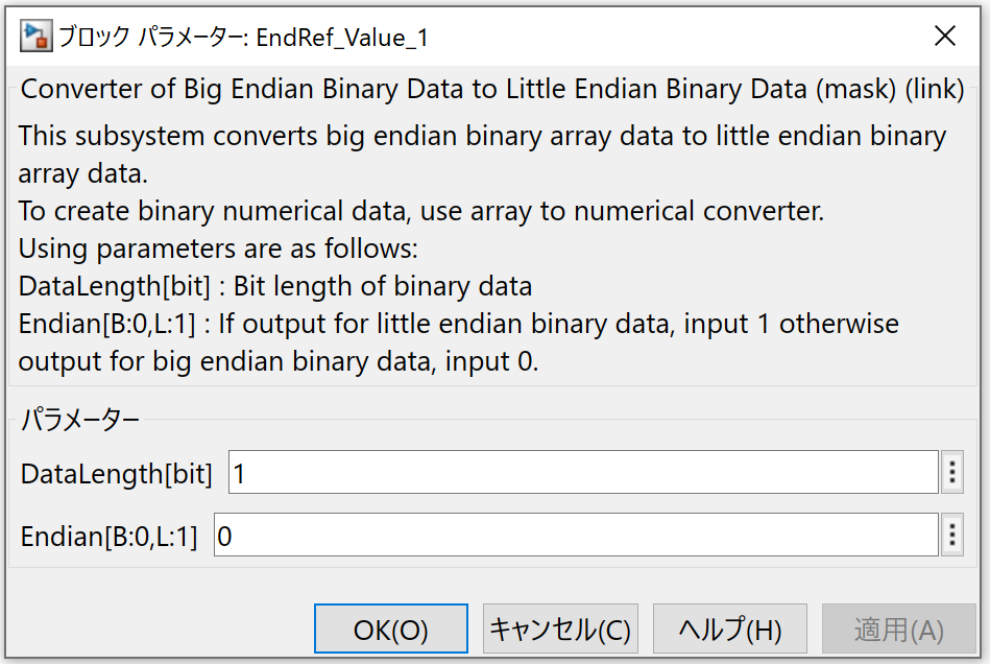


図 11-11. EndRef_Value パラメータ

表 11-6. EndRef_Value パラメータの意味

パラメータ名	意味
DataLength [bit]	データの bit 長[bit]
Endian [B:0, L:1]	エンディアン選択（Big：0、Little：1）

A2N_Value サブシステムのパラメータを図 11-12 に示す。各パラメータの意味を表 11-7 に示す。

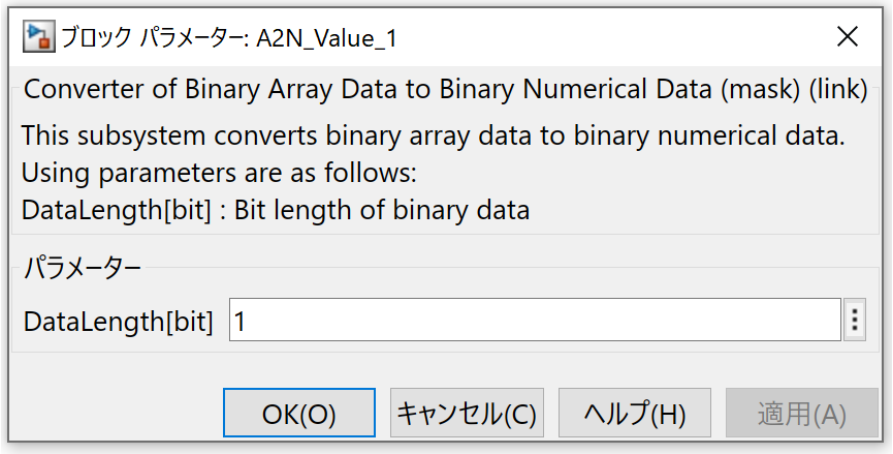


図 11-12. A2N_Value パラメータ

表 11-7. A2N_Value パラメータの意味

パラメータ名	意味
DataLength [bit]	データの bit 長[bit]

基数変換（10 進数）アダプタの内部を図 11-13 に示す。バイナリからデシマルへの基数変換は内部に存在する N2A_Value サブシステム、InvalidEnd_Value サブシステム、B2D_Value サブシステムが担っており、使用する場合は必要な信号の数だけ入力ポートを作成し、上記 3 つのサブシステムと出力バスのポートを増設する。

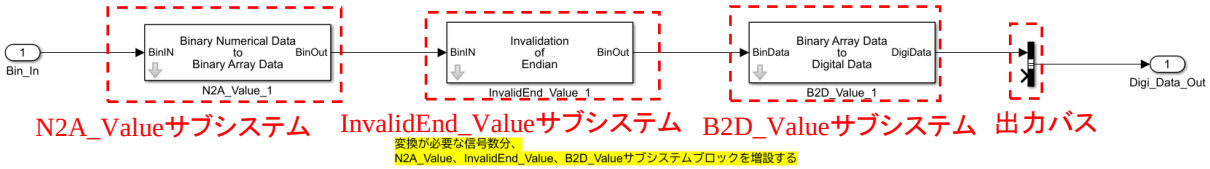


図 11-13. 基数変換（10 進数）アダプタ内部

3 つのサブシステムそれぞれの機能を表 11-8 に示す。

表 11-8. 基数変換（2 進）アダプタ内サブシステム機能

サブシステム名	機能
N2A_Value	バイナリ数値データをバイナリ配列データに変換する。
InvalidEnd_Value	バイナリ配列がリトルエンディアンの場合、ビッグエンディアンに再配列する。
B2D_Value	バイナリ配列データをデシマル数値データに変換する。

N2A_Value サブシステムのパラメータを図 11-14 に示す。各パラメータの意味を表 11-9 に示す。

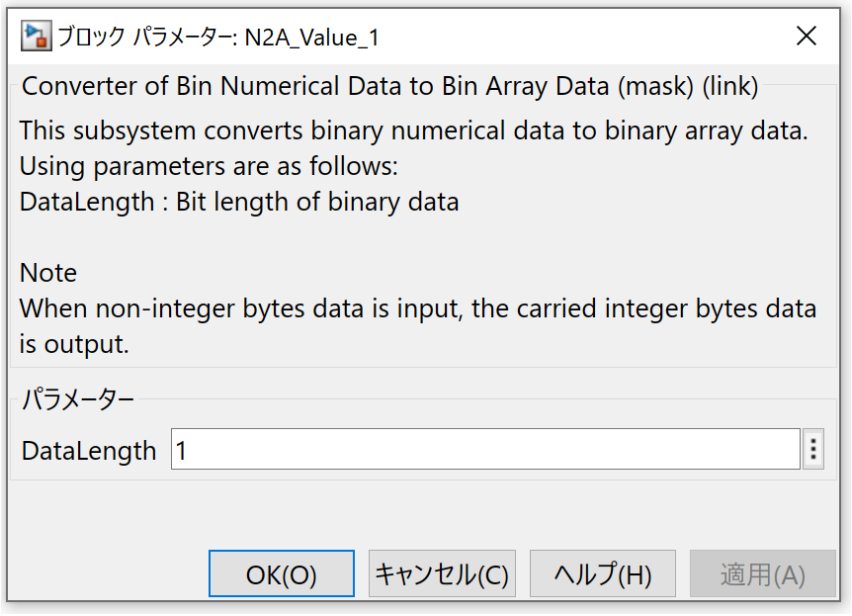


図 11-14. N2A_Value パラメータ

表 11-9. N2A_Value パラメータの意味

パラメータ名	意味
DataLength [bit]	データの bit 長[bit]

InvalidEnd_Value サブシステムのパラメータを図 11-15 に示す。各パラメータの意味を表 11-10 に示す。

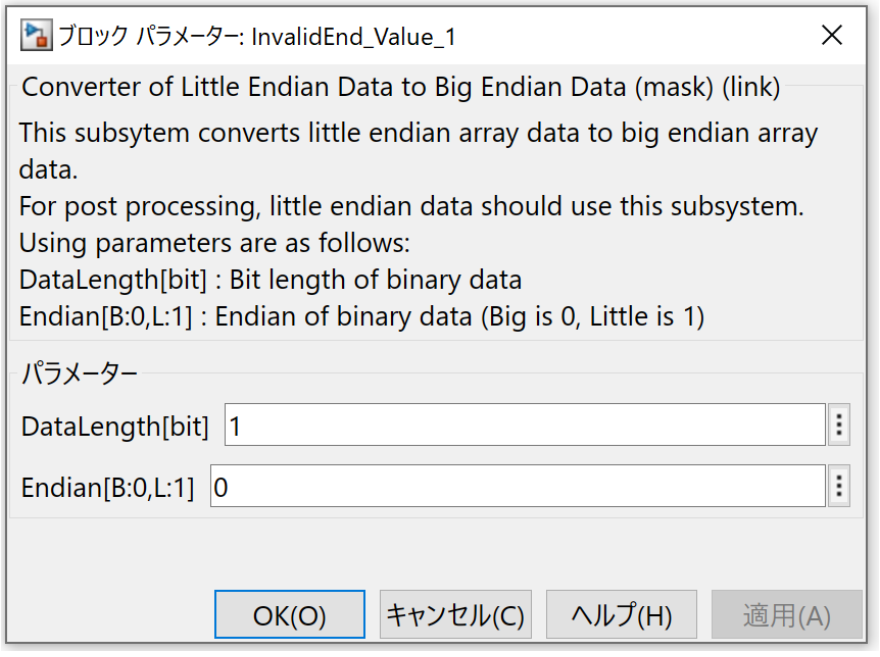


図 11-15. InvalidEnd_Value パラメータ

表 11-10. InvalidEnd_Value パラメータの意味

パラメータ名	意味
DataLength [bit]	データの bit 長[bit]
Endian [B:0, L:1]	エンディアン選択（Big：0、Little：1）

B2D_Value サブシステムのパラメータを図 11-16 に示す。各パラメータの意味を表 11-11 に示す。

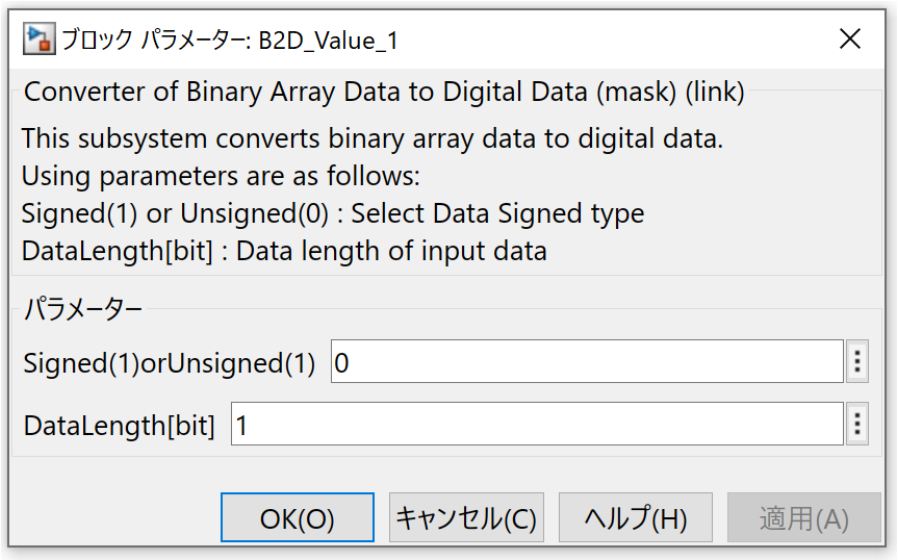


図 11-16. B2D_Value パラメータ

表 11-11. B2D_Value パラメータの意味

パラメータ名	意味
Signed(1)orUnsigned(0)	データの符号有無（有:1、無:0）
DataLength[bit]	データの bit 長[bit]

11.3. 通信プロトコル変換/配列分解アダプタ

通信プロトコル変換アダプタの内部を図 11-17 に示す。ライブラリのアダプタ内部は 2 つのブロックにより構成され、赤枠で示すのが CAN プロトコル生成を担う部分であり灰枠で示すのが SPILS の通信先の I/F に応じて ID、DLC、CAN データフィールドの 1Byte 単位のデータに分解した通信用ブロックである。灰枠の部分は対象とするシミュレータや I/F によって変わることに留意すること。

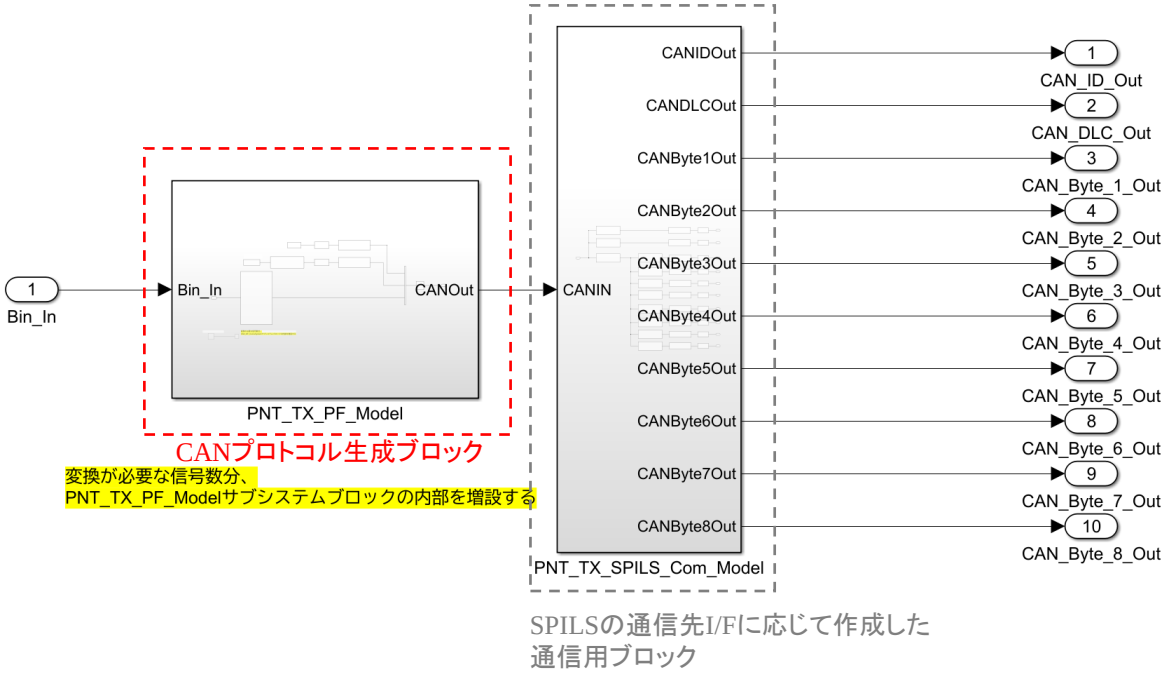


図 11-17. 通信プロトコル変換アダプタ内部

CAN プロトコル生成ブロックである PNT_TX_PF_Model の内部を図 11-18 に示す。実施している内容は CANID、DLC と CAN データフィールドのバイナリ配列を結合し CAN プロトコルを生成するものである。CANID、DLC は Constant ブロックにより定義されており、開発対象に応じて設定すること。

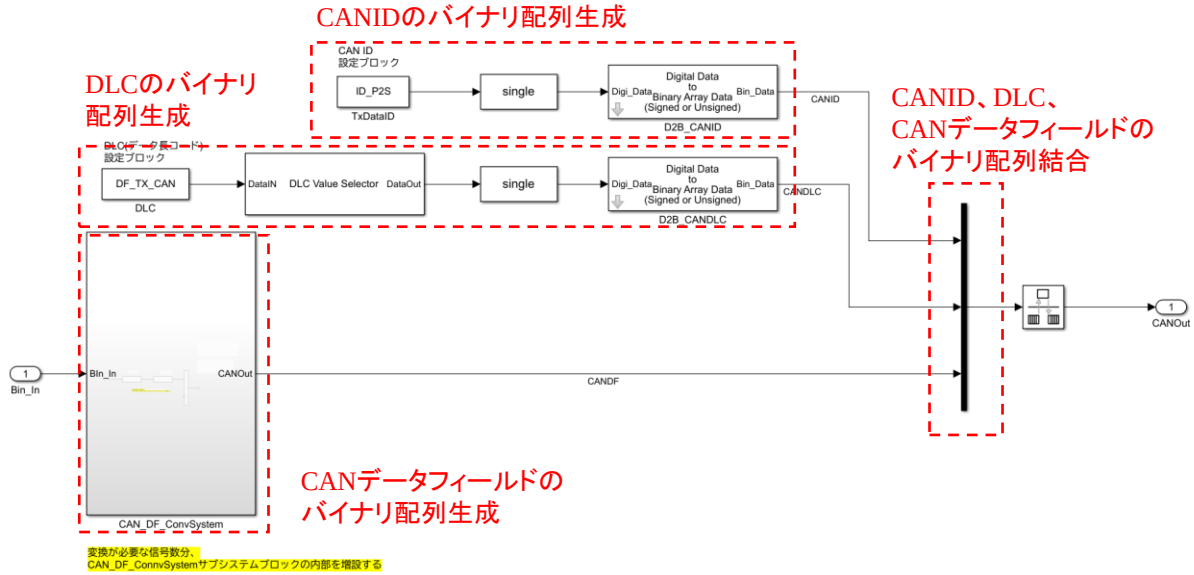


図 11-18. PNT_TX_PF_Model 内部

CAN データフィールドのバイナリ配列生成を行う CAN_DF_ConvSystem の内部を図 11-19 に示す。バイナリ数値から CAN データフィールドのバイナリ配列への変換は内部に存在する N2A_Value サブシステム、A2DF_Value サブシステムが担っており、使用する場合は必要な信号の数だけ入力ポートを作成し、上記 2 つのサブシステムと OR ブロックの入力ポートを増設する。N2A サブシステムの意味やパラメータは表 11-8、図 11-14、表 11-9 を参照のこと。

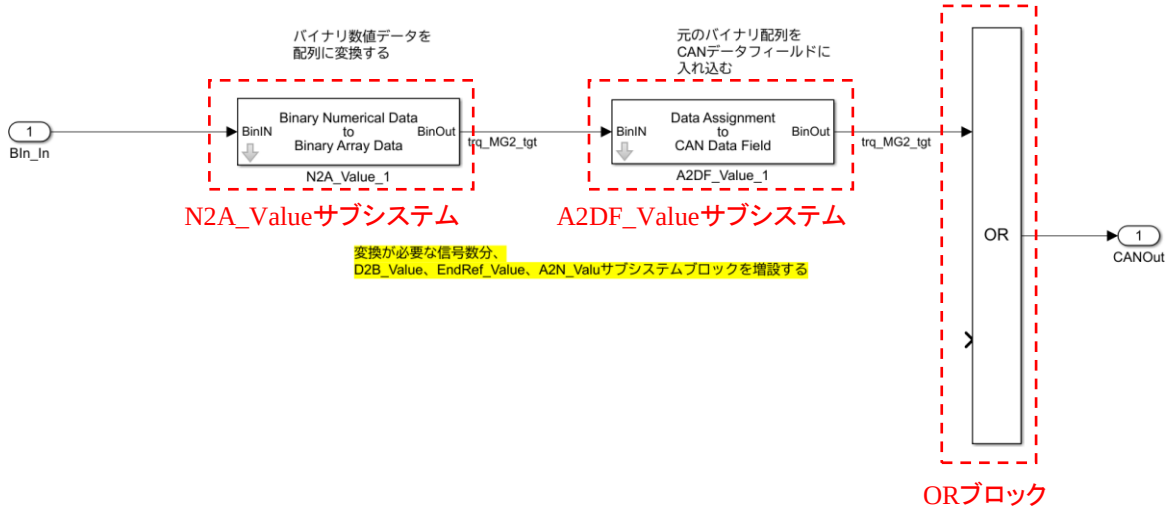


図 11-19. CAN_DF_ConvSystem 内部

A2DF_Value サブシステムの機能を表 11-12 に示す。

表 11-12. A2DF_Value サブシステム機能

サブシステム名	機能
A2DF_Value サブシステム	入力バイナリ配列をデータフィールド内の指定した位置に配置する。

A2DF_Value サブシステムのパラメータを図 11-20 に示す。各パラメータの意味を表 11-13 に示す。

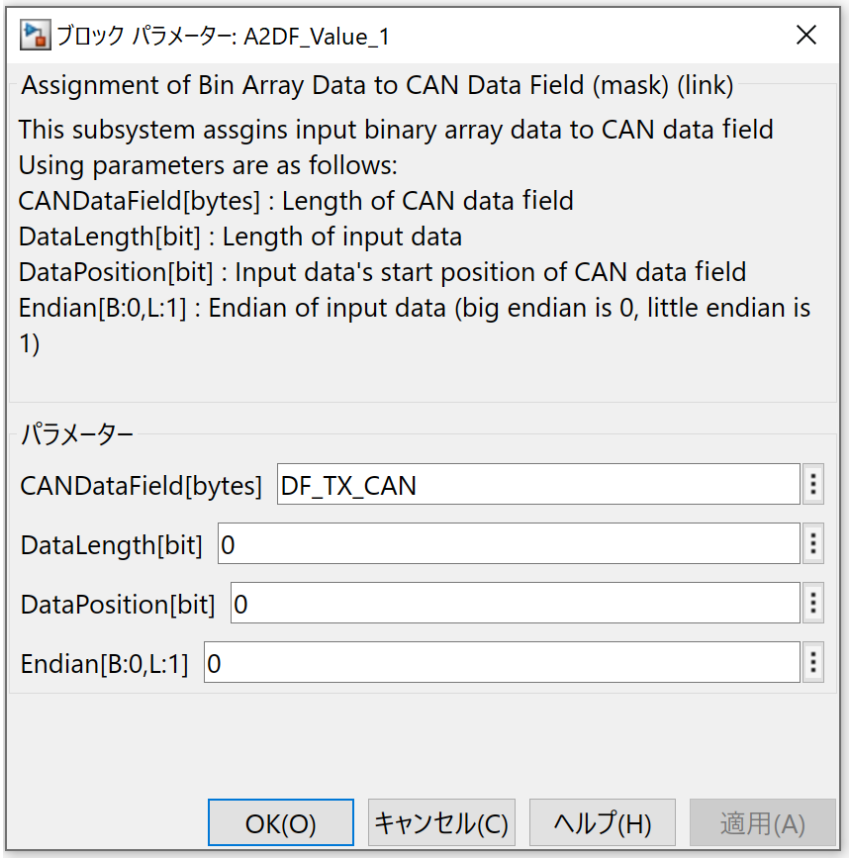


図 11-20. A2DF_Value パラメータ

表 11-13. A2DF_Value パラメータの意味

パラメータ名	意味
CANDataField[bytes]	CAN データフィールド長[bytes]
DataLength[bit]	入力データのデータ長[bit]
DataPosition[bit]	入力データのデータ開始位置[bit]
Endian [B:0, L:1]	エンディアン選択（Big：0、Little：1）

12. 出典

(1) "*No.20002　自動車開発におけるプラントモデル I/F ガイドライン(ver.4.0)*", MBD 推進センター, 3 月, 2022 年,　https://www.jambe.jp/system/download.

(2) "*No.20009　(1)シリーズハイブッド自動車用燃費モデル_Simulink 及び解説書*", MBD 推進センター, 3 月, 2022 年, https://www.jambe.jp/system/download.

(3) "*SmartSE Recommendation V2, Smart Systems Engineering, Simulation Model Exchange Version 2.0*", p7, Figure 7: SmartSE Use Cases around V-Model, Prostep ivip, April, 2018.