Vol.2016-SE-192 No.4
Vol.2016-EMB-41 No.4
2016/6/2

AUTOSARにおける複合デバイスドライバのマイコン依存記述生成手法

廣瀬 秀樹[1]　高瀬 英希[1]　高木 一義[1]　高木 直史[1]

概要：車載ソフトウェア開発の大規模化および複雑化を解消するため，複数の自動車関連企業から成るコンソーシアムによって AUTOSAR（AUTomotive Open System ARchitecture）仕様が策定されている．AUTOSAR 仕様に基づく車載ソフトウェアの開発に際しては，複合デバイスドライバや OS などの ECU ハードウェアに依存する層に位置するコンポーネントをターゲットとなる ECU ハードウェアごとに開発する必要がある．さらに，複合デバイスドライバのコンポーネントの設計情報は，可読性に乏しい ARXML 形式を用いて記述することが規定されている．そこで本研究では，複合デバイスドライバ SW-C において，ターゲットとするマイコンに依存する記述を自動生成するツールを提案する．ツールの使用局面として，既にある ECU ハードウェアで使用され，デバイスドライバフォームファイルが用意されているデバイスを，別の ECU ハードウェアで使用するための開発を想定している．ターゲットとする ECU ハードウェアに依存する箇所は，ドメイン固有言語を用いて抽象化して記述される．提案するツールの適用事例として，RC カーキットおよび RC カー制御プログラムを対象とした自動ブレーキ機能の開発を行った．提案するツールによって，AUTOSAR 仕様に対応した複合デバイスドライバ SW-C を ECU ハードウェアごとに生成することができるようになる．これにより，開発者の負荷が軽減され，車載ソフトウェア開発における生産性の向上が期待される．

Hirose Hideki[1]　Takase Hideki[1]　Takagi Kazuyoshi[1]　Takagi Naofumi[1]

1. はじめに

自動車の電子制御技術の向上により，車載ソフトウェアの高機能化に対する要求が高まっている．さらに，自動車に搭載される ECU（Electronic Control Unit）の数は増加傾向にあり，近年の高級車では搭載される ECU の個数は 140 以上にものぼる [1]．ECU とは，自動車に搭載される電子制御ユニットであり，エンジン，ブレーキおよびヘッドライトの操作をはじめとする自動車に備わる基本機能から，カーナビゲーションやエアコンなどの快適装備まで，あらゆる車載システムの制御を担っている．車載ソフトウェアの高機能化や ECU 数の増加に伴い，車載ソフトウェアの開発も大規模化および複雑化の一途を辿っている．

車載ソフトウェア開発の大規模化および複雑化を解決するため，複数の自動車関連企業から成るコンソーシアムによって AUTOSAR（AUTomotive Open System ARchitecture）仕様 [2] が策定されている．AUTOSAR の目的は，車載ソフトウェアアーキテクチャの標準化および車載ソフトウェアのコンポーネント指向設計の実現である．AUTOSAR では，車載ソフトウェアを複数の層に分割し，それぞれの層内にコンポーネントが配置される構成となっている．アプリケーション機能を実現する層は，ハードウェアに依存しない構造として規定されており，異なる車載システム間で再利用可能である．一方，複合デバイスドライバや OS の一部は，ハードウェアに依存する層に位置する．このため，これらの層に位置するコンポーネントは，使用するハードウェアごとに移植開発する必要がある．

複合デバイスドライバを開発する際には，複合デバイスドライバのコンポーネントの設計情報を記述する必要がある．設計情報の記述には，AUTOSAR で規定されている XML 形式の拡張である ARXML 形式を使用する．しかし，ARXML 形式は可読性に乏しいため，設計情報の記述が非常に煩雑な手順となっている．

そこで本研究では，複合デバイスドライバ SW-C を生成するツールを提案する．デバイスドライバのうち，動作周波数やピン配置などの ECU ハードウェアの構成に依存

[1] 京都大学
Kyoto University

しない記述のみが実装されたファイル（デバイスドライバフォームファイル）とマイコン依存の情報を記述したファイルを統合することで，複合デバイスドライバ SW-C を生成する．さらに，既存の車載システムへ追加したい SW-C の設計情報をドメイン固有言語（DSL）によって記述し，統合元のシステムで使用されている ARXML ファイルと共にツールに入力することで，追加する SW-C に関する記述を付加した ARXML ファイルを生成する．提案手法が有用となる使用局面としては，既にある ECU ハードウェアで使用され，デバイスドライバフォームファイルが用意されているデバイスを，別の ECU ハードウェアで使用するための開発が想定される．開発する際に開発者が用意する必要があるのは，ECU ハードウェア依存情報記述ファイルのみである．ECU ハードウェアに依存する情報を読み書きの容易な DSL を使用して記述するだけで，AUTOSAR の仕様に対応した複合デバイスドライバ SW-C を得ることができる．提案手法は，開発した SW-C を既存の車載システムに統合する場合にも有用である．開発者が用意する必要があるのは，追加する SW-C の設計情報記述ファイルのみである．開発者が自ら ARXML ファイルを編集する必要がなくなるため，複合デバイスドライバの開発にあたっての労力が軽減されることが期待できる．

複合デバイスドライバ SW-C 生成ツールの適用事例として，RC カーキットおよび RC カー制御プログラムを対象とした自動ブレーキ機能の開発を行う．自動ブレーキ機能とは，センサによって取得した前方の障害物との距離が一定の値を下回ると自動的にブレーキをかけ，車両を停止させる機能である．複合デバイスドライバ SW-C 生成ツールを python を用いて実装し，距離を取得するための測距センサのデバイスドライバフォームファイルを C 言語を用いて開発した．次に，HSBRH850F1L100 と DE0-Nano の 2 種類の ECU ハードウェアを対象に ECU ハードウェア依存情報記述ファイルを作成した．作成したファイルをツールに入力し，それぞれの ECU ハードウェアを対象とした測距センサの複合デバイスドライバ SW-C を生成した．さらに，測距センサから得た距離に基づき自動ブレーキを行うアプリケーション SW-C を開発した．測距センサの複合デバイスドライバ SW-C および自動ブレーキのアプリケーション SW-C の設計情報を DSL で記述したファイルを作成し，ツールを使用して既存の ARXML ファイルと統合した．統合した車載システムを RC カーキット上で実行し，正常に動作することを確認した．

本稿の構成は次の通りである．第 2 章では，AUTOSAR と AUTOSAR における車載ソフトウェアの開発フローについて説明する．第 3 章では，提案する複合デバイスドライバ SW-C 生成ツールについて述べる．第 4 章では，RC カーキットを用いた，複合デバイスドライバ SW-C 生成ツールの適用事例を紹介する．第 5 章では，本稿のまとめを述べる．

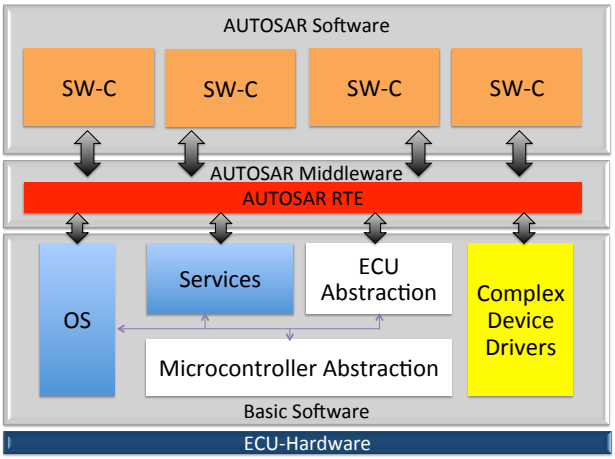

図 1 AUTOSAR の定める ECU アーキテクチャ

2. AUTOSAR

2.1 AUTOSAR で定められた車載ソフトウェアの構造

AUTOSAR で規定されている ECU アーキテクチャは，**図 1** に示したように，AUTOSAR ソフトウェア層，AUTOSAR ミドルウェア層，基盤ソフトウェア層，ECU ハードウェアの 4 つの層から構成される．

AUTOSAR では，それぞれの層内のコンポーネントに対するインタフェースも規定されている．SW-C 同士は，AUTOSAR RTE を介して「ポート」によって他の SW-C と接続される．SW-C 間の通信には，Sender-Receiver 連携と Client-Server 連携の 2 種類のプロトコルが存在する．Sender-Receiver 連携は，ある SW-C から別の SW-C への何らかのデータの送受信をサポートする．このとき，一対多あるいは多対一の非同期通信によってやり取りする．Client-Server 連携は，ある SW-C から別の SW-C が提供するサービスへの呼出しをサポートする．多対一の同期通信によって実現している．

AUTOSAR は，コンポーネント指向開発の概念を取り入れている．すなわち，AUTOSAR では車載ソフトウェアの開発はコンポーネントごとの開発およびコンポーネント間のインタフェースの開発が基本となる．さらに，上位に位置するコンポーネントの開発段階において，ECU ハードウェアや ECU 間のネットワークの構造といったハードウェアの構成情報を考慮する必要がない．このことによって，コンポーネントの改良や異なる製品での再利用が容易となり，開発コストの削減が可能となる [3]．

AUTOSAR ソフトウェア層，AUTOSAR ミドルウェア層，基盤ソフトウェア層および ECU ハードウェアについてそれぞれ説明する．

AUTOSAR ソフトウェア層

ECU アプリケーションは規定のインタフェースを

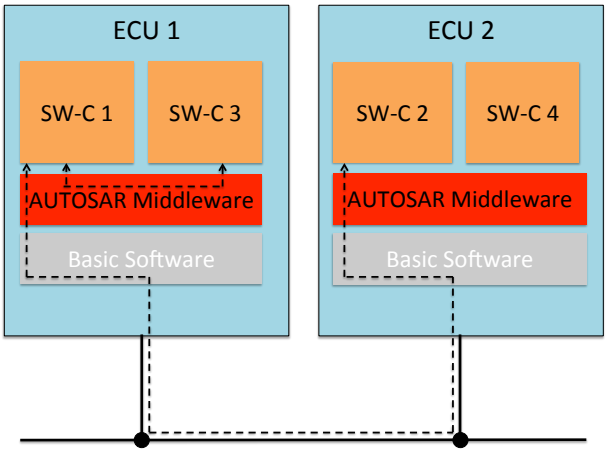

図 2 ECU 同士の接続関係

持ったソフトウェアコンポーネント（SW-C）として存在する．これらの SW-C は AUTOSAR ソフトウェア層に位置する．それぞれの SW-C は，ECU の仕様情報やネットワークのトポロジなどの情報をもとに，**図 2** のように各 ECU に割り当てられる．SW-C 同士は，ポートを通して他の SW-C と接続される．SW-C は複数の「ランナブル」と呼ばれる単位であるアプリケーション処理から構成され，起動周期や排他制御はランナブルごとに設定される．AUTOSAR ミドルウェア層より下位に存在するハードウェア部分はすべて抽象化されて AUTOSAR ソフトウェア層に提供されるため，AUTOSAR ソフトウェア層がハードウェアに依存しない構造となっている．

AUTOSAR ミドルウェア層

AUTOSAR ソフトウェア層と基盤ソフトウェア層をつなぐミドルウェア層であり，AUTOSAR RTE（Run-Time Environment）が含まれる．SW-C と基盤ソフトウェア層の間の通信や，SW-C 同士の通信はすべて AUTOSAR RTE を通して行われる．前述の通り，SW-C 同士はポートにより接続されている．AUTOSAR RTE は，ポートを通して行われる SW-C 間の通信が，同一 ECU 内の通信であるか別の ECU との通信であるかに関わらず，統一された通信インタフェースを AUTOSAR ソフトウェア層に提供する．すなわち，図 2 において，SW-C 1 から SW-C 3 への通信（同一 ECU 内の通信）も SW-C 1 から SW-C 2 への通信（別の ECU との通信）も，SW-C 1 にとっては同一の通信方法をとっているように見える．

基盤ソフトウェア層

ECU ハードウェアを抽象化して AUTOSAR ミドルウェア層に提供する層である．OS 層，サービス層，ECU 抽象化層，マイコン抽象化層および複合デバイスドライバ層による階層構造を成しており，各モジュールもさらに細かいコンポーネントから構成されている．基盤ソフトウェア層の中でも上位に位置するサービス層は，ネットワークの管理やエラー診断機能，入出力管理などの基本機能を提供する．複合デバイスドライバ層は，マイコン抽象化層を経由せずにハードウェアへの接続することができるため，周辺装置をアプリケーションで直接使用したり，処理速度向上のために独自モジュールを導入したりする場合に用いられる．OS は，スケーラビリティクラスとして SC1 から SC4 の 4 つのクラスが定義されており，各クラスで備えているべき機能が異なっている．

ECU ハードウェアに直接アクセスしないコンポーネントに関しては，SW-C と同様に，異なる製品での再利用が可能である．一方，マイコン抽象化層，OS および複合デバイスドライバ層に含まれるコンポーネントは ECU ハードウェア依存であるため，使用する ECU ハードウェアに応じて開発する必要がある．

ECU ハードウェア

マイコンボードや物理デバイスなどの，ECU を物理的に構成するコンポーネントを指す．

2.2 AUTOSAR における車載ソフトウェアの開発フロー

AUTOSAR に基づく車載ソフトウェアの開発フローは次の通りである [4].

(1) SW-C ディスクリプション
SW-C によって実現する機能を整理し，SW-C 同士の接続関係を設計する．

(2) システムディスクリプション
配置する ECU の定義，各 ECU で動作させる SW-C の定義などのシステム全体の定義を行う．

(3) ECU コンフィギュレーション
使用する ECU ごとに，基盤ソフトウェアと RTE を生成するために必要な情報を整理する．

(4) ECU インテグレーション
ECU 依存の初期化処理などの実装，基盤ソフトウェアと RTE の生成およびプログラムのビルドを行う．

システム全体の設計情報や，ECU コンフィギュレーションで整理した情報は，すべて AUTOSAR の規定する XML 形式（ARXML 形式）で記述する必要がある．**図 3** に，ARXML 形式でのシステムディスクリプションファイルの記述例を示す．AUTOSAR の定めるタグを使用して，SW-C のポート，ランナブル，SW-C 間インタフェースなどの設計情報を記述している．

3. 複合デバイスドライバ SW-C 生成ツール

3.1 全体像

複合デバイスドライバ開発における問題点を解決するため，複合デバイスドライバ SW-C のマイコンに依存する記

```
<COMPLEX-DEVICE-DRIVER-SW-COMPONENT-TYPE>
  <SHORT-NAME>CddRadarControl</SHORT-NAME>
    <PORTS>
      <P-PORT-PROTOTYPE>
      <SHORT-NAME>DistanceSrv</SHORT-NAME>
      <PROVIDED-INTERFACE-TREF DEST="CLIENT-SERVER-INTERFACE">
        /RcCar/IfDistance
      </PROVIDED-INTERFACE-TREF>
      </P-PORT-PROTOTYPE>
    </PORTS>
    <INTERNAL-BEHAVIORS>
      <SWC-INTERNAL-BEHAVIOR>
        ......
      <RUNNABLES>
        <RUNNABLE-ENTITY>
          <SHORT-NAME>MeasureDistance</SHORT-NAME>
          <MINIMUM-START-INTERVAL>0.0</MINIMUM-START-INTERVAL>
          <CAN-BE-INVOKED-CONCURRENTLY>false</CAN-BE-INVOKED-CONCURRENTLY>
          <SYMBOL>MeasureDistance</SYMBOL>
        </RUNNABLE-ENTITY>
      </RUNNABLES>
    </SWC-INTERNAL-BEHAVIOR>
  </INTERNAL-BEHAVIORS>
</COMPLEX-DEVICE-DRIVER-SW-COMPONENT-TYPE>
```

図 3 ARXML 形式による記述

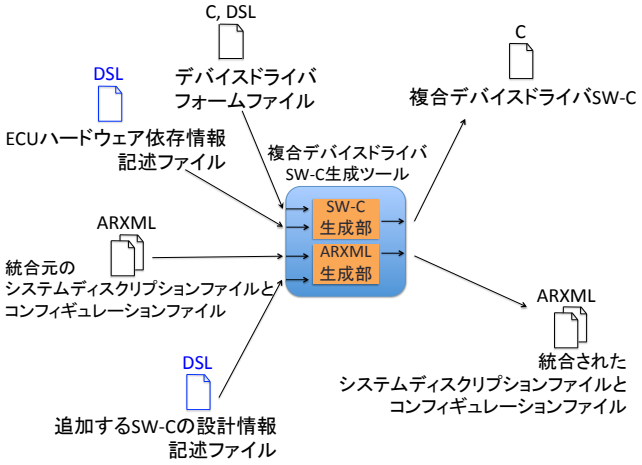

図 4 複合デバイスドライバ SW-C 生成ツール

デバイスドライバフォームファイル

```
<driver.c>
#include “driver.h”

void sensor_sense(void){
    ...
    pulse = @READ(PWM);
    ...
}

void sensor_trigger(void){
    ...
    @WRITE(C_T);
    wait(CYCLES);
}

<driver.h>
#define CYCLES @FREQ / 1000
#define C_T @COMP_TRIG
#define PWM @PWM
```

ECUハードウェア依存情報記述ファイル

```
/* HSB-RH850F1L100 */
FREQ : 80000000$$
COMP_TRIG : 0x0001$$
PWM : 0x0002$$
READ : sil_reh_mem$$
WRITE : sil_wrh_mem$$
```

```
/* DE0-Nano */
FREQ : 50000000$$
COMP_TRIG : 0x0021$$
PWM : 0x0022$$
READ : sil_rew_iop$$
WRITE : sil_wrw_iop$$
```

図 5 SW-C 生成部に入力するファイルの記述例

述を自動生成するツールを提案する．

図 4 に，複合デバイスドライバ SW-C 生成ツールの全体像を示す．生成ツールへの入力として，デバイスドライバフォームファイル，ECU ハードウェア依存情報記述ファイル，統合元のシステムディスクリプションファイルとコンフィギュレーションファイルおよび追加する SW-C の設計情報記述ファイルを与える．ツールの SW-C 生成部では，デバイスドライバフォームファイルに ECU ハードウェア依存情報記述ファイルによって与えられたマイコン依存の情報を統合することで，複合デバイスドライバ SW-C を生成する．ARXML 生成部では，統合元の ARXML ファイルに，複合デバイスドライバ SW-C の情報を書き加えた ARXML ファイルを生成する．ツールを使用する際に開発者が用意する必要があるのは，ECU ハードウェア依存情報記述ファイルおよび追加する SW-C 設計情報記述ファイルのみである．

3.2 SW-C 生成部

図 5 に，デバイスドライバフォームファイルおよび ECU ハードウェア依存情報記述ファイルの記述例を示す．

デバイスドライバフォームファイルとは，デバイスドライバのうち，動作周波数やピン配置などの ECU ハードウェアの構成に依存しない記述のみが実装されたファイルである．センサやアクチュエータなどのデバイスごとに既に用意されていることを前提としている．ECU ハードウェアに依存する部分は，`@XXXX` のような抽象表現を用いて記述される．ECU ハードウェアに依存しない部分は，C 言語をはじめとする，統合元のシステムで使用されている言語で記述される．ECU ハードウェアに依存する記述を最小限にするため，タイマや PWM 制御などのマイコン固有の機

```
#DESCRIPTION
##INTERFACE
IfDistance:
 Protocol: ClientServer
 Operations:
  OpGetIfDistance:
   Args:
    command:
     Type: IDT_Distance
     Direction: OUT$$
##SWCS
###Applications
###ComplexDeviceDrivers
CddRadarControl:
 P-Ports:
  DistanceSrv:
   Interface: IfDistance
 Internal:
  ...
  MeasureDistance:
   MinimumStartInterval: 0.0
   CanBeInvokedConcurrently: false$$
##CONNECTORS
CddRadarControl_DistanceSrv_to_AutoBrakeManager_DistanceClt:
 Provider: CddRadarControl/DistanceSrv
 Requester: AutoBrakeManager/DistanceClt$$
#CONFIGURATION
CddRadarControl:
 MeasureDistance:
  Event: DistanceSrv_OpGetIfDistance$$
```

図 6 追加する SW-C の設計情報記述ファイルの記述例

能は使用せず，汎用的な機能のみを使用して実装される．

ECU ハードウェア依存情報記述ファイルには，プロセッサの動作周波数やピン配置などの，使用する ECU ハードウェアごとに固有の値を DSL を用いて記述する．デバイスドライバフォームファイル中の抽象表現`@XXXX` に対応するコードを，

```
XXXX : <code>$$
```

のように，区切り記号:と終了記号`$$`を使用して記述する．一つの抽象表現に対して，複数行のコードを記述することも可能である．

提案手法の SW-C 生成部が有用となる使用局面としては，デバイスドライバフォームファイルが用意されているデバイスを，新たな ECU ハードウェアに移植開発する場合が想定される．複合デバイスドライバ SW-C を開発するために開発者が用意する必要があるのは，ECU ハードウェア依存情報記述ファイルのみである．提案手法は，実装済みの複合デバイスドライバ SW-C が存在し，そこから新たな ECU ハードウェアに移植する場合にも有用である．その場合は，開発者が実装済みのものを DSL 記述によって修正することで，新たなデバイスドライバフォームファイルを用意することも可能である．

3.3 ARXML 生成部

図 6 に，追加する SW-C の設計情報記述ファイルの記述例を示す．赤字で示した部分は，図 3 に示した ARXML による設計情報の記述に対応する．システムディスクリプションファイルおよびコンフィギュレーションファイルに追記する情報を，それぞれディスクリプション部およびコンフィギュレーション部に分けて記述する．

表 1 ディスクリプション部に記述する項目

項目	設計情報
Protocol	SW-C 間インタフェースが提供する SW-C 間通信のプロトコルを指定する
Operations	ClientServer 連携の SW-C 間インタフェースを介して呼出すオペレーションを列挙する
Args	オペレーションとともにやり取りするデータを列挙する
Type	データの型を指定する
Direction	データの方向を指定する
P-Ports	SW-C のデータ送信用ポートを列挙する
R-Ports	SW-C のデータ受信用ポートを列挙する
Interface	ポートが接続する SW-C 間インタフェースを指定する
Internal	SW-C 内部の構成情報を列挙する
Provider	データ送信側の SW-C およびポートを /で区切って指定する
Requester	データ受信側の SW-C およびポートを /で区切って指定する

表 2 コンフィギュレーション部に記述する項目

項目	設計情報
Event	ランナブルの起動トリガーとなるイベントを指定する

ディスクリプション部の 1 行目には，`#DESCRIPTION` と記述する．2 行目以降に，SW-C 間インタフェース，SW-C，および，インタフェースとポートをつなぐコネクタの情報を，それぞれブロックに分けて記述する．各ブロックの 1 行目には，`##INTERFACE`，`##SWCS`，`##CONNECTORS` とそれぞれ記述する．さらに，SW-C の情報を記述するブロックを，アプリケーション SW-C の情報を記述するサブブロックと複合デバイスドライバ SW-C の情報を記述するサブブロックに分割し，それぞれ最初の行に `###Applications`，`###ComplexDeviceDrivers` と記述する．それぞれのブロックでは，コンポーネント名，区切り記号:および改行に続けて，コンポーネントの情報を一段インデントを下げて記述し，終了記号`$$`を使用してコンポーネントごとに分割する．インデント一段を空白 2 文字とする．ディスクリプション部に記述する項目を**表 1** に示す．項目と設計情報は，区切り記号:で区切って記述する．オペレーションやデータなどの構成情報を列挙する際には，項目を記述したのち改行し，一段インデントを下げ，名称，区切り記号:および改行に続けて必要なパラメータを記述する．

コンフィギュレーション部の 1 行目には，`#CONFIGURATION` と記述する．2 行目以降に，SW-C がもつランナブルの起動条件を SW-C ごとにブロックに分

けて記述する．区切り記号，終了記号およびインデントはディスクリプション部と同一の規則を使用する．コンフィギュレーション部に記述する項目を**表 2** に示す．

統合元のシステムディスクリプションファイルとコンフィギュレーションファイルは，統合元の車載システムで使用されているものをそのまま入力として使用する．

提案手法の ARXML 生成部が有用となる使用局面としては，開発した SW-C を既存の車載システムに統合する場面が想定される．開発者が用意する必要があるのは，追加する SW-C の設計情報記述ファイルのみである．

3.4 提案する生成ツールの有用性

提案する複合デバイスドライバ SW-C のマイコン依存記述生成ツールの有用性を，一般的な開発方法と比較して議論する．ターゲットとなるハードウェアが複数存在する開発においては，ターゲット依存部に ifdef 文を使用して各ターゲット向けの定義を列挙して記述される．複合デバイスドライバも，ECU ハードウェア依存部分を ifdef 文によって ECU ハードウェアごとに列挙することで開発される．しかし，ifdef 文を使用する方法は，似通った内容のコードが繰り返し記述されることになる．このため，可読性と保守性に乏しく，開発対象となっていない ECU ハードウェアに関する記述もソースコード中に残ってしまうためコードの行数が増大してしまうという問題がある．一方，デバイスドライバフォームファイルへの記述は，ECU ハードウェアに依存する部分を抽象化して一箇所に保持しているため，可読性および保守性に優れている．マイコンに依存する記述は，ターゲットとする ECU ハードウェアごとに DSL ファイルを用意すればよい．

ツールを使用する際に開発者が用意する必要があるのは，ECU ハードウェア依存情報記述ファイルおよび追加する SW-C 設計情報記述ファイルのみである．これにより，ある ECU ハードウェア向けに開発された車載システムの複合デバイスドライバを別の ECU ハードウェアに移植する開発において，ソースコード中に存在する ECU ハードウェアに依存した記述を探し出し，開発対象となっている ECU ハードウェアに適したコードに書き直す手間を省くことができる．さらに，ECU ハードウェア依存情報記述ファイルおよび追加する SW-C 設計情報記述ファイルは，いずれも読み書きの容易な DSL を使用して記述するため，複合デバイスドライバの開発における労力の削減が期待できる．

4. 適用事例

複合デバイスドライバ SW-C 生成ツールを，python を用いて実装した．実装したツールを使用し，HSBRH850F1L100[5] と DE0-Nano[6] の 2 種類の ECU ハードウェアを対象に測距センサの複合デバイスドライバ SW-C を開発した．さらに，測距センサを使用した自動ブレーキ機能を開発し，RC カーキット上での動作を確認した．

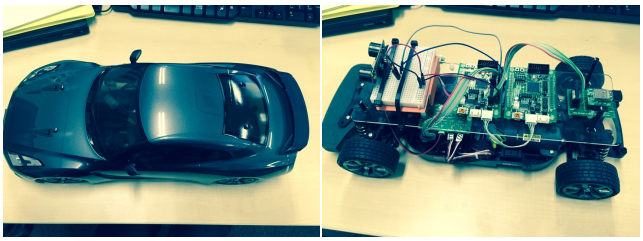

図 7 RC カーキットの外観

4.1 適用対象

4.1.1 RC カーキット

マイコンや FPGA を搭載したボードを ECU ハードウェアとして使用し，市販の RC カーの制御を実現する RC カーキット [7] が販売されている．ECU ハードウェアとして北斗電子社製の HSBRH850F1L100 と，アルテラ社製の DE0-Nano の 2 種類が使用可能である．このキットを用いることで，実車向けの車載ソフトウェア開発を RC カー上で再現することができる．**図 7** に，RC カーキットの外観を示す．

RC カーキットでは，タミヤ製のシャーシ TT-01 TYPE-E に ECU ハードウェアを接続し，プレイステーション 3 のコントローラ（PS3 コントローラ）で RC カーを操作する．PS3 コントローラとは汎用 Bluetooth アダプタを通して Bluetooth 接続によって通信する．受信したデータは，専用のマイコン基板によって UART に変換されて ECU ハードウェアに送られる．RC カーの駆動は，車速を制御する ESC（Electronic Speed Controller）および舵角を制御するサーボに対して，PWM による制御信号を送信することで行われる．また，RC カーのボディには，LED としてブレーキランプやヘッドライトが搭載されており，これらの灯火類を制御することも可能である．

さらに，2 枚以上の ECU ハードウェアを使用し，ECU ハードウェアごとにアプリケーションを割り付けることもできる．その場合は，ECU ハードウェア同士の通信に CAN を使用する．

4.1.2 RC カー制御プログラム

RC カーキットの ECU ハードウェアに書き込んで実行されるプログラムは，名古屋大学大学院情報科学研究科附属組込みシステム研究センターが中心となって，AUTOSAR のフレームワークに基づいて開発されたものが提供されている．OS 部分には TOPPERS プロジェクト [8] によって公開されている，TOPPERS/ATK2[9] を使用している．HSBRH850F1L100 向けのものと DE0-Nano 向けのものの 2 種類が提供されている．HSBRH850F1L100 向けのプログラムには，アプリケーションをすべて OS のタスクとし

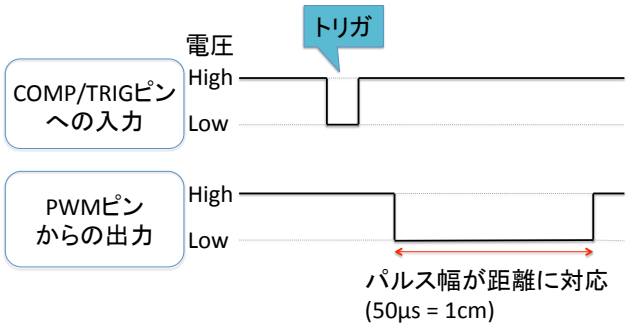

図 8 URM37 の動作

て実装した OS タスク版と，SW-C として実装した SW-C 版が存在する．DE0-Nano 向けのものは，OS タスク版のみ公開されている．

AUTOSAR の仕様上，SW-C を実行するためには RTE が必要となる．RC カー制御プログラムでは，TOPPERS プロジェクトによって公開されている RTE ジェネレータ [10] を使用して RTE を生成し，SW-C とともに OS に統合する．

4.2 生成ツールの適用

本研究で使用する測距センサ URM37[11] は，超音波によって前方に存在する物体からの距離を測定するモジュールである．**図 8** に，URM37 の動作の時系列図を示す．COMP/TRIG ピンにトリガとしてパルスを送信すると距離を計測し，PWM ピンからその結果に応じたパルスを返却する．返却されるパルスのパルス幅が距離を表しており，50μs が 1cm に対応する．

測距センサ URM37 のデバイスドライバに求められる機能は，COMP/TRIG ピンへのパルスの送信と PWM ピンからのパルスの受信を行うパルス送受信機能，および，受け取ったパルスを解析して距離を算出するパルス幅測定機能の 2 つに分けられる．以上のことを踏まえて，URM37 との接続初期化，パルス送受信およびパルス幅測定を行う処理をそれぞれ関数として実装したデバイスドライバフォームファイルを作成した．ピンの位置や動作周波数，および ECU ハードウェアごとに異なる処理部分のコードは，3.2 節で述べた記法を使用し，抽象化して記述した．デバイスドライバフォームファイルの行数は合計 112 行であった．14 種類の抽象表現によって 15 箇所の記述を抽象化した．

次に，HSBRH850F1L100 および DE0-Nano に対応する ECU ハードウェア依存情報記述ファイルをそれぞれ作成した．デバイスドライバフォームファイルで抽象化して記述されている部分に対応する具体的なコード，すなわち URM37 と ECU ハードウェアを接続するために使用するピンの位置，ECU ハードウェアの動作周波数，ピンの初期化処理およびピンの読み書きに関する処理を 3.2 節で述べた記法を用いて記述した．ECU ハードウェア依存情報記述ファイルの行数は，HSBRH850F1L100 のファイルが 46 行，DE0-Nano のファイルが 14 行であった．

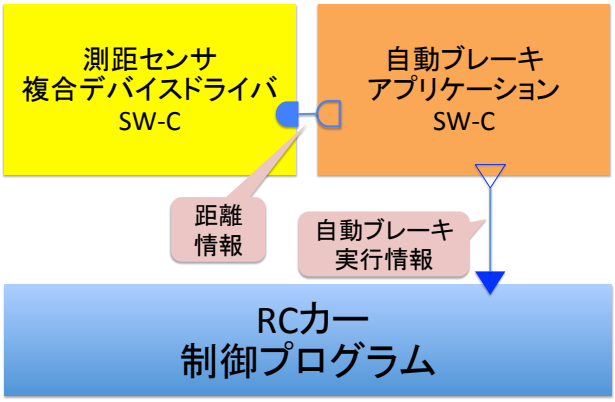

図 9 自動ブレーキ機能と RC カー制御プログラムの構成

デバイスドライバフォームファイルと ECU ハードウェア依存情報記述ファイルを複合デバイスドライバ SW-C 生成ツールの SW-C 生成部に入力し，各 ECU ハードウェアに対応した測距センサの複合デバイスドライバ SW-C を得た．

4.3 自動ブレーキ機能の実装

前方の障害物との距離が一定の値を下回ると自動的にブレーキをかけて車両を停止させる，自動ブレーキ機能をもつアプリケーションの SW-C を開発し，RC カー制御プログラムに統合した．**図 9** に，自動ブレーキ機能と RC カー制御プログラムの構成を示す．測距センサの複合デバイスドライバ SW-C と自動ブレーキアプリケーション SW-C は，Client-Server 連携によって通信する．自動ブレーキアプリケーション SW-C は，測距センサの複合デバイスドライバ SW-C を呼び出し，前方の物体との距離の情報を取得する．自動ブレーキアプリケーション SW-C と RC カーキットは，Sender-Receiver 連携によって通信する．取得した距離情報をもとに自動ブレーキを実行するかどうか判断し，実行情報を RC カー制御プログラムに送信する．

HSBRH850F1L100 向けの SW-C 版 RC カー制御プログラムと統合する際には，提案したツールの ARXML 生成部を使用した．追加する SW-C の設計情報記述ファイルに，測距センサの複合デバイスドライバ SW-C と自動ブレーキのアプリケーション SW-C の設計情報を記述した．既存の ARXML ファイルとともに生成ツールの ARXML 生成部に入力し，統合された ARXML ファイルを得た．DE0-Nano 向けの RC カー制御プログラムには SW-C 版が存在しないため，ツールは使用せずに必要な ARXML ファイルを作成した．

統合した車載ソフトウェアを HSBRH850F1L100 および DE0-Nano に書き込み，それぞれ RC カーキット上で実行した．いずれも正常に自動ブレーキ実行情報が送信されていることを確認した．

5. おわりに

複合デバイスドライバの開発における労力を軽減して車載ソフトウェア開発の効率化を図るため，複合デバイスドライバ SW-C 生成ツールを提案した．提案したツールを使用して複合デバイスドライバ SW-C を開発する際には，まずターゲットとなる ECU ハードウェアに依存する情報を ECU ハードウェアごとに読み書きの容易な DSL を用いて記述する．記述した ECU ハードウェア依存情報記述ファイルを，デバイスごとに用意されているデバイスドライバフォームファイルと組み合わせて複合デバイスドライバ SW-C 生成ツールに入力することで，複合デバイスドライバ SW-C を得ることができる．さらに，追加する SW-C の設計情報を記述し，統合元のシステムの ARXML ファイルとともに提案したツールに入力することで，追加する SW-C についての ARXML による記述が追記された ARXML ファイルを得ることができる．

提案手法によって，ある ECU ハードウェア向けに開発された車載システムの複合デバイスドライバを別の ECU ハードウェアに移植する開発において，ソースコード中に存在する ECU ハードウェアに依存した記述を探し出し，開発対象となっている ECU ハードウェアに適したコードに書き直す手間を省くことができる．さらに，開発した SW-C を既存の車載システムに統合する場面においては，ARXML ファイルを開発者自身が編集することなく，適切に統合された ARXML ファイルを得ることができる．

今後の方針として，本研究で提案した複合デバイスドライバ SW-C 生成ツールの評価が挙げられる．具体的には，ツールを使用せずに開発した複合デバイスドライバ SW-C とツールを使用して開発した複合デバイスドライバ SW-C を，コード量や実行時間などの基準で比較することが考えられる．さらに，追加する SW-C の設計情報記述ファイルの仕様の簡略化が挙げられる．現在の仕様では，一つの SW-C に関する情報をシステムディスクリプションファイルに統合する内容とコンフィギュレーションファイルに統合する内容に分けて記述する必要がある．開発の簡単化のため，一つの SW-C に関する情報は一箇所にまとめて記述する仕様に変更することが考えられる．

参考文献

[1] 櫻井　剛：自動車の組込みソフトウェアの現状: AUTOSAR および ISO 26262 (組込みシステムの信頼性・安全性)，日本信頼性学会誌， Vol. 36, No. 4, pp. 197–205 (2014).
[2] AUTOSAR: AUTomotive Open System ARchitecture(online), http://www.autosar.org/ (2016.05.11).
[3] 鈴村延保，香月伸一：車載組み込み技術開発の欧州全体俯瞰と動向，研究報告組込みシステム (EMB)， Vol. 9, pp. 1–12 (2009).
[4] 鴫原一人：安全に使い回す！車載ソフトウェアの世界 第 2 回 AUTOSAR 準拠ソフトウェアの基本開発ステップ，*Interface*, Vol. 464, CQ 出版社，pp. 157–159 (2016).
[5] 北斗電子：HSBRH850F1L100(online), http://www.hokutodenshi.co.jp/7/HSBRH850F1L100.htm (2016.05.11).
[6] Terasic: DE0-Nano(online), http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=165&No=593&PartNo=2 (2016.05.11).
[7] 北斗電子：RC カーキット (online)，http://www.hokutodenshi.co.jp/7/HSBRH850F1L100.htm#rccar (2016.05.11).
[8] TOPPERS プロジェクト：https://www.toppers.jp/index.html (2016.05.11).
[9] TOPPERS プロジェクト：TOPPERS/ATK2(online), https://www.toppers.jp/atk2.html (2016.05.11).
[10] TOPPERS プロジェクト：TOPPERS/A-RTEGEN(online), https://www.toppers.jp/a-rtegen.html (2016.05.11).
[11] DFRobot: URM37 V3.2 Ultrasonic Sensor(online), http://www.dfrobot.com/wiki/index.php/URM37_V3.2_Ultrasonic_Sensor_(SKU:SEN0001)#Specification (2016.05.11).