情報処理学会論文誌　コンシューマ・デバイス & システム　Vol.7　No.2　41–50 (May 2017)

コンシューマ・システム論文

車載ECU向け差分更新方式

寺岡 秀敏[1,a)]　中原 章晴[2]　黒澤 憲一[2]

受付日 2016年9月30日, 採録日 2017年2月27日

概要：近年，自動車分野においても機能追加やセキュリティリスクへの対策，ソフトウェア不具合に起因するリコール対応等のため，電子制御ユニット（ECU）に搭載されたソフトウェアを販売後に更新する機会が増大している．そのような中で，デジタルテレビや携帯電話等のコンシューマエレクトロニクス分野で実用化されている無線によるソフトウェアの遠隔更新技術（Over the Air Firmware Update: OTA）の自動車分野への適用が検討されている．OTA による遠隔更新では，更新時間の短縮が重要となりそのための要素技術として差分更新が必要になる．本研究では，ECU に適用可能な差分更新方式を提案する．また，提案方式を，自動車向けマイコン RH850 上に実装し，当該マイコン上に搭載可能なレベルに消費メモリを削減し，更新時間を短縮できることを確認した．

キーワード：車載 ECU，差分更新，bsdiff

Incremental Update Method for In-vehicle ECUs

Hidetoshi Teraoka[1,a)]　Fumiharu Nakahara[2]　Kenichi Kurosawa[2]

Received: September 30, 2016, Accepted: February 27, 2017

Abstract: Recently, the increase of amount of program code on Electric Control Units (ECUs) in vehicles causes the increase of firmware update after sales which stems from bugs in the program code. In this situation, automakers are studying to introduce over the air firmware update technology which is used in consumer electronics field such as digital TV and mobile phone. In this paper, we propose an incremental update method which is applicable to ECUs. We implement the method on RH850 evaluation board and show that the proposed method is applicable to ECU and is able to shorten the update time.

Keywords: In-vehicle ECU, incremental update, bsdiff

1. はじめに

1.1 背景

Advanced Drive Assist System（ADAS）や自動運転の導入により，自動車の価値を差別化する要素として，ソフトウェアの価値が高まっている．また，自動車に搭載される車載電子制御装置（Electronic Control Unit: ECU）のソフトウェア規模増大にともないプログラム不具合が要因のリコール（回収・無償修理）が増加している．さらに，自動車システムがネットワークに接続され，セキュリティリスクが高まっている．たとえば，Chrysler 社はセキュリティ対策のため，2015 年に米国で約 140 万台のリコールを実施している [1]．

現在，自動車が販売された後に車載 ECU のプログラム更新を行う場合，顧客（自動車所有者）がディーラに車両を持ち込み，ディーラの整備員が専用ツールを使ってマニュアル作業で 1 台 1 台 ECU 上のプログラムを書き換えることが一般的である．しかし，この方法は，車両の持ち込み等時間がかかることでユーザの利便性が悪い．この問題を解決するため，車載 ECU に対して遠隔ソフトウェア更新を適用し，プログラム更新時のユーザ利便性を向上させることが検討され始めている．

携帯電話網等の無線接続を利用してソフトウェア/ファー

[1] 株式会社日立製作所研究開発グループ
Hitachi, Ltd. Research & Development Group, Yokohama, Kanagawa 244–0817, Japan
[2] 日立オートモティブシステムズ株式会社
Hitachi Automotive Systems, Ltd., Hitachinaka, Ibaraki 312–8503, Japan
[a)] hidetoshi.teraoka.rf@hitachi.com

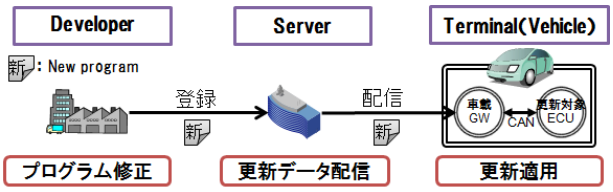

図 1　OTA システム構成

Fig. 1　System configuration of an OTA system.

ムウェアを端末に配信し，遠隔で更新する技術は OTA（Over the Air software/firmware update：無線によるソフトウェアの更新）と略称される．デジタルテレビや携帯電話分野では OTA は一般化され，多量の端末のソフトウェアが効率的に更新されている．OTA システムを自動車に適用する場合，修正した新プログラムをセンタサーバに登録し，サーバは車両に修正した新プログラムを配信する．車両では，車載 Gateway（GW）が新プログラムを受信し，これを更新対象 ECU に送信して更新を適用する（**図 1**）．

近年，車載機（カーナビ）においても，遠隔での地図やソフトウェアの更新が導入されている．さらに，テスラモーターズ社は，OTA を自社車両に適用し，走行に関連する機能の遠隔更新を実現している．同社は 2015 年 8 月には販売済みの車両に対して，OTA ソフトウェア更新によって有料で自動運転機能を追加するという，新たなビジネスモデルを試行している [2]．

1.2　課題

OTA では，通常一般顧客（車両所有者）が更新を行うため，車両を使用できない時間をできるだけ短くすることが要求される．また，停車中も長時間電力供給が可能な電気自動車と異なり，ガソリン車では更新時間の短縮はより重要となる．すなわち，エンジン動作中に更新を行う場合は，更新時間の長さが環境負荷に直結する一方，エンジン停止状態で更新する場合は，電力供給可能な時間が限られる．

ディーラの整備員が専用ツールを使って ECU 上のプログラムを書き換える従来のソフトウェア更新プロセスにおける処理時間の内訳を**図 2** に示す．従来の更新プロセスでは，車両外の診断装置から Controller Area Network（CAN）[3] を経由してプログラム全体を更新対象 ECU に送信し，ECU 上で Flash ROM を書き換えることでソフトウェアの更新を行う．図 2 に示すとおり，ECU のソフト更新時間の大半は CAN による伝送時間が占める．ここでは，2 MB のプログラムを 500 Kbps の CAN の帯域のうち 10%を利用して伝送した場合の例を示した．

OTA によるソフト更新において，車載 Gateway への配信までを走行中等に実施すると仮定すると，車両を利用できない時間は図 1 の更新適用処理の間である．更新適用処理は，診断装置が車載 Gateway に置き換わることを除いては従来の更新プロセスと同様であるため，処理時間の内訳

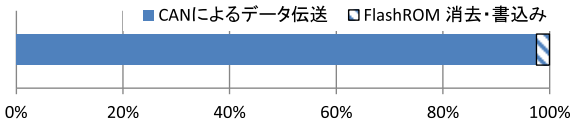

図 2　ECU のソフト更新時間の処理内訳

Fig. 2　Breakdown of ECU program update time.

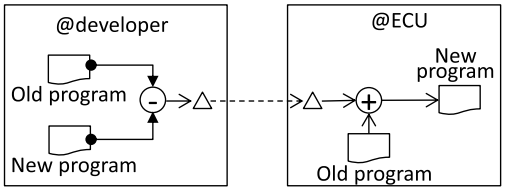

図 3　差分更新の概要

Fig. 3　Overview of incremental update.

は上記と同様になる．このため，OTA によるソフトウェア更新時間の短縮には CAN の伝送時間の短縮が課題となる．このようなプログラム更新のためのネットワーク伝送時間短縮技術として，差分更新技術がある [4], [5], [6], [7]．

差分更新では，開発元等で新旧プログラムから差分ファイルを生成して更新対象に転送し，更新対象上の旧プログラムと差分ファイルから新プログラムを復元する（**図 3**）．このようにプログラム全体ではなく差分のみを送信することでネットワーク上を伝送するデータ量を削減できる．

本研究では，CAN におけるデータ転送量を削減し更新時間を短縮する技術として，車載 ECU に適用可能な差分更新方式を提案する．

2.　関連研究

清原 [4] は，携帯電話のプログラムの遠隔更新に関して，Flash ROM の書き換え領域の局所化による書き換え時間の短縮および，差分更新を用いた通信データ量削減による配信時間の短縮を提案している．前述のとおり，ECU の書き換え処理においては，Flash ROM の書き換え時間が占める割合は相対的に小さく，通信量を削減できる差分更新の時間短縮効果が高いと考えられる．プログラムの差分更新については他にも，商用ソフト [5], [6] に加え，オープンソースソフトウェア（OSS）である bsdiff [7] が PC や携帯電話，カーナビに適用され広く利用されている．しかしながら，ECU に搭載されたメモリは PC や携帯電話と比較すると小さいため，これらのデバイス向けに開発されたソフトウェアをそのまま搭載することは困難であると考えられる．

Nakanishi ら [8] は，前述の bsdiff で生成した差分を復元する際に，有向グラフを用いて復元時の書き込み干渉を回避し，インプレース更新を適用してメモリ使用量を削減する方法を提案している．また，野澤ら [9], [10] は，圧縮率向上を目的として，プログラムの変更によるバイナリモジュールの変化の内容に着目して bsdiff の差分抽出方法の

改善を提案している．これらの方式はメモリ領域の削減や圧縮率の向上に有効な手段であると考えるが，実際の車載ECU のメモリサイズ等を考慮した搭載性評価がなされていない．

また，ECU 向けの更新時間の短縮に関しては，車内ネットワークとして FlexRay，CAN-FD や Ethernet を導入し，通信の広帯域化と複数 ECU の並列更新によって更新時間を短縮する方法も提案されている [11], [12], [13]．通信の広帯域化は更新時間短縮の有効な手段であるが，コストの観点からは，ソフト更新時間の短縮のみを目的として導入することは難しい．

本研究では，前述の研究をふまえ，bsdiff をベースとして車載 ECU に搭載可能な省メモリな差分更新方式を提案する．また，提案する方式を車載マイコン上に実装して評価を行い，メモリ使用量および更新時間の短縮効果を確認する．

3. 車載 ECU への差分更新適用課題

3.1 bsdiff の差分圧縮処理概要

bsdiff は，バイナリファイル間の差分を抽出する OSS である．**図 4** に，bsdiff による差分ファイルの生成手順および生成される差分ファイルのフォーマットを示す．bsdiff では，最初に，Suffix Sorting を用いて生成した接尾辞配列を用いて新旧バイナリデータの一致部分を検索する．次に，当該一致部分をもとに，近似一致部分と新規追加部分の分離を行う．近似一致部分とは，ソースコード上の変更は行われなかったが，参照アドレスの変更等により，プログラムのバイナリデータに変更が生じた部分に相当する．新規追加部分は，ソースコード上も新規に追加された部分に該当する．その後，隣接する近似一致部分と新規追加部分を 1 つの区画として，この区画に対する復元命令（Control）を生成する．抽出した近似一致部分については，新データと旧データの減算を行う．これにより，変更部分以外が 0 となる圧縮効率の良いデータ（DIFF）が生成される．復元命令は，DIFF 部のサイズ a_n，新規追加部分（EXTRA）のサイズ b_n，および復元時の旧プログラム上のポインタ移動量 c_n から構成する（図 4 の①）．bsdiff では，これら各区画の Control，DIFF，EXTRA をそれぞれひとまとめにして bzip2 [14] を用いて圧縮し，これにヘッダを追加して差分ファイルを生成している（図 4 の②）．

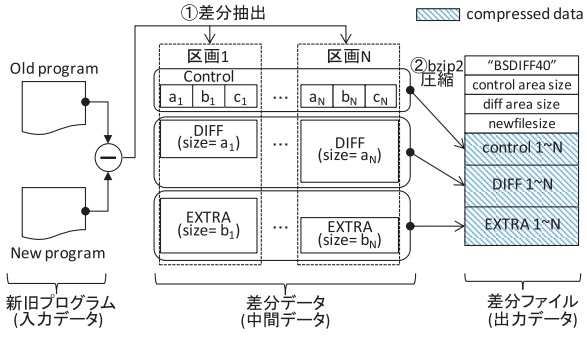

図 4　bsdiff の差分ファイル生成手順とデータ構成

Fig. 4　Delta file generation process and data structure of bsdiff.

3.2 bspatch における復元処理

bsdiff を用いて生成された差分ファイルは，bspatch と呼ばれるプログラムを用いて復元する．**図 5** に bspatch による差分復元手順と復元時に使用するメモリ領域を示す．

bspatch による差分復元では，Flash ROM に格納されている旧プログラムと，受信した差分ファイルから新プログラムを復元し，復元領域に格納する（①）．次に，Flash ROM 上の旧プログラムを復元した新プログラムで上書きする（②）．このため，復元に際しては式 (1) で示すメモリ量が必要となる．

$$M = d + w + o + n \tag{1}$$

ここで，d は差分ファイル格納領域のサイズ，w は復元に必要なワークメモリ，o は旧プログラムサイズ，n は新プログラムサイズである．以下，各領域の必要量について説明する．

(1) 差分ファイル格納領域 d

bsdiff により出力される差分ファイルのサイズは，比較対象の類似度によって大きさが異なる．まったく異なるデータを比較する場合等，十分に圧縮されない可能性もあり，復元側では保守的に領域を確保しておく必要がある．ECU のプログラムでは，メモリを動的に確保することはまれであり，基本的には用途に応じたメモリを静的に確保しておく．このため，差分ファイル格納領域としてプログラムサイズの 1/4 を確保すると仮定した場合，2 MB のプログラムでは 500 KB の領域が必要となる．

(2) ワークメモリ w

bsdiff では，前述のとおり，Control 部，DIFF 部，EXTRA 部がそれぞれ別々に圧縮されており，復元時にこれらを並行して解凍する必要がある．このため，復元に際しては圧縮アルゴリズムが必要とする量の 3 倍のメモリが必要となる．bsdiff では，圧縮アルゴリズムとして bzip2 を利用しており，復元には 100 kB + (2.5 × ブロックサイズ) のメモ

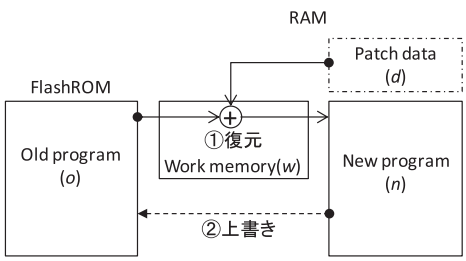

図 5　差分復元手順と必要メモリ

Fig. 5　Patch process and memory layout.

リが必要である．ブロックサイズを 4 KB に設定した場合でも，110 kB のメモリが必要となるため，bspatch における復元では 330 kB のメモリが必要となる．

(3) 新旧データ格納領域 o，n

差分更新では，旧プログラムを参照しながら新プログラムを復元するため，旧プログラムの格納領域と復元した新プログラムの格納領域が必要となる．旧プログラムを Flash ROM から読み出すとして，復元用には，新プログラムと同じサイズの RAM が必要となる．

3.3　適用課題と目標仕様

車載 ECU では，部品コストを抑制するため，OTA が先行導入されている携帯電話等と比較してメモリリソースが少ない．そのため，基本的にはプログラムコードは Flash ROM 上で実行される．また，プログラムを格納する Flash ROM のサイズと比較し，差分ファイル格納，新プログラム復元およびワーク領域として利用可能な RAM のサイズが小さい．たとえば，Renesas 社製のボディ系 ECU 向けマイコンである RH850/F1L [15] では，ROM サイズは 256 KB～2 MB，RAM サイズは 32 KB～192 KB である．

このような特性から，差分更新を車載 ECU に適用するためには，復元処理における使用メモリ量を削減することが重要となる．一方，一般的に，メモリ使用量を削減した場合，圧縮率が下がる．そこで，本研究では，RH850/F1L に搭載可能なレベルにメモリ使用量を削減しつつ，bsdiff 相当のデータ削減率を実現し，ECU のソフトウェア更新の更新時間を短縮することを目標とする．

4.　提案方式

本章では，ECU における復元処理に必要なメモリ量を削減するための方式について，差分の取り方（差分生成単位），差分ファイル構造，圧縮アルゴリズムに着目して検討した結果を示す．

差分生成単位の検討では，広域ブロック差分方式により，差分サイズを抑えつつ復元した新データを一時格納する領域 n を削減する方式を提案する．差分ファイル構造の検討では，データを復元する順番に並べて圧縮し，逐次復元を可能とすることで差分ファイル格納領域 d とワークメモリ w を削減する方式を提案する．圧縮アルゴリズムの検討では，2 段階圧縮方式を適用してワークメモリ w を削減する方式を提案する．

4.1　差分生成単位

Nakanishi らの提案 [8] では，新プログラムを復元する前に必要な旧プログラムの一部を削除してしまう Read-Write Conflict を，有向グラフを用いて回避しインプレース更新を適用することで新旧プログラム格納領域 o，n を削減する方式が示された．この方式では，前述の領域を大きく削減できる一方で，グラフを生成し格納するための領域がワークメモリ w に追加となる．また，プログラム全体を比較しているため，差分格納領域 d として十分な領域を準備しようとした場合，たとえばプログラムサイズの 1/4 などの大きな領域の確保が必要になる．本節では，前者の課題に対する方式提案を行う．後者の課題への対応については，次節で詳細を説明する．新旧プログラム格納領域 o，n を削減するため，本研究では Flash ROM の消去ブロックを基本単位として差分を生成するブロック差分方式を提案する．これにより，復元時に必要とする Work メモリの追加なしでインプレース更新を適用でき，復元した新プログラムを格納する領域 n を削減できる．消去ブロックを基本単位とした差分生成方法には，以下の 2 通りがある．

(1a)　新旧プログラムの同一アドレス範囲を比較（ブロック差分）

(1b)　新プログラムの 1 ブロックと，旧プログラムの未復元領域に相当する複数ブロックを比較（広域ブロック差分）

図 6 に，各方式の概要を示す．

方式 (1a) では，新旧プログラムの同一ブロックを比較し，差分を生成する．たとえば，新プログラムの Block 1 は旧プログラムの Block 1 と比較され，Block 1 を復元するための差分ファイルが生成される．各ブロックについてこれを繰り返すことで，プログラム全体を復元するためのパッチファイルを生成する．

方式 (1b) では，新プログラムの 1 ブロックに対して，旧プログラムの複数ブロックを比較対象として差分を生成する．たとえば，新プログラムの Block 2 については旧プログラムの Block 0，1，2 全体と比較し，Block 2 を復元するための差分ファイルが生成される．各ブロックについてこれを繰り返すことで，プログラム全体を復元するためのパッチファイルを生成する．方式 (1b) は，新プログラムに大きな新規コードが追加される場合に有効であると考え

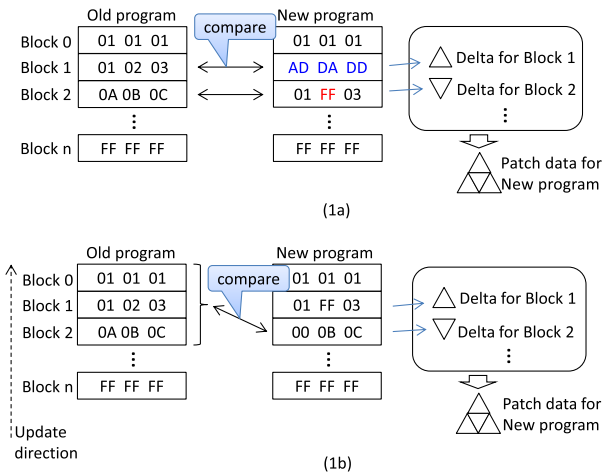

図 6　ブロック差分方式概要

Fig. 6　Overview of block delta encoding.

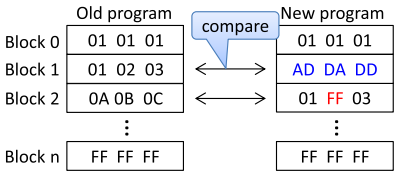

図 7　モジュールが挿入される事例
Fig. 7　Example of inserting a large module.

る．すなわち，**図 7** に示すように新プログラムの Block 1 に相当する部分に新規コードが追加された場合，新旧プログラムの Block 1 どうしや Block 2 どうしを比較しても，一致する部分がほとんどなくなってしまう．これは，追加された部分以降の全ブロックに影響するため，結果として差分サイズが大きくなるという懸念がある．プログラムのコードが削除された場合も発生すると考えられるが，ECU のプログラムは PC や携帯電話のプログラムとは異なり，出荷後に機能削減やメモリマップの変更が行われることは少ないと考えられる．このため，本方式では，図 6 の (1b) に示すように，Block n から降順に差分生成を進め，旧プログラムの未復元領域全体を比較対象として利用する．

4.2　差分ファイル構造

bzip2 を含む多くの圧縮アルゴリズムでは，ランダムな位置からの解凍はできず，データ先頭から，解凍位置や状態をワークメモリ領域に保持しながら解凍処理を行う必要がある．このため，図 4 の ② に示した Control 部，DIFF 部，EXTRA 部をそれぞれひとまとめにして並列に圧縮する場合（並列圧縮）では，復元時に要するワークメモリ量は圧縮アルゴリズムが解凍に要するサイズの 3 倍となってしまう．本問題を解決する方式として，以下の 2 通りが考えられる．

(2a)　並列圧縮ではなく，差分データを 1 つに連結して圧縮（以下，直列圧縮と呼ぶ）し，復元時に必要な状態保持領域を削減する．

(2b)　復元データへのランダムアクセスが可能な圧縮アルゴリズムを利用し，並列に復元する際にもそれぞれの復元状態保持を不要にする．

ここで，(2b) については，一般化されたアルゴリズムが少ないことと，一般的な圧縮アルゴリズムでランダムアクセスを可能にする場合は，細かいブロック単位に分割する必要があるなどで圧縮効率が下がる可能性が高いことから，(2a) の方法を詳細検討することとした．

差分データを直列圧縮する場合のデータの直列化方式として，以下の 2 通りが考えられる．

(2a-1)　従来のデータ列のまま Control 部，DIFF 部，EXTRA 部をそれぞれまとめてから全データを 1 度に圧縮する．

(2a-2)　データを復元処理の順番に整列した後で圧縮する．

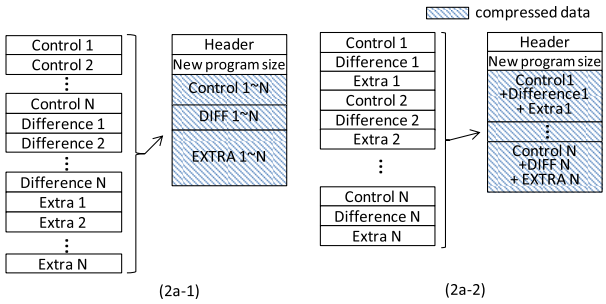

図 8　データ構成概略
Fig. 8　Overview of data structure.

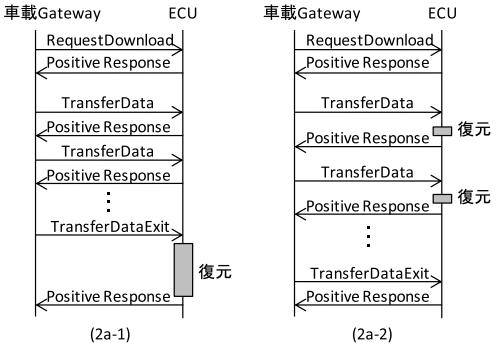

図 9　差分復元手順の比較
Fig. 9　Comparison of delta decompression process.

表 1　データ構成の比較
Table 1　Comparison of data structure.

方式	圧縮率	メモリ使用量
(2a-1)	○ 類似データが集合	× 一括復元
(2a-2)	△	○ 逐次復元可

図 8 にこれらのデータ構成の概略図，**図 9** に差分復元手順の概略図を示す．圧縮率の観点では，(2a-1) の方式では，類似の構造のデータが集まっているために圧縮率が良いと見込まれる．一方で，(2a-2) の方式は異なる構造のデータを整列しているため圧縮効率は良くない可能性がある．復元時のメモリ使用量の観点では，(2a-1) の方式では，差分ファイル全体を受信するまで復元ができない（図 9 (2a-1)）ため，差分ファイル格納領域 d が大きくなる．一方，(2a-2) の方式は，復元する順番にデータを受信し，逐次復元できる（図 9 (2a-2)）ので，差分ファイルを格納するために必要な領域は最小限に抑えられる．

表 1 に本比較のまとめを示す．本研究では，メモリ使用効率を優先し，(2a-2) の方式を採用することとした．

(2a-2) の方式を用いて逐次復元を行う場合，ECU における復元処理は**図 10** に示す手順で行う．

すなわち，固定長（たとえば，256 byte）の領域を差分格納領域 d として受信したデータを格納し，当該領域の残量がなくなった時点で復元を開始する．ワークメモリで復元された新プログラムの断片は，復元領域 n に格納し，1 ブ

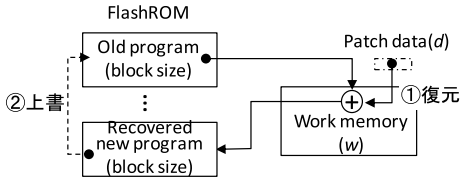

図 10　逐次復元処理

Fig. 10　Stream delta decompression process.

ロック分を復元したところで，対応する領域の旧プログラムを上書きする．ここでは，更新中の電源断等発生時にも旧プログラム・新プログラムのいずれかが存在する状態にして信頼性を向上させるため，復元領域 n は Flash ROM 上に確保する例を示した．

4.3　圧縮アルゴリズム

bsdiff で利用されている bzip2 は，Burrows–Wheeler 変換 [16] および Move To Front 法によるブロックソートと，Huffman 符号 [17] による符号化を組み合わせて deflate [18] 等と比較して高い圧縮率を実現している．一方で，復元時間が遅い，復元時の消費メモリが多い，などの課題がある．そのため，前述のように圧縮を直列化したとしても，依然として 110 Kbyte のワークメモリ w が必要となる．

そこで，本研究では，圧縮アルゴリズムとして圧縮効率が高く，復元が高速な LZMA [19] をベースとした 2 段階圧縮方式を適用する．すなわち，抽出した差分データに対し，LZ77 [20] ベースの辞書式符号化に加え，Binary Range Coder [21] を利用して圧縮率の向上を図る（**図 11**）．

LZMA の辞書式符号化では，入力データは LZMA Packet に符号化される．LZMA Packet は，一致する文字列がある場合に，単純にオフセットと長さを表現するのではなく，一致が繰り返される場合にこれを短く表現できるように 7 種類の符号が定義されている．この LZMA Packet をさらに符号化する Binary Range Encode ではコンテキストに基づく確率情報を利用し，圧縮率を向上させている．コンテキストとしては，符号化対象の Packet 種別に加え，直前の符号化の状態や符号対象の位置情報などを利用する．これにより圧縮率の向上が図れる一方で，確率情報管理のために必要なメモリが増大する．LZMA では，確率の管理に約 14 Kbyte のメモリが必要となる．本提案では，利用する確率情報を限定し，必要なワークメモリ w を削減する．

4.4　提案方式

以上より，車載 ECU への搭載性を考慮して復元時のメモリ使用量を削減した差分更新方式として，方式 (1b) の広域ブロック差分方式，(2a-2) の直列化圧縮化方式，2 段階圧縮方式を適用した改良版 bsdiff を提案する．**表 2** に bsdiff と提案方式の比較を示す．

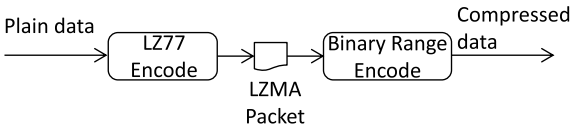

図 11　LZMA に基づく 2 段階圧縮方式

Fig. 11　Two stage compress method based on LZMA.

表 2　bsdiff と提案方式の比較

Table 2　Comparison of bsdiff and proposed method.

項目	bsdiff	提案方式
差分生成単位	プログラム全体比較	広域ブロック差分 (1b)
差分ファイル構造	並列圧縮	復元順序を考慮したデータ直列化(2a-2)
圧縮アルゴリズム	bzip2	LZMA ベースの 2 段階圧縮方式

表 3　評価対象プログラムとサイズ

Table 3　Type and size of test programs.

Test program	有効 ROM サイズ S_R[byte]	備考
A	1,998,848	パワートレイン系
B	1,572,864	パワートレイン系
C	1,310,720	走行系
D	1,310,720	走行系
E	98,304	TOPPERS/ATK2[22] SC1 1.3.2→SC2 1.0.0

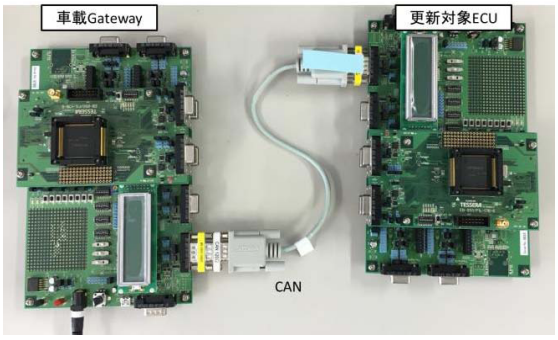

図 12　評価環境外観

Fig. 12　Appearance of evaluation platform.

5.　評価

5.1　評価環境

本研究では，実際の車両に搭載されている 4 種類の ECU 用プログラムと車載 ECU 向けのオープンソースソフトウェアを対象として評価を実施した．評価対象プログラムとサイズを**表 3** に示す．

ここで，有効 ROM サイズとは，未使用のブロックを除いた，更新対象のプログラムが書き込まれているブロックの合計サイズを示す．

評価に用いた環境条件を**図 12** および**表 4** に示す．

本研究では，テセラ・テクノロジー社製のマイコン評価ボード FL-850/F1L-176-S [23] にルネサスエレクトロニクス社製の RH850/F1L を搭載し評価環境を構成した．本評

表 4　評価環境
Table 4　Specification of evaluation platform.

項目		値と説明	
CPU		80MHz	Renesas RH850/F1L
ROM		2MB	70 ブロックで構成 （8Kbyte×8, 32Kbyte×62）
RAM		160KB	
CAN	帯域	50Kbps	HI-Speed CAN(500Kbps)利用 ただし，利用帯域 10%
	プロトコル	ISO15765-2, ISO14229(UDS)	
	パラメータ	Separation Time=0, Block size=1	

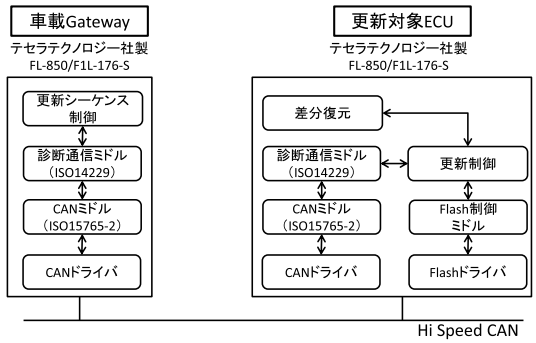

図 13　評価環境機能配置
Fig. 13　Functional layout of evaluation platform.

価では，車載 Gateway および更新対象 ECU に同じ評価ボードを用いそれぞれに異なるソフトウェアを実装した．評価用車載 Gateway および更新対象 ECU の機能配置を**図 13** に示す．車載 Gateway には UDS（Unified Diagnostic Services）[24] で規定された手順に従い更新シーケンスの制御を行う更新シーケンス制御部，UDS コマンドの構成や解析を行う診断通信ミドル，ISO15765-2 [25] に対応した CAN 通信を行う CAN ミドルおよび CAN ドライバを搭載する．また，更新対象 ECU には，車載 Gateway からの指示に従って差分復元部や診断通信ミドル，Flash 制御ミドルを制御して ECU ファームウェアの更新を制御する更新制御部と，差分復元処理を行う差分復元部，Flash ROM の消去や書き込みを制御する Flash 制御ミドルと Flash ドライバ，UDS コマンドの構成や解析を診断通信ミドル，ISO15765-2 に対応した CAN 通信を行う CAN ミドルおよび CAN ドライバを搭載する．Flash 書き込み制御機能およびこれらを制御する更新制御機能を搭載する．CAN の通信帯域は 500 Kbps であるが，通常ソフト更新のための診断通信に割り当てられる帯域は少ないため，10%が利用可能であると仮定した．ここで，ISO15765-2 に規定された CAN の通信パラメータとして Separate time は 0，Block size は 1 を利用した．

評価項目としては，プログラム更新に必要なメモリ量，圧縮率，更新時間を評価した．**表 5** に評価項目および評価方法をまとめる．

本研究では，従来方式の bsdiff/bspatch [8] との比較を基本として提案方式の効果を検証する．また，広域ブロック差分の効果を検証するため，ブロック差分とも比較を行う．

表 5　評価項目と方法
Table 5　Evaluation items and methods.

評価項目	評価方法
メモリ量	ソースコード及びメモリマップを利用して積算
圧縮率	PC 上で圧縮アルゴリズムを実装， 評価対象プログラムの差分ファイルを生成し，サイズ評価
更新時間	RH850/F1L 評価環境での実測

表 6　通常更新時の更新時間 [s]
Table 6　Update time of normal update [s].

Test program	T_{update}	T_{com}	T_{GW}	T_{ECU}		
					T_{decomp}	T_{flash}
A	**1231.0**	**1215.7**	**1.1**	**14.2**	0	14.2
B	**970.1**	**957.9**	**0.8**	**11.2**	0	11.2
C	**808.2**	**798.1**	**0.7**	**9.4**	0	9.4
D	**808.0**	**798.0**	**0.7**	**9.4**	0	9.4
E	**47.0**	**46.3**	**0.1**	**0.6**	0	0.6

プログラム更新に必要なメモリ量は，ECU に差分復元ソフトを組み込んで動作させるために必要な ROM サイズおよび RAM サイズである．これらの必要量については，ソースコードおよびメモリマップからの積算により評価する．圧縮率は，差分生成アルゴリズムを実装して各評価対象プログラムの差分圧縮を行った際にどれだけデータを削減できたかを評価する．更新時間については，差分復元アルゴリズムを更新対象 ECU 上に実装して実測する．更新時間は，式 (2) のように表現できる．

$$T_{update} = T_{GW} + T_{ECU} + T_{com} \tag{2}$$

ここで，T_{GW} は車載 Gateway 内の処理時間，T_{ECU} は ECU 内処理時間，T_{com} は通信時間である．ECU 内の処理時間 T_{ECU} は，式 (3) で算出する．

$$T_{ECU} = T_{decomp} + T_{Flash} \tag{3}$$

ここで，T_{decomp} は ECU 上で差分を復元する時間であり，T_{Flash} は FlashROM の消去・書き込みに要する時間である．本評価においては，車載 Gateway 側で T_{update}，T_{GW} を計測し，更新対象 ECU 側で T_{decomp} および T_{Flash} を計測する．

CAN 通信時間 T_{com} はこれら計測結果を用いて式 (4) で算出する．

$$T_{com} = T_{update} - (T_{GW} + T_{ECU}) \tag{4}$$

前述の評価環境における通常更新時の更新時間計測結果を**表 6** に示す．

実際の更新にあたっては，前述の時間に加えて，更新データをサーバからダウンロードする時間，受信したデータの暗号復号・検証等を行う時間が必要である．一方で，これらの処理はバックグラウンドで実施し，ユーザは意識しないように構成することが可能であるため，ここでは更新時間には含めない．

表 7　提案方式のメモリ使用量 [byte]
Table 7　Specification of evaluation platform [byte].

Item	bspatch([8])	提案方式
プログラム(ROM)	16,350〜*1	8,110
差分格納 d＋ワーク w (RAM)	862,208〜*2	7,680
一時復元領域 (RAM または ROM)	32,768*3	32,768*3

*1:オリジナルの bspatch に加え，有向グラフ生成ロジックが必要
*2:有向グラフ生成・格納領域が必要
*3:インプレース更新における一時復元/退避領域
(表 4 の評価環境における最大ブロックサイズ)

表 8　提案方式のパッチファイルサイズ S_d [byte]
Table 8　Comparison of path file size S_d [byte].

Test program	Bsdiff	ブロック差分	広域ブロック差分
A	12,929	24,617	23,090
B	86,640	130,054	100,549
C	146,740	351,473	157,279
D	37,509	73,951	45,178
E	8,132	8,167	7,671

5.2　評価

(1) 使用メモリ量

表 7 に，提案方式のメモリ使用量の評価結果を示す．

本評価においては，bspatch では，$d = 500$ Kbyte とし，bzip2 のブロックサイズを 4 Kbyte とした．また，提案方式の辞書式符号化用 Window Size は 4 Kbyte とした．

提案方式について，図 13 の更新制御部および差分復元部のプログラムサイズは約 8 KB，RAM 上の差分格納領域およびワークメモリは約 7.5 KB であるため，たとえば，評価環境とした RH850/F1L 等の車載 ECU に搭載可能なレベルのメモリ使用量で差分を復元できることを確認できた．なお，これらのメモリ使用量は，入力データの内容に依存しない．なお，広域ブロック差分・ブロック差分では，参照先が変わるだけのため復元時のメモリ使用量は変わらない．差分データと旧プログラムから再構成した新プログラムを一時的に格納する一時格納領域は，RAM 上，ROM 上いずれに用意してもよい．更新のロバスト性を向上させるためには ROM 上に確保することが望ましいが，Flash ROM 寿命への影響や更新速度への影響を考慮する必要がある．

(2) 圧縮率

前述の評価対象プログラムについて，バージョン間の差分を生成して元プログラムとの比較を行い，圧縮率を評価する．圧縮率 $r_{compression}$ は式 (5) で計算する．

$$r_{compression} = S_d / S_R \tag{5}$$

ここで，S_d は生成されたパッチファイルのサイズ，S_R は有効 ROM サイズである．**表 8** にオリジナルの bsdiff，ブロック差分方式および広域ブロック差分方式で生成したパッチファイルのサイズを示す．また，各方式の圧縮率の比較を**図 14** に示す．

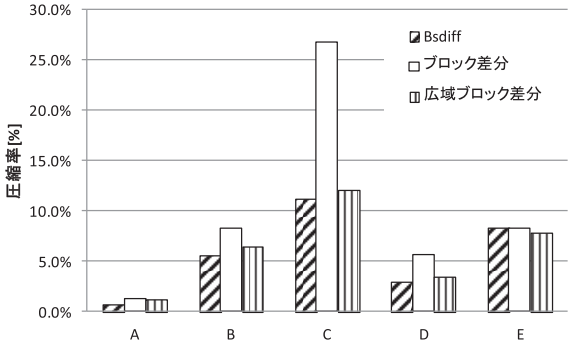

図 14　提案方式の圧縮率 $r_{compression}$
Fig. 14　Compression ratio of proposed method.

表 9　差分更新適用時の更新時間 [s]
Table 9　Update time of incremental update [s].

Test program	T_{update}	T_{com}	T_{GW}	T_{ECU}		
					T_{decomp}	T_{Flash}
A	**28.7**	**14.8**	0.1	13.8	1.7	12.1
B	**73.7**	**61.8**	0.1	11.8	1.8	10.0
C	**105.8**	**96.2**	0.2	9.5	1.7	7.8
D	**37.1**	**28.0**	0.1	9.0	1.2	7.8
E	**5.5**	**4.7**	0.04	0.8	0.1	0.7

提案方式ではメモリ使用量を削減できた一方で，ブロック単位に分割して差分を生成したことによって，圧縮率が低下している．特に，ブロック差分方式では，C のケースで約 15%圧縮率が低下している．これは，C のケースでは新規に追加されたコード部分が大きかったためである．しかし，広域ブロック差分を適用した場合は，評価したすべてのケースにおいて圧縮率の低下が 1%以内であった．これにより，提案方式では ECU に適用可能なサイズにメモリ使用量を削減しつつ，圧縮性能を維持できていることを確認できた．

(3) 更新時間の評価

更新時間の評価は，圧縮率の評価の結果最も効果の高かった広域ブロック差分で生成したデータについて，評価を実施した．本評価では，一時復元領域は RAM 上に確保した．また，従来手法 [8] による復元処理は，表 7 に示したとおり，RAM が 160 KB の車載 ECU 向けマイコンを用いた本評価環境ではメモリ不足のため搭載できない．このため，更新時間時間の評価は，提案手法についてのみ実施した．

表 9 に，本提案による差分更新時の更新時間の評価結果を示す．

差分更新の適用により CAN を転送するデータ量が削減されるため，通信時間 T_{com} を 1/10 以下に短縮できる．一方で，差分更新時に必要となる差分復元時間 T_{decomp} についても，最大で 1.8 秒であり，すべての評価プログラムにおいて更新時間全体の 1/10 以下である．これにより，本提案方式による差分復元時間は，更新に必要な時間に対して十分に小さいことが確認できた．**表 10** に通常更新および本提案による差分更新における更新時間の比較を示す．

表 10　更新時間の比較 [s]
Table 10　Comparison of update time [s].

Test program	T_{update_full} (通常)	T_{update_inc} (提案方式)	$T_{update\_inc} / T_{update\_full}$ [%]
A	**1231.0**	**28.7**	**2.3**
B	**970.1**	**73.7**	**7.6**
C	**808.2**	**37.1**	**4.6**
D	**808.0**	**105.8**	**13.1**
E	**47.0**	**5.5**	**11.7**

通常更新時と差分更新時の更新時間の比較により，差分復元時間を含めても，差分更新を適用することによって更新時間を短縮できることが確認できた．

6. おわりに

本研究では，bsdiff をベースに ECU に適用可能な差分更新方式を提案した．提案方式では，ブロック単位の差分生成とデータ構造および圧縮アルゴリズムの変更により，圧縮率を維持しつつ，ECU に搭載可能なレベルへの省メモリ化を実現した．具体的には，評価したテストケースについては圧縮率低下 1%以内で，入力データの内容によらず約 7.5 KB の RAM での差分復元を可能とした．また，提案方式を自動車向けマイコン上に実装し ECU に搭載可能なレベルに省メモリ化できたことを確認した．さらに，実機評価により，本方式を適用することで，ECU ソフトウェアの更新時間を短縮できることを確認した．

今後は，更新の高信頼化やセキュリティなど，自動車システムへの OTA 適用課題についての検討を進めていく．

参考文献

[1] Chrysler，車の遠隔操作問題で 140 万台のリコール発表，入手先 ⟨http://www.itmedia.co.jp/enterprise/articles/1507/27/news038.html⟩（参照 2016-04-16）.
[2] available from ⟨https://www.teslamotors.com/sites/default/files/pdfs/release_notes/tesla_model_s_software_7_1.pdf⟩ (accessed 2016-04-16).
[3] Bosch, R.: CAN specification version 2.0, Rober Bousch GmbH, Postfach, 300240 (1991).
[4] 清原良三：携帯電話のソフトウェアアドインに関する研究，大阪大学大学院情報科学研究科博士論文 (2008).
[5] available from ⟨http://www.pocketsoft.com/⟩ (accessed 2016-04-16).
[6] available from ⟨http://www.redbend.com/en⟩ (accessed 2016-04-16).
[7] available from ⟨http://www.daemonology.net/bsdiff/⟩ (accessed 2016-04-16).
[8] Nakanishi, T., Shih, H.H., Hisazumi, K. and Fukuda, A.: A software update scheme by airwaves for automotive equipment, *2013 International Conference on Informatics, Electronics & Vision* (*ICIEV*), pp.1–6, IEEE (May 2013).
[9] 野澤優尚，小沼　寛，清原良三：車載 ECU 向けソフト更新のためのデータ圧縮方式，研究報告マルチメディア通信と分散処理（DPS），Vol.2015, No.4, pp.1–6 (2015).
[10] 小沼　寛，野澤優尚，清原良三：bsdiff を応用した ECU ソフトウェア高速ダウンロード，情報処理学会第 78 回全国大会 (2015).
[11] Lee, Y.S., Kim, J.H., Van Hung, H. and Jeon, J.W.: A parallel re-programming method for in-vehicle gateway to save software update time, *2015 IEEE International Conference on Information and Automation*, pp.1497–1502, IEEE (Aug. 2015).
[12] Jang, S.J. and Jeon, J.W.: Software reprogramming performance analysis of CAN FD and FlexRay protocols, *2015 IEEE International Conference on Information and Automation*, pp.2535–2540, IEEE (Aug. 2015).
[13] Lee, Y.S., Kim, J.H., Jang, S.J. and Jeon, J.W.: Automotive ECU Software Reprogramming Method Based on Ethernet Backbone Network to Save Time, *Proc. 10th International Conference on Ubiquitous Information Management and Communication*, p.39, ACM (Jan. 2016).
[14] available from ⟨http://www.bzip.org/⟩.
[15] available from ⟨http://japan.renesas.com/products/mpumcu/rh850/rh850f1x/rh850f1l/index.jsp⟩.
[16] Burrows, M. and Wheeler, D.: A block-sorting lossless data compression algorithm, *DIGITAL SRC RESEARCH REPORT* (1994).
[17] Huffman, D.A.: A method for the construction of minimum-redundancy codes, *Proc. IRE*, Vol.40, No.9, pp.1098–1101 (1952).
[18] available from ⟨https://tools.ietf.org/html/rfc1951⟩.
[19] available from ⟨http://www.7-zip.org/sdk.html⟩.
[20] Ziv, J. and Lempel, A.: A universal algorithm for sequential data compression, *IEEE Trans. Inf. Theory*, Vol.23, No.3, pp.337–343 (1977).
[21] Martin, G.N.N.: Range encoding: an algorithm for removing redundancy from a digitised message, *Proc. Institution of Electronic and Radio Engineers International Conference on Video and Data Recording* (July 1979).
[22] available from ⟨https://www.toppers.jp/atk2.html⟩ (accessed 2016-04-22).
[23] available from ⟨http://www.tessera.co.jp/fl/f1l-176.html⟩ (accessed 2016-04-16).
[24] ISO 14229-1: 2013 Road vehicles—Unified diagnostic services (UDS)—Part 1: Specification and requirements (2013).
[25] ISO 15765-2: 2011 Road vehicles—Diagnostic communication over Controller Area Network (DoCAN)—Part 2: Transport protocol and network layer services (2011).

1　CAN は，Bosch 社の登録商標です．
2　FlexRay は，Daimler Chrysler AG の登録商標です．
3　Ethernet は，富士ゼロックス株式会社の登録商標です．

寺岡 秀敏

2002 年京都大学大学院工学研究科修士課程修了．（株）日立製作所．組み込みシステム，ネットワーク，車載システムに関連する研究に従事．

中原 章晴

1994年東京工科大学工学部卒業．（株）日立オートモティブシステムズ．携帯電話，車載制御装置に関連する研究開発に従事．

黒澤 憲一

1980年東北大学大学院工学研究科博士前期課程修了．（株）日立オートモティブシステムズ．車載制御装置に関連する研究開発に従事．