(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)公開番号
特開2024-70053
(P2024-70053A)
(43)公開日　令和6年5月22日(2024.5.22)

(51)国際特許分類　ＦＩ
Ｇ０６Ｆ　13/10　(2006.01)　Ｇ０６Ｆ　13/10　３１０Ｚ

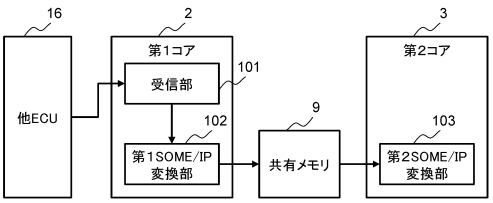
審査請求　未請求　請求項の数　14　ＯＬ　（全21頁）

(21)出願番号　特願2022-180412(P2022-180412)
(22)出願日　令和4年11月10日(2022.11.10)

(71)出願人　509186579
日立Ａｓｔｅｍｏ株式会社
茨城県ひたちなか市高場２５２０番地
(74)代理人　110001678
藤央弁理士法人
(72)発明者　永沼 新之介
茨城県ひたちなか市高場２５２０番地
日立Ａｓｔｅｍｏ株式会社内
(72)発明者　飯田 隆博
茨城県ひたちなか市高場２５２０番地
日立Ａｓｔｅｍｏ株式会社内

(54)【発明の名称】　電子制御装置、及びコア間通信方法

(57)【要約】
【課題】データ転送の信頼性が保証しつつ、大容量のデータを高速で転送する。
【解決手段】電子制御装置であって、第１プロセッサコアと、第２プロセッサコアと、前記第１プロセッサコア及び前記第２プロセッサコアがアクセス可能な共有メモリとを備え、前記第１プロセッサコアは、ＣＡＮ通信で受信したデータをＳＯＭＥ／ＩＰメッセージに変換する第１ＳＯＭＥ／ＩＰ変換部を有し、前記第１ＳＯＭＥ／ＩＰ変換部で変換されたＳＯＭＥ／ＩＰメッセージを前記共有メモリに格納し、前記第２プロセッサコアは、ＳＯＭＥ／ＩＰメッセージをアプリケーションで利用可能な形式に変換する第２ＳＯＭＥ／ＩＰ変換部を有し、前記共有メモリに格納されたＳＯＭＥ／ＩＰメッセージを取得し、前記第２ＳＯＭＥ／ＩＰ変換部で前記取得したＳＯＭＥ／ＩＰメッセージをアプリケーションで利用可能な形式に変換して、前記アプリケーションに提供する。
【選択図】図３



【特許請求の範囲】
【請求項１】
電子制御装置であって、
ＣＡＮ通信でデータを受信する第１プロセッサコアと、
アプリケーションが動作する第２プロセッサコアと、
前記第１プロセッサコア及び前記第２プロセッサコアがアクセス可能な共有メモリとを備え、
前記第１プロセッサコアは、
ＣＡＮ通信で受信したデータをＳＯＭＥ／ＩＰメッセージに変換する第１ＳＯＭＥ／ＩＰ変換部を有し、
前記第１ＳＯＭＥ／ＩＰ変換部で変換されたＳＯＭＥ／ＩＰメッセージを前記共有メモリに格納し、
前記第２プロセッサコアは、
ＳＯＭＥ／ＩＰメッセージを前記アプリケーションで利用可能な形式に変換する第２ＳＯＭＥ／ＩＰ変換部を有し、
前記共有メモリに格納されたＳＯＭＥ／ＩＰメッセージを取得し、
前記第２ＳＯＭＥ／ＩＰ変換部で前記取得したＳＯＭＥ／ＩＰメッセージを前記アプリケーションで利用可能な形式に変換して、前記アプリケーションに提供することを特徴とする電子制御装置。
【請求項２】
請求項１に記載の電子制御装置であって、
前記第１プロセッサコアは、前記第１ＳＯＭＥ／ＩＰ変換部で変換されたＳＯＭＥ／ＩＰメッセージに基づいて誤り検出符号を計算し、
前記計算された誤り検出符号をＳＯＭＥ／ＩＰメッセージと関連付けて前記共有メモリに格納することを特徴とする電子制御装置。
【請求項３】
請求項２に記載の電子制御装置であって、
前記第１プロセッサコアは、前記ＳＯＭＥ／ＩＰメッセージのヘッダーから誤り検出符号を計算し、前記計算された誤り検出符号を前記共有メモリへ格納することを特徴とする電子制御装置。
【請求項４】
請求項２に記載の電子制御装置であって、
前記第１プロセッサコアは、前記ＳＯＭＥ／ＩＰメッセージのペイロードから誤り検出符号を計算し、前記計算された誤り検出符号を前記共有メモリへ格納することを特徴とする電子制御装置。
【請求項５】
請求項４に記載の電子制御装置であって、
前記第２プロセッサコアは、前記共有メモリから取得した誤り検出符号によって、前記ＳＯＭＥ／ＩＰメッセージの異常を検出した場合、利用不可能であることを示すデータを当該ＳＯＭＥ／ＩＰメッセージに書き込むことを特徴とする電子制御装置。
【請求項６】
請求項４に記載の電子制御装置であって、
前記第２プロセッサコアは、前記共有メモリから取得した誤り検出符号によって、前記ＳＯＭＥ／ＩＰメッセージの異常を検出した場合、当該ＳＯＭＥ／ＩＰメッセージの再送を前記第１プロセッサコアに要求することを特徴とする電子制御装置。
【請求項７】
請求項１に記載の電子制御装置であって、
前記第２ＳＯＭＥ／ＩＰ変換部は、前記共有メモリから取得したＳＯＭＥ／ＩＰメッセージと、Ｅｔｈｅｒｎｅｔ経由で他のＥＣＵから転送されたＳＯＭＥ／ＩＰメッセージの両方を変換して、前記アプリケーションに提供可能であることを特徴とする電子制御装置

。
【請求項８】
請求項１に記載の電子制御装置であって、
前記第１ＳＯＭＥ／ＩＰ変換部は、ＣＡＮ通信で受信した複数のＣＡＮデータを纏めて一つのＳＯＭＥ／ＩＰメッセージに変換することを特徴とする電子制御装置。
【請求項９】
請求項１に記載の電子制御装置であって、
前記第１ＳＯＭＥ／ＩＰ変換部は、ＣＡＮ通信で受信したデータが欠損している場合、前後のデータから計算された値で欠損値を補完することを特徴とする電子制御装置。
【請求項１０】
請求項１に記載の電子制御装置であって、
前記第１ＳＯＭＥ／ＩＰ変換部は、ＣＡＮ通信で受信したデータが欠損している場合、前回値を複製して欠損値を補完することを特徴とする電子制御装置。
【請求項１１】
請求項１に記載の電子制御装置であって、
前記第２プロセッサコアは、前記共有メモリに格納されたＳＯＭＥ／ＩＰメッセージを受信する機構を有することを特徴とする電子制御装置。
【請求項１２】
請求項１１に記載の電子制御装置であって、
前記第２プロセッサコアは、前記共有メモリから取得したＳＯＭＥ／ＩＰメッセージを格納するバッファを有することを特徴とする電子制御装置。
【請求項１３】
請求項１１に記載の電子制御装置であって、
前記第２プロセッサコアで動作するアプリケーションは、前記共有メモリから取得したＳＯＭＥ／ＩＰメッセージを格納するバッファを有することを特徴とする電子制御装置。
【請求項１４】
電子制御装置が実行するコア間通信方法であって、
前記電子制御装置は、ＣＡＮプロトコルに従ってデータを受信する第１プロセッサコアと、アプリケーションが動作する第２プロセッサコアと、前記第１プロセッサコア及び前記第２プロセッサコアがアクセス可能な共有メモリとを有し、
前記コア間通信方法は、
前記第１プロセッサコアが、ＣＡＮ通信で受信したデータをＳＯＭＥ／ＩＰメッセージに変換し、前記変換されたＳＯＭＥ／ＩＰメッセージを前記共有メモリに格納し、
前記第２プロセッサコアが、前記共有メモリに格納されたＳＯＭＥ／ＩＰメッセージを取得し、前記取得したＳＯＭＥ／ＩＰメッセージを前記アプリケーションで利用可能な形式に変換して、前記アプリケーションに提供することを特徴とするコア間通信方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、自動車用通信規格のＳＯＭＥ／ＩＰを用いたデータ転送方法に関する。
【背景技術】
【０００２】
近年はＳＯＡ（Service-oriented architecture）の概念が車載ネットワークにも取り入れられ、ＳＯＭＥ／ＩＰと呼ばれるプロトコルが使用されるようになった。このＳＯＭＥ／ＩＰはＥｔｈｅｒｎｅｔ（Ｅｔｈｅｒｎｅｔは登録商標、以下同じ）に基づいて作成されたプロトコルであるため、通常はＥｔｈｅｒｎｅｔを経由してデータを転送する。ＳＯＭＥ／ＩＰを転送する際、Ｅｔｈｅｒｎｅｔ、ＩＰ、ＴＣＰと経由する各プロトコルでデータの誤りが検出され、データ転送の信頼性を保証している。
【０００３】
ＳＯＭＥ／ＩＰメッセージをマルチコア上で動作する異なるＯＳ間において転送する場

合、Ｅｔｈｅｒｎｅｔを使用して送ると、大容量データでは処理時間が長くなる。ＳＯＭＥ／ＩＰメッセージを共有メモリを介してコア間で転送する方法は確立していない。また、Ｅｔｈｅｒｎｅｔを経由することで保証されていたデータ転送の信頼性の補償が困難である。

【０００４】

ＥｔｈｅｒｎｅｔではＣＲＣと呼ばれるデータ誤り検出が採用されており、ＣＲＣによってデータ転送の信頼性を保証している。ＣＲＣはＣＡＮでも採用されており、フレームフォーマットにＣＲＣ用の領域が用意されている。そのため、これらのプロトコルはプロトコル自体にＣＲＣが実装されている。一方、ＳＯＭＥ／ＩＰメッセージフォーマットはＣＲＣ等の誤り検出を行うための値が実装されておらず、ＳＯＭＥ／ＩＰメッセージ単体ではデータの信頼性の保証が困難である。

【０００５】

本技術分野の背景技術として、以下の先行技術がある。特許文献１（国際公開２０２０／２１７２０２号）には、乗物のオンボードの通信ネットワークにおいて、ＳＯＭＥ／ＩＰ通信プロトコルを用いて、サービスインスタンスの要求側エンティティ及び提供側エンティティの間でメッセージを伝送する。サービスインスタンスに関連付けられた通信を考慮して、要求側及び提供側の相互認証を行う。本ステップは、サービスインスタンスへのアクセスを認可する、要求側及び提供側の予め割り当てられた証明書の存在及び相互の有効性を検証することと、提供側によってサービスが提供されるセキュリティレベルが、要求側及び提供側においてサービスに予め割り当てられた最低のセキュリティレベル未満ではないことを検証することとを含む。証明書及びセキュリティレベルの検証に成功した場合、サービスインスタンスに関連付けられた少なくとも１つの通信メッセージを提供側から要求側に、及びその逆に伝送する。ＳＯＭＥ／ＩＰ通信プロトコルを用いるメッセージ又はデータの伝送方法が記載されている。

【先行技術文献】

【特許文献】

【０００６】

【特許文献１】国際公開２０２０／２１７２０２号

【発明の概要】

【発明が解決しようとする課題】

【０００７】

マルチコアのコア間通信では、大容量のデータを高速で転送できる方法が求められており、また、データ転送の信頼性の保証も重要である。前述したように、通常、ＳＯＭＥ／ＩＰメッセージはＥｔｈｅｒｎｅｔを経由して転送されるが、共有メモリを使用する方が大容量のデータを高速で転送できる。

【０００８】

しかし、共有メモリを使用して、Ｅｔｈｅｒｎｅｔを経由しないことにより、Ｅｔｈｅｒｎｅｔ、ＩＰ、ＴＣＰ等のプロトコルにおけるＣＲＣ等のデータ誤り検出が実施されないため、データ転送の信頼性が保証されない。

【０００９】

特許文献１は、ＳＯＭＥ／ＩＰ通信プロトコルを用いる乗り物上におけるデータまたはメッセージの伝送の改良を開示しており、本発明で解決しようとしている、ＣＡＮデータのＳＯＭＥ／ＩＰメッセージへの変換や、ＳＯＭＥ／ＩＰメッセージを利用し、異なるＯＳのコア間通信を実現するものではなく、また、ＳＯＭＥ／ＩＰプロトコルの信頼性を高めるものでもない。

【００１０】

また、ＣＡＮで転送された受信データをコア間で転送する場合、受信側コアではアプリケーションでデータを使用可能にするサービス変換が必要となる。このような処理を行うＳ２Ｓ（signal to service）は、処理が遅いという問題がある。ＳＯＭＥ／ＩＰプロトコルを用いる場合は、このＳ２Ｓを使用する必要がなくなる。

【００１１】
本発明は、共有メモリを使用したコア間通信で、データ転送の信頼性が保証しつつ、大容量のデータを高速で転送する方法の提供を目的とする。
【課題を解決するための手段】
【００１２】
本願において開示される発明の代表的な一例を示せば以下の通りである。すなわち、電子制御装置であって、ＣＡＮ通信でデータを受信する第１プロセッサコアと、アプリケーションが動作する第２プロセッサコアと、前記第１プロセッサコア及び前記第２プロセッサコアがアクセス可能な共有メモリとを備え、前記第１プロセッサコアは、ＣＡＮ通信で受信したデータをＳＯＭＥ／ＩＰメッセージに変換する第１ＳＯＭＥ／ＩＰ変換部を有し、前記第１ＳＯＭＥ／ＩＰ変換部で変換されたＳＯＭＥ／ＩＰメッセージを前記共有メモリに格納し、前記第２プロセッサコアは、ＳＯＭＥ／ＩＰメッセージを前記アプリケーションで利用可能な形式に変換する第２ＳＯＭＥ／ＩＰ変換部を有し、前記共有メモリに格納されたＳＯＭＥ／ＩＰメッセージを取得し、前記第２ＳＯＭＥ／ＩＰ変換部で前記取得したＳＯＭＥ／ＩＰメッセージを前記アプリケーションで利用可能な形式に変換して、前記アプリケーションに提供することを特徴とする。
【発明の効果】
【００１３】
本発明の一態様によれば、共有メモリを使用したコア間通信で、データ転送の信頼性が保証しつつ、大容量のデータを高速で転送できる。前述した以外の課題、構成及び効果は、以下の実施例の説明によって明らかにされる。
【図面の簡単な説明】
【００１４】
【図１】実施例１のマルチコアマイコンの構成を示す図である。
【図２】実施例１のＥＣＵの構成を示す図である。
【図３】実施例１の各コアで実行される処理を示すブロック図である。
【図４】実施例１のＣＡＮとＣＡＮ　ＦＤのフレームフォーマットを示す図である。
【図５】実施例１のＳＯＭＥ／ＩＰのメッセージフォーマットを示す図である。
【図６】実施例１のＣＡＮデータを使用して作成されたＳＯＭＥ／ＩＰメッセージを示す図である。
【図７】実施例２のＳＯＭＥ／ＩＰメッセージに付加されるＣＲＣ値を示す図である。
【図８】実施例２のＣＲＣの演算処理を示す図である。
【図９】実施例２のＰａｙｌｏａｄ　ＣＲＣ誤りを検出した場合にアプリに通知する処理のフローチャートである。
【図１０】実施例２のＰａｙｌｏａｄ　ＣＲＣ誤りを検出した場合に再送を要求する処理のフローチャートである。
【図１１】実施例３の第２ＳＯＭＥ／ＩＰ変換部の周辺の構成を示す図である。
【図１２】実施例４の連続データに生じた欠損値の補完方法を示す図である。
【図１３】実施例４の連続データに生じた欠損を補完するデータ補完処理のフローチャートである。
【図１４】実施例４の離散データに生じた欠損値の補完方法を示す図である。
【図１５】実施例４の離散データに生じた欠損を補完するデータ補完処理のフローチャートである。
【図１６】実施例５の第２ＳＯＭＥ／ＩＰ変換部の周辺の構成を示す図である。
【図１７】実施例５の第２ＳＯＭＥ／ＩＰ変換部の周辺の別の構成を示す図である。
【発明を実施するための形態】
【００１５】
＜実施例１＞
図１は、マルチコアマイコン１の構成を示す図である。本実施例のマルチコアマイコン１は、処理を実行する第１コア２、処理を実行する第２コア３、第１コアＲＡＭ４、第２

コアＲＡＭ５、第１コア不揮発性メモリ６、第２不揮発性メモリ７、マルチコアマイコン外部とデータを送受信する通信コントローラ８、及び第１コア２と第２コア３とで参照できるＲＡＭである共有メモリ９を有する。また、各構成要素は、内部バス１０を通して接続される。

【００１６】

マルチコアマイコン１は、二つ以上のコアが搭載されているマイクロコンピュータである。コアが一つのマイコンは、シングルコアマイコンと呼ばれる。マイコンのパフォーマンスを高める場合、マイコン内コアの動作周波数を高くして処理のスピードを上げる方法と、単に処理に用いるマイコン内のコアの数を増やす方法とがある。動作周波数を高くする場合は電力消費が大きくなるが、マルチコアマイコンでは低い動作周波数でもパフォーマンスが向上し消費電力を抑制できる。本実施例では２コアのマイコンを使用して説明する。

【００１７】

第１コア２及び第２コア３は、実際に処理を実行するデバイスであり、様々なデバイスからデータを受け取り、様々な演算を行う。前述したとおり、コアの動作周波数を上げることでパフォーマンスが向上するが、消費電力が増加する。

【００１８】

第１コアＲＡＭ４は、第１コア２が処理を実行する際に使用する作業用のメインメモリである。また、第２コアＲＡＭ５は、第２コア３が処理を実行する際に使用する作業用のメインメモリである。揮発性メモリデバイスで構成され、マイコンの電源が切れると使用していたデータが消去される。

【００１９】

内部バスとは、マルチコアマイコン１の内部の回路をつなぐ伝送路である。

【００２０】

マルチコアマイコン１は、車両に搭載される電子制御装置（ＥＣＵ）に実装されている。ＥＣＵは、エンジン、パワーステアリング装置、制御ブレーキ装置などの車両搭載機器を制御する。

【００２１】

図２は、ＥＣＵ１０の構成を示す図である。ＥＣＵ１０は、図１で述べたマルチコアマイコン１の他に、ＣＡＮとの通信を可能とするＣＡＮトランシーバ１１やＥｔｈｅｒｎｅｔ接続を可能とするＬＡＮポート１２等を有する（Ｅｔｈｅｒｎｅｔは登録商標、以下同じ）。ＣＡＮトランシーバ１１は、後述するＣＡＮバス１５に接続される。他ＥＣＵ１６からＣＡＮバス１５を経由してＣＡＮトランシーバ１１が受け取る信号やＬＡＮ１２からＥｔｈｅｒｎｅｔを経由してＬＡＮポート１２が受け取る信号は、マルチコアマイコン１の通信コントローラ８に送られる。通信コントローラ８は、ＣＡＮ経由の信号を扱うＣＡＮコントローラ１３とＥｔｈｅｒｎｅｔ経由の信号を扱うイーサスイッチ１４等を有する。

【００２２】

ＣＡＮバス１５は、ＣＡＮプロトコルでデータを転送する伝送路であり、ＣＡＮ　ＨｉｇｈとＣＡＮ　Ｌｏｗと呼ばれる２本の通信線で構成される。ＣＡＮではこの２本の通信線に印加される電圧の差で０と１のＣＡＮ信号を送信する。このような形式によって、ノイズ耐性を向上できる。ＣＡＮプロトコルの詳細は後述する。

【００２３】

ＣＡＮコントローラ１３は、ＣＡＮバス１５で転送されるデータのうち、必要なデータは割り込みを発生させて受信し、不要なデータは無視するなどの機能を備えたモジュールである。

【００２４】

イーサスイッチ１４は、Ｅｔｈｅｒｎｅｔ　Ｓｗｉｔｃｈの略であり、ＬＡＮポート１２から受け取ったＥｔｈｅｒｎｅｔのデータの転送先を制御するモジュールである。イーサスイッチ１４は、Ｅｔｈｅｒｎｅｔで転送されるデータに含まれる送信元や宛先などの

情報に基づいてデータの転送先を制御する。

【００２５】

図３は、各コアで実行される処理を示すブロック図である。マルチコアマイコン１は、後述するＣＡＮデータを受信する受信部１０１を有する第１コア２と、第１コア２で受け取ったデータを利用する第２コア３と、二つのコア間２、３の間のデータの移動を可能とするための共有メモリ９を有する。本実施例では、第１コア２に搭載された受信部１０１が、他ＥＣＵ１６からＣＡＮデータを受信し、第１ＳＯＭＥ／ＩＰ変換部１０２が、後述するＣＡＮの実データを取り出して、ＳＯＭＥ／ＩＰのメッセージフォーマットに従ったＳＯＭＥ／ＩＰメッセージを作成し、共有メモリ９を介して、第２コア３に転送する。第２コア３では、第２ＳＯＭＥ／ＩＰ変換部１０３が、サービス変換を行い、アプリケーションへのデータ転送を可能とする。

【００２６】

共有メモリ９は、複数のコアのいずれからも参照できるようなＲＡＭ領域である。共有メモリ９を使用することで、外部の通信経路を経由せずに、データを異なるコアに転送できる。

【００２７】

ＣＡＮは、Ｃｏｎｔｒｏｌｌｅｒ　Ａｒｅａ　Ｎｅｔｗｏｒｋの略であり、車載ネットワークプロトコルのうち、現在最も普及しているプロトコルである。

【００２８】

ＣＡＮを経由して転送されるデータは、フレーム単位で定義されており、ＣＡＮデータの１フレームにはＣＡＮを経由して転送されるフレームを識別するためのＩＤ等の様々な情報が含まれる。フレームに含まれるデータのうち、実際にアプリケーションなどで制御に使用されるデータは、データフィールドと呼ばれる領域に格納されており、０－８ｂｙｔｅの範囲で転送される。ＣＡＮのフレームフォーマットは後述する。この他にもＣＡＮを拡張したＣＡＮ　ＦＤと呼ばれるプロトコルもある。ＣＡＮ　ＦＤは自動車の機能の増化に伴って開発された通信プロトコルで、ＣＡＮより高速に多くのデータを転送できる。ＣＡＮ　ＦＤプロトコルによって転送されるフレームのデータフィールドは、０－６４ｂｙｔｅの範囲でデータが格納される。ＣＡＮ　ＦＤのフレームフォーマットは後述する。

【００２９】

プロトコルとは、コンピュータ上でデータのやり取りをするために取り決められた、通信の形式や手順であり、取り決められたプロトコルに従うことによって、異なるメーカー同士の部品でも通信できる。

【００３０】

ＳＯＭＥ／ＩＰは、後述するＳＯＡという概念を車載ネットワークで実現するためにＥｔｈｅｒｎｅｔをベースとして作成されたプロトコルであり、後述するＯＳＩ参照モデルでは上位層に分類される。ＳＯＭＥ／ＩＰでは、車載プリケーションの機能や処理がサービス単位で定義され、これらのサービスを利用するときは、車載ネットワーク上からサービスを検索する。サービスを検索するＥＣＵをクライアント、サービス提供するＥＣＵをサーバーとすると、検索されたサービスを提供するサーバーは、サービスが提供可能であることをクライアントへ通知する。クライアントは、通知によってサービスを確認したら、データの送信をサーバーへ要求する。サーバーは、問題なければ、要求を承認し、周期的又は値が変更されるタイミングで、クライアントへデータを送信する。

【００３１】

ＳＯＡは、Ｓｅｒｖｉｃｅ　Ｏｒｉｅｎｔｅｄ　Ａｒｃｈｉｔｅｃｔｕｒｅの略であり、コンピュータシステムを構築する際の一つの考え方である。ＳＯＡでは、アプリケーションの機能をサービス単位で部品化し、アプリケーションの外部からサービスを利用できる状態にする。部品化されたサービスを組み合わせて、システムを構築し、実装できる。

【００３２】

ＯＳＩ参照モデルは、異なるネットワークアーキテクチャを統一するための概念である。ＯＳＩ参照モデルでは７階層に分類されている。

７層：アプリケーション層
６層：プレゼンテーション層
５層：セッション層
４層：トランスポート層
３層：ネットワーク層
２層：データリンク層
１層：物理層

【００３３】

上位層ほどソフトウェアによる設定が必要となり、下層ほどハードウェアに関する設定が必要となる。例えば、Ｅｔｈｅｒｎｅｔでは、物理層はＬＡＮケーブル等を差し込む接続口であり。データリンク層はＥｔｈｅｒｎｅｔのフレームを受け取る層である。ネットワーク層とトランスポート層はＥｔｈｅｒｎｅｔに乗せられるＴＣＰやＩＰである。このように細かく分類されているが、ＳＯＭＥ／ＩＰはセッション層からアプリケーション層を包含するプロトコルである。このように複数の層をまたぐプロトコルも存在し、例えばＣＡＮも物理層とデータリンク層をまたぐプロトコルである。

【００３４】

図４は、ＣＡＮとＣＡＮ　ＦＤのフレームフォーマットを示す図である。他のＥＣＵから受け取るＣＡＮのデータは、ＣＡＮプロトコル（ＣＡＮ　ＦＤを含む）のフォーマットで転送されるデータであり、このデータの中には、ＩＤ、コントロールフィールド、データフィールド２０１，２０２、ＣＲＣ等のＣＡＮプロトコル特有の情報が含まれる。

【００３５】

ＳＯＭＥ／ＩＰのメッセージフォーマットに変換して使用されるＣＡＮの実データは、ＣＡＮデータのうちデータフィールドに格納されたデータ２０１ａ、２１１ａ、２１１ｂ等である。ＣＡＮとＣＡＮ　ＦＤ共に、複数のデータを転送可能であり、両者はデータの最大値と通信速度が異なる。

【００３６】

ＳＯＭＥ／ＩＰメッセージフォーマットは、ＳＯＭＥ／ＩＰプロトコルで使用できるデータの形式である、ＳＯＭＥ／ＩＰメッセージフォーマットに従って作成されたデータの纏まりをＳＯＭＥ／ＩＰメッセージと定義する。以下、ＳＯＭＥ／ＩＰメッセージフォーマットを説明する。

【００３７】

図５は、ＳＯＭＥ／ＩＰのメッセージフォーマットを示す図である。第１ＳＯＭＥ／ＩＰ変換部１１３は、ＣＡＮの実データに基づいて、ＳＯＭＥ／ＩＰメッセージを作成する処理を実行する。ＳＯＭＥ／ＩＰのメッセージフォーマットは、下記の情報を必要とする。

Service ID
Method ID
Length
Client ID
Session ID
Protocol Version
Interface Version
Message Type
Return Code
Payload

【００３８】

それぞれの領域を説明する。

【００３９】

Ｓｅｒｖｉｃｅ　ＩＤは、後述するサービスの種類を示す。Ｍｅｔｈｏｄ　ＩＤは、Ｍｅｔｈｏｄの種類を示す。Ｓｅｒｖｉｃｅ　ＩＤとＭｅｔｈｏｄ　ＩＤを纏めてＭｅｓｓ

ａｇｅ　ＩＤと呼ぶこともある。
【００４０】
　Ｌｅｎｇｔｈは、Ｃｌｉｅｎｔ　ＩＤからＰａｙｌｏａｄまでのメッセージ長である。
【００４１】
　Ｃｌｉｅｎｔ　ＩＤは、ＥＣＵ内の呼び出し元を識別するための識別情報である。Ｓｅｓｓｉｏｎ　ＩＤは、同一の送信者から転送されたメッセージを区別するための識別情報である。Ｃｌｉｅｎｔ　ＩＤとＳｅｓｓｉｏｎ　ＩＤを纏めてＲｅｑｕｅｓｔ　ＩＤと呼ぶこともある。
【００４２】
　Ｐｒｏｔｏｃｏｌ　Ｖｅｒｓｉｏｎは、ＳＯＭＥ／ＩＰのプロトコルバージョンを示す。
【００４３】
　Ｉｎｔｅｒｆａｃｅ　Ｖｅｒｓｉｏｎは、後述するサービスインターフェイスのメジャーバージョンを示す。Ｍｅｓｓａｇｅ　Ｔｙｐｅは、ＳＯＭＥ／ＩＰのメッセージのタイプを示す。
【００４４】
　Ｒｅｔｕｒｎ　Ｃｏｄｅは、ｒｅｑｕｅｓｔが正常に処理されたかを示す戻り値である。Ｐａｙｌｏａｄは、データである。
【００４５】
　ここで、サービスとは、各ＥＣＵが提供するアプリケーションであり、アプリケーション内に含まれる関数がＭｅｔｈｏｄである。
【００４６】
　Ｍｅｔｈｏｄは、ＳＯＭＥ／ＩＰを使用する際の一つの考え方であり、サービス内に含まれる関数を指す。具体的には、前述したＳＯＭＥ／ＩＰプロトコルで説明した、サービスを要求するアプリケーションをクライアント、サービスと提供するアプリケーションをサーバーとしている。サービスは、１又は複数の関数の組み合わせで構成され、関数はＭｅｔｈｏｄという単位で管理され、各ＭｅｔｈｏｄにはＭｅｔｈｏｄ　ＩＤが付与される。ＳＯＭＥ／ＩＰでは、どのサービスのどのＭｅｔｈｏｄ（関数）から取得したデータであるかをヘッダー情報に記載する。
【００４７】
　サービスインターフェイスは、サービスを持つ機能にアクセスするためのインターフェースである。ＳＯＭＥ／ＩＰでサービスを使用する場合、サービスインターフェイスを通してデータを取得する。
【００４８】
　前述したＳＯＭＥ／ＩＰメッセージフォーマットのうちＰａｙｌｏａｄ以外の情報３０１をＨｅａｄｅｒと呼び、ＣＡＮの実データが格納されるＰａｙｌｏａｄ部分３０２を単にＰａｙｌｏａｄと呼ぶ。
【００４９】
　図６に、ＣＡＮデータを使用して作成されたＳＯＭＥ／ＩＰメッセージを示す。ＳＯＭＥ／ＩＰ作成の段階ではアプリケーションへ向けたシリアライズは行われないため、実データをそのまま変数として格納できる。第１ＳＯＭＥ／ＩＰ変換部では、ＣＡＮデータに含まれるＩＤに基づいてヘッダーを作成し、図４に示すＣＡＮの実データ２０１ａ、２１１ａ、２１１ｂが、ＳＯＭＥ／ＩＰメッセージフォーマットのＰａｙｌｏａｄ部分へ格納される。本実施例では、ＣＡＮデータのＩＤによりＳＯＭＥ／ＩＰのＨｅａｄｅｒを作成しており、ＩＤが異なるＣＡＮの実データは同一のＳＯＭＥ／ＩＰパケットに纏めることはできない。ＣＡＮデータのＩＤにより、ＳＯＭＥ／ＩＰメッセージを通して、アプリケーション側で受信するＣＡＮデータの種類を判別できる。
【００５０】
　ＳＯＭＥ／ＩＰは、通常、Ｅｔｈｅｒｎｅｔを経由して転送されるデータの上位プロトコルとなっている。本実施例では、共有メモリ９を介してＳＯＭＥ／ＩＰメッセージを転送することで、大容量データのコア間通信において、Ｅｔｈｅｒｎｅｔ通信経由の場合に

比べ、高速で通信できる。また、第１コア側でＳＯＭＥ／ＩＰに変換することにより、言語環境が異なるコア間においても、二つのコアでデータを利用可能となり、データ利用における利便性を向上できる。

【００５１】

＜実施例２＞

本実施例において、作成したＳＯＭＥ／ＩＰメッセージと合わせて、ＣＲＣを計算し共有メモリ９に格納する。ＣＲＣは、巡回冗長検査とも呼び、データの誤り検出符号の一つである。ＣＡＮのフレームや、Ｅｔｈｅｒｎｅｔのフレームでも、ＣＲＣは用いられており、ＣＡＮやＥｔｈｅｒｎｅｔのフレームにはＣＲＣ値が含まれている。フレームに含まれる情報に基づいてＣＲＣ値を算出し、付加されたＣＲＣ値が受信時に検査される。しかし、ＳＯＭＥ／ＩＰのメッセージフォーマットにはＣＲＣフィールドが定義されていないので、ＳＯＭＥ／ＩＰ単体での誤り検出は不可能である。前述したようにＳＯＭＥ／ＩＰは通常Ｅｔｈｅｒｎｅｔで送られることを前提として作成されたプロトコルであり、ＳＯＭＥ／ＩＰがアプリに届くまでに経由する、Ｅｔｈｅｒｎｅｔ、ＩＰ、ＴＣＰ等の各プロトコルで誤り検出が実施される。そのため、作成したＳＯＭＥ／ＩＰメッセージを共有メモリ９を介して転送する場合、同様の誤り検出を実施することで、データの信頼性を保証できる。そこで、本実施例では、ＳＯＭＥ／ＩＰメッセージにＣＲＣ値が含まれないため（ＣＲＣ値を含めた場合、ＳＯＭＥ／ＩＰのメッセージフォーマットと異なるため受信不可となる）、作成したＳＯＭＥ／ＩＰメッセージと別に算出したＣＲＣ値を共有メモリ９へ格納することとした。

【００５２】

図７は、ＳＯＭＥ／ＩＰメッセージに付加されるＣＲＣ値を示す図である。本実施例では、Ｈｅａｄｅｒ３０１に基づいて算出されるＨｅａｄｅｒ　ＣＲＣと、Ｐａｙｌｏａｄ３０２に基づいて算出されるＰａｙｌｏａｄ　ＣＲＣの二つのＣＲＣをＳＯＭＥ／ＩＰメッセージに付加する。例えば、作成されたＳＯＭＥ／ＩＰメッセージの後続するアドレスにＣＲＣを格納することによって、ＣＲＣとＳＯＭＥ／ＩＰメッセージを関連付けて、ＣＲＣをＳＯＭＥ／ＩＰメッセージに付加する。実施例では、二つのＣＲＣのそれぞれに４ｂｙｔｅを使用している。これにより、転送されたデータに誤りがあるかを判定でき、ＨｅａｄｅｒとＰａｙｌｏａｄのいずれに誤りがあるかを区分けできる。

【００５３】

図８を参照して、本実施例におけるＣＲＣの演算処理を説明する。

【００５４】

ステップ４０１：第１コア２がＣＡＮのデータを受信後、ＣＡＮフレームに含まれるＩＤに基づいて、ＳＯＭＥ／ＩＰのメッセージフォーマット合わせてデータを変換し、ヘッダーなどの情報を作成して、ＳＯＭＥ／ＩＰメッセージを作成する。

【００５５】

ステップ４０２：ステップ４０１で作成されたＳＯＭＥ／ＩＰメッセージのうち、先頭１６ｂｙｔｅのＨｅａｄｅｒの情報に基づいてＣＲＣを算出する。本実施例ではＨｅａｄｅｒ　ＣＲＣは４ｂｙｔｅの固定長とする。

【００５６】

ステップ４０３：ステップ４０１で作成されたＳＯＭＥ／ＩＰメッセージのうち、先頭１６ｂｙｔｅより後のＰａｙｌｏａｄの情報に基づいてＣＲＣを算出する。本実施例ではＰａｙｌｏａｄ　ＣＲＣはＰａｙｌｏａｄのデータ長により変化することなく４ｂｙｔｅの固定長とする。

【００５７】

ステップ４０４：作成されたＳＯＭＥ／ＩＰメッセージを共有メモリに格納し、当該ＳＯＭＥ／ＩＰメッセージの後ろに続けて４ｂｙｔｅずつＨｅａｄｅｒ　ＣＲＣとＰａｙｌｏａｄ　ＣＲＣの順番で格納する。

【００５８】

Ｈｅａｄｅｒ　ＣＲＣによって、ＳＯＭＥ／ＩＰ変換部１０３で読み込む前に不正デー

タを検出できる。また、Ｐａｙｌｏａｄ　ＣＲＣによって、データがアプリで使用可能な正しいデータかを判別できる。

【００５９】

Ｐａｙｌｏａｄ　ＣＲＣによって、データに誤りがあると判定される場合、二つの処理（図９、図１０）が可能である。ＰａｙｌｏａｄのＣＲＣで誤りを検出した際の対応について説明する。

【００６０】

図９は、Ｐａｙｌｏａｄ　ＣＲＣ誤りを検出した場合にアプリに通知する処理のフローチャートである。

【００６１】

ステップ５０１：まず、第１コア側の処理完了を検出する。本実施例で用いるマイコンには、処理結果が共有メモリ９に格納された後に格納完了を通知する割り込み処理機能があるため、この割り込みによって処理完了を検出する。処理完了検出の実装方法は様々な方法を採用できる。

【００６２】

ステップ５０２：ステップ５０１において第１コア側の処理完了を検出したら、共有メモリ９からＳＯＭＥ／ＩＰデータとＣＲＣデータを取り出す。データは、先頭アドレスから１６ｂｙｔｅがＳＯＭＥ／ＩＰのＨｅａｄｅｒとなっている。Ｐａｙｌｏａｄは可変長データであるため、Ｈｅａｄｅｒに含まれるＬｅｎｇｔｈに記録されたデータに基づいてＰａｙｌｏａｄの長さを計算する。また、本実施例において、ＣＲＣは、４ｂｙｔｅの固定長のＨｅａｄｅｒ　ＣＲＣと４ｂｙｔｅの固定長のＰａｙｌｏａｄ　ＣＲＣがＰａｙｌｏａｄの直後に格納されているため、Ｐａｙｌｏａｄの直後の８ｂｙｔｅをＣＲＣとして取り出して、４ｂｙｔｅのＨｅａｄｅｒ　ＣＲＣと４ｂｙｔｅのＰａｙｌｏａｄ　ＣＲＣに分離する。

【００６３】

ステップ５０３：ステップ５０２により取り出されたＳＯＭＥ／ＩＰとＨｅａｄｅｒ　ＣＲＣを用いてＣＲＣチェックを行う。ＳＯＭＥ／ＩＰメッセージのＨｅａｄｅｒは先頭１６ｂｙｔｅの固定長であり、この１６ｂｙｔｅのデータに基づいてＣＲＣを計算する。ＣＲＣを計算したら、共有メモリからＳＯＭＥ／ＩＰメッセージと同時に取り出したＨｅａｄｅｒ　ＣＲＣと値を比較し、両者の値が一致すれば「１」、一致しなければ「０」のように、結果を判定するためのフラグを格納する。

【００６４】

ステップ５０４：ステップ５０３の結果が一致を示す「１」である場合、ステップ５０６へ遷移する。ステップ５０３の結果が不一致を示す「０」である場合、ステップ５０５へ遷移する。

【００６５】

ステップ５０５：ステップ５０４においてＳＯＭＥ／ＩＰのＨｅａｄｅｒに異常があると判定されたので、ＳＯＭＥ／ＩＰのデータを正常に読み込めないため、受け取ったＳＯＭＥ／ＩＰメッセージを破棄する。破棄の後、特に処理を行うことなく処理を終了する。

【００６６】

ステップ５０６：ステップ５０４においてＳＯＭＥ／ＩＰのＨｅａｄｅｒに異常がないと判定されたので、ＳＯＭＥ／ＩＰのＰａｙｌｏａｄのＣＲＣチェックを行う。Ｈｅａｄｅｒ情報に含まれるＬｅｎｇｔｈに記録されたデータに基づいてＳＯＭＥ／ＩＰメッセージのＰａｙｌｏａｄ部分を読み込み、読み込んだＰａｙｌｏａｄに基づいてＣＲＣを計算する。ＣＲＣを計算したら、共有メモリからＳＯＭＥ／ＩＰメッセージと同時に取り出したＰａｙｌｏａｄ　ＣＲＣと比較し、両者の値が一致すれば「１」、一致しなければ「０」のように、結果を判定するためのフラグを格納する。

【００６７】

ステップ５０７：ステップ５０６の結果が一致を示す「１」である場合、ステップ５０９へ遷移する。ステップ５０６の結果が不一致を示す「０」である場合、ステップ５０８

へ遷移する。
【００６８】
ステップ５０８：ステップ５０７でＰａｙｌｏａｄに異常があると判定された場合、ＳＯＭＥ／ＩＰのＰａｙｌｏａｄ部分に利用不可というステータスを上書きする。Ｐａｙｌｏａｄの長さはこの時点で決まっているため、利用不可のステータスとして、エラーが分かる数値をＰａｙｌｏａｄのデータ長に書き込む。なお、エラーが分かる数値は、Ｐａｙｌｏａｄの一部（例えば、先頭から所定長）に書き込んでもよい。
【００６９】
ステップ５０９：ステップ５０２で取り出したＳＯＭＥ／ＩＰのＨｅａｄｅｒと、ステップ５０７の判定結果によってステップ５０６で読み込んだＰａｙｌｏａｄ又はステップ５０８でエラーが分かる数値が上書きされたＰａｙｌｏａｄをアプリケーションへ転送する。アプリケーションへ転送されるデータは、ＳＯＭＥ／ＩＰのＨｅａｄｅｒとＰａｙｌｏａｄが組み合わされたＳＯＭＥ／ＩＰメッセージのみで、ＣＲＣは転送不要のため破棄する。
【００７０】
本実施例によってＰａｙｌｏａｄのデータに異常がある場合、アプリケーションへデータが利用不可となっていることを通知できる。
【００７１】
また、ＰａｙｌｏａｄのＣＲＣに異常がある場合、再送を要求してもよい。
【００７２】
図１０は、Ｐａｙｌｏａｄ　ＣＲＣ誤りを検出した場合に再送を要求する処理のフローチャートである。
【００７３】
図１０に示す処理において、ステップ５０１～ステップ５０７及びステップ５０９は、図９に示す処理と同じである。
【００７４】
ステップ６０１：ステップ５０７でＰａｙｌｏａｄに異常があると判定された場合、第１コア側へ再送を要求する。再送要求の実装方法は様々な方法を採用できる。再送要求の後、アプリケーションへデータを転送することなく処理を終了する。
【００７５】
図１０に示す処理では、Ｐａｙｌｏａｄに異常がある場合、第１コア側へデータの再送を要求できる。
【００７６】
また、ＨｅａｄｅｒとＰａｙｌｏａｄの二つのＣＲＣを付加することで、ＳＯＭＥ／ＩＰメッセージの信頼性を高め、さらに、異常がある場合にＳＯＭＥ／ＩＰメッセージのどの部分に誤りがあるかを検出できるようになり、フレームの破棄、アプリへの利用不可の通知、再送要求などの、異常の原因に適する処理を実行できる。
【００７７】
＜実施例３＞
図１１は、実施例３の第２コア３に含まれる第２ＳＯＭＥ／ＩＰ変換部１０３の周辺の構成を示す図である。
【００７８】
実施例１及び実施例２において、第２コア３の第２ＳＯＭＥ／ＩＰ変換部１０３は、予め実装されているＳＯＭＥ／ＩＰの処理関数によって構成でき、本実施例のために新たに作成される必要はない。第２ＳＯＥＭ／ＩＰ変換部１０３の実装方法はコアの仕様によって変わるが、入力と出力はＳＯＭＥ／ＩＰの規格に従っている。そのため、共有メモリ９からＣＲＣ演算部７０１を通して転送されるＳＯＭＥ／ＩＰメッセージと、通常のＥｔｈｅｒｎｅｔ受信部７０２を経由して転送されるＳＯＭＥ／ＩＰメッセージは、必要とするインターフェースが同一であり、同一の第２ＳＯＭＥ／ＩＰ変換部１０３で処理できる。
【００７９】

Ｅｔｈｅｒｎｅｔ経由で転送されるＳＯＭＥ／ＩＰメッセージは、ＥｔｈｅｒｎｅｔフレームのＰａｙｌｏａｄ部分に含まれるＳＯＭＥ／ＩＰメッセージであり、第２ＳＯＭＥ／ＩＰ変換部１０３で受信する際にＥｔｈｅｒｎｅｔ、ＩＰ、ＴＣＰ等のプロトコルで使用されるヘッダー情報が削除された状態である。また、インターフェースが同一とは、共有メモリ９から受信するＳＯＭＥ／ＩＰメッセージと、Ｅｔｈｅｒｎｅｔから受信するＳＯＭＥ／ＩＰメッセージで扱いが同じであり、第２ＳＯＭＥ／ＩＰ変換部１０３で使用される引数や出力方法は同一であることを意味する。

【００８０】

このような構成にすることで、構成の複雑化を抑制できる。構成の複雑化の抑制によって、ソフトウェアの保守性が向上し、不具合発生時に容易にデバックできる。

【００８１】

＜実施例４＞

本実施例において、第１コア２が受信するＣＡＮのデータは必ずしも正常な値を受信できるとは限らない。データ送信元で異常があった場合に欠損値が生じることがある。ＣＡＮの規格に、欠損値がある場合に再送する処理は設けられていない。そのため欠損値が生じた場合、アプリケーションで正しく処理できない可能性がある。そのため、そのようなケースに対応するため、第１ＳＯＭＥ／ＩＰ変換部１０２にデータを補完する機能を持たせる。欠損値は、ＣＡＮのフレームで転送されるデータに基づいて検出できる。欠損値を検出した場合、アプリケーションで対応できるが、本実施例では、アプリケーションより手前で、欠損値に対する処理を実行する。

【００８２】

図１２は、連続データに生じた欠損値の補完方法を示す図である。

【００８３】

実際に受信した連続値８０１ａ、８０１ｂ、８０１ｃ、８０１ｄにおいて、値８０１ｂと値８０１ｃの間の３番目に値の欠損が生じた場合、アプリケーションの仕様に合わせたデータ８０２（例えば、アプリケーションの動作に影響しないように前後値から計算される値８０２）で欠損値を補完する。

【００８４】

図１３は、連続データに生じた欠損を補完するデータ補完処理のフローチャートである。

【００８５】

ステップ９０１：ＣＡＮのデータを受信する。ＣＡＮのデータを受信するＥＣＵのＣＡＮのＩＤを登録しておき、指定のＩＤを持つＣＡＮの信号を到来したら、ＣＡＮのデータを受信する。

【００８６】

ステップ９０２：受信したＣＡＮのデータのＩＤやデータフィールドを確認し、データの種類や、欠損値の有無を判定する。ＣＡＮのフレームのデータフィールドには、複数のデータが含まれていることがあり、各データの区切りはバイトポジションなどで、静的に定義される。バイトポジションとは、ＣＡＮのフレーム内のデータフィールドに含まれるデータの始まりの位置を示すものであり、ＣＡＮで転送されるフレームの内容は静的に定義されている。

【００８７】

ステップ９０３：受信したＣＡＮのデータが値の欠損を示す場合、ステップ９０４へ進む。受信したＣＡＮのデータが値の欠損がないことを示す場合、ステップ９０５へ進む。

【００８８】

ステップ９０４：データが欠損している場合、受信したＣＡＮのデータのＩＤに基づいて連続値か離散値かを判定し、連続値であればＣＡＮのデータフィールドの値を、前後値から計算した値（例えば、前回値と次回値との平均値、近似関数を用いて前回値と次回値から計算される値）で書き換え、ステップ９０５へ進む

【００８９】

ステップ９０５：ステップ９０３又はステップ９０４から受け取ったデータに基づいてＳＯＭＥ／ＩＰメッセージを作成する。受け取ったＣＡＮのデータから、ＩＤに基づいてヘッダーを作成し、データフィールドの値をＰａｙｌｏａｄへ格納する。受け取るデータに応じて処理が変わらない。

【００９０】

このように実装することで、受信したデータが、物理値のような連続データであるときに、欠損したデータを補完するデータの付与が可能となる。

【００９１】

また、転送されるデータが離散値の場合についても考慮が必要である。

【００９２】

図１４は、離散データに生じた欠損値の補完方法を示す図である。

【００９３】

実際に受信した離散値１００１ａ、１００１ｂ、１００１ｃ、１００１ｄにおいて、値１００１ｂと値１００１ｃの間の３番目に値の欠損が生じた場合、アプリケーションの仕様に合わせたデータ１００２（例えば、アプリケーションの動作に影響しないように前回値を複製したデータ１００２）で欠損値を補完する。

【００９４】

図１５は、離散データに生じた欠損を補完するデータ補完処理のフローチャートである。

【００９５】

図１５に示す処理において、ステップ９０１～ステップ９０３及びステップ９０５は、図１３に示す処理と同じである。

【００９６】

ステップ１１０１：データが欠損している場合、受信したＣＡＮのデータのＩＤに基づいて連続値か離散値かを判定し、離散値であればＣＡＮのデータフィールドの値を前回値を複製したデータで書き換え、ステップ９０５へ進む。

【００９７】

前述したいずれの補完方法も、アプリに影響が生じない範囲で実装されるため、必ずしも有効な補完が可能とは限らない。しかし、これにより、アプリケーション側で特別な処理を実行することなく、通常のデータとして処理が可能となる。

【００９８】

＜実施例５＞

通常、ＳＯＭＥ／ＩＰは車載ネットワーク上のサービスを検索し使用するものであり、初めに受信要求が必要であるため、一方的なデータ送信ができない。本実施例では、受信要求と関係なく、受信したＣＡＮデータから作成したＳＯＭＥ／ＩＰメッセージを第２コア３側へ送るための受信機構が必要となる。本実施例では後述する二つの方法を提案する。

【００９９】

図１６は、実施例５の第２コア３に含まれる第２ＳＯＭＥ／ＩＰ変換部１０３の周辺の構成を示す図である。

【０１００】

初めに、ＣＲＣ演算部７０１は、ＳＯＭＥ／ＩＰのメッセージとＣＲＣデータを共有メモリ９から読み出す。取り出したデータに基づいて、前述の実施例に記載したＨｅａｄｅｒ　ＣＲＣとＰａｙｌｏａｄ　ＣＲＣを用いて誤り検査を行う。ＣＲＣによってデータに異常がないことを確認した後、ＳＯＭＥ／ＩＰメッセージをバッファ１２０１に格納する。

【０１０１】

次に、第２ＳＯＭＥ／ＩＰ変換部１０３は、アプリケーション１２０２から受信要求１２０３を受け取ったら、バッファ１２０１にデータが格納されているかを判定する。データが格納されていると判定されると、第２ＳＯＭＥ／ＩＰ変換部１０３がバッファ１２０

１に格納されたデータをデシリアライズして、アプリケーション１２０２へ転送する。
【０１０２】
本実施例では、共有メモリ９から読み出したＳＯＭＥ／ＩＰメッセージを、直ぐにアプリケーション１２０１へ転送できないため、一時的にバッファ１２０１に格納するＳＯＭＥ／ＩＰメッセージ受信機構を設けた。
【０１０３】
バッファ１２０１は、データを一時的に保存する領域であり、前後の処理部の処理速度の差を吸収する。バッファ１２０１にはサイズが定義される。サイズを定義するために、バッファ１２０１が取り扱うデータ長を把握する必要がある。静的に定義する場合は、バッファのサイズは一つのＳＯＭＥ／ＩＰメッセージのサイズの最大値以上にするとよい。本実施例では、ＳＯＭＥ／ＩＰメッセージのサイズの最大値はＥｔｈｅｒｎｅｔで転送可能なデータ量の最大値から、Ｅｔｈｅｒｎｅｔ、ＩＰ、ＴＣＰのヘッダーに必要なデータ量を減じた、１４６０ｂｙｔｅを最大値とするとよい。
【０１０４】
本実施例でバッファ１２０１に格納されるＳＯＭＥ／ＩＰメッセージは可変長データであるため、動的にバッファ用のメモリを確保してもよい。可変長データを格納するメモリの使用量を節約できるというメリットがある。
【０１０５】
バッファ１２０１用の記憶領域の確保は、様々な方法で実装できる。
【０１０６】
また、バッファ１２０１には、キューやリングバッファなど様々な形態がある。キューは、ＦＩＦＯとも呼ばれ、先に登録されたデータが先に取り出される構造のバッファである。順序性を伴ってデータが格納されるため、後から入力されたデータが先に処理されない特徴がある。リングバッファは円環状バッファとも呼ばれ、循環構造を取っている。先頭から順番に読み出すとと、末尾の次が次回のアクセスする先頭となる構造のバッファである。リングバッファでは最も古い内容から更新される処理となる。
【０１０７】
デシリアライズとは、バイト列にシリアルに並んだデータを、アプリケーションで利用できる形式の並列データを変換する処理である。逆に、シリアライズ処理によって、アプリケーションから受け取る並列データをバイト列に並んだシリアルデータに変換できる。
【０１０８】
バイト列とは、特定の形式が与えられていないデータの並びであり、このままではアプリケーションが使用できない。デシリアライズによって、バイト列からデータ構造を作成し、アプリケーションで扱いやすいデータ形式に変換する。
【０１０９】
このようにバッファ１２０１を設けることで、アプリケーション側が任意のタイミングでデータを取得できる。
【０１１０】
図１７は、実施例５の第２コア３に含まれる第２ＳＯＭＥ／ＩＰ変換部１０３の周辺の別の構成を示す図である。図１７に示す態様では、アプリケーション１２０２内にアプリケーションバッファ１３０１が設けられる。
【０１１１】
はじめに、ＣＲＣ演算部７０１は、ＳＯＭＥ／ＩＰメッセージとＣＲＣデータを共有メモリ９から読み出す。取り出したデータに基づいて、前述の実施例に記載したＨｅａｄｅｒ　ＣＲＣとＰａｙｌｏａｄ　ＣＲＣを用いて誤り検査を行う。ＣＲＣによってデータに異常がないことを確認した後、ＳＯＭＥ／ＩＰメッセージを第２ＳＯＭＥ／ＩＰ変換部１０３へ送る。
【０１１２】
次に、第２ＳＯＭＥ／ＩＰ変換部１０３は、データをアプリケーションで使用できる形式に変換した後、アプリケーション１２０２内のアプリケーションバッファ１３０１にデ

ータを格納する。アプリケーション１２０２は所定のタイミング（例えば、所定の時間間隔）でアプリケーションバッファ１３０１にデータが格納されているかを判定する。データが格納されていると判定されると、データ受信機能によってアプリケーション１２０２を受信待機状態にして、アプリケーションバッファ１３０１からデータを受け取る。

【０１１３】

アプリケーション１２０２が管理する記憶領域であるアプリケーションバッファ１３０１を設ける構成によって、余計なバッファを設けることなく、データを転送でき、リソースを節約できる。

【０１１４】

以上に説明したように、本発明の実施例によると、ＳＯＭＥ／ＩＰメッセージを用いてコア間通信する際に算出されたＣＲＣをＳＯＭＥ／ＩＰと同時に共有メモリを介して転送するので、データ転送の信頼性が保証しつつ、大容量のデータを高速で転送できる。

【０１１５】

なお、本発明は前述した実施例に限定されるものではなく、添付した特許請求の範囲の趣旨内における様々な変形例及び同等の構成が含まれる。例えば、前述した実施例は本発明を分かりやすく説明するために詳細に説明したものであり、必ずしも説明した全ての構成を備えるものに本発明は限定されない。また、ある実施例の構成の一部を他の実施例の構成に置き換えてもよい。また、ある実施例の構成に他の実施例の構成を加えてもよい。また、各実施例の構成の一部について、他の構成の追加・削除・置換をしてもよい。

【０１１６】

また、前述した各構成、機能、処理部、処理手段等は、それらの一部又は全部を、例えば集積回路で設計する等により、ハードウェアで実現してもよく、プロセッサがそれぞれの機能を実現するプログラムを解釈し実行することにより、ソフトウェアで実現してもよい。

【０１１７】

各機能を実現するプログラム、テーブル、ファイル等の情報は、メモリ、ハードディスク、ＳＳＤ（Ｓｏｌｉｄ　Ｓｔａｔｅ　Ｄｒｉｖｅ）等の記憶装置、又は、ＩＣカード、ＳＤカード、ＤＶＤ等の記録媒体に格納することができる。

【０１１８】

また、制御線や情報線は説明上必要と考えられるものを示しており、実装上必要な全ての制御線や情報線を示しているとは限らない。実際には、ほとんど全ての構成が相互に接続されていると考えてよい。

【符号の説明】

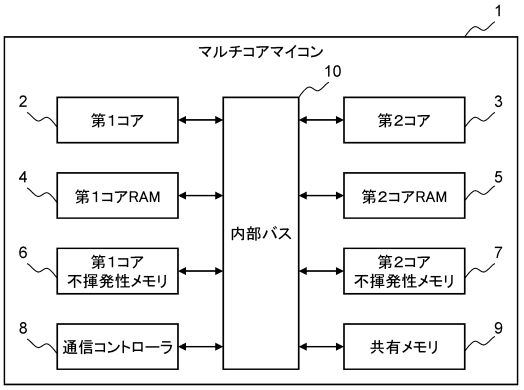
【０１１９】

１　マルチコアマイコン
２　第１コア
３　第２コア
４　第１コアＲＡＭ
５　第２コアＲＡＭ
６　第１コア不揮発性メモリ
７　第２コア不揮発性メモリ
８　通信コントローラ
９　共有メモリ
１０　内部バス
１１　ＣＡＮトランシーバ
１２　ＬＡＮ
１３　ＣＡＮコントローラ
１４　イーサネットスイッチ
１５　ＣＡＮバス
１６　他ＥＣＵ

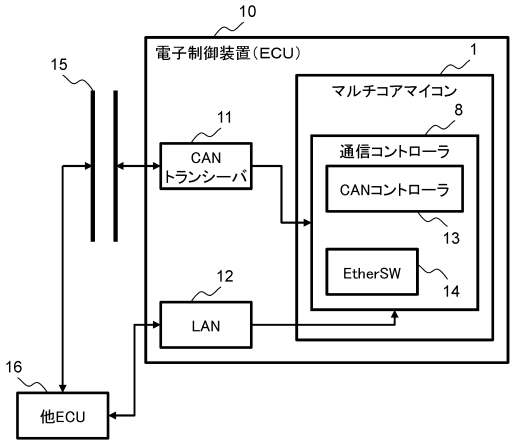
１０１　受信部
１０２　第１ＳＯＭＥ／ＩＰ変換部
１０３　第２ＳＯＭＥ／ＩＰ変換部
２０１　ＣＡＮデータフィールド
２０１ａ　ＣＡＮ実データ
２１１　ＣＡＮ　ＦＤデータフィールド
２１１ａ－ｂ　ＣＡＮ　ＦＤ実データ
３０１　ＳＯＭＥ／ＩＰヘッダー情報
３０２　ＳＯＭＥ／ＩＰ　Ｐａｙｌｏａｄ
７０１　ＣＲＣ演算部
７０２　Ｅｔｈｅｒｎｅｔ受信部
１２０１　バッファ
１２０２　アプリケーション
１２０３　受信要求
１３０１　アプリケーションバッファ

【図面】

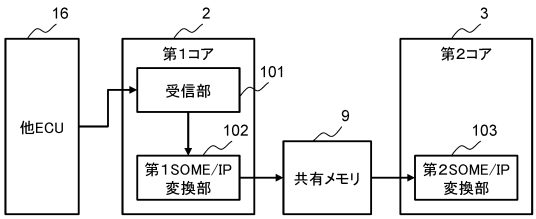
【図１】



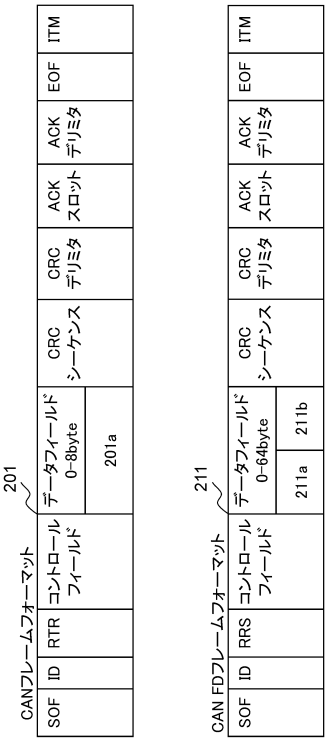
【図２】



【図３】



【図４】



【図５】

byte	1	2	3	4
0	Service ID		Method ID	
4	Length			
8	Client ID		Session ID	
12	Protocol Version	Interface Version	Message Type	Return Code
16	Payload			
...				

301
302

【図６】

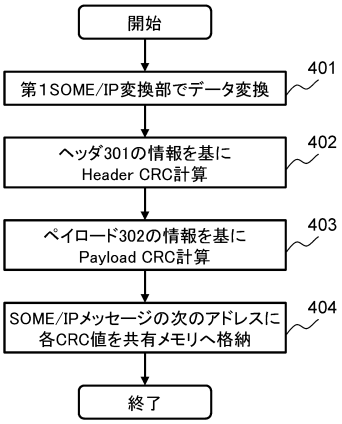
byte	1	2	3	4
0	Service ID		Method ID	
4	Length			
8	Client ID		Session ID	
12	Protocol Version	Interface Version	Message Type	Return Code
16	201a			
...	211a		211b	

301
302

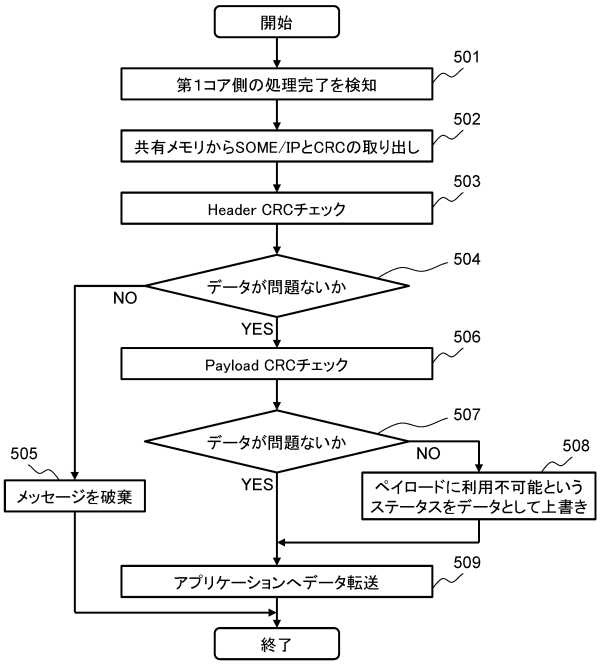
【図７】

byte	1	2	3	4
0	Header CRC			
4	Payload CRC			

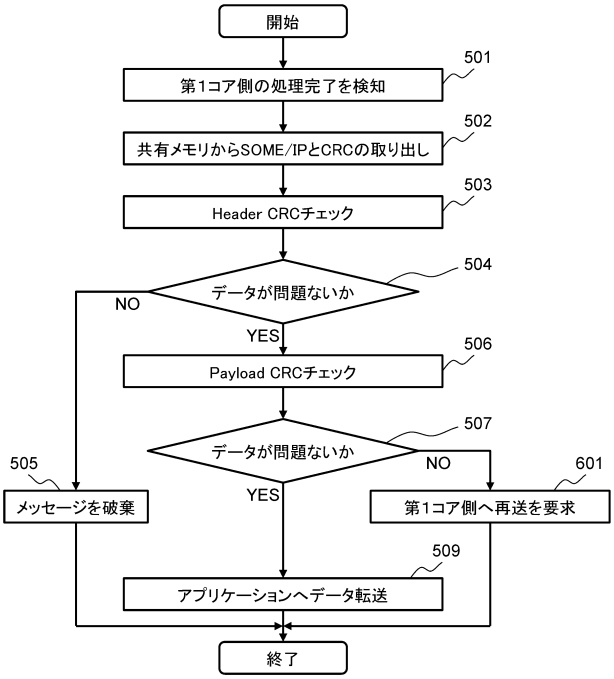
【図８】



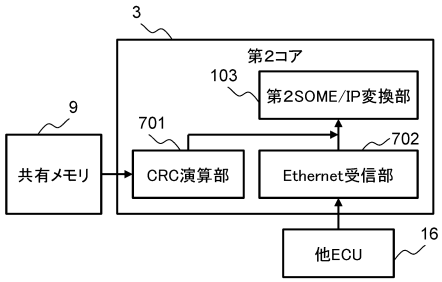
【図９】



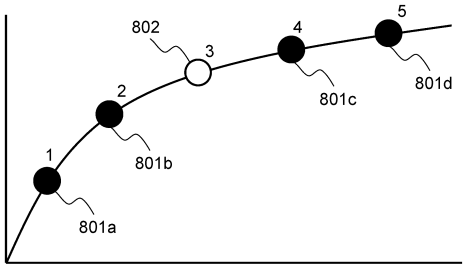
【図１０】



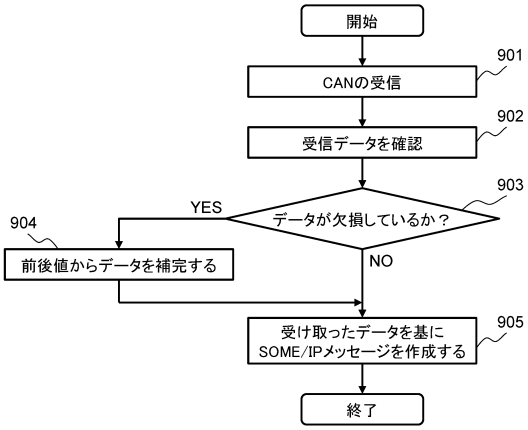
【図１１】



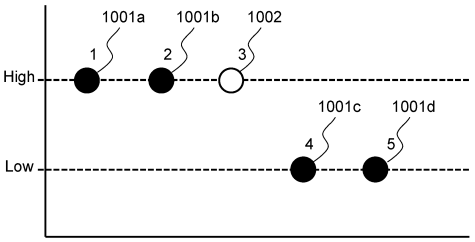
【図１２】



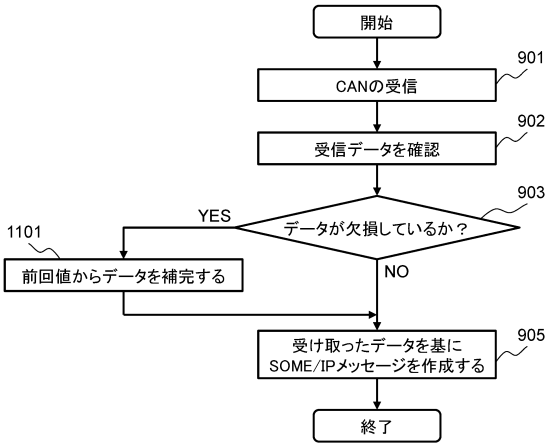
【図１３】



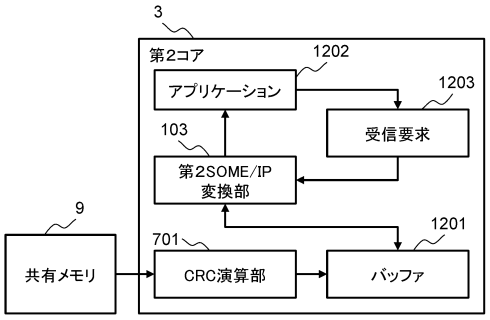
【図１４】



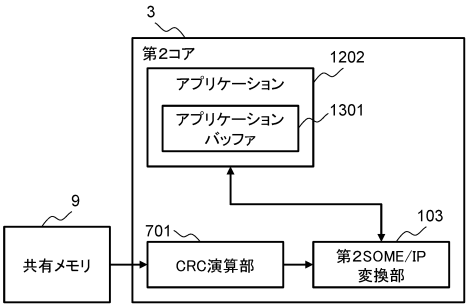
【図１５】


開始
CANの受信
901
受信データを確認
902
データが欠損しているか？
903
YES
NO
1101
前回値からデータを補完する
受け取ったデータを基に
SOME/IPメッセージを作成する
905
終了

【図１６】


3
第2コア
1202
アプリケーション
1203
受信要求
103
第2SOME/IP
変換部
9
1201
701
共有メモリ
CRC演算部
バッファ

【図１７】


3
第2コア
1202
アプリケーション
1301
アプリケーション
バッファ
9
103
701
共有メモリ
CRC演算部
第2SOME/IP
変換部