編集: Prof. Dr. Ing. Andreas Grzemba

MOST®

車載マルチメディア ネットワーク

MOST25からMOST150へ

The MOST System

This book is based on the MOST Specification Version 3.0 E2, 07/2010. Since the specification is under constant development there could have been changes from what is described in the book. Should there be any differences between the book and the specification, the specification should be followed. The latest version of the specification can be found at www.mostcooperation.com.

ISBN 978-1-62077-774-9

第2版序文

2008年3月に発行された本書の初版は、第1世代のMOST25 (MOST仕様2.4)をベースにしていました。MOST25のデータレートは25 Mbpsです。この第2版は、第2および第3世代のMOSTもカバーしています。MOSTはMedia Oriented Systems Transportの略です。

第2世代のMOST仕様Rev. 2.5 (MOST50)により、車載インフォテインメント ソリューションの帯域幅は25 Mbpsから50 Mbpsへ拡大しました。MOST50で追加された重要な仕様が『MOST Specification of Electrical Physical Layer Rev.1.1 07/2006』で、これにより車載環境での厳しい電磁両立性の要件を満たしつつ銅線のシールドなしツイストペア(UTP)ケーブルでデータ転送が可能になりました。

第3世代のMOST150では、帯域幅が150 Mbpsまで拡大しました。MOST150は従来のPOF/LEDの光物理層でこの転送レートを実現しており、OEMはこれまでのPOFとLED光源をそのまま使ってMOST25およびMOST50からスムーズに移行できます。帯域幅の拡大に加え、MOST150では広範なビデオ アプリケーションをサポートするアイソクロナス転送メカニズム、そしてIPベースのパケットデータを効率よく転送するEthernetチャンネルも導入しました。このチャンネルは、従来のIEEE 802.3に準拠したEthernetパケットをそのまま伝送できます。このように、新世代のMOSTは車載Ethernetに対応した物理層を備えています。これにより、幅広い種類のIPベース アプリケーションからMOSTを利用できます。また、従来のストリーミング接続とIP通信の帯域幅を個々の要件に合わせて調整する事もできます。これに加え、MOST150は既存の非同期チャンネルもサポートしており、MOST25アプリケーションとの下位互換性も維持しています。MOST150では、オーディオおよびビデオ信号をアドレッシング、コリジョン検出/リカバリ、ブロードキャストのオーバーヘッドなしで効率よく広帯域転送できます。このため、複数の高精細(HD)ビデオストリームとマルチチャンネル サラウンド サウンドを高いサービス品質で送信しながら、同時に大量のパケットデータを転送できます。

MOSTネットワークは、以下のようなデジタル オーディオ、ビデオ、データサービスを等しくサポートする必要があります。

- QoSの高い同期転送(PCMオーディオ…)
- QoSの高いアイソクロナス転送(MPEG Transport Streams/audio/video over IP)
- IPをサポートした高性能パケット転送
- リアルタイム制御

インフォテインメントのバックボーンにMOSTを採用した自動車は現在、100車種を数えています。短期間でMOST搭載車が100車種まで拡大したのは、MOSTの実用性と信頼性の高さを物語っています。1998年にMOST Cooperationが設立されてからわずか3年後の2001年に、最初のMOST搭載車が発表されました。翌年には、さらに13車種がインフォテインメント バックボーンにMOSTを採用しました。ネットワーク技術のリーダーとして、MOST搭載車は2007年10月には50車種、2009年3月には68車種に達しました。それからわずか1年でMOST搭載車の数は50%増加し、MOSTテクノロジは大衆市場への普及を果たしています。

アジアのOEMが初めてMOST搭載車を発表したのは2007年でした。現在、アジアのOEMの22車種にMOSTが採用されています。また、現在米国のOEMがMOSTテクノロジを採用した新車の開発を進めており、アメリカ大陸にもMOSTテクノロジが広がろうとしています。

ヨーロッパおよび韓国市場ではMOST25テクノロジが広く普及しています。日本および米国市場ではMOST50に人気があります。

現在、最新のMOST仕様Rev. 3.0 (MOST150)に基づいた自動車の量産が間近に迫っています。Audi社とDaimler社がMOST150を量産車に採用する最初のOEMとなる見込みです。

この第2版では、本書の構成を一新しました。初版ではMOST以外の車載通信システムも取り上げていましたが、第2版はMOSTの説明のみに絞っています。

謝辞

本書は多くの共著者のご協力によって実現しました。貴重なお時間を割いて頂きました著者の皆様に感謝いたします。またこの第2版の制作を開始し、積極的にサポートしてくださったMOST Cooperation運営委員会の全ての会員の皆様にもお礼申し上げます。

本書の制作に全面的にご協力を頂いたMOST Cooperationのコーディネータ Wolfgang Bott氏、および原稿の校正に当たって頂いたRenato Machelett氏にも感謝します。

最後に、本書の完成まで様々な面からご支援を頂きましたFranzis出版社に感謝の意を捧げます。

2011年1月

カールスルーエ　　デッゲンドルフ

MOST Cooperation　　Andreas Grzemba(編集者)

初版序文

自動車業界が車載通信システムを初めて導入してから約20年が経過しました。最初に開発されたのはCAN-Busで、1990年代から量産車で使われています。その後、車載インフォテインメント分野はここ数年で急速な発展を遂げました。AM/FMチューナさえあれば十分だった頃とは違い、現在では多くの自動車にハイグレードのサウンドおよびナビゲーション システムが搭載されています。これに伴い接続される機器の数も増え、無線、携帯電話、CDチェンジャ、ナビゲーション システム、音声操作、マルチチャンネル アンプ等が初期のシステムに追加されるようになりました。これらの機器は相互に連携した動作が求められる上、運転者の注意を必要以上にそらさないように中央の操作パネルだけで操作できる事が必要です。CAN-Busの帯域幅で転送できるのは制御信号だけで、オーディオおよびビデオ信号は転送できないため、これらの要求を満たす事はできません。

新しい柔軟なアーキテクチャを備えた通信システムであるMOST (Media Oriented Systems Transport)はこうした要求を満たすために開発され、多くのOEMに採用されています。MOSTはオーディオ信号を電話のように同期転送でき、現在では最も広く使われている車載マルチメディア システムです。

MOSTテクノロジは、MOST Cooperationに参加するOEMとサプライヤ各社が共通の標準を共同で確立、改良する取り組みから生まれました。MOST Cooperationの設立企業は、MOST Cooperationの最高意思決定機関である運営委員会を構成します。MOSTシステムに関心のあるその他全ての企業は、ワーキング グループでの協業に参加できます。MOST Cooperationについては、第1章で紹介します。

MOSTシステムは、通信およびコンピュータ ネットワーク設計に関するOSI (Open System Interconnect)参照モデルの全ての階層を含むだけでなく、アプリケーション(例: AM/FMチューナ)とのインターフェイスもファンクション ブロックとして定義し、標準化しています。これらの実装は、サンプルの相互動作モデルとして記述されています。また、オーディオおよびビデオ信号の符号化方法についても仕様で定義されています。このため、比較的高い抽象度でマルチメディア システムを設計できます。

本章は、MOSTテクノロジの入門書です。MOST仕様の内容を置き換えるものではありません。本書はMOST仕様2.4をベースに、今後の開発動向を加味して作成しています。最新の仕様は、MOST Cooperationのウェブサイト(http://www.mostcooperation.com)で入手できます。

本書は2部構成です。

第1部では、MOST仕様の内容を取り上げます。まずMOST Cooperationの組織構成を紹介した後、自動車内の通信アーキテクチャについて説明し、MOSTテクノロジを分類します。第3章「システム アーキテクチャ」でMOSTテクノロジの概要を説明します。特に重要な事項については、その後の章で詳しく説明します。

第2部では、MOSTテクノロジのアプリケーションを取り上げます。まず、開発ツールを紹介します。その次の章で、これらのツールを使ったシンプルなMOSTシステムの構造と実装について説明します。次に、MOSTゲートウェイの基本構造、MOST部品のプロセス、現在の量産車におけるMOSTの利用について説明します。最後の章は、自動車で使われているサウンドシステムについて説明します。

補遺に略語一覧、MOST用語集、そしてMOST、Ethernet、IEEE1394の比較を掲載しています。現在自動車で使われている他の通信システムを概観する事で、MOSTテクノロジへの理解が深まるものと考えます。ゲートウェイ関連で特に重要なのが、CAN-BusとBluetoothです。

本書がMOSTテクノロジ普及の一助となり、設計者とユーザにとって便利な参考書となれば幸いです。MOSTシステムはこれからも継続的な改良と新しい開発が進められます。本書に対するご意見、ご感想をお待ちしております。以下のメールアドレスまでお寄せください。andreas.grzemba@fh-deggendorf.de

謝辞

本書は多くの共著者のご協力によって実現しました。貴重なお時間を割いて頂きました著者の皆様に感謝いたします。

このプロジェクトを開始し、積極的にサポートしてくださったMOST Cooperation運営委員会の全ての会員の皆様にもお礼申し上げます。

FlexRayの章を確認して頂きましたBMW社のJosef Berwanger氏、第9章を校正して頂きましたSMSC EuropeのRoland Trißl氏、第14章と第15章の執筆にご協力頂きましたSMSC EuropeのMatthias Karcher氏とVector Informatik社のCarsten Kellner博士、そして第16章を校正して頂きましたHarman社Automotive DivisionのJochen Klaus-Wagenbrenner氏、BMW社のMartin Arend氏にも感謝いたします。

特に、多くのご意見とご助言を頂きましたBMW社のRobert Reiter氏、そして原稿全体の校正をご担当頂きましたMicrochip Technology, Inc.社のChristian Thiel博士、Armin Schmidt氏、Henry Muyshondt氏とその同僚の皆様、Daimler社のAlexander Leonhardi博士には深く感謝いたします。

また、本書を英訳して頂きました翻訳者の皆様にも感謝いたします。

最後に、原稿全体を最初にチェックしてくれた妻のJana、そして本書の完成まで様々な面からご支援を頂きましたFranzis出版社のAstrid Lehmann氏とその同僚の皆様に心から感謝します。

デッゲンドルフ、2008年1月

Andreas Grzemba

(編集者)

本書の執筆者一覧(アルファベット順)

Dr.-Ing.Steffen Abbenseth
VOLKSWAGEN AG
第14章

Dipl.-Ing.(FH) Frank Bähren
Harman Automotive Division
第9章

Dipl.-Ing.(FH) Patrick Blum
Harman Automotive Division
第13章

Dr.-Ing.Wolfgang Bott
Ingenieurbüro BOTTCO
第3、12章

Dr. Simon Burton, BEng. (Hons), Phd.
Vector Informatik GmbH
第10章

Dipl.-Ing.(FH) Markus Dittmann
Harman Automotive Division
第16章

Dr.-Ing.Volker Feil
Daimler AG
第8章

Prof. Dr.-Ing. Andreas Grzemba
FH Deggendorf
第2、5、9、11、15、17章
補遺B、C、D

Dr.-Ing.Dieter Jurzitza
Herrmann Ultraschalltechnik GmbH
第16章

Dipl.-Ing.(FH) Stefan Kerber
SMSC Europe GmbH
第9章

Dr.-Ing.Thomas Kibler
Daimler AG
第6章

Dipl.-Physiker Jochen Kirschbaum BMW AG	第7章
Dr. Rainer König Daimler AG	第1章
Dipl.-Ing.Joachim Leonhard K2L GmbH	第17章
Dr. Alexander Leonhardi Daimler AG	第4章
M.Sc.CS Markus Nordgren SMSC Sweden	セクション5.8.2～5.8.4
Dipl.-Ing.Tilo Nothacker Daimler AG	第6章
Dipl.-Ing.Stefan Poferl Daimler AG	第6章
Dipl.-Ing.(FH) Armin Schmidt SMSC Europe GmbH	第10章
Dipl.-Ing.(FH) Alexander Schneidenbach AUDI AG	第14章
Dr. M. Eng. Andreas Schramm BMW AG	第15章 補遺A
Dipl.-Ing.Harald Schöpp SMSC Europe GmbH	第18章
Dr.-Ing.Christian Thiel SMSC Europe GmbH	第1章

目次

1 MOST Cooperation

1.1 設立の動機と沿革

複数の機器を制御し、多くの信号を1本のケーブルで伝送する車載ネットワークが初めて実装されたのは1980年代終わり頃でした。当時は、どのOEMもこのような技術は自社製品の差別化に大きく寄与する可能性があると考えていました。Mercedes社はCANシステムを開発し、PSA (Peugeot Citroën)社とRenault社は車両インフラストラクチャをVANで制御していました。一方、米国でも独自の市場ニーズに応える形でJ1850が開発されました。BMW社も独自ネットワークとしてI-Busを開発していました。こうした状況がしばらく続いた後、各社はこれらの技術が同じ目的をもったものであると認識するに至りました。すなわち、デバイス間でデータを転送する事です。しかしOEMごとに別々の技術を開発するには多大な時間とコストがかかります。結局、これらのOEMは形は違えど似たような機能を持つネットワーク技術を開発していたのです。その後約20年を経て、CANが広く採用されるようになりました。現在では、世界中のほぼ全ての自動車が車内機能とパワートレインの制御に各種CANネットワークを使っています。CAN以外のネットワーク技術の開発は無駄だったとも言えますが、1980年代にその結論に達するのはまだ無理がありました。

1996年、BMW社はBecker社(現在のHarman社Automotive Division)およびOASIS Silicon Systems社(現在のMicrochip社)と共同で、当時Mercedes社が開発、生産していたD2Bシステムをベースにした MOSTの原型について議論を開始し、この開発を他のOEMと共同で継続していく事が決定されました。各社が独自にソリューションを開発すると、再び開発の負担がのしかかる事を懸念しての決定でした。この提案はまずDaimler Benz社に持ちかけられましたが、最初から密接な協業が行われた訳ではなく、両社の接近はどちらかと言えば用心深いものでした。

その後、慎重に作業を進めて成功を重ねるうちに両社の関係は次第に緊密になり、MOSTの開発に共同で取り組む事が決定されました。この協業の第1段階は、物理層やネットワーク管理等のネットワーク機能に関する仕様の策定の協業のみを想定していました。協業が進んで信頼関係が深まるにつれ、バスシステムの上位のアプリケーション要件に関する議論をしやすい環境が整いました。BMW社とDaimler Benz社によるこの共同作業を基盤として、物理層の定義と機能仕様の開発が行われました。その後、これらの仕様を拡張するものとして、物理層のコンプライアンス、ネットワーク管理のコンプライアンス (コア コンプライアンス)、アプリケーション レベルのコンプライアンスをテストするための仕様が作成されました(第12章参照)。

こうした初期の協業の結果、以下に示す基本的な合意が形成されました。これは現在もMOST Cooperationの活動を特徴づけています。

MOSTは、量産開発を実現する目的で設立されました。このため、机上の標準化ではなく量産に必要な事項の定義と実装を目指し、現実に即した協業を行います。MOST Cooperationに参加する他の当事者の妨げとなる行為は禁止されています。個々の機能コンポーネントのうち、最も時間のかかる開発を手がけている当事者が開発ペースを決定します。このタイムフレームで実現できない当事者の開発要求は、次のリリースで取り扱われます。

初期の仕様が作成され、共同開発の最初の段階が順調に滑り出した時点で、この活動をさらに拡大しようという声が持ち上がりました。そこで、MOSTの開発と標準化のためのオープンな協同組合を設立する事が決定されました。他のOEM、機器メーカー、部品メーカーが規制の障壁なしに自社の利益のためにこの協同組合に参入できるようにしました。

1998年、MOST CooperationはドイツJ民法上の組合(GbR)としてBMW、Daimler Benz、Becker、OASIS Silicon Systemsの4社によって設立されました。その後間もなくAudi社が設立グループに加わりました。

MOST Cooperationの認知が広まるにつれ、会員数は短期間で約80にまで増加しました。世界中のほぼ全てのOEMとインフォテインメント機器メーカー、そして多くの部品メーカーも加入しました。

MOST CooperationはMOSTの技術面の開発とそれに対応する仕様だけでなく、MOSTを車載マルチメディア ネットワークの世界標準とするための技術のプロモーション活動にも力を入れています。その一環として、MOST Cooperationは米国、ヨーロッパ、アジア各地で開催される展示会または会議のトレードショー イベントに積極的に参加してきました。これまで、かなり多くの展示会でMOST Cooperationの会員企業が出展を行ってきました。例えばトリノで開催されたITS World Congress 2000では、OEMが共同開発したMOSTテクノロジを世界初公開しました。当時開発に携わっていたOEM (Audi、BMW、Mercedes、Volkswagen)が各社の新車を展示し、設計担当者が出席しました。

現在は、OEM間の協業が非常に活発です。ユーザの目からは見えないインフラストラクチャの技術については、共同で開発する事が当然となっています。

図1.1:ITS World Congress 2000のMOST Cooperation展示ブースにAudi、BMW、Mercedes、Volkswagen車を展示

1.2 迅速な標準化

MOST Cooperationが設立された当初は、ネットワーク インターフェイス コントローラOS8104とネットワーク管理およびトランスポート層に関するいくつかの仕様しかありませんでした。MOSTの準備は量産の期限までに整える必要がありました。短期間での標準化が必要でしたが、通常これは両立しません。標準化は通常、何年もかかる作業と果てしない議論を伴います。では、迅速な標準化を実現するにはどうすれば良いでしょうか。

広範な標準化を実現するには、多くの企業の参加が必要ですが、議論を効率的に行う必要があります。意思決定は迅速に行う必要があるため、小規模なグループ単位で行います。議論には、その分野に長けている企業のみが参加するようにします。また、MOST Cooperationの運営方法は複数の要件に柔軟に適応できる必要があります。この他、関連のないトピックは議題に含めないように注意します。MOST Cooperationは、量産車の標準となる事を目指したプロジェクトのみを扱います。全ての参加企業は、MOST Cooperationの理念(セクション1.1参照)を受け入れ、協力する必要があります。

MOST Cooperationの組織体制は、「迅速」と「標準化」という相反するものを両立できるように定義されています。この体制は信頼性が実証されたため、他の多くの組織の模範となりました。

1.3 組織体制と仕様策定の流れ

ここからは、図1.2に示した組織体制について詳しく説明します。

1.3.1 運営委員会

MOST Cooperationは、設立企業が運営委員会を組織し、それ以外の企業は準会員として参加しています。運営委員会の会員のみがMOST Cooperationのパートナです。運営委員会の会員はMOST Cooperationの活動への協力義務がありますが、準会員は任意です。しかし、MOST Cooperationの成果物は全ての会員が無制限かつ完全に利用できます。各会員は、自社が標準化に貢献した内容を他の会員が無償で利用できるように相互ライセンスを供与します。

MOST Cooperationの協力の枠組みは協力協定で大まかに定義されているだけで、これを具現化するのは運営委員会の役割です。運営委員会は、この枠組みの中で標準化作業の進め方を定義します。すなわち、運営委員会はMOST Cooperationの立

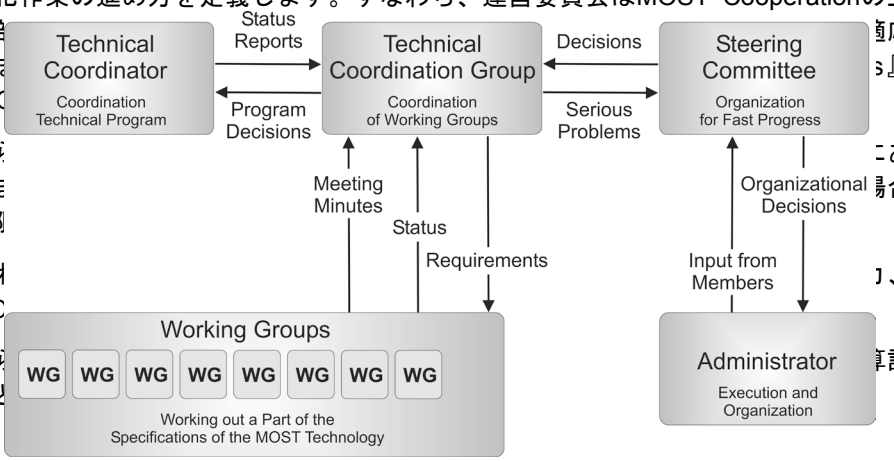


図1.2:MOST Cooperationの組織体制と仕様策定の流れ

1.3.2 アドミニストレータ

運営委員会が1名のアドミニストレータを任命します。アドミニストレータは、運営委員会での決定事項をMOST Cooperationの日常業務に反映させます。また、会員企業および加入申請企業との連絡窓口もアドミニストレータが務めます。さらに、MOST Cooperation主催イベントの企画と実行、ウェブサイトの管理もアドミニストレータが行います。

1.3.3 技術調整グループ

MOST Cooperationの技術作業は、運営委員会の会員とその他全てのOEMによって構成される技術調整グループ(TCG)が遂行します。これにより、MOST Cooperationのプロジェクトが一貫性のない単なる標準化にならず、OEMの要件に沿って実践されます。

TCGはプロジェクトを定義し、ワーキング グループ(WG)を設置し、WGのコーディネータを決定します。通常、コーディネータはTCGの中でそのWGのプロジェクトの成果に対して最も大きな利害関係を持つ会員企業の従業員が選ばれます。これにより、そのプロジェクトに最も密接に関係する会員企業がWGを運営および推進するようにしています。TCGはまた、新しいWGで協業する他の会員企業を招待します。こうする事で、WGの参加企業が有益な貢献を行い、WGが目的指向型のアプローチで作業を進められるようにしています。このため、WGへの参加会員の数には妥当な上限も設定されています。

TCGはWGの目的を定義し、WGの活動成果に関して定期的に報告を受けます。必要であれば、WGの目標を新しい要件に合わせて調整する事もあります。WGの目標に競合が発生し、これを解決できない場合、WGがTCGに報告し、TCGが新たな決定を下します。

1.3.4 ワーキング グループ

MOST Cooperationの各技術プロジェクトには、ワーキング グループ(WG)を設置し、各WGに1名のコーディネータを選任します。WGは、TCGが設定した目標に従ってプロジェクトを遂行します。実際の作業の段取りは、コーディネータとWGに一任されます。WGには、『MOST Cooperation Organizational Procedures』[MOST Org]で定義された枠組みの条件に従う事のみが要求されます。これにより、個々のWGが独自の理念に基づき、独自の方法で作業を進める事が可能です。このようなアプローチが必要なのは、例えば物理層のバスに関する作業とソフトウェア レベルの作業では適用される要件が異なるためです。

通常、1つのWGでは1つまたは複数の仕様が作成されます。WGが仕様を完成させるとただちに、MOST Cooperationの中心的なWGであるWG Device Architectureが仕様規則への適合性をチェックします。修正の必要がなければその仕様はプレリリース段階へ進み、全ての会員による査読と意見提出が行われます。ここでも修正の必要がなければ8週間後に仕様がリリースされ、正式なMOST仕様の一部となります。

1.3.5 技術コーディネータ

MOST Cooperationには1名の技術コーディネータが存在します。技術コーディネータはプロジェクト マネージャの中から選ばれ、技術面に関するMOST Cooperationの全体的な活動を統括します。技術コーディネータはWGとの連絡窓口であり、

WGの全体的な進捗状況を監視し、TCGに報告します。異なるWGが作成した仕様のリリース等のプロセスを調整します。技術コーディネータの最大の役割は、技術作業の進行を迅速化する事にあります。

MOST仕様の構成については、第3章で説明します。

1.4 コンプライアンス検証プログラム

MOST Cooperationは、標準の確立によって、全ての会員に主要部品の量産効果等の相乗的利益がもたらされる事を目指しています。このため、各会員はMOST仕様の実装に必要となる各社保有の知的財産権(例: 著作権、商標権、特許権)を無償で使える権利を相互に許諾しています。MOSTの全ての「オーナー」は、MOSTの仕様がMOST本来の目的のみに使われる事、他のプロジェクトの一部に使われない事を望んでいます。

この目的を保証するため、コンプライアンス検証のプロセスを定義しています。あるデバイスに対してメーカーが会員の保護された権利の下でライセンスを取得できるのは、MOST仕様への適合が証明されているデバイスに限られます。このためにMOST CooperationはMOSTコンプライアンス テストを定義しており、全てのデバイスがこのテストへの合格を義務付けられています。このテストに合格したデバイスのみにライセンスが許諾され、MOSTの商標を使う事が許可されます。

コンプライアンス テストには、ライセンス供与という法的側面以外にもMOSTデバイス間の相互接続性を確保し、デバイスの品質を確認する役割があり、そのために常に拡張が行われています。MOSTロゴの使用は強制ではありません。しかし、ベンダが広告にMOSTロゴを掲載した場合、それは高い品質と相互接続性の証となります。

コンプライアンス検証の手順については、第12章で説明します。この手順は、主にMOST Cooperationの仕様策定グループから独立した複数の委員会によって定義されています。ただし、MOST Cooperationで仕様策定グループとコンプライアンスグループを同じ委員が兼務する事もあります。

2 システム アーキテクチャの概要

この章では、MOSTのシステム アーキテクチャ概要および基本的な特徴について説明します。詳細は、この後の章で説明します。

2.1 MOSTの階層モデル

MOSTシステムを開発するにあたり、OEMにはインフォテインメント バスシステムの機能について以下の2つの基本的な要求がありました。

1. 機能指向の視点に基づいたシンプルなシステム設計とする事

2. 制御情報に加え、ストリーミング データとパケットデータを転送できる事

1の要求を満たすため、ファンクション ブロックを基本的要素としたMOSTフレームワークが開発されました。ファンクション ブロックには、MOSTデバイス(例: CDチェンジャ)の制御に必要なプロパティとメソッドが全て含まれます。ファンクション ブロックとの通信は、アプリケーション プロトコルを使って行います。このプロトコルは人間にも理解しやすいニーモニックで構成され、MOSTデバイスのアドレスは必要ありません。このため、インフォテインメント システムを高い抽象度で簡単かつ迅速に設計できます。

ファンクション ブロックはアプリケーションとのインターフェイスを構成するもので、ISO/OSI参照モデルでは第7層に属します(図2.1参照)。ファンクション ブロックとアプリケーション プロトコルについては、セクション2.2で概要、第4章で詳細を説明します。

上記2の課題は、フレーム構造で解決しています。このフレーム構造は、マルチメディア データを同期転送できる他、同期転送に影響を与えずに大量のデータを非同期転送でき、制御コマンドとステータス メッセージを転送するための制御チャンネルを備えています。制御チャンネルはアプリケーション プロトコルに属します。

上記2の要求により、データリンク層はデータフレームの同期転送に基づいています。データリンク層はMOSTネットワーク インターフェイス コントローラに実装されます。一般には光物理層が多く使われますが、電気物理層も利用できます。

データリンク層の概要はセクション2.5で説明します。詳細は第5章で説明します。物理層の概要はセクション2.6で説明します。詳細は第6章で説明します。

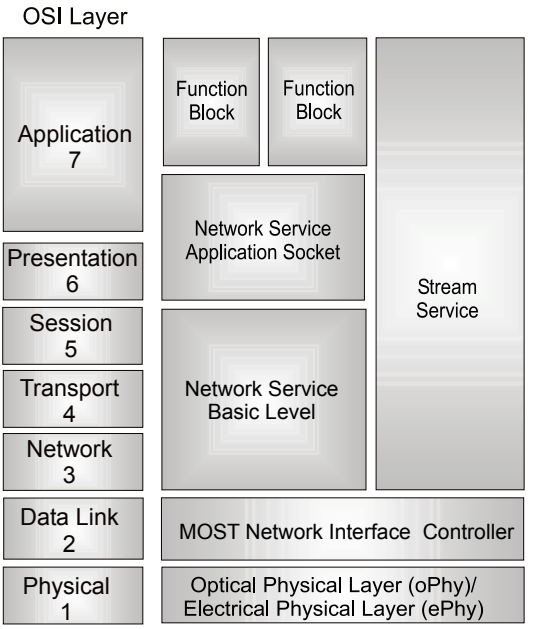


図2.1:MOSTとISO/OSI参照モデルの対応

MOSTシステムのドライバ ソフトウェア、すなわちネットワーク サービスは、ファンクション ブロックとデータリンク層の中間に位置するミドルウェアです。これは、ISO/OSI参照モデルの第3層から第7層の一部までに対応します。NetServicesは、Microchip社が実装したネットワーク サービスの製品名です。

MOSTシステムはオーディオおよびビデオデータの同期転送を主な目的として開発されているため、これらの転送に使うストリーム サービスもあります。ストリーム サービスは、図2.1に記載していますが、実際のOSI参照モデルには属しません。

2.2 アプリケーション フレームワーク

アプリケーション フレームワークの中心的な要素は、前述のファンクション ブロックとその動的挙動です。

ファンクション ブロックは、MOSTネットワークで特定目的の機能を制御するオブジェクトです。ファンクション ブロックには、CDチェンジャやオーディオアンプ等のアプリケーションを制御するものと、ネットワークを管理するものがあります。複雑なオーディオ機能も、ファンクション ブロックの組み合わせとして実装できます。ファンクション ブロックを使うと、アプリケーション設計エンジニアは分散型オーディオ アプリケーションを高い抽象度で比較的簡単に開発できます。

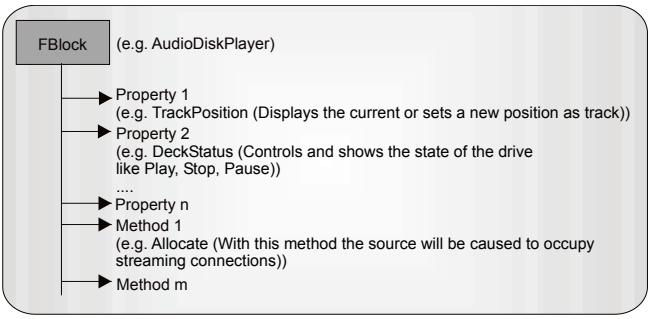


図2.2:CDプレーヤのファンクション ブロック (AudioDiskPlayer)

ファンクション ブロックは、制御対象のアプリケーションのインターフェイスを定義します。ファンクション ブロックはプロパティとメソッドで構成されます。プロパティは制御対象のファンクションの状態を記述または変更します。メソッドは何らかのアクションを実行し、一定時間の後、何らかの結果を生じます。MOST仕様では、プロパティとメソッドを総称して「ファンクション」という用語を使います。

実装の一貫性を確保するため、全てのファンクション ブロックは交換可能なフォーマットのXMLとして記述されています。このXML記述は、MOST Editor(セクション11.2.2参照)で編集できます。

図2.2に、CDプレーヤのファンクション ブロック(AudioDiskPlayer)のプロパティとメソッドの例(一部抜粋)を示します。

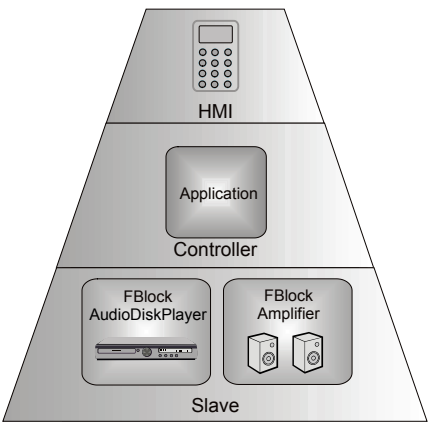


図2.3:MOSTシステムの階層

MOST仕様では、ファンクション ブロックとの相互動作を以下の3つの視点で区別しています(図2.3参照)。

- HMI
- コントローラ
- スレーブ

スレーブとは、コントローラで制御されるMOSTデバイスです。スレーブの機能はファンクション ブロックのプロパティとメソッドで提供します。スレーブはシステムの情報を全く持ちません。つまり、スレーブには他のデバイスに関する情報が静的に格納されません。従って、スレーブが他のスレーブを制御する事もありません(図2.4参照)。スレーブはMOSTシステムに簡単に追加または削除できます。ソフトウェアへの変更は不要で、他のスレーブに影響を与えません。つまりCDチェンジャまたはアンプをスレーブとして実装しておけば、異なる車両プラットフォームで容易に使えます。

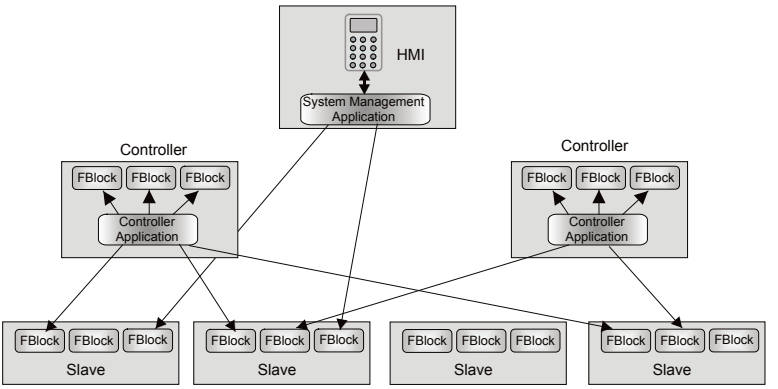


図2.4:MOSTシステムの階層における相互作用

コントローラはMOSTシステムの機能部分を管理するためのアプリケーションで、スレーブのファンクション ブロックを制御します(図2.4参照)。例えば、チューナは対応するアンプを制御できます。そのために、コントローラはシステムに関する部分的な情報、つまり制御対象のファンクション ブロックに関する情報を必要とします。コントローラがアクセスするファンクション ブロックのプロキシ(またはスタブ)を、このファンクション ブロックのシャドーとも呼びます。これ以外に、コントローラは自分自身のファンクション ブロックを持つ事もできます(セクション4.1参照)。

ヒューマンマシン インターフェイス(HMI)は、MOSTシステムのユーザ インターフェイスでシステム機能を高い抽象度で提示します。HMIは各種コントローラを調整します。

コントローラは、アプリケーション プロトコル(図2.5およびセクション4.2参照)を使ってファンクション ブロックを制御します。アプリケーション メッセージは、制御チャンネルで転送するか、パケットデータ チャンネルでMOST High Protocolを使って転送します。アドレス指定したファンクション ブロックのデバイスアドレスはネットワーク サービスによって確認されるため、コントローラは知る必要がありません(セクション9.3.3参照)。

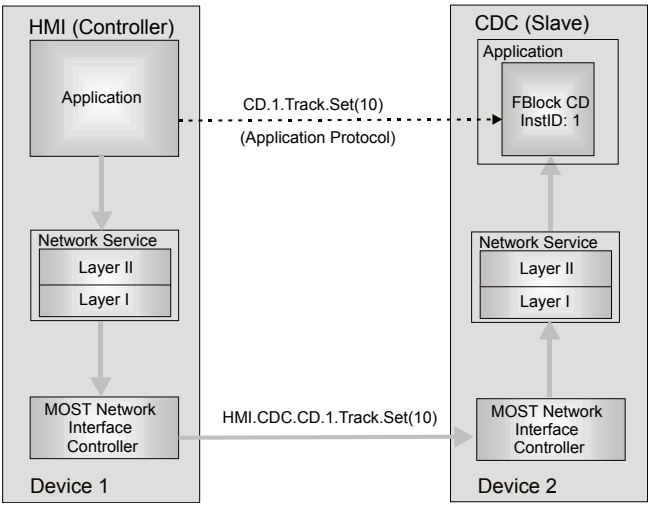


図2.5:アプリケーション プロトコル(出典: [MOST 3.0])

ファンクション ブロックを使った時の動的挙動は、記述言語のMessage Sequence Chart (MSC)を使って記述します。MSCは、システム コンポーネント同士、およびシステム コンポーネントと周囲環境との通信挙動を定義および記述する目的で国際電気通信連合(ITU)によって定義された技術言語です。この言語は、使う通信プロトコルに依存しません。MSCについては、補遺Aで説明します。

2.3 MOST150で導入されたアイソクロナス転送メカニズム

MOST150では、アプリケーションに応じて3種類のアイソクロナス メカニズムが扱えます。

A/Vパケット化アイソクロナス ストリーミング: MOSTタイムベースへの参照情報なしで到着したビデオ データストリームを転送できます。このデータは既に小さなデータパケットに集約されており、任意でタイムスタンプを含める事ができます。この代表例として、VBR(可変ビットレート)またはCBR(固定ビットレート)で符号化されたMPEGデータストリームがあります。

DiscreteFrameアイソクロナス ストリーミング: データとタイムベース情報を並列に転送します。データはフレームに配置されます。これらのデータはMOSTのタイムベースには同期しません。PCMストリームは全てこのカテゴリに該当します。つまり、PCMデータとフレーム同期信号を並列に転送します。例えば、48 kHzのMOSTネットワークで44.1 kHzの信号を転送する場合がこれに該当します。

QoS IP(ストリーミング): パケット送信でこのモードを使う事により、サービス品質要件を完全に満たす事ができます。通常、パケットデータ チャンネルのリソースはネットワーク上の全てのアプリケーションで共有します。しかし場合によっては帯域幅の保証が必要になる事があります。例えばMOST Ethernetチャンネルでビ

デオデータをIPストリームとして転送する場合、大容量のデータ転送が突然発生するとビデオ信号に利用できる帯域幅が低下し、ビデオ転送に影響が出る事があります。このような場合、QoS IP(ストリーミング) メカニズムを使えば突然データ転送が発生してもビデオ転送に干渉せず、画質の乱れを防ぐ事ができます。

2.4 ネットワーク サービス

ネットワーク サービスは、MOSTの標準化されたプロトコル スタックで、OSI参照モデルの第3～第7層に対応します。ネットワーク サービスは通常、2つのレイヤに分かれます。ネットワーク サービスには、トランスポート プロトコルのMOST High Protocol、TCP/IPスタックをMOSTにマッピングするアダプテーション層のMAMAC (MOST25とMOST50のみ)、MOST150の純粋なTCP/IPスタック(いわゆるEthernet over MOST) (セクション5.8.2参照)も含まれます。

ネットワーク サービスはMOSTネットワーク インターフェイス コントローラに接続し、アプリケーション(基本的にファンクション ブロックで構成)に対するプログラミング インターフェイスを提供します(図2.10参照)。ネットワーク サービスには、パケットデータ チャンネルでパケットデータを転送するためのモジュールと制御チャンネルでネットワークを制御するためのモジュールが含まれます。ネットワーク サービスは外部ホスト コントローラ(EHC)に実装されます。ネットワーク サービスはストリーミング接続も制御します。しかし、同期/アイソクロナスデータの転送は制御しません。

2.5 MOSTのデータリンク層

データリンク層は、MOSTバスの基本的な転送メカニズムを定義します。データリンク層では、データフレームの構造と、TimingMasterとTimingSlaveのファンクションを定義しています。

2.5.1 MOSTフレーム

フレーム構造には、MOSTシステムに対する前述の2番目の要求、すなわちマルチメディア データの同期転送に対する要求が直接反映されています。1つのフレームには、ストリーミング データの同期転送または(MOST150のみ)アイソクロナス転送用チャンネル、パケットデータの非同期転送用チャンネル、制御データ転送用チャンネルの3つがあります(図2.6参照)。ストリーミング データチャンネルでは、サンプリング レート44.1 kHz (MOST25とMOST50)または48 kHz (MOST150)でストリーミング ソースとストリーミング シンクを1対1または1対多で静的に接続できます。

接続と切断の制御、およびファンクション ブロックに対する制御メッセージの交換は、制御チャンネルで行います。制御チャンネルで転送されるコマンドのデータは、後続の複数のフレームに分けて送信されます。この制御データの送信用として、MOST25では1フレーム当たり2バイト、MOST50とMOST150では1フレーム当たり4バイトが確保されています。

パケットデータは、同期データの転送に全く影響を与えずにパケットデータ チャンネルで転送されます。パケットデータ チャンネルで転送されるデータとしては、ナビゲーション システムの設定データ等があります。パケットデータ チャンネルでの転送には、データリンク層プロトコルを使います。MOST150では、パケットデータ チャンネルで純粋なEthernetデータを転送できます。

ストリーミング チャンネルとパケットデータ チャンネルの境界は、バウンダリディスクリプタを使って設定します。

図2.6に示すように、MOST25は1フレームが合計64バイトで構成されます。MOST50では帯域幅が2倍に拡大されているため、一度に128バイトを転送できます。MOST150ではさらにその3倍の384バイトを転送できます。

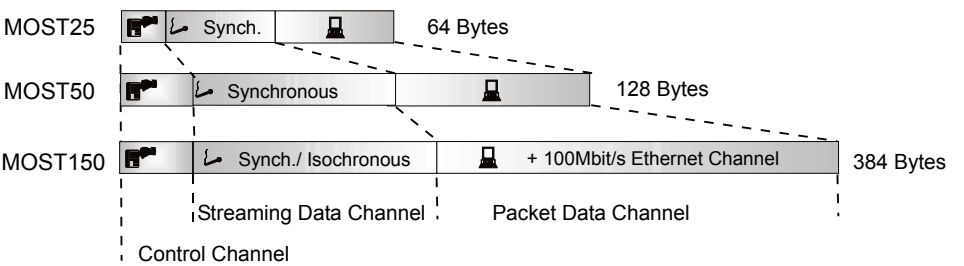


図2.6:MOSTフレームの基本構造

データリンク層のMOSTプロトコルの詳細は、第5章で説明します。

2.5.2 TimingMaster、TimingSlave

MOSTは一対多のポイント ツー ポイント データフロー システムで、ストリーミング データには1つのソースと任意の数のシンクがあります。これらのデバイスは全てデータストリームからリカバリした共通システムクロック パルスを使います。こうして全てのデバイスの位相が互いに揃い、全てのデータを同期送信できるため、信号のバッファリングと信号処理のための仕組みは不要です。

システムクロックは、システム内のいずれか1つのノードに実装されたTimingMasterによって生成されます。自動車では、通常インフォテインメント システムのヘッドユニットにTimingMasterがあります。それ以外のノードは全てTimingSlaveと呼ばれ、PLL接続を使ってシステムクロック パルスに同期します(図2.7参照)。

TimingMasterが送信したフレームがリングを1周してTimingMasterに戻ると、TimingMasterはPLL接続を使って信号を回復した後、次のフレームを生成します。

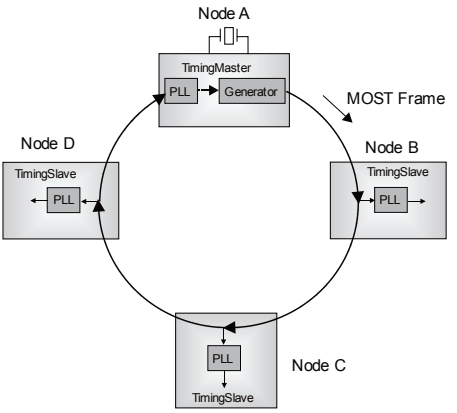


図2.7:MOSTリング内のTimingMasterとTimingSlave

ロック/アンロック

TimingSlaveが受信した入力信号にPLLを使って同期できる場合をロック状態と呼び、同期できない場合はアンロック状態へ移行します。

TimingMasterがリングを1周して戻ってきた信号からフレームを再生成できる場合、TimingMasterはロック状態です。

光物理層の場合、ロック状態はリングで光が点灯している事(点灯状態)を意味します。

詳細は、セクション7.4.1で説明します。

2.6 物理層

表2.1に、物理層の主要なパラメータをまとめます。MOST25はデータストリームのビットレートが約25 Mbit/sである事を意味しており、光物理層を使います。実際のデータレートは、システムのサンプリング レートで決まります。サンプリング レート44.1 kHzでは、MOSTフレーム (フレーム長512ビット)が毎秒44,100回送信されるため、データレートは22.58 Mbit/sとなります(セクション6.5.1参照)。これはMOST50の場合も同じです。ただしフレーム長が1,024ビットのため、同じサンプリング レートならデータレートは2倍です。MOST50では電気物理層も追加で規定されました。

第1世代のネットワーク インターフェイス コントローラ(NIC)はMOST25でのみ使えます。第2世代のインテリジェント ネットワーク インターフェイス コントローラ(INIC)は、MOST25、MOST50、MOST150で使えます。これら2つの世代のMOSTネットワーク インターフェイス コントローラは、第10章で説明します。

	MOST25	MOST50	MOST150
ビットレート	約25 Mbit/s	約50 Mbit/s	約150 Mbit/s
物理層	光	電気 光	電気(同軸ラインドライバを使用) 光
NIC	OS8104; OS8104A	-	-
INIC	OS81050; OS81060	OS81082; OS81092	OS81110; OS81120

表2.1:物理層の主要なパラメータ

2.6.1 光物理層

MOSTは物理伝送媒体としてプラスチック光ファイバ(POF)を使います。このため、EMC特性は非常に良好です。

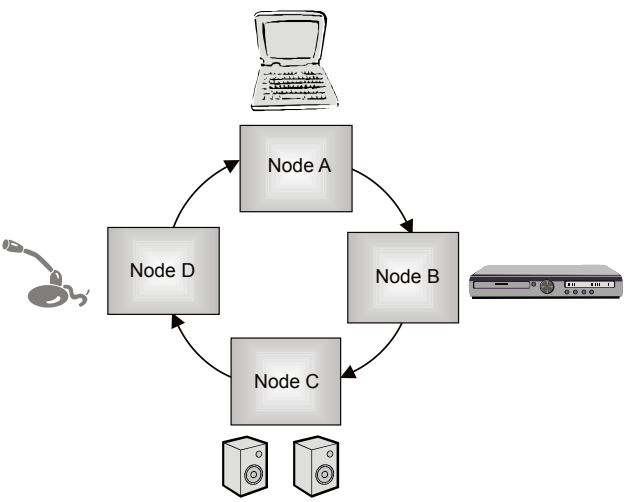


図2.8:シンプルなMOSTリング

MOSTネットワークの基本的なアーキテクチャは論理リングで、デバイスからデバイスへとデータを受け渡していきます。論理リング構造は通常、物理リングとして実装されます(図2.8参照)。しかしこれは必須ではありません。例えば、1つのハブを使ってリング型とスター型を組み合わせたトポロジも可能です。スター型トポロ

ジではネットワークの他の部分に影響を与えずに簡単にノードを追加または削除できるため、システムの信頼性が向上します。

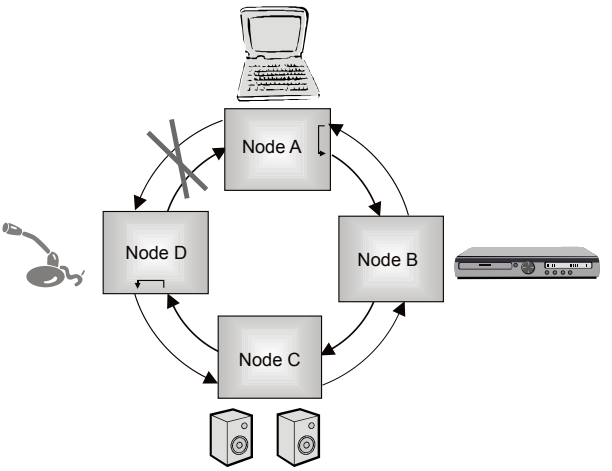


図2.9:ダブルリング構造

スター型、またはリング型とスター型の組み合わせ等、どのようなトポロジであってもノードからノードへとデータストリームを転送していく原理は常に維持されます。

MOSTでエラーが発生した場合の可用性を高めるため、ダブルリング構造とする事ができます。このためには、各ノードに光レシーバとトランスミッタが2つずつ必要です(図2.9参照)。2つのノード間に問題が発生した場合、冗長セグメントを使ってリングを閉じる事ができます。Bosch社の会議システムPraesideo [Praesideo]は、このようなダブルリング構造を採用しています。この製品は大規模なビルの複合体で採用されており、欧州安全規格EN 60849に準拠しています。

光物理層の詳細は第6章で説明します。

2.6.2 電気物理層

MOST50とMOST150では、同軸ラインドライバを用いた電気物理層も定義されています。これについてはセクション6.6で説明します。

2.7 MOSTデバイス

図2.10に、図2.1に基づくMOSTデバイスの基本アーキテクチャを示します。この図に示したデバイスは光物理層(oPhy)を使っています。光ファイバレシーバ(FOR)は光信号を電気信号へ変換し、この電気信号がインターフェイス コントローラのRx入力へ送信されます。コントローラからのTx信号は光ファイバ トランスミッタ(FOX)で再び光信号へ変換されます。レシーバとトランスミッタを総称して光ファイバ トランシーバ(FOT)または物理インターフェイスと呼びます。

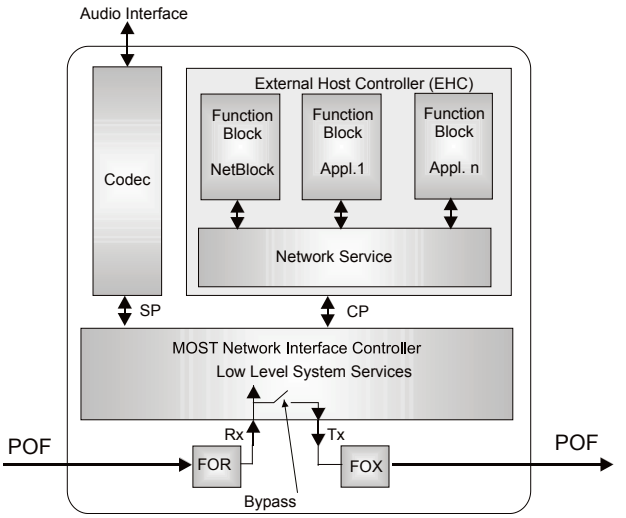


図2.10:光物理層を使った MOSTデバイスのモデル

データリンク層はMOSTネットワーク インターフェイス コントローラに実装され、ここで3種類のチャンネルへのアクセスを制御します。ネットワーク サービスとファンクション ブロックはマイクロコントローラに実装されます。このマイクロコントローラは、外部ホスト コントローラ(EHC)とも呼びます。EHCからパケットデータ チャンネルと制御チャンネルへのアクセスは、MOSTネットワーク インターフェイス コントローラの制御ポート(CP)経由で実行します。

ネットワーク サービスについては第9章で説明します。

ストリーミング サービスには、プロセッサを使わないコーデック (シンプルなシステムの場合)またはシグナル プロセッサ (ハイグレード システムの場合)が使えます。MOSTネットワーク インターフェイス コントローラとのインターフェイスは、NICの場合はソースポート(SP)、INICの場合はストリーミング ポートです。これに加え、INICには全てのチャンネルをアドレス指定できる汎用の効率的なインターフェイスとしてMediaLBがあります(セクション10.11参照)

MOSTネットワーク インターフェイス コントローラは、バイパスを内蔵しています。コントローラが通常動作モードでない場合はバイパスが閉じたままとなり(バイパス有効)、受信した入力信号をほとんど遅延なしにそのままTxから送信します。例えばエラー発生時やデバイス起動時がこれに該当します。

2.8 ネットワーク管理

アプリケーション機能(例: CDチェンジャ、チューナ)のためのファンクション ブロック以外に、NetBlock、PowerMaster、NetworkMaster、ConnectionMaster等、ネットワーク管理機能のためのファンクション ブロックも多数あります。これらについては第7章で説明します。

NetBlockは全てのデバイスに実装する必要があります。ネットワーク管理に関するその他のファンクション ブロックは、システム全体で1つだけ存在します。これらのファンクションをどのデバイスに実装するかは、仕様には定義されていません。システム内の複数のデバイスに分散する事もできますが、通常は全てヘッドユニットに実装します。

2.8.1 NetBlock

NetBlockは、デバイスの管理を司ります。例えば、NetBlockはデバイスに実装されている全てのファンクション ブロックのリストを持っており、デバイスの全てのアドレス (ノード位置アドレス、論理アドレス、グループアドレス)を管理します。詳細はセクション4.3.4で説明します。

2.8.2 PowerMaster

PowerMasterとは、MOSTネットワークの中でMOSTネットワークの起動とシャットダウンを司り、電源のステータスを監視するデバイスのファンクションをいいます。PowerMaster自体にはファンクション ブロック仕様はなく、これらの機能を持つデバイスを示すファンクション ブロックIDのみが定義されています。PowerMasterファンクション ブロックについては、セクション4.3.4で説明します。

ネットワークは、任意のノードから復帰させられます。PowerMaster(例: ゲートウェイ)以外によってトリガされた復帰の事をスレーブ復帰と呼びます。

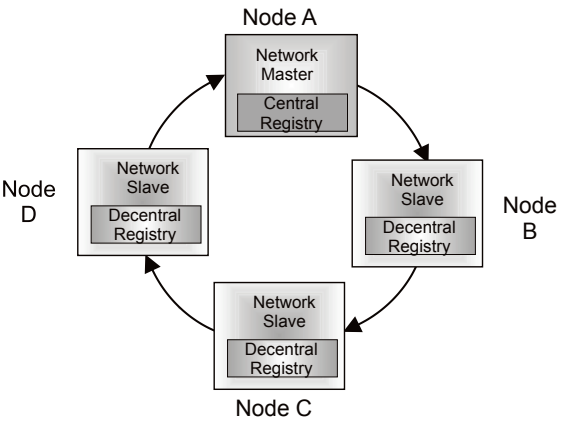


図2.11:NetworkMasterとNetworkSlave

2.8.3 NetworkMaster、NetworkSlave

NetworkMasterはシステムを起動させ、ネットワーク ステータスとセントラル レジストリを管理します。セントラル レジストリとは、ネットワーク内に存在する

全てのファンクション ブロックと、そのファンクション ブロックが実装されているデバイスのアドレスをリストにしたものです。

リング内のその他全てのデバイスをNetworkSlaveと呼びます。NetworkSlaveは、セントラル レジストリの一部の情報をデセントラル レジストリとして持つ事ができます。デセントラル レジストリには、そのデバイスの通信相手のファンクション ブロックが登録されます。

図2.11に、NetworkMasterとNetworkSlaveで構成されるMOSTリングを示します。

第7章では、NetworkMasterの機能を中心に説明します。

NetworkMasterに対応するファンクション ブロックは、NetworkMasterです(セクション4.3.4参照)。

2.8.4 ConnectionMaster

ConnectionMasterは、Connection Managerとのインターフェイスを行うファンクション ブロックです。ストリーミング接続のセットアップと切断は、Connection Managerが行います。ConnectionMasterファンクション ブロックの実装は任意です(セクション4.3.4参照)。

図2.12に、ConnectionMasterの基本的な動作モードを示します。イニシエータはConnectionMasterに対して接続の確立を命令します。その結果、ConnectionMasterは同期領域に必要な帯域幅を確保し、データソースとデータシンクのチャンネル番号を報告します。切断は、同様の手順で実行されます。

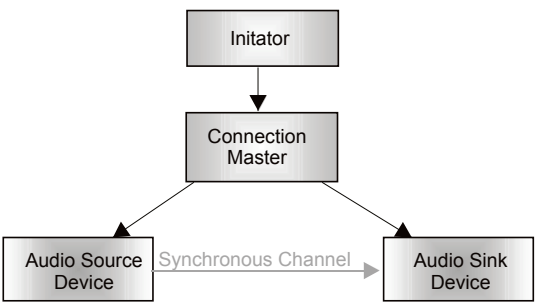


図2.12:ConnectionMasterの基本原理

2.9 エラー管理および診断

MOST仕様では、リングブレーク診断を定義しています。リングブレーク診断はPOFの破損箇所を特定し、その結果を専用のデバイスへ送信できます。MOST仕様はオプションでECL (Electrical Control Line)もサポートしています。詳細は第8章で説明します。

『Diagnostic Protocols Adaption Specification』[DPA]では、UDS (Unified Diagnostic Services)等の各種診断プロトコルをMOSTに適応させるための要件を定義しています。

2.10 マルチメディア データの転送

ストリーミング接続でコンテンツを転送するには、符号化手順を定義すると共に、コピープロテクトを徹底する事が必要です。特に商用のエンターテインメント コンテンツは保護する必要があります。

MOSTはPCMやMPEG等、既存の符号化手順を採用しており、これらの実装方法は『MOST Specification for Stream Transmission』[Stream]で説明しています。

MOSTは、著作権保護されたコンテンツの送信にはDigital Transmission Content Protection (DTCP)規格を適用します。この技術は、Intel、パナソニック、東芝、ソニー、日立のデバイスメーカー5社が共同で設立した5C Groupによって開発されました。DTCP規格はDigital Transmission Licensing Administrator (DTLA)のウェブサイト(www.dtcp.com)で入手できます。DTCPは、セットトップ ボックスからDVDビデオレコーダへ信号を安全に転送するために開発されました。ハリウッドを代表する映画会社のソニーとWarnerもデジタル ネットワークでの映画配信にこの保護方式を採用しています。

MOST仕様の『MOST Content Protection Scheme DTCP Implementation』[ContProt]は、この技術を使うために必要なファンクションとサービスを定義しています。

3 MOST仕様の概要

この章では、MOST仕様の全体構造について説明します。各仕様の最新バージョンは、MOST Cooperationのウェブサイト(http://www.mostcooperation.com)に掲載されている最新文書リストから入手できます。

3.1 構造

図3.1に、MOSTフレームワークの構造と、各文書間の依存関係を示します。MOSTフレームワークの中核の文書が、MOST仕様です。矢印で示したように、MOSTファンクション ブロック(FBlock)の仕様やそれに対応する動的仕様等、全ての文書がMOST仕様を参照します。

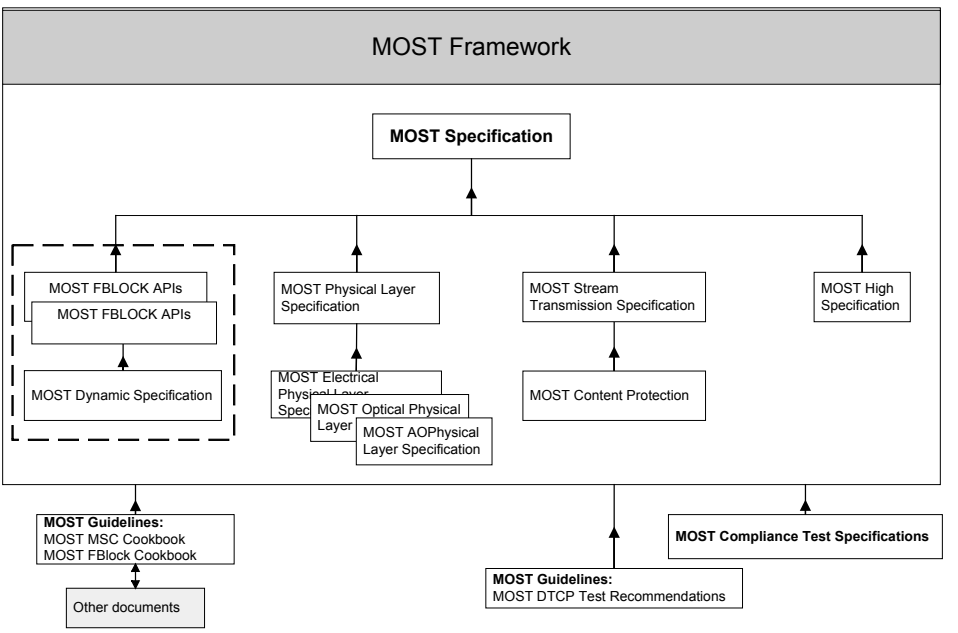


図3.1:MOSTフレームワークの文書体系

3.2 MOST仕様

MOST仕様[MOST 3.0]は、MOSTの階層モデルに基づいて基本的に以下のセクションで構成されています。

- アプリケーション
- MOSTのシステム コンセプト
- デバイスモデル
- MOSTデバイス間の通信
- ファンクション クラス
- ネットワーク
- データ トランスポート
- 制御チャンネル
- パケットデータ チャンネル
- 同期チャンネル
- アイソクロナス チャンネル
- Ethernetチャンネル
- デバイスの動的挙動
- エラー管理
- 診断

MOSTシステムは、物理ネットワークを高い抽象度で表現したファンクション モデルを使います。通信は、ファンクション ブロック(FBlock)とファンクションを使って行います。これらは、物理デバイスから切り離した形で物理ネットワークに存在できます。

ファンクション モデルに基づき、MOSTシステムは厳密な階層構造をとります。最上位の階層はSystem Masterで、この中にある1つまたは複数のコントローラが多数のスレーブを制御します。この階層構造により、システムに関する情報をSystem Masterといくつかのコントローラに集約します。

System Masterは通常ヒューマンマシン インターフェイス(HMI)に実装され、高い抽象度でシステム全体を制御し、ユーザにシステム機能を提供します。

コントローラはシステムに関する部分的な情報のみを持ちます。コントローラはシステムの複雑な拡張機能を制御し、これらをシンプルな形で上位の階層へ提供します。全てのコントローラは、特定の機能(例: 全オーディオ機能)の処理に特化しています。

MOST仕様では、システム コンセプトの説明に続いてMOSTデバイス間の通信およびファンクション クラスについて説明しています。

ネットワークに関するセクションでは、MOSTのテレグラムの構造、および5種類のデータ トランスポートの概念(制御チャンネル、パケットデータ チャンネル、同期チャンネル、アイソクロナス チャンネル、Ethernetチャンネル)について説明しています。

デバイスの動的挙動に関するセクションでは、システム状態、システムの起動、NetworkMasterへのファンクション ブロックの登録について説明しています。これ以外に、システムの復帰、エラー管理、診断に関するセクションがあります。

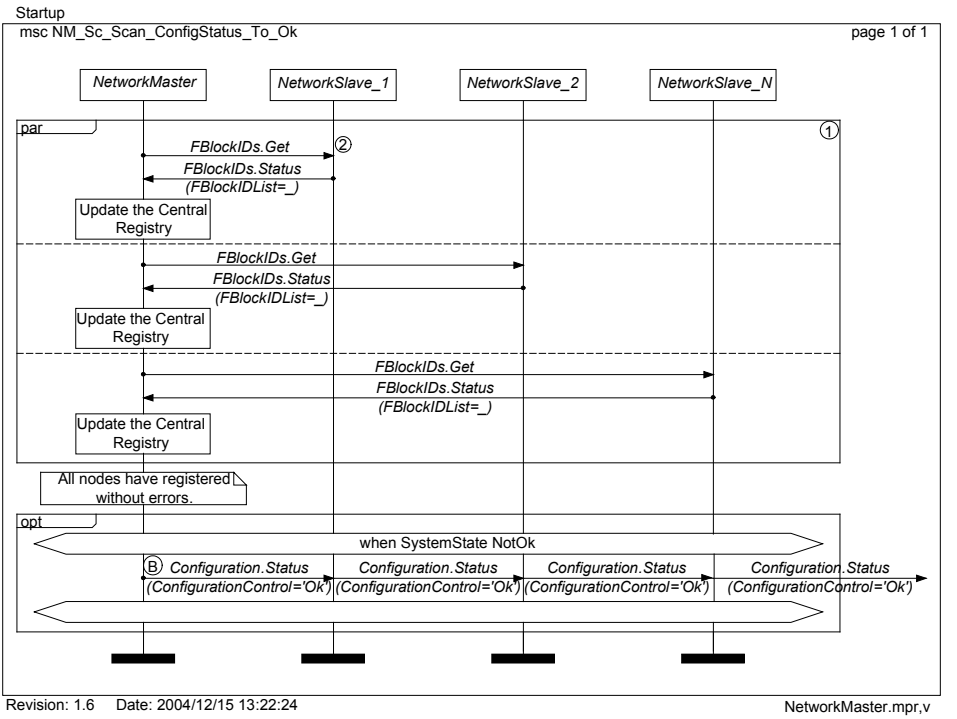


図3.2:シナリオ例:通常のシステムスキャン

エラー管理のセクションでは、低電圧、過熱、デバイスエラーの場合の挙動について説明しています。また、同期の失敗(アンロック)、登録情報の変更(ネットワーク変更イベント) (セクション7.1.2参照)等のシステムエラーについても説明しています。

診断のセクションでは、リングブレーク診断、突発的信号OFFの場合の挙動、および符号エラーカウンタについて説明しています。これらの診断により、機能別車載要件を満たします。

3.3 MOST動的仕様

MOST動的仕様[DynSpec]は、MOST仕様と緊密に結びついています。この仕様では、MOST仕様の中からネットワーク管理、接続管理、電源管理の項目を取り上げ、Message Sequence Chart (MSC)形式で説明しています。

図3.2に、通信パートナであるNetworkMasterと複数のNetworkSlave間での通常のシステムスキャンのシナリオを例として示します。parセクションでは、NetworkMasterが制御メッセージ`FBlockIDs.Get()`を使って、全てのMOSTデバイスに対して利用可能なファンクション ブロックに関する情報を問い合わせています。この情報に基づき、セントラル レジストリを構築します。optセクションでは、ネットワークがNotOK状態(セクション7.1.2参照)の時にNetworkMasterが制御メッセージ`Configuration.Status(OK)`を送信する場合を記述しています。

3.4 MOSTファンクション カタログ

MOSTファンクション カタログ(FBlockライブラリ)には、定義済みの全てのファンクション ブロックが記述されています。表4.7(セクション4.4.1)に、利用可能なファンクション ブロックの一覧を示します。

全てのFBlockには、そのFBlockの静的および動的挙動を定義した専用の文書があります。静的情報の構造は、XMLで記述します。その構造はファンクション カタログのDTD (Document Type Definition)で定義されており、『MOST DTD Cookbook』[DTD Cook]で参照されています。動的挙動は、MOST動的仕様[DynSpec]に従いMSCを使ってシナリオ形式で記述します(セクション3.2参照)。

図3.3に、FBlock EnhancedTestabilityのファンクションNotificationMatrixSizeの静的記述モードの概要を示します。このファンクションは、あるデバイスの全てのFBlockのノーティフィケーション マトリクスの最小サイズと最大サイズの値を通知します。このファンクションは必須であり、全てのデバイスに実装する必要があります。このファンクションはOpTypeとしてGet、Status、Errorを使います。Statusが使うパラメータMinSizeとMaxSizeのデータ型はUnsigned Byteです。

3.5 MOSTコンプライアンス テスト仕様

認証テストプロセス(第12章参照)の構造に従い、各コンプライアンス レベルに対してそれぞれ専用のテスト仕様が定義されています。

認証テストプロセス全般の作業手順は、『MOST Compliance Requirements』[ComReq]として文書化されています。

光物理層のテストは『MOST Compliance Test of Physical Layer』[ComPhyL 1.0]と『MOST Compliance Verification Procedure – Physical Layer』[ComPhyLP 1.0] (MOST25)、および『MOST150 oPhy Measurement Guideline』[ComPhyL 150]と『MOST150 oPhy Compliance Verification Procedure – Physical Layer』

[ComPhyLP 150]で定義しています。訂正または変更がある場合、それぞれのエラッタシート([ComPhyL_Err]、[ComPhyLP_Err])に記載されます。電気物理層のテストは『MOST ePhy Compliance Test Specification (ePhy Compliance Test Patterns)』[Com_ePhyL]で定義しています。

2.1.11 NotificationMatrixSize (0x213)

Occurrence: Mandatory

Function has to be implemented in every node.

The NotificationMatrixSize property contains the minimum and maximum values of the size of the Notification Matrix across all FBlocks.

2.1.11.1 Format of Function

Function classes: Unclassified Property

FBlock	Function	OPType	Parameter
EnhancedTestability (0x0F)	NotificationMatrix Size (0x213)	Get	-
		Status	MinSize, MaxSize
		Error	ErrorCode, ErrorInfo

2.1.11.2 Parameter

MinSize

Decribes the minimum size of all Notification Matrices. If at minimum one FBlock supports notification then MinSize is greater than 0.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

MaxSize

Decribes the maximum size of all Notification Matrices. The value 0xFF indicates that the device uses dynamic Notification Matrices

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

図3.3:FBlock EnhancedTestabilityの静的記述の例[FBET]

コア コンプライアンスに関連するテストケースは、『MOST Core Compliance Test Specification』[Core]とエラッタ[Core_Err]で定義しています。

プロファイル コンプライアンスに関連する汎用テストは、『MOST Profile Compliance Test Specification』[ComProf]で定義しています。

個々のファンクション ブロックとAPIのテストは、それぞれ専用のテスト仕様に基づいて行います(例: ConnectionMasterの場合は『MOST Profile ConnectionMaster Compliance Test Specification』)。

3.6 その他の仕様、ガイドライン、クックブック

MOSTフレームワークの仕様には、以下の項目も含まれます。

- 物理層仕様
- MOST High Protocol [MHP]: パケットデータ チャンネルでのアプリケーション メッセージの転送を定義した仕様
- MAMAC (MOST Asynchronous Medium Access Control) [MAMAC]: TCP/IP、IPX、NetBEUI、ARP等のネットワーク プロトコルを転送するための仕様
- MOSTコンテンツ保護仕様
- 『MOST Stream Transmission Specification』: オーディオおよびビデオデータ転送用のストリーミング フォーマットを定義
- アプリケーション推奨文書 (例: [MOST 2003])
- ガイドラインとクックブック

仕様以外に、蓄積したノウハウおよび手順の手法を適切な方法で文書化しておく事が重要です。1つには、システム インテグレータによるテスト手順のベスト プラクティスをまとめたアプリケーション推奨文書があります。さらに、Message Sequence Chart(補遺A参照)の使い方を説明した『MOST MSC Cookbook』[MSC]の他、『MOST DTD Cookbook』[DTD Cook]、『MOST FIBEX Cookbook』[FBX Cook]等の各種ガイドラインとクックブックもあります。これらは経験の集大成であり、幅広い記述の可能性を本質的な面に絞ったものです。

4 アプリケーション フレームワーク

MOST仕様では、データ転送とネットワーク管理の基礎となる下層のMOSTネットワークだけでなく、上位階層にアプリケーションを実装するためのプロトコルとメカニズムについても定義しています。MOST仕様では、アプリケーションのインターフェイスをファンクション ブロック (FBlock)と呼びます。これは、Bluetooth規格のプロファイルに相当します([Bluetooth]参照)。

MOSTアプリケーション フレームワークでは、特定のデバイスへの実装に依存しないアプリケーションを作成できます。また、MOSTシステム内で相互通信する相手側アプリケーションにおいても実装から独立した開発が可能です。MOST仕様で標準化されたFBlockを使う事で、リユース性と相互動作性の高いアプリケーションを開発できます。例えば、ラジオのヒューマンマシン インターフェイス(HMI)を実装したアプリケーションは、ラジオチューナの標準化されたアプリケーション インターフェイスを持つ任意のデバイスと組み合わせて使えます。また、互換性のあるインターフェイスを実装していれば別々のメーカーのデバイスを組み合わせて使えます。

この章では、MOST仕様でアプリケーション フレームワークを定義している部分について説明します。最初に、基礎であるデバイスモデルについて説明します。次に、MOSTシステムでアプリケーションがデータ通信を行う際に使うプロトコルについて説明します。最後に、MOSTシステムの管理タスクに必要なインターフェイス、および代表的な車載インフォテインメント システムのアプリケーション インターフェイスについて説明し、その使用例を紹介します。

4.1 デバイスモデル

図4.1に、MOST仕様が定義しているMOSTデバイスのモデルを示します。ハードウェア レベルでは、デバイスはMOSTネットワーク インターフェイス コントローラ(NIC)経由で物理層にアクセスします。ネットワーク サービスはドライバ層を実装します。これにより、インターフェイス コントローラ チップへのアクセスを制御し、アプリケーションがメッセージの送受信等の基本的な機能を実行できるようにします。

MOST仕様では、特定の機能(すなわち関連するファンクションの集合)を提供するアプリケーション インターフェイスをFBlockと呼びます。MOSTシステム内での管理タスクに必要なFBlockであるNetBlock (セクション4.3.4参照)と、コンプライアンスのためのFBlockであるEnhancedTestabilityは、全てのデバイスに実装する必要があります。これ以外に、各デバイスはそのタイプに応じて、対応するアプリケーション機能を実現するFBlockを1つまたは複数実装できます。よくある例として、

アンプとラジオチューナの機能を備えたデバイスでは、システム内でこれらの機能がアンプのFBlockとラジオチューナのFBlockで表現されます。

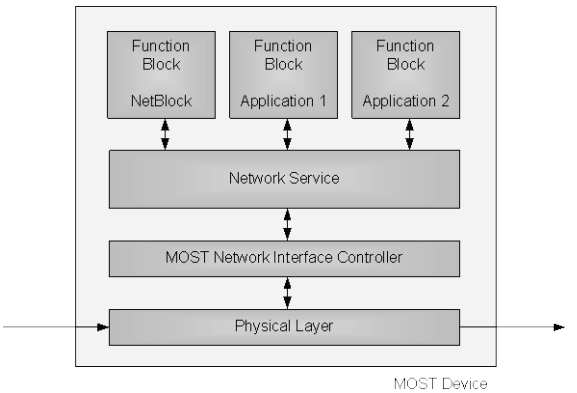


図4.1: MOSTデバイスのモデル

MOSTシステムでは、ファンクション アドレスを使ってFBlockをアドレス指定します。これにより、ファンクションの物理的な位置には関係なくその機能にアクセスできます。つまり、FBlockで表現されるファンクションがどのデバイスに実装されているかは重要ではありません。

アプリケーションが使用、制御するFBlockをスレーブと呼びます。FBlockを使うアプリケーション等のインスタンスをコントローラと呼びます(図4.2参照)。

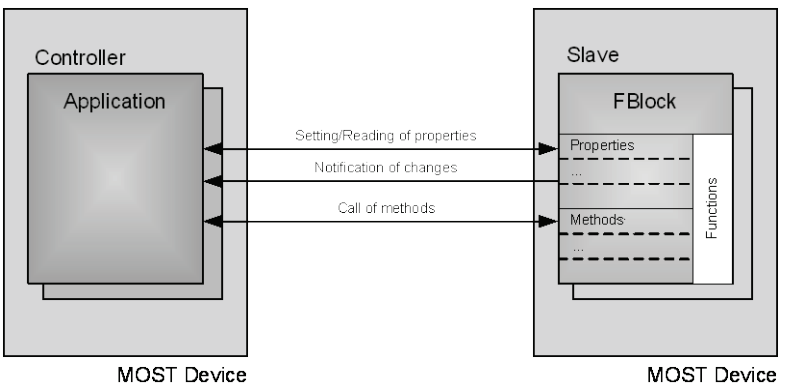


図4.2:FBlockとのやりとり

FBlockの中では、2種類のファンクションを区別します。1つはFBlockの特定の属性(例: ラジオチューナの現在選局中の周波数)を記述した「プロパティ」で、もう1つは特定のアクション(例: 利用可能なラジオ局のスキャン)をトリガする「メソッド」です。

コントローラは、スレーブのプロパティの値を読み出す事ができ、変更をサポートしているプロパティの場合は変更できます。また、MOST仕様はノーティフィケーション メカニズムを定義しており、コントローラが必要に応じてスレーブのプロパティを登録しておくと、そのプロパティ値が変化した時にスレーブから通知を受け取る事ができます。

コントローラがメソッドを呼び出すと、メソッドはスレーブの対応するアクションをトリガします。メソッドの実行が完了すると、スレーブはコントローラへ実行結果を返します。

4.2 アプリケーション プロトコル

MOST仕様はMOSTシステム内におけるアプリケーション間の相互通信に関するプロトコルを定義したもので、MOSTアプリケーション フレームワークの中心に位置づけられます。アプリケーション プロトコルは、下層チャンネルのプロトコルがベースとなっており、これらの影響を受けます。初期のMOSTシステムでは、MOST制御チャンネルの組み合わせが一般的でした。送信する内容のサイズ増大に伴い、現在ではパケットデータ チャンネルを使うケースが増えています。

制御チャンネルでは、アプリケーション メッセージを送信するためのプロトコルをアプリケーション メッセージ サービス(AMS)と呼びます。パケットデータ チャンネルでは、MOST High Protocol (MHP)を使ってアプリケーション メッセージを送信できます。これらのプロトコルは、セグメンテーションやフロー制御等を処理します。詳細は第5章を参照してください。

MOSTアプリケーション プロトコルは、MOSTシステムのアプリケーションが提供するサービスへのアクセス方法を決定します。また、アクセスの際に交換するメッセージのデータ フォーマットを定義します。さらに、FBlockのプロパティとメソッドに対する読み書きアクセスの際のオペレーション シーケンスと条件、およびノーティフィケーション メカニズムの使い方も定義します。

4.2.1 データ形式

このセクションでは、アプリケーション メッセージ サービス(AMS)プロトコルのデータ形式について説明します。データは以下の要素から成ります。

[DeviceID.]FBlockID.InstID.FktID.OPType (Data)

通信シーケンスまたは例を記述する際は、各要素をドット区切りで表記した上記の記法を用います。

基本的に、このデータ形式には以下の3つの領域があります。

1. オプションのデバイスアドレス(DeviceID)、FBlockのID (FBlockID)、MOSTシステムにおけるインスタンスのID (InstID)から成るアドレッシング領域

2. ファンクションID (FktID)でアドレス指定されたファンクション、およびそのファンクションに適用されるオペレーション(OPType)
3. 長さの情報(Length)とファンクションのパラメータ用のデータ領域(Data)から成るデータフィールド

下表に、個々の要素とそのサイズの概要を示します。

要素	サイズ	説明
DeviceID[1]	16ビット	ターゲット デバイスのアドレス
FBlockID	8ビット	FBlock ID
InstID	8ビット	FBlockのインスタンス
FktID	12ビット	ファンクションID
OPType	4ビット	オペレーション
Length[2]	16ビット	データフィールドの長さ
Data	0～65535バイト	データフィールド

表4.1:MOSTアプリケーション プロトコルの要素

デバイスアドレス(DeviceID)

デバイスアドレス(DeviceID)のサイズは16ビットで、状況に応じて個々のレシーバ、複数レシーバのグループ、メッセージのセンダのいずれかを表します。

アプリケーション レベルで個々のデバイスをアドレス指定する場合、論理デバイスアドレスを使います。ノード位置アドレスを使ってアドレス指定するのは、ネットワーク管理またはデバッグの場合のみです。グループアドレスを使うと、MOSTシステムの複数のデバイスを同時にアドレス指定できます。ブロードキャスト アドレスには0x03C8(ブロッキング)と0x03FF(ノンブロッキング)があり、これらを使ってMOSTシステム内の全てのデバイスをアドレス指定できます。デバイスアドレスとアドレッシング モードの種類については、第5章「MOSTプロトコル」で詳しく説明します。

アプリケーション レベルでプログラミングを行う場合、デバイスアドレスを使う事はそれほど多くありません。アプリケーション レベルでは、ファンクション アドレス(FBlockIDとInstID)のみを使ってアドレスを指定できます。通常、アプリケーションは通信相手のネットワーク アドレスを知りません。この場合、アプリケーションは参照用にワイルドカード0xFFFFを送信メッセージに挿入し、このメッセージをネットワーク サービスに渡す事ができます。ネットワーク サービス

[1] FBlockはファンクション アドレスでアドレス指定されるため、DeviceIDは高次のMOSTネットワーク サービス(層II)では使いません。

[2] Lengthは直接送信されず、下層のプロトコルに基づきます。

は通信相手のアドレスを決定し、そのアドレスでワイルドカードを置き換えてからメッセージを送信します(セクション9.2.3参照)。

FBlock ID (FBlockID)

FBlockIDはFBlockのタイプを記述し、サイズは8ビットです。0xA0未満のFBlockIDは、MOST仕様が定義しています。この中には、管理タスクに必要なFBlock(システムFBlock)用に予約されたFBlockIDと、特定のデバイス ファンクションまたはデバイスタイプを表すFBlockIDが含まれます(表4.2参照)。MOST Cooperationで標準化されているシステムFBlockとアプリケーションFBlockについては、この章のセクション4.3と4.4で説明します。

0xA0以上のFBlockIDは、システム インテグレータ(0xC8を除く0xA0～0xEFのFBlockID)またはデバイスメーカー(0xFCを除く0xF0～0xFEのFBlockID)が独自のFBlockを定義できます。FBlockID 0xFFは、1つのデバイスに含まれる全てのFBlock(NetBlockを除く)をアドレス指定するワイルドカードとして使います。

FBlock名 または領域	FBlockID	説明
予約済み	0x00	ネットワーク サービス用に予約済み
NetBlock	0x01	全てのデバイスに実装されるシステムFBlock
NetworkMaster	0x02	NetworkMasterのシステムFBlock。システム内の1つのデバイスにのみ実装
ConnectionMaster	0x03	ConnectionMasterのシステムFBlock。任意でシステム内の1つのデバイスにのみ実装
PowerMaster	0x04	システム内のPowerMasterを示すFBlockID
EnhancedTestability	0x0F	全てのデバイスに実装
予約済み	0x70～0x9F	将来用に予約済み
システム固有	0xA0～0xEF	システム インテグレータが割り当て(0xC8は予約済み)
サプライヤ固有	0xF0～0xFB	デバイスメーカーが割り当て
予約済み	0xFC	
サプライヤ固有	0xFD、0xFE	デバイスメーカーが割り当て
ワイルドカード、全て	0xFF	FBlockブロードキャスト

表4.2:FBlock領域の構造

FBlockのインスタンス(InstID)

内蔵型のCDドライブと外付けCDチェンジャ等、1つのMOSTシステム内で同じタイプのFBlockを複数使う事もあるため、InstIDで個々のインスタンスをさらに区別します。InstIDのサイズは8ビットで、既定値は0x01です。FBlockIDとInstIDを総称してFBlockのファンクション アドレスと呼びます。

FBlockIDとInstIDの組み合わせは、システム全体で重複してはいけません。まず、デバイスが自分自身のFBlockに適切なInstIDを割り当てます。割り当ては、0x01から昇順で割り当てる等の方法があります。システム全体でファンクション アドレスが重複しないようにするのは、NetworkMasterの役割です(セクション4.3.4参照)。例えば、2つの異なる物理デバイスがFBlock AudioDiskPlayerを持つCDプレーヤを実装しており、どちらのAudioDiskPlayerもInstIDが0x01で初期化されたとします。この場合、1つのMOSTシステムにファンクション アドレス0x31.0x01が2つ存在してしまいます。このような競合が発生した場合、NetworkMasterは片方のInstIDをリセットする必要があります。InstIDを動的に割り当てるのではなく、システム インテグレータが固定値にプリセットできます。

0x00と0xFFは、デバイスが同じタイプのFBlockを複数実装した場合にワイルドカードとして使う特別なInstIDです。0x00は、デバイス内の対応するFBlockの任意のインスタンスをアドレス指定します(ドントケア)。0xFFはデバイス内の全てのインスタンスをアドレス指定します(ブロードキャスト)。ワイルドカードが使えるのは要求の場合のみです。応答メッセージには応答元のFBlockの正しいInstIDを含める必要があります。

システムFBlockについては、MOST仕様でInstIDの特別な取り扱いを定めています。全てのデバイスへの実装が必須のFBlock(すなわちNetBlockとEnhancedTestability)は、InstIDがリング位置に対応します。TimingMasterの0x00から開始し、リング内の位置に合わせて各デバイスに1つずつ大きいInstIDを割り当てます。

システム内で1つのインスタンスしか存在できないFBlock NetworkMasterのInstIDは既定値で0x01です。0x00とする事も可能です。要求の場合、ワイルドカードである0x00を使う必要があります。

ファンクションID (FktID)

FktIDはFBlockの個々のファンクションを区別するもので、サイズは12ビットです。ファンクションは、プロパティ(すなわちFBlockの変数)またはメソッドのどちらかです。プロパティかメソッドかを問わず、ファンクションがどのオペレーションをサポートし、どのパラメータを使うか(詳細は後述)は、FBlock仕様のインターフェイス定義に記述されています。

プロパティとメソッドの違い、およびそれぞれのファンクション タイプと使い方は、セクション4.2.2で詳しく説明します。

FktIDでは、いくつかの領域が区別されます(表4.3参照)。「調整」領域のファンクション(FktID 0x000～0x1FF)は、全てのMOSTデバイスに同じ方法で実装する必要があります。この領域のファンクションは、GeneralFBlock仕様に基づいてMOST Cooperationが調整します(セクション4.4.2参照)。「アプリケーション」領域(FktID 0x200～0x9FF)には、各FBlock仕様においてFBlockの主要なアプリケーション機能を表すファンクションが属します。「一意」領域(0xA00～0xBFF)は、MOSTシステム全体で1つしか存在できないファンクションを含みます。この場合、システム全体での使用をシステム インテグレータが調整する必要があります。

FBlockID同様、システム インテグレータ(FktID 0xC00～0xEFF)またはデバイスメーカー(FktID 0xF00～0xFFE)が独自のファンクションを定義できる領域もあります。

名称	FktIDの領域	説明
調整	0x000～0x1FF	FBlockで管理用に使われるファンクション(MOST Cooperationが調整する)
アプリケーション	0x200～0x9FF	FBlockの中心的なアプリケーション機能を表すファンクション
一意	0xA00～0xBFF	システム全体で一意に定義されるファンクション
システム固有	0xC00～0xEFF	システム インテグレータが使用できるファンクション
サプライヤ固有	0xF00～0xFFE	デバイスメーカーが使用できるファンクション

表4.3:FktID領域の構造

オペレーション タイプ(OPType)

ファンクションに適用できる基本的なオペレーションの種類は、ファンクションのタイプ (プロパティまたはメソッド)で決まります。オペレーションは、アプリケーション プロトコルで定義されたプロパティとメソッドの相互通信パターンの基本的要素です。オペレーションは、サイズが4ビットのOPTypeで指定します。

下表に、合計16個のオペレーションを示します。これらのオペレーションは、FBlockのファンクションがプロパティかメソッドかによって意味が異なります。最初の9つのオペレーションはファンクションに対する要求を表しており、コントローラからスレーブへ送信するメッセージで使います。残りの7つのオペレーションはスレーブからの応答を表しており、対応するレポートで使います。

	プロパティに対するOPType	メソッドに対するOPType	OPType
要求:	Set	Start(廃止)[1]	0x0
	Get	Abort(廃止)[1]	0x1
	SetGet	StartResult(廃止)[1]	0x2
	Increment	予約済み	0x3
	Decrement	予約済み	0x4
	GetInterface[2]	GetInterface[2]	0x5
	-	StartResultAck	0x6
	-	AbortAck	0x7
	-	StartAck	0x8
応答:	ErrorAck[3]	ErrorAck	0x9
	-	ProcessingAck	0xA
	予約済み	Processing (廃止)[1]	0xB
	Status	Result (廃止)[1]	0xC
	-	ResultAck	0xD
	Interface[2]	Interface[2]	0xE
	Error	Error[4]	0xF

表4.4:オペレーション タイプ

各OPTypeの意味と、オペレーション シーケンスでの使い方は、セクション4.2.2で説明します。

データ領域のサイズ(Length)

Lengthはデータ領域のサイズをバイト単位で示す要素で、サイズは16ビットです。従って、データバイトの最大サイズは65,535バイトです。Length = 0の場合、データフィールドがエンプティである事を示します。

要素Lengthは直接は送信されず、プロトコルの種類(例: AMSまたはMOST High Protocol)に応じて下位のプロトコル層の情報を使ってNetwork Serviceが計算します。例えばAMSを使う場合、制御チャンネルのテレグラムの数と長さからLengthを求めます。

---

[1] MOST仕様3.0より、メソッドに対するOPTypeでパラメータSenderHandleを使わないものは全て廃止されました。

[2] InterfaceとGetInterfaceは任意選択のOPTypeです。

[3] 末尾に「Ack」の付くOPTypeの要求をプロパティに対して誤って使った場合、プロパティがOPType ErrorAckを返す事があります。

[4] SenderHandleが該当しないか不明な場合、メソッドがOPType Errorを返す事があります。

データとパラメータ型

ファンクションに1つまたは複数のパラメータがある場合、対応するパラメータ値がデータフィールド(Data)で送信されます。パラメータの型とシーケンスは、FBlock仕様内のファンクション インターフェイス仕様であらかじめ決まっています。

MOST仕様では、下表に示したパラメータ型を定義しています。

パラメータ	サイズ	説明
Boolean	1バイト	ブール値(真または偽)
BitField	1、2、4、8バイト	マスク付きビットフィールド
Enum	1バイト	Enumeration(列挙)
Unsigned Byte	1バイト	整数値(符号なし)
Signed Byte	1バイト	符号付き整数値
Unsigned Word	2バイト	整数値(符号なし)
Signed Word	2バイト	符号付き整数値
Unsigned Long	4バイト	整数値(符号なし)
Signed Long	4バイト	符号付き整数値
String	可変	符号化情報を含んだNULL終端文字列
Stream	可変	任意のデータストリーム
Classified Stream	可変	型情報を持つストリーム
Short Stream	可変 (最大255バイト)	長さの情報を持つショート ストリーム

表4.5:MOSTアプリケーション プロトコルのパラメータ型

以下に、個々のパラメータの概要を示します。詳細は、MOST仕様[MOST 3.0]を参照してください。これらパラメータ型の特徴は、他のプログラミング言語で使われるパラメータと似ています。

- **Boolean**: Boolean型のパラメータは、真または偽のどちらか1つをとります。対応するバイトの最下位ビットのみを使います。
- **BitField**: BitField型のパラメータは、必要に応じて1、2、4、8バイトのサイズでビットシーケンスを指定します。MOST仕様に従い、BitField型のパラメータは前半がマスクで後半がデータです。オペレーションを実行する際、データ部のどのビット位置に対してオペレーションを適用するかをマスク部のビットで指示します。実際には、このパラメータ型をマスクなしで使う事もあります。

- **Enum**: Enum型のパラメータは、事前に定義された最大256個の値のいずれか1つをとる事ができます。個々の値とその意味は、ファンクションのインターフェイス定義で決まっています。
- **Signed/Unsigned Byte** 、 Signed/Unsigned Word 、 **Signed/Unsigned Long**(整数値): MOST仕様は、符号付きまたは符号なしの1、2、4バイト整数値をサポートしています。符号付き数値の場合、最上位ビットが符号で、値は2の補数で符号化されます。(MOST仕様は、現在のバージョン3.0まで浮動小数点用のパラメータ型を定義していません。数値パラメータの小数点の位置は、インターフェイス定義で指数値を指定して定義できます。)
- **String**: 文字列を送信する場合、String型のパラメータを使います。文字列の1バイト目は符号化方式(例: 0x00ならUTF-16、0x01ならASCII)を表します。2バイト目以降は、NULL終端文字列です。各文字に対して何バイトを使うかは、符号化の種類で決まります。文字列の最後は1バイトのNULL文字です。UTF-16のように2文字を使う符号化の場合、最後は2バイトのNULL文字です。
- **Stream**: Stream型のパラメータは、任意のバイトシーケンスを定義します。他のパラメータタイプでは適切に記述できない場合、または動的なデータ構造が必要な場合にこのパラメータタイプを使います。Stream型のパラメータを他のパラメータと組み合わせて使う場合、最後かつ1つだけとする必要があります。
- **Short Stream**: Short Streamは、短いバイトシーケンスを表現するためのものです。バイトシーケンスの長さは最大256文字で、最初の文字で後続のバイトシーケンスの長さ(最大255バイト)を示します。このパラメータは長さの情報を持つためStream型パラメータのような制約はなく、最後のパラメータとする必要はありません。
- **Classified Stream**: Classified Streamは最初の2バイトでストリームの長さを示し、後続のNULL終端ASCII文字列(符号化情報なし)でメディアタイプを示します。メディアタイプは、HTTP1.1仕様([RFC 2616]参照)のメディアタイプに準拠します。

より柔軟なデータ構造を扱えるようにするため、**ストリームケース**と**ストリーム信号**の概念を導入しました。どちらもファンクションのインターフェイス定義で定義しています。

ストリームケースとストリーム信号を使うと、Stream型のパラメータを2つの異なる方法で構造化できます。

- ストリームケースの場合、1つのStream型パラメータがセレクタの値に応じて複数の異なるStream型パラメータのシーケンスに構造化されます。これは、プログラミング言語におけるバリアント レコードの概念に似ています。セレクタは任意の型のパラメータで、Stream型パラメータより前に現れる必要があります。多くの場合、セレクタにはEnum型のパラメータを使います。事前に定義したセレクタの値に応じて、ストリームは型と値の異なる複数のStream型パラメータのシーケンスから成ります。
- ストリーム信号の場合、1つのストリームを名前の異なる複数の連続する要素に分割できます。個々の要素のビット数は任意です。

ストリームケースとストリーム信号の詳細は、MOST仕様[MOST 3.0]を参照してください。

4.2.2 動的挙動

ここまで、MOSTアプリケーション プロトコルのメッセージ形式とその要素について説明しました。このセクションでは、その使い方と、それによって発生するメッセージ シーケンスについて説明します。

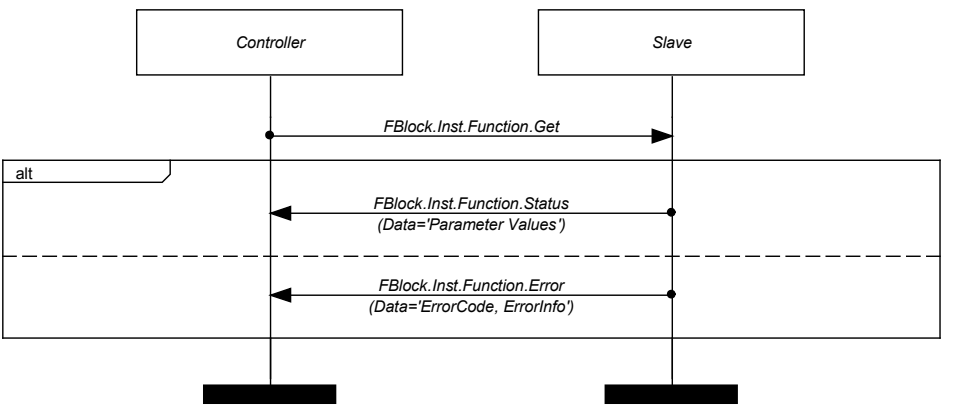


図4.3:Getオペレーションによるプロパティ値の問い合わせ

プロパティの問い合わせと設定

前述の通り、ファンクションはプロパティまたはメソッドのどちらかです。プロパティはFBlockの変数を記述したもので、1つまたは複数のパラメータ値から成ります。FBlockの現在のステータスは、そのFBlockの全てのプロパティ値で表します。プロパティの値は、MOSTアプリケーション プロトコルを用いて問い合わせまた

は設定が可能です。さらにMOST仕様では、プロパティの値が変化した時に、そのプロパティに関心のあるデバイスに自動で通知を送るノーティフィケーション メカニズムを定義しています。

プロパティの現在のステータスを問い合わせるには、**Get**オペレーション(OPType: 0x1)を用いた制御メッセージをFBlockの該当するファンクションに送信します。このファンクションは、**Status**オペレーション(OPType: 0xC)を用いたメッセージで応答します(Dataフィールドはプロパティの現在値を格納)。エラーが発生した場合、Errorオペレーション(OPType: 0xF)を用いたエラーメッセージを返します(エラー処理については次のセクション参照)。

図4.3は、このメッセージ シーケンスをMSC 2000の記法[Z.120 99]に準拠したMessage Sequence Chart (MSC)として表したものです。MOST Cooperationはこの記法を採用しており、以下の例もこの記法を使って説明します。MOST仕様で使うMSCの詳細は、補遺Aと[MSC]を参照してください。

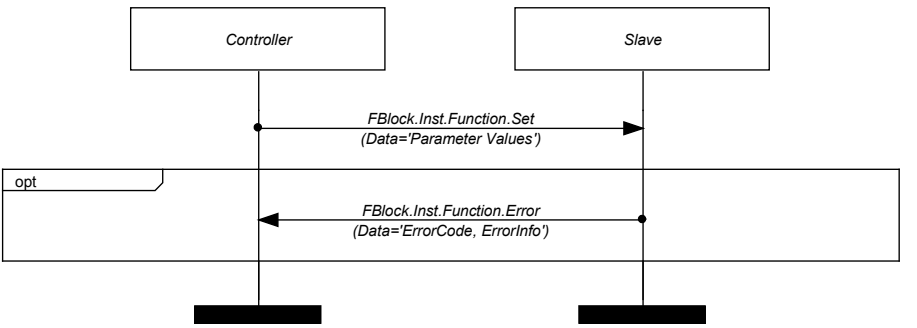


図4.4:Setオペレーションによるプロパティ値の設定

Setオペレーション(OPType: 0x0)は、プロパティの値を設定後にフィードバックが不要な場合に使います(図4.4参照)。プロパティ値の設定中にエラーが発生した場合のみ、FBlockはそのエラーに関する情報を返します。Setオペレーションは、コントローラがノーティフィケーションを設定してあるプロパティに対して実行してください(ノーティフィケーション メカニズムについては後述)。

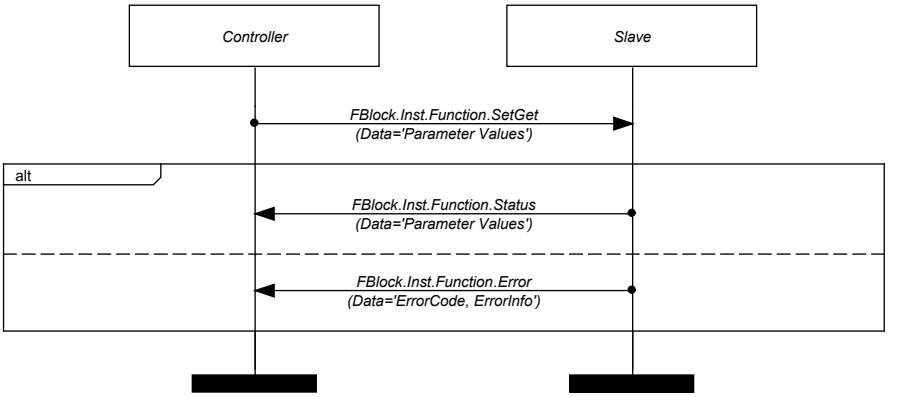


図4.5:SetGetオペレーションによるプロパティ値の設定

SetGetオペレーション(OPType: 0x2)はSetに似ていますが、変更が実行された事の確認として新しいプロパティ値をステータス メッセージとして返す点が異なります(図4.5参照)。このオペレーションは、ノーティフィケーションを設定していないプロパティの値を設定する場合に適しています。

これ以外に、MOST仕様では**Increment**オペレーション(OPType: 0x3)と**Decrement**オペレーション(OPType: 0x4)を定義しています。これらのオペレーションを使うと、プロパティの値をインターフェイス仕様で指定した量だけ増分または減分できます。

しかし、IncrementオペレーションとDecrementオペレーションはそれまでに説明した他のオペレーションほど多くは使われません。また、意図しないリトライによって値が1回ではなく2回増分または減分するため、エラーが生じやすい性質があります。

メソッドの実行

メソッドは、FBlockのステータスを変更するプロセス、または長時間継続するプロセス(例: ラジオ局のスキャン)を開始するために使います。

コントローラはメソッドを開始でき、必要に応じてメソッドを中止できます。メソッドを開始する際、メソッドの挙動(例: ラジオ局スキャンの方向)に影響するパラメータを任意で追加できます。メソッドの実行で発生した結果(この場合は放送局のリスト)を、メソッドを開始したコントローラ (イニシエータ)に返します。

デバイス内のどのコントローラ(例: アプリケーション)がメソッドを開始したかを区別するため、メソッドに対する全てのオペレーションに16ビットのパラメータSenderHandleを追加します。呼び出されたメソッドは、必ずこのパラメータを含めて応答を返します。これにより、イニシエータは自分自身のコマンド メッセージと応答メッセージを関連付ける事ができます。MOST仕様3.0では、

SenderHandleを使わないオペレーション タイプ(末尾に「Ack」の付かないオペレーション タイプ)が廃止されました。

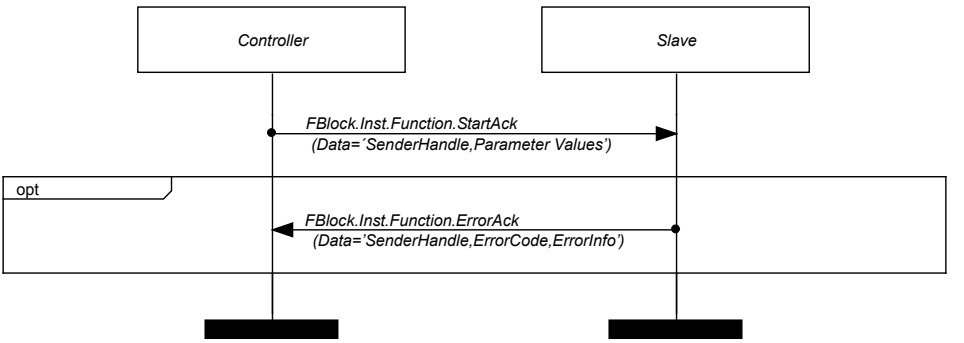


図4.6:StartAckオペレーションによるメソッドの開始

オペレーションに対する直接の応答が不要なメソッド、または連続動作をトリガするメソッドを開始する場合、**StartAck**オペレーション(OPType: 0x8)を使います(図4.6参照)。これはプロパティに対するSetオペレーションに相当するものです。StartAckに対するフィードバックは、エラー発生時のエラー メッセージのみです。

メソッドの実行結果をイニシエータに返す場合、**StartResultAck**オペレーション(OPType: 0x6)でメソッドをトリガします(図4.7参照)。実行中にエラーが発生しなければ、**ResultAck**オペレーション(OPType: 0xD)を用いたメッセージでイニシエータに実行結果を通知します。

結果の計算に時間がかかる場合(> 100 ms)、100 ms[1]ごとに**ProcessingAck**オペレーション(OPType: 0xA)を用いたメッセージをイニシエータに返し、計算が進行中である事を通知します。メソッドによっては、ProcessingAckメッセージで中間結果を送信し、ファンクション実行の進捗状況を示す事もできます。対話型HMIの実装でアプリケーションが良好に応答する事を保証するには、ProcessingAckメッセージとエラーを適切に処理する事が重要です。

[1] ここに示したタイミングはMOST仕様が定義している標準値です。個々のメソッドに異なる値を定義する事も、システム インテグレータがシステム全体に異なる値を定義する事もできます。

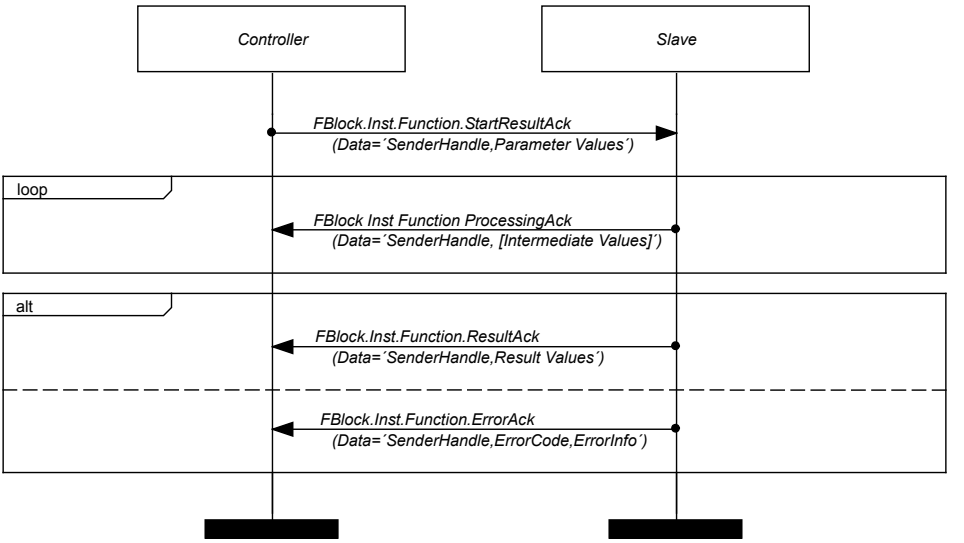


図4.7:StartResultAckオペレーションによるメソッドの開始

実行中のメソッドに対して**AbortAck**オペレーション(OPType: 0x7)を呼び出すと、現在の処理を中止できます。この場合、メソッドを開始したコントローラとAbortAckオペレーションの送信元(開始したコントローラと異なる場合)の両方に**Method Aborted**のエラーコードを持つエラーメッセージが送信されます。

ノーティフィケーション メカニズム

MOST仕様はノーティフィケーション メカニズム(図4.8参照)を定義しており、コントローラはGetメッセージを用いて常時ポーリングを実行しなくてもプロパティの現在値を取得できます。どのプロパティがノーティフィケーション メカニズムをサポートするかは、FBlockの定義時に暗黙的に決まります。

FBlockは、ノーティフィケーションをサポートした全てのプロパティについて**ノーティフィケーション マトリクス**と呼ばれるテーブルを管理しています。このテーブルには、各プロパティに対してノーティフィケーションを設定したデバイスのアドレスを入力します。これらプロパティのいずれか1つの値が変化すると、FBlockはこのテーブルに登録された全てのデバイスに対してそのプロパティの新しい値を含むステータス メッセージ (ノーティフィケーション)をただちに送信します。

コントローラがノーティフィケーションを設定または削除するには、目的のFBlockの**Notification**ファンクション(FktID: 0x401)を呼び出します。最初のパラメータ(Control)でオペレーションを示します(以下の箇条書き参照)。次のパラメータ(DeviceID)でノーティフィケーションを要求するデバイスのアドレスを示します。最後に、どのファンクションに対してノーティフィケーションを要求するかを、必要に応じてFktIDのリストで記述します。

- **SetAll**: FBlockのプロパティのうち、ノーティフィケーションをサポートしている全てのプロパティから指定したデバイスへノーティフィケーションが送信されるようにします。
- **SetFunction**: FBlockのプロパティのうち、後続のリストで指定した全てのプロパティからデバイスへノーティフィケーションが送信されるようにします。

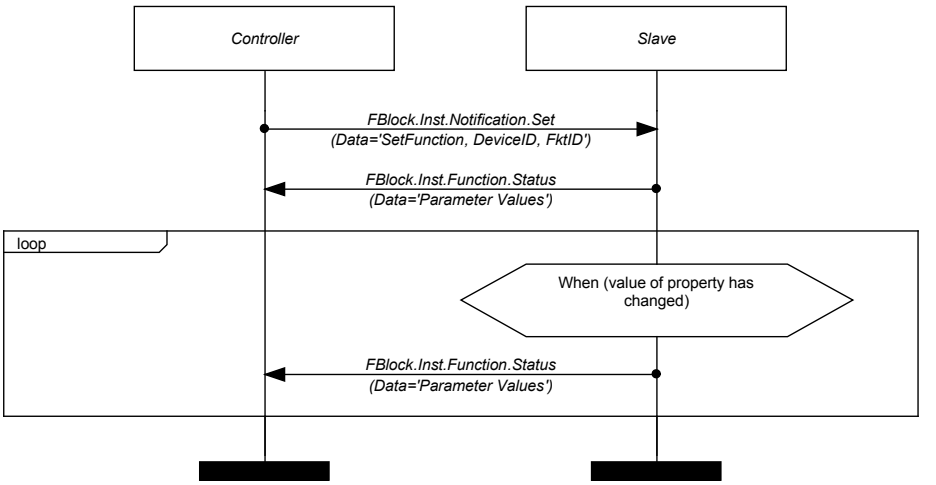


図4.8:ノーティフィケーション メカニズム

- **ClearAll**: 指定したデバイスに設定されたFBlockのプロパティのノーティフィケーションを全て削除します。
- **ClearFunction**: デバイスに設定されたFBlockのプロパティのうち、後続のリストで指定したプロパティのノーティフィケーションを削除します。

プロパティに対するノーティフィケーションが正しく設定されると、その確認としてFBlockは現在のプロパティ値をステータス メッセージで送信します。ノーティフィケーション メカニズムの使用例は、セクション4.4.4で紹介します。

エラー名	エラー コード	説明
FBlockIDが利用不可	0x01	構文エラー: FBlockが存在しない
InstIDが利用不可	0x02	構文エラー: インスタンスが存在しない
FktIDが利用不可	0x03	構文エラー: ファンクションが存在しない
OPTypeが利用不可	0x04	構文エラー: オペレーションが定義されていない
無効な長さ	0x05	データフィールドの長さがインター

エラー名	エラーコード	説明
		フェイス仕様の値と一致しない
パラメータが不正またはレンジ外	0x06	パラメータが不正、すなわちファンクションに指定された境界を超えていた
パラメータが利用不可	0x07	パラメータが(ファンクションに指定された境界を超えていないが)利用できなかった
予約済み	0x08	利用廃止
予約済み	0x09	利用廃止
予約済み	0x0A	利用廃止
デバイスの動作異常	0x0B	一般的なデバイスエラー
セグメンテーションエラー	0x0C	AMSによるメッセージのデセグメンテーション中のエラー
ファンクション固有	0x20	ファンクション固有のエラー
ビジー	0x40	ファンクションは利用可能だが、現時点で実行中
利用不可	0x41	ファンクションは原理上実装されているが、現時点で利用不可
処理エラー	0x42	メソッド実行中に発生したエラー全般
メソッドの中止	0x43	AbortAckオペレーションによりメソッドが中止された事を示す
システム固有	0xC0～0xEF	システム インテグレータが独自に定義したエラーコード
サプライヤ固有	0xF0～0xFE	デバイスメーカーが独自に定義したエラーコード

表4.6:MOSTプロトコルのエラーコード

エラー情報

プロパティかメソッドかを問わず、要求の実行中にエラーが発生すると、オペレーション タイプ**Error** (OPType: 0xF)または**ErrorAck** (OPType: 0x9)のメッセージが返されます。データフィールドの最初のバイトでエラーコード (パラメータErrorCode)が送信されます。エラーコードによっては、2バイト目にパラメータErrorInfoで追加の情報が報告される事があります。これについては、ここでは説明を割愛します。詳細は、MOST仕様[MOST 3.0]を参照してください。

MOSTシステムでは、エラー応答に関していくつかの条件を定めています。エラー情報が周期的に送信されるのを防ぐため、エラー メッセージを送信できるのはスレーブからコントローラのみです(セグメンテーション エラーの場合を除く)。

表4.6に示した一般的なエラー情報は、MOST仕様が定義しています。

4.3 MOSTファンクション ブロック

このセクションでは、FBlockの構造について詳しく説明し、その定義で守るべき重要な点について説明します。まずFBlockの全体的な構造を説明した後、FBlockの個々のファンクションを定義する際のガイドラインとなるファンクション クラスの概念を紹介します。

また、MOSTシステムで管理タスクを実行するのに必要なNetBlock等のシステムFBlockも、このセクションで紹介します。最後に、ストリーミング接続の管理とマルチメディア データの送信に必要なファンクションについて説明します。

4.3.1 ファンクション ブロックの構造

FBlock仕様は、どのタイプのFBlockにどのファンクションが属するかを定義したものです。FBlockのタイプはFBlockIDで決まります。

MOST Cooperationはいくつかの FBlockタイプを標準化しており、その中にはシステムタスクに使うものと、特定のアプリケーション機能を記述したものがあります。デバイスメーカーとシステム インテグレータはしばしば後者を拡張して独自ファンクションを組み込みます。独自のファンクションを定義する場合、または標準化されたFBlock仕様が存在しないファンクションを定義する場合、システム インテグレータまたはデバイスメーカーが新しいFBlock仕様全体を定義する必要があります。MOST Cooperationで標準化されたFBlockは、ファンクション ライブラリに集約されています(セクション4.4.1参照)。

MOST仕様では、全てのFBlockに以下のファンクションが必須とされています。

- **FktIDs** (FktID: 0x000): Fblockが実装している全ファンクションのリストは、FktIDsプロパティで問い合わせる事ができます。FktIDsのリストをそのまま送ると長くなるため、実装されているファンクションと実装されていないファンクションの連続する領域が切り替わる位置のFktIDのみをリストに含める符号化でリストを圧縮します(詳細は[MOST 3.0]参照)。
- **FBlockInfo** (FktID: 0x011): この必須ファンクションは、システム内に複数のインスタンスが存在するFBlock(例: リアシート エンターテイメント システムのディスプレイ ユニットLとR)を区別する情報として、FBlockの名前とバージョン、実装されているFBlockが準拠しているMOST仕様のバージョン、FBlockを使うシステムのシステム インテグレータ、システム インテグレータで割り当てられた文字列(FBlockType)を返します。
- **Notification** (FktID: 0x401): Notificationプロパティは全てのFBlockへの実装が必要です。FBlockの1つまたは複数のプロパティに対してノーティフィケーションを設定する際に使います。このプロパティを使って、デバイスはノーティフィケーションを設定または削除できます(セクション4.2.2参照)。
- **NotificationCheck** (FktID: 0x402): どのノーティフィケーションが現在設定されているかを確認するために使う任意のプロパティです。

これらのファンクションに加え、「調整」レンジのファンクションで「条件付き」と記載されているものも、GeneralFBlock仕様[GenFB]が定義している個々の前提条件に応じて全てのFBlockに実装する必要があります。さらに、該当するFBlock仕様で「必須」と定義されている全てのファンクションを実装する必要があります。「任意」と表記されているファンクションは、実装を省略できます。

MOSTシステムの定義と導入では、システムを構成するデバイス、および各デバイスで実装されるFBlockをシステム インテグレータが詳細に指定します。また、個々のFBlockでどのファンクションを実装するかも通常はシステム インテグレータが詳細に指定します。

4.3.2 ファンクション クラス

ファンクション クラスはファンクションのデータ型とオペレーションに関するガイドラインとなるもので、これによってFBlockの指定を簡略化します。FBlock仕様では、各ファンクションに何らかのファンクション クラスを割り当てます。このファンクション クラスによって、文字列の最大長さ等、ファンクションのパラメータ設定が決まります。また、ファンクション クラスはそのファンクションに対してどのオペレーションを実行できるかを定義します。さらに、ファンクション クラスはRecordやArray等のより複雑な合成データ型の基盤になります。

ファンクション クラスにより、ファンクションに対する統一された構造化モデリングが可能です。また、ファンクション クラスはモデリングとコード生成を行うツールの使用をサポートし、テスト自動化を簡略化します。

下記のファンクション クラスのいずれにも割り当てできないファンクションは、**Unclassified Property**ファンクション クラス (プロパティの場合)または**Unclassified Method**ファンクション クラス (メソッドの場合)に割り当てます。

MOST仕様では、1つのパラメータでステータスを記述できるプロパティに関して以下のシンプルなファンクション クラスを定義しています。

- **Switch**: Switchファンクション クラスはBoolean型パラメータをカプセル化したもので、2つの状態を持つ「スイッチ」を記述します。これらは設定と読み出しが可能です。
- **Number**: Numberファンクション クラスは、符号付きまたは符号なしの各種数値型に使います。このファンクション クラスには、パラメータ型、最大値と最小値、小数点の位置を定義する指数、Incrementオペレーションと Decrementオペレーションの増分値と減分値を設定できます。
- **Text**: Textファンクション クラスは、String型パラメータを用いて文字列を定義します。その設定により、文字列の最大の長さが決まります。
- **Enumeration**: Enumerationファンクション クラスは、Unsigned Byte型を用いてEnum型を記述します。このファンクション クラスは、Enum型に何個の要素があるか、これら要素がどのような値をとるかを文字列での記述を含めて

定義します。例えば、FBlock AuxInのDeckStatusファンクションは「Play」が0x00、「Stop」が0x01、「Pause」が0x02の値を持ちます。

- **BoolField**: BoolFieldファンクション クラスは、合計8、16、32ビットの長さのビットシーケンスをそれぞれUnsigned Byte、Unsigned Word、Unsigned Long型で定義します。このビット シーケンスを、連続する複数ビットから成るアドレス指定可能な要素に構造化する事もできます。余りの上位ビットは使わず、空のままです。
- **BitSet**: BitField型のパラメータでステータスを記述できるプロパティは、同じ長さのBitSetファンクション クラスに割り当てます。BitField型パラメータが定義しているように、前半にマスク、後半にデータを格納します。
- **Container**: 上記以外の非構造化データには、Containerファンクション クラスを使います。このファンクション クラスは、Stream型、Classified Stream型、Short Stream型のパラメータを使います。ファンクション クラスの設定で、最大長さを定義します。

さらに、複数のパラメータから成る複雑なデータ構造を表すファンクション クラスがあります。これらのファンクション クラスでは、どの要素にアクセスするかを先頭のパラメータ**Position**で示します。Positionは、各1バイトの**PosX**と**PosY**の2つから成ります。PosXは外部データ構造における位置を示し、内部データ構造がある場合はPosYでその位置を示します。2次元配列の場合、PosXが行を示し、PosYが列を示します。PosXが0でなくPosYが0の場合、配列の1行全体を使います。PosXとPosYが両方0の場合、配列全体をアドレス指定します。

プロパティには、以下の構造化ファンクション クラスがあります。

- **Record**: Recordはファンクション クラスの異なる任意の数の要素のシーケンスから成り、個々の要素にはパラメータPosXでアクセスできます。Array等の構造化ファンクション クラスも要素として使えます(この場合、PosYを使って配列の要素をアドレス指定します)。
- **Array:** Arrayは、最大256個の同じ型の要素から成るシーケンスを記述し、個々の要素にはパラメータPosXでアクセスできます。ファンクション クラスArrayの設定は、個々の要素のファンクション クラスとArrayの最大長さを示します。
- **DynamicArray**: DynamicArrayはArrayから派生したもので、PosXではなく16ビットのタグを用いて個々の要素にアクセスします。このため、より長い配列が可能です。また、要素を挿入および削除するファンクションを定義します。要素のタグは、位置のように挿入または削除によって変化する事がないため、DynamicArrayは動的なデータ構造に適しています。
- **LongArray**: LongArrayも16ビットのタグを用いて個々の要素にアクセスします。さらに、LongArrayではウィンドウ メカニズムが定義されており、コントローラは配列の特定のセクション (ウィンドウ)を定義できます。その後、このセクション内で変化があった場合のみコントローラに通知されます。これ

により、携帯電話で電話帳を表示する場合等、画面サイズに応じて現在表示しているセクションのみを更新する事ができ、HMIのリスト表示を効率よく実装できます。

- **Map**: Mapファンクション クラスはDynamicArrayに似ています。ただし、ノーティフィケーションで1行または複数行の挿入または削除を送信できる点が異なります。従って、Mapファンクション クラスに対する更新は非常に効率よく送信できます。
- **Sequence Property**: Sequence Propertyファンクション クラスはシンプルなファンクション クラスで、個々の要素のシーケンスに対して設定と読み出しが可能です。多次元データ型と個々の要素への個別アクセスはサポートしません。

メソッドに定義されているファンクション クラスは、Unclassified Method以外には以下の2つのみです。

- **Trigger**: Triggerファンクション クラスはパラメータを持たないファンクション クラスで、特定のアクションをトリガします。
- **Sequence Method**: Sequence Methodファンクション クラスは、シンプルなパラメータのリストを持つメソッド用のファンクション クラスです。

4.3.3 ファンクション インターフェイス

MOST仕様3.0 [MOST 3.0]からは、ファンクション インターフェイスのメカニズムは任意です。通常、ファンクション クラスの概念が必要となるのはファンクションを定義する際のみです。システムの動作中にファンクション インターフェイス(すなわちファンクション クラスとその設定)に関する情報を取得する事はほとんどありません。

MOST仕様では、ファンクションが持つファンクション インターフェイスを要求する任意のオペレーションとして**GetInterface** (OPType: 0x5)を定義しています。その結果は、**Interface**オペレーション(OPType: 0xE)で送信します。情報を送信する際の符号化の詳細は、MOST仕様を参照してください。

4.3.4 システムFBlock

このセクションでは、システム管理用にMOST仕様で必須とされているFBlocksの概要を説明します。先に述べた通り、FBlockにはMOSTシステムの全てのデバイスが実装するものと、システム内の1つのデバイスのみが提供するものがあります。

NetBlock (FBlockID:0x01)

NetBlockは、個々のMOSTデバイスに関するシステム管理タスクを一元的に実行します。NetBlockはMOSTシステム内の全てのデバイスに実装する必要があります。

以下に、NetBlockで特に重要なファンクションについて説明します。NetBlockの全てのインターフェイスは[NetBlock]で定義されています。

- **FBlockIDs** (FktID: 0x000):FBlockIDsファンクションは、そのデバイスが提供する全てのアクティブなアプリケーションFBlockについて、FBlockIDとInstIDをペアにしたリストを返します(NetBlock自身とEnhancedTestability等のシステムFBlockは含みません)。このリストには、正しく初期化され、MOSTシステムのコントローラに対して現在その機能を提供できるFBlockのみが含まれます。また、FBlockIDsファンクションを使ってFBlockのInstIDを設定できます。このファンクションは、主にNetworkMasterによるシステムスキャン中に使います(NetworkMasterのセクション参照)。
- **DeviceInfo** (FktID: 0x001): メーカー名やシリアル番号等、デバイスを特徴付けるパラメータを問い合わせます。
- アドレッシング ファンクション: デバイスのアドレス情報を提供するファンクションとして、ノード位置アドレスを提供する**NodePositionAddress** (FktID: 0x002)、論理アドレスを提供する**NodeAddress** (FktID: 0x003)、グループアドレスを提供する**GroupAddress** (FktID: 0x004)があります。
- **ShutDown** (FktID: 0x006): このファンクションは、MOSTシステムのシャットダウン手順を制御します(下記参照)。
- **ShutDownReason** (FktID: 0x009): MOSTシステムの再起動後、唯一のコントローラは各デバイスに対してシャットダウンの原因を問い合わせる事ができます。障害(突発的信号OFFまたはクリティカル アンロック)に起因するシャットダウンの場合、障害を検出した全てのデバイスがこのファンクションでそれぞれの情報を返します。
- **ImplFBlockIDs** (FktID: 0x012): このファンクション(任意実装)は、デバイスに実装されている全てのFBlockのFBlockIDをリストで返します。FBlockIDsファンクションとは異なり、現在利用できないFBlockも問い合わせできます。
- **RBDResult** (FktID: 0x405): リングブレーク診断の結果をこのファンクションで送信します。詳細は第8章を参照してください。
- **Boundary** (FktID: 0xA03): このファンクションがあるのは、TimingMasterとして動作するデバイスのみです。このファンクションにより、データリンク層での帯域幅の割り当てを管理します(読み出しと設定)。このメカニズムはセクション5.1で説明します。

FBlock EnhancedTestability (FBlockID: 0x0F)

FBlock EnhancedTestability (ET)は、MOSTデバイスのコア コンプライアンス([Core 3.0]参照)に必要なテストを実行するのに必要で、全てのデバイスに実装する必要があります。これらのテストでは、アドレスの割り当て等、デバイスの基本的なシステム ファンクションをチェックします。FBlock ETのファンクションの詳細は[FBET]に記述されており、本書では取り上げません。

FBlock ETには、テスト自動化のためのファンクションがあります。例えば、安定状態から開始できるようにデバイスをリセットするResetファンクション(FktID: 0x221)があります。FBlock ETは、デバイスの自発的な復帰をシミュレートするAutoWakeupファンクション(FktID: 0x201)のように、他の手段では自動的にトリガされない機能を実行するファンクションも含みます。

FBlock NetworkMaster (FBlockID: 0x02)

FBlock NetworkMasterは、MOSTシステム全体のシステム構成を一元的に管理します。このため、各MOSTシステムに必須で、指定されたデバイスに実装する必要があります。多くの場合、MOSTシステムにおける全てのマスタ機能(NetworkMaster、TimingMaster、ConnectionMaster、PowerMaster)を1つのデバイスが受け持ちます。しかし、NetworkMasterとTimingMaster以外は同じデバイスが受け持つ必要はありません。

NetworkMasterはシステム構成、すなわちMOSTシステムの現在のシステムステータスを管理します。現在のシステム構成(システムに存在するデバイスとそのFBlock)をNetworkMasterが把握している場合システムステートは**OK**で、それ以外の場合は**NotOK**です。起動直後、ネットワーク変更イベントによって無効な構成(例: 無効なデバイスアドレス)が発生した場合、システムステートはNotOKです。

起動中またはネットワーク変更イベント後、NetworkMasterはシステム構成を確認するため、FBlockIDsファンクションを用いて全てのデバイスのFBlockを問い合わせます(システムスキャン)。この問い合わせは、物理デバイスアドレスに対して送ります。

システムステータスがNotOKの場合に有効な構成を確認すると、システムステートがOKに変化し、NetworkMasterはMOSTシステム内のデバイスに対してConfigurationファンクションをブロードキャスト送信します。このファンクションのパラメータConfigurationControlでシステムステート(この場合はOK)を通知します。図4.9に、システムスキャンのオペレーション シーケンスを示します。実際には、複数のFBlockに対して並列に問い合わせを実行できます。

最初のシステムスキャンの後、デバイスがFBlockを追加するか無効にすると、そのデバイスはFBlockIDsファンクションを用いたステータス メッセージで新しいFBlockのリストを送信し、NetworkMasterに通知できます。システムステートがOKの状態でシステムスキャンを実行した場合、またはデバイスからFBlockの変更についてNetworkMasterに通知があった場合、NetworkMasterはFBlockの追加または無

効化に関する情報をConfigurationファンクションを用いてMOSTシステム全体に通知します。FBlockを追加した場合のConfigurationControlのパラメータ値は**NewExt**で、FBlockを無効化した場合のパラメータ値は**Invalid**です。パラメータにDeltaFBlockIDListを追加し、各FBlockのFBlockIDとInstID(およびNewExtの場合は論理アドレス)のリストを送信します。

システムスキャン中に複数のFBlockのファンクション アドレス間で競合が発生した場合(すなわち、あるFBlockの複数のインスタンスが同じInstIDを持つ場合)、NetworkMasterは前回報告されたInstIDをリセットして競合を解決しようとします。無効なデバイスアドレス等、InstIDのリセットでは解決できないエラーが発生した場合、NetworkMasterはシステムステートをNotOKに設定し、新規にシステム起動をトリガします。

MOSTシステムの現在の構成に関する情報は、NetworkMasterが中央のデータベース (セントラル レジストリ)に格納します。セントラル レジストリには、システム内の全てのデバイスの論理アドレス、および実装されているFBlockとそのInstIDのリストを格納します。システム内のNetworkMaster以外のデバイスは、CentralRegistryファンクションを用いて現在の構成を問い合わせ、通信相手のファンクション アドレスに対応するデバイスアドレスを確認できます。高次のネットワーク サービスを使っている場合、このメカニズムは自動的に実行されます。

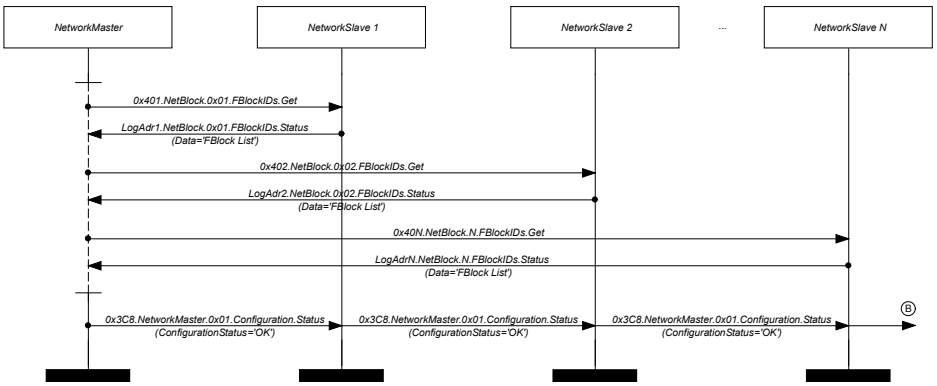


図4.9:NetworkMasterによるシステムスキャンのオペレーション シーケンス

FBlock NetworkMasterは、以下のファンクションを実装します。

- **Configuration** (FktID: 0xA00): NetworkMasterは、Configurationファンクションを用いてMOSTシステムのシステムステートに関する情報をブロードキャストで一斉送信します(ConfigurationControlのパラメータ値はOKまたはNotOK)。また、新たに追加されたFBlockまたは無効になったFBlockをアナウンスします(ConfigurationControlのパラメータ値はNewExtまたはInvalid)。

- **CentralRegistry** (FktID: 0xA01): 現在のセントラル レジストリに対する問い合わせを有効にします。セントラル レジストリ全体を問い合わせるか、特定のFBlockの全てのインスタンスを取得するか、特定のFBlockの特定のインスタンスを問い合わせる事ができます。いずれの場合も、CentralRegistryファンクションは、論理デバイスアドレス、FBlockID、InstIDの3つの要素をリストにして返します。
- **SystemAvail** (FktID: 0xA10): このメカニズム(任意実装)は、システムが利用可能かどうかに関する情報を提供します。この場合、NetworkMasterはシステム内の全てのデバイスのImplFBlockIDsファンクションを問い合わせます。パラメータのDeviceAvailとFBlockAvailを用いて、スレーブデバイスは全てのデバイスがNetworkMasterに正しく応答したかどうか、実装されている全てのFBlockが現在利用可能かどうかを判断します。

FBlock ConnectionMaster (FBlockID: 0x03)

FBlock ConnectionMasterはMOSTシステムにおけるストリーミング接続の管理に使います。詳細は、次のセクションで説明します。

FBlock PowerMaster (FBlockID: 0x04)

PowerMaster自身には、FBlock仕様がありません。FBlockID 0x04は、PowerMasterの機能を持つデバイスを示すためだけに使います。

PowerMasterは、MOSTシステムのシャットダウンを調整します。シャットダウンには、通常のシャットダウン(例: 自動車のエンジンを停止した場合)と非常時のシャットダウン(例: 温度が異常に上昇した場合)があります。

PowerMasterは、NetBlockのShutDownファンクションを使ってMOSTシステム内の他のデバイスをシャットダウンします。まず、PowerMasterはShutdown.StartAck (SenderHandle, Query)メッセージをブロードキャスト送信してシャットダウンをアナウンスします。シャットダウンの準備ができていないデバイスは、ShutDown.ResultAck (SenderHandle, Suspend)メッセージを送信してシャットダウンを遅らせる事ができます。システム アーキテクチャによっては、例えば通話中のシャットダウンを抑止する事ができます。

PowerMasterは一定間隔でシャットダウンのアナウンスを繰り返します。ShutDown.ResultAck (SenderHandle, Suspend)メッセージの応答がなくなると、PowerMasterはShudown.StartAck (SenderHandle, Execute)メッセージをブロードキャスト送信して実際のシャットダウン手順を開始します。

デバイスが高温を検出すると、ShutDown.ResultAck (SenderHandle, TemperatureShutdown)メッセージをブロードキャスト送信してシステムのシャットダウンを要求します。すると、PowerMasterが前述の通常シャットダウン手順を実行します。

4.3.5 ストリーミング接続の管理

MOSTシステムはマルチメディア データの転送が中心的なアプリケーションであるため、オーディオデータとビデオデータを送信するストリーミング接続の管理が、MOSTシステムの中でも特に重要です。このセクションでは、アプリケーション層におけるストリーミング接続管理用のメカニズムとインターフェイスについて説明します。

概要

MOST仕様では、マルチメディア データ (オーディオとビデオ)のソースとシンクは、概念上FBlockに関連付けられています。各FBlockには1つまたは複数のソースがあり、それと同時に1つまたは複数のシンクがあります。例えば電話のFBlockには、車載スピーカからの音声出力に対するソースと、車載マイクからの音声入力に対するシンクがあります。

FBlockの各ソースは、Unsigned Byte型の**SourceNr**タグで一意にアドレス指定されます。同様に、FBlockのシンクは**SinkNr**タグで一意に識別されます。

個々のFBlockにどのようなソースとシンクが実装されているかは、StreamDataInfoファンクションで確認できます。

- **StreamDataInfo** (FktID: 0x116): FBlockのソース番号のリストとシンク番号のリストを返します。

1つのソースに対して、1つのストリーミング接続を作成できます。ストリーミング接続は、接続ラベルと帯域幅で特徴付けられます。接続ラベルは一意の識別子で、Unsigned Word型のパラメータ**ConnectionLabel**で記述します。帯域幅はこの接続で1 MOSTフレーム当たり何バイトを送信できるかを定義したもので、Unsigned Word型のパラメータ**BlockWidth**で記述します。既に存在する1つの接続に1つまたは複数のシンクを接続できます。

図4.10に、あるFBlockのソースと別のFBlockのシンクとの間でストリーミング接続を確立する際の手順を示します。その後、ソースのファンクションとシンクのファンクションをそれぞれ説明します。

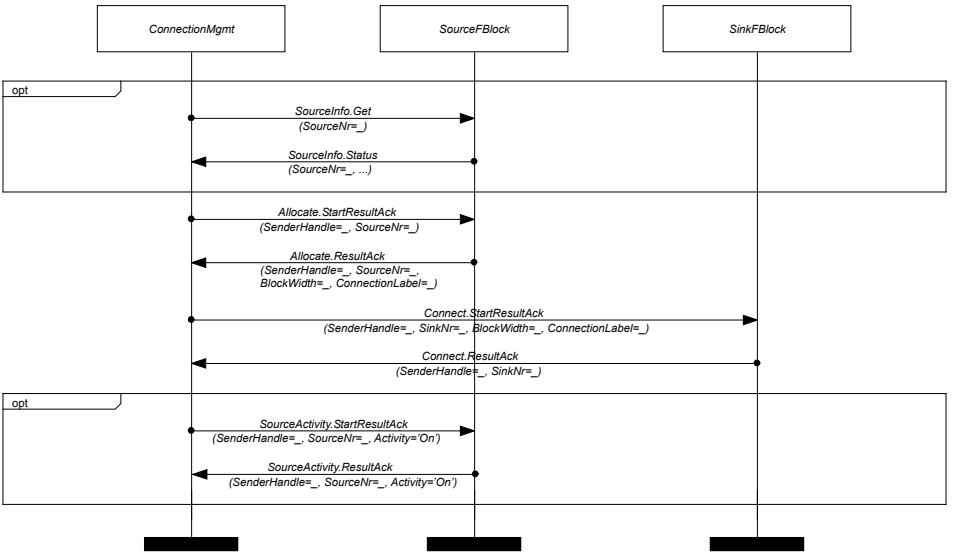


図4.10:ストリーミング接続の確立

ソースのファンクション

MOST仕様では、マルチメディア データのソースとなるFBlockで以下のファンクションが必須です。これらのファンクションはGeneralFBlock仕様[GenFB]が定義しており、ソース機能を持つ全てのFBlockに実装されます。

- **SourceInfo** (FktID: 0x100): SourceInfoプロパティは、ソースの内容タイプ(例: PCM、SPDF、MPEG2)等、指定されたソースのプロパティに関する情報を返します。
- **SourceName** (FktID: 0x104): 指定されたソースを識別する文字列を返します。
- **SourceActivity** (FktID: 0x103): ソースをアクティブまたは非アクティブに設定するファンクション(任意実装)です。このファンクションのパラメータに応じて、ソースは接続されたストリーミング接続へデータを送信します。
- **Allocate** (FktID: 0x101): FBlockは、Allocateファンクションによる要求を受けるとストリーミング接続を作成し、指定したソースをそのストリーミング接続に接続します。1つのストリーミング接続に複数のソースを同時に接続する事はできません。
- **AllocateExt** (FktID: 0x108): AllocateファンクションにパラメータClkSrcLabelを追加したものです。このパラメータは、アイソクロナス接続に必要なクロック源の接続ラベルを示します。
- **DeAllocate** (FktID: 0x102): 指定したソースをストリーミング接続から切断し、TimingMasterでの割り当てを解除します。

シンクのファンクション

マルチメディア データのシンク機能を持つFBlockは、以下のファンクションを定義します。

- **SinkInfo** (0x110): SourceInfoと同様に、指定したシンクに関する情報を格納したプロパティです。
- **SinkName** (0x114): シンクを識別する文字列です。
- **Mute** (0x113): 接続のミュート ステータスを切り換えるファンクションです。また、エラー処理のためにFBlockで自動的にステータスが設定された場合等、このファンクションを使って対応するシンクの現在のステータスを問い合わせる事もできます。システムによっては、より複雑なメカニズムを使ってこの機能を実行する事があるため、Muteファンクションの実装は任意です。
- **Connect** (0x111): FBlockの指定したシンクを、対応するストリーミング接続に接続します。1つのストリーミング接続に、同時に複数のシンクを接続できます。
- **DisConnect** (0x112): 指定したシンクをストリーミング接続から切断します。

FBlock ConnectionMaster (FBlockID: 0x03)

MOST仕様では、ストリーミング接続を一元的に管理するFBlockとしてConnectionMasterを定義しています。ConnectionMasterは、要求に応じて特定のソースとシンク間のストリーミング接続を確立または削除します。また、ConnectionMasterはシステム内に存在する全ての接続に関する情報を管理します。

この目的のため、FBlock ConnectionMasterは以下のファンクションを実装しています([FBCM]参照)。

- **BuildConnection** (FktID: 0x200): FBlockIDとInstID、およびSourceNrとSinkNrで指定されたソースとシンクの間で接続を確立します。必要なら、ストリーミング接続を予約できます。基本的に、ConnectionMasterは図4.9に示した手順を実行します。
- **RemoveConnection** (FktID: 0x201): 指定したソースとシンクの間で確立されている接続を削除します。対応するストリーミング接続は必要に応じて解放されます。
- **ConnectionTable** (FktID: 0x400): システム内で現在確立されている全てのストリーミング接続のリストを保持します。
- **AvailableChannels** (FktID: 0x401): 1つのMOSTフレームでストリーミング接続に利用できる残りの帯域幅をバイト単位で示します。
- **MoveBoundary** (FktID: 0x402): MOSTシステムで、ストリーミング接続に利用可能な帯域幅とパケット接続に利用可能な帯域幅を決定する境界を調整するファンクションです。

- **BoundaryChange** (FktID: 0x403): デバイスがこのプロパティにノーティフィケーションを設定しておくと、境界が変化した時に通知を受け取る事ができます。

インフォテインメント システムの中心的なインスタンスが、アプリケーションの優先度等の条件も考慮してシステム全体でストリーミング接続を管理する場合、このインスタンスがConnectionMasterの役割も受け持つのが一般的です。通常、このような場合はFBlock ConnectionMasterをデバッグ用にのみ実装します。

4.4 標準化されたアプリケーション インターフェイス

システムFBlock以外に、MOST Cooperationは車載インフォテインメント システムの代表的なアプリケーションのインターフェイスに関するFBlockを標準化しています。

このセクションでは、仕様が定義しているアプリケーション インターフェイスの概要を説明します。一部のインターフェイスについては、例を挙げながらさらに詳しく説明します。最後に、携帯型MP3プレーヤへのアクセスを提供するAuxIn FBlockを例に、MOSTシステムにおけるアプリケーションFBlockの使い方を詳しく説明します。

4.4.1 ファンクション ライブラリ

ファンクション カタログは、MOST仕様が定義している全てのFBlockの最新仕様を含みます。この中には、セクション4.3.4で説明したシステムFBlockだけでなく、アプリケーション用に標準化したインターフェイスも含みます。表4.7に、最新のファンクション カタログに含まれる全てのFBlockとそのバージョン番号を示します。

種類	FBlock名	FBlockID	バージョン[1]	説明
	GeneralFBlock	-	3.0.2	調整された共通ファンクション
	GeneralPlayer	-	2.5.1	再生デバイスに関する調整されたファンクション
管理	NetBlock	0x01	3.0.1	全てのデバイスに実装されるシステムFBlock
	NetworkMaster	0x02	3.0.1	NetworkMaster
	ConnectionMaster	0x03	3.0.1	ConnectionMaster

[1] 2010年第2四半期現在

種類	FBlock名	FBlockID	バージョン[1]	説明
	Vehicle	0x05	1.6	車両関連データに関するインターフェイス
	Diagnosis	0x06	3.0	診断機能へのアクセス
	DebugMessages	0x07	1.0.1	デバッグ出力を制御するためのファンクション
	Tool	0x0E	1.0	テスト用のFBlock
	EnhancedTestability	0x0F	3.0.1	コンプライアンステストに必要
オーディオ	AudioAmplifier	0x22	2.4.2	アンプ
	AuxIn	0x24	3.5	民生用携帯電子機器用のインターフェイス
	MicrophoneInput	0x26	2.3.1	マイク用インターフェイス
ドライブ	AudioTapePlayer	0x30	2.3.1	テープデッキ
	AudioDiskPlayer	0x31	2.4	CDプレーヤまたはCDチェンジャ
	DVDVideoPlayer	0x34	3.0	DVDプレーヤ
レシーバ	AmFmTuner	0x40	2.4.2	ラジオチューナ
	TMCTuner	0x41	2.3.1	交通情報メッセージ(TMC)用チューナ
	TVTuner	0x42	2.3.2	テレビチューナ
	DABTuner	0x43	3.0、4.0	DABチューナ
	SatRadio	0x44	2.4	衛星デジタルオーディオ ラジオ サービス(SDARS)用チューナ
通信	Telephone	0x50	2.3.2	携帯電話接続用インターフェイス
	GeneralPhonebook	0x51	2.3.2	電話帳
	NavigationSystem	0x52	1.11	ナビゲーションユニット

[1] MOST Cooperationによる非公式リリースのみ

種類	FBlock名	FBlockID	バージョン[1]	説明
	GraphicDisplay	0x60	2.3	ディスプレイ ユニット
	UniqueFunctions	0xFF	2.3	「独特な」領域のファンクションを集めたもの

表4.7:MOST Cooperationで標準化されたFBlock

MOST仕様[MOST 3.0]の表2-2は、ファンクション カタログに記載がないFBlockIDを定義しています(例: FBlockID 0x54のBluetooth)。これらは公式に発表されていない仕様であるか、過去または将来の定義に対するプレースホルダのどちらかです。

4.4.2 共通のファンクション(GeneralFBlock)

「調整」領域のファンクション(0x000～0x1FF)は、全てのFBlockに同じ方法で実装する必要があります。これらのファンクションはMOST Cooperationが管理しており、GeneralFBlock仕様(最新版はバージョン3.0) [GenFB]で定義しています。

GeneralFBlockには、FBlockに必須のファンクション(FktIDs、Notification、NotificationCheck)以外に、配列に要素を挿入する等、データ構造を管理するためのファンクションを定義しています(セクション4.3.2参照)。ストリーミング接続を管理するためのファンクションも、ここで定義しています(セクション4.3.5参照)。

GeneralFBlockは、テンプレートとしてのみ使います。FBlockを定義する際は、現在のバージョンのGeneralFBlockのファンクションから必要なものをFBlock仕様の一部として採用します(すなわち、GeneralFBlockの新しいバージョンで内容が変更されても影響を受けません)。しかし、FBlockに「調整」領域のファンクションを実装した場合、このファンクションはGeneralFBlockの仕様から逸脱しないようにする必要があります。

4.4.3 アプリケーション インターフェイス

MOSTファンクション カタログで標準化されたアプリケーション インターフェイスへの理解を深めるため、ここでは車載インフォテインメント システム分野における主要なアプリケーションのインターフェイスをいくつか例にとって詳しく説明します。紙面の都合上、ファンクション カタログの全てのFBlockについては詳細な説明を割愛します。

CDプレーヤ、CDチェンジャ(AudioDiskPlayer、FBlockID: 0x31)

FBlock AudioDiskPlayer [FBADiskPl 2.4]は、シンプルなCDプレーヤまたはCDチェンジャを定義したものです。MOST Cooperationは、CDプレーヤ以外にもカセットプレーヤ等の各種再生デバイス用のFBlockを定義しており、汎用的なファンクションはFBlock GeneralPlayer [FBGenPl 2.5]にまとめています。AudioDiskPlayerは、GeneralPlayerの一部機能を実装したものです。GeneralPlayerは、DVD再生等のファンクションも含みます。

FBlock AudioDiskPlayerは、以下の重要なファンクション グループを定義しています。

- オーディオソースにアクセスできるように、セクション4.3.5で説明したファンクションが実装されています。このソースから、現在再生中のトラックのオーディオデータをアンプに出力する等して、MOSTシステムで利用できます。
- デバイスの現在のステータスはDeckStatus等のプロパティに格納されており、対応するオペレーションで読み出しまたは設定できます。DeckStatusを使うと、再生、一時停止、停止等、再生デバイスの一般的なアクションを実行できます。この他、ランダム、スキャン、リピート等のデバイスモードを設定するファンクションがあります。
- AudioDiskInfo等のファンクションは、CDのタイプ (オーディオCD、ビデオCD、CD-ROM)とフォーマット(例: CDDA、ISO9660)等、挿入された1枚または複数枚のCDに関する情報を提供します。CDチェンジャの場合、マガジンのステータス(MagazineStatus)とアクティブCD (ActiveDisk)を問い合わせできます。
- この他、再生動作に関する情報を提供するファンクションとして現在のトラック番号を知らせるTrackPosition、再生時間を知らせるTimePosition等があります。コントローラは通常、これらのファンクションに対してノーティフィケーションを要求します。これにより、例えばユーザ インターフェイスにトラック再生の残り時間を表示したり、プレーヤが次のトラックに移動した時に通知を受けてこのトラックに関するメタ情報を更新したりできます。再生動作またはTrackPositionを利用した現在のトラックの選択は、これらのファンクションによってある程度影響を受ける事があります。

アンプ(AudioAmplifier、FBlockID: 0x22)

オーディオをデータを受信、処理、増幅して自動車のスピーカから出力するオーディオアンプ機能を持ったデバイスは、FBlock AudioAmplifier [FBAudioA 2.4]を実装します。

- オーディオデータは、1つまたは複数のシンクで受信します。FBlockには、シンクに必要なファンクション (セクション4.3.5参照)を実装します。

- オーディオ信号の出力に対する代表的な設定(例: ボリューム、低音、フェーダ、高音)は、対応するファンクションで制御します。
- この他、イコライザ機能の有効化や設定(EqualizerOnOff、EqualizerSettings)等、より専門的なアンプ機能の調整を実行するFBlockもあります。

ラジオチューナ(AmFmTuner、FBlockID: 0x40)

MOST仕様では、デジタルラジオ規格DABに対応したFBlock DABTunerやデジタル衛星ラジオに対応したFBlock SDARS等、各種のラジオチューナ向けのFBlockを定義しています。ここでは、その一例としてAM/FMチューナに対応したFBlock AmFmTuner [FBAmFmT 2.4]について説明します。

FBlock AmFmTunerの仕様では、主に以下のファンクションが定義されています。

- ATFrequency、ATWaveband、ATStationInfoの各ファンクションは、現在設定されている周波数、受信バンド、受信放送局(例: 放送局名または番組タイプ)に関する情報を提供します。
- 放送局スキャンは、ATSeekファンクションで開始します。このファンクションはいくつかの設定が可能です(例: スキャン方向の上下)。
- FBlock AmFmTunerは、放送局メモリを利用した複数リストの管理をサポートしていますが、シンプルなチューナでは1つの放送局リストだけ実装すれば十分です。ATPresetSaveファンクションを使うと、現在の周波数リストを保存できます。ATPresetList1ファンクションとATPresetList2ファンクションを使うと詳細な放送局リストにアクセスでき、ATPresetShortList1 ～ ATPresetShortList8ファンクションを使うと簡単な情報のみの放送局リストにアクセスできます。
- この他、道路交通情報が利用できる場合のAM/FMチューナの挙動を制御するファンクションがあります。TAInfoファンクションは、現在の道路交通情報番組の名前を示します。動的な道路交通情報は、TAMessageファンクションで示されます。中断するには、TAEscapeファンクションを使います。TPSwitchファンクションは、道路交通情報の受信機能のONとOFFを切り換えます。
- さらに、FBlock AmFmTunerには各種方法でチューナの挙動を制御するファンクションがあります。例えば、RDSSwitchファンクションはRDS信号の使用のONとOFFを切り換えます。

モバイル機器の接続(AuxIn、FBlockID: 0x24)

FBlock AuxInもFBlock GeneralPlayerをベースにしており([FBAuxIn 3.5]参照)、携帯型MP3プレーヤと音楽データを格納したその他のストレージ デバイスをMOSTシステムに接続するためのゲートウェイとして機能するMOSTデバイスが実装します。

FBlock AuxInは、Apple社のiPodのように音楽データを保存して自分自身で管理できるモバイル機器、およびUSBメモリのように汎用ファイルシステムにデータを保

存するだけのモバイル機器をサポートしています。FBlock AuxInの仕様では、前者をデータベース デバイス、後者をマスストレージ デバイスと呼びます。また、アナログ オーディオ入力経由でデバイスを接続する事もできます。

FBlock AudioDiskPlayerのセクションで説明したGeneralPlayerのファンクションに加え、以下の機能が拡張されています。

- トラック数の増加またはアクセスモードの多様化に対応するため、GeneralPlayerのファンクションを改変したファンクションがいくつか存在します。このようなファンクションとしては、TrackPositionに由来するAuxTrackPosition、TrackInformationに由来するAuxTrackInformation、TimePositionに由来するAuxTimeInformationがあります。
- 接続した機器についての情報を提供するファンクションが追加されています。接続した機器のデバイスクラスとデバイスタイプは、AuxDeviceInfoファンクションで確認できます。DeviceBrowsingCapabilitiesファンクションは、あるアーティストの全てのトラック、あるいは特定のジャンルの全てのトラックを問い合わせる等、データベース デバイスに対して問い合わせが可能な音楽ファイルの属性情報を通知します。
- デバイスに格納された音楽トラックに対するメタデータの問い合わせは非常に大規模な処理になるため、FBlock AuxInはデータベースのクエリに匹敵するクエリ機能をサポートしたSelectAudioListInfoファンクションを定義しています。どのトラックを問い合わせの対象とするか、そしてトラックに関するどの情報(例: 曲名、アーティスト名、アルバム名)を返すかを、2つのフィルタで指定します。さらに、並び替えの順序およびどのセクションのデータを送信するかを指定して、ディスプレイ上での表示を最適化できます。SelectCurrentAudioListInfo は SelectAudioListInfo に似ています。SelectAudioListInfoは接続した機器に格納されている全ての音楽ファイルを対象とするのに対し、SelectCurrentAudioListInfoは現在選択されているトラックリストのみを対象にします。
- SelectAudioListFilterファンクションでは、SelectAudioListInfoで問い合わせに使うのと同様のフィルタを設定して、現在の再生トラックリストを選択できます。現在アクティブなフィルタは、CurrentAudioListFilterファンクションで読み出します。
- AuxIn FBlockのバージョン3.5では、モバイル機器の音楽ファイルに格納されたカバーアート情報を要求するためにSelectCoverArtファンクションとSelectCurrentCoverArtファンクションが追加されました。RetrieveCoverArtファンクションを使うと、対応する画像ファイルを取得できます。
- その他の各種設定を決定するファンクションとして、例えばメタデータを制御チャンネルとパケットデータ チャンネルのどちらで送信するかを決定するAsyncControlSwitchファンクション等があります。

4.4.4 使用例(AuxIn)

このセクションでは、FBlock AuxInの例を挙げてMOSTシステムにおけるFBlockの使い方を説明します。特に、FBlockにアクセスする際の重要なメッセージ シーケンスについて見ていきます。ここに示したシーケンスは、FBlock AuxInの仕様[FBAuxIn 3.5]に記載されたものを引用しています。これらはあくまでも例であり、実際の実装では異なる箇所があります。

システム初期化時に、コントローラはHMI表示の更新等で必要となるノーティフィケーションを設定します(図4.11参照)。FBlock AuxInで必要なノーティフィケーションとしては、AuxInデバイスのステータス(DeckStatus)、現在アクティブなトラック番号(AuxTrackPosition、TrackInformation)、再生時間(TimePosition)等があります。ノーティフィケーションを設定すると、デバイスが次のトラックに移動するたびにコントローラに通知があり、それに基づいてコントローラはHMIの情報を更新できます。

接続したモバイル機器の機能をユーザが要求した場合、コントローラはこの要求をFBlock AuxInの対応コマンドに変換する必要があります。図4.12に、DeckStatusプロパティを用いてデバイスを再生モードにする例を示します。この変更結果は、事前に通知されたプロパティで返されます。コントローラは、このノーティフィケーションによる情報および自分自身でスレーブに問い合わせた現在再生中のトラックのメタデータに基づいてHMIの表示を更新します。

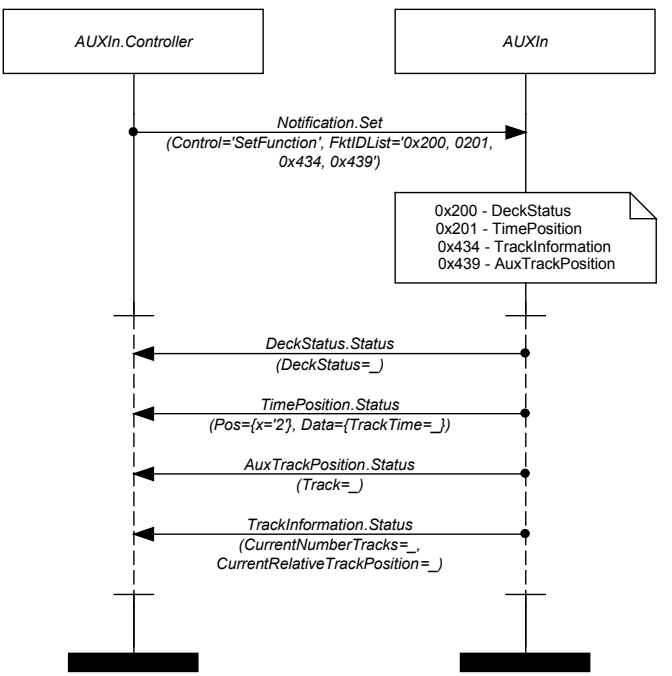


図4.11:初期化の際のノーティフィケーション設定

図4.13に、現在再生中の音楽トラックに関する情報の問い合わせを示します。SelectCurrentAudioListInfoファンクションにより、現在再生中のトラックのID(タグ)、タイトル、アーティスト、メディアタイプを問い合わせます。パラメータStartの値を0とすると、現在のトラックを表します。応答は、現在のトラックリストにおけるトラックの実際の位置を含みます(図の例では、現在のトラックリストの最初のトラック)。

コントローラがこのシーケンスを開始するのは、例えばコントローラがFBlock AuxInに対して何らかのアクション(上述)を要求した場合、またはスレーブがステータスの変化(例: 曲の再生が次のトラックに移動)を通知した場合です。

ユーザがコントローラのHMIを操作してデバイスのメディア データベースをブラウズして目的のトラックを探す場合、コントローラはFBlock AuxInに対して対応するメディアファイル リストの情報を問い合わせる必要があります。通常、HMIではトラックの属性(アーティスト名、アルバム名、ジャンル、曲目)を指定して何種類かの方法でトラックリストを表示できます。

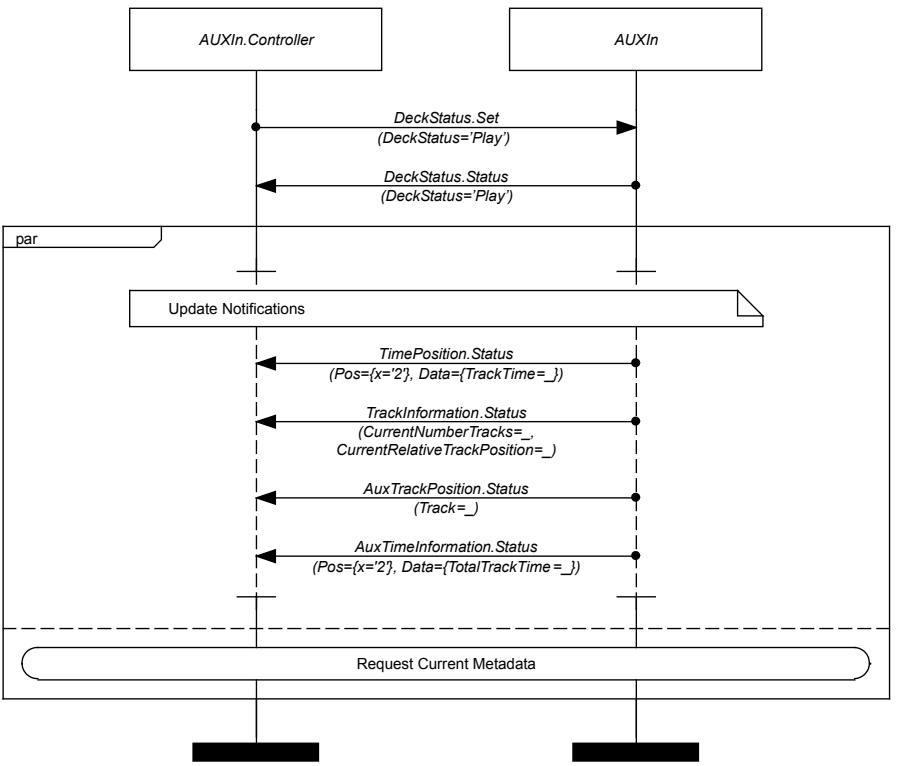


図4.12:再生モードの設定

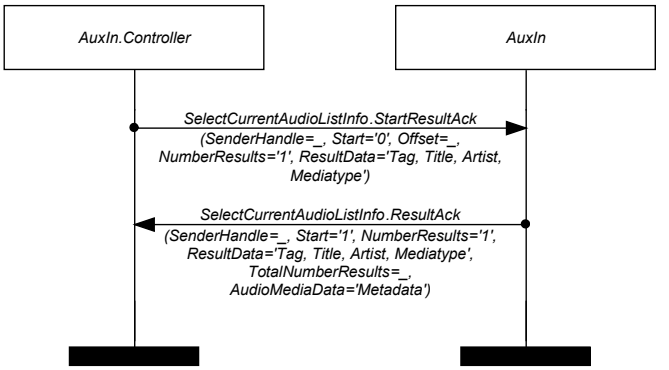


図4.13:現在再生中のトラックのメタデータを問い合わせ

図4.14に、ユーザがある特定のアーティストを探す場合の例を示します。ここでは、HMIに1ページ当たり3件のエントリを表示できるものとします。まず、SelectAudioListInfoファンクションを用いて最初の3件のエントリに関する情報を問い合わせます。その結果、選択したトラックリストの最初のトラックを先頭に、タグ、アーティスト名、メディアタイプの情報がアーティスト名でソートされた形で返されます。ユーザがリストを下へスクロールすると、コントローラは先頭トラックの位置に3を加算して次ページのエントリを問い合わせます。通常、コントローラはスクロール速度を高めるために後続トラックの情報を事前にキャッシュします。

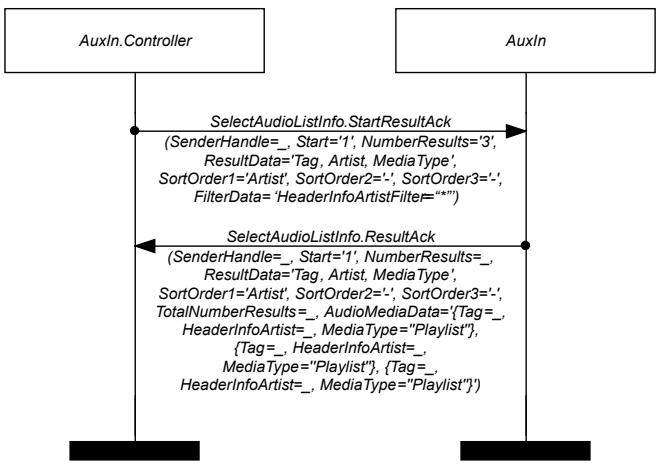


図4.14:アーティスト リストの問い合わせ

5 MOSTプロトコル

MOSTシステムは、44.1 kHzまたは48 kHzのサンプルレートを使ってHi-Fi品質のデジタル オーディオを送信します。サンプルレートの異なるデバイスは、サンプルレート変換を利用してネットワークに接続できます。MOSTシステムでは、48 kHzの使用を推奨します。

MOSTシステムはオーディオデータを同期的に送信するため、バッファ処理を追加する必要はありません。これによりデバイスの複雑さを抑え、コストを削減しています。

5.1 MOSTフレーム

MOSTシステムには、MOST25、MOST50、MOST150の3種類のフレームがあり、それぞれ機能が異なります。

5.1.1 MOST25フレーム

MOST25フレームは512ビット(64バイト)から成ります。このうち60バイトをストリームおよびパケットデータの送信に使います。1フレーム2バイトで構成される合計32バイトの制御メッセージをネットワークとノードの管理に使います。制御メッセージを16フレームに分散して転送し、これを1ブロックに結合します。最初と最後のバイトは、そのフレームの制御情報を格納します。表5.1に、概要を示します。

バイト	ビット	説明
0	0-3	プリアンブル
	4-7	境界ディスクリプタ
1	8-15	データバイト0
2	16-23	データバイト1
…	…	
60	480-487	データバイト59
61	488-495	制御フレームバイト0
62	496-503	制御フレームバイト1
63	504-510	フレーム制御およびステータスビット
	511	パリティビット

表5.1:MOST25フレームの構造

図5.1に、MOST25フレームの構造を示します。

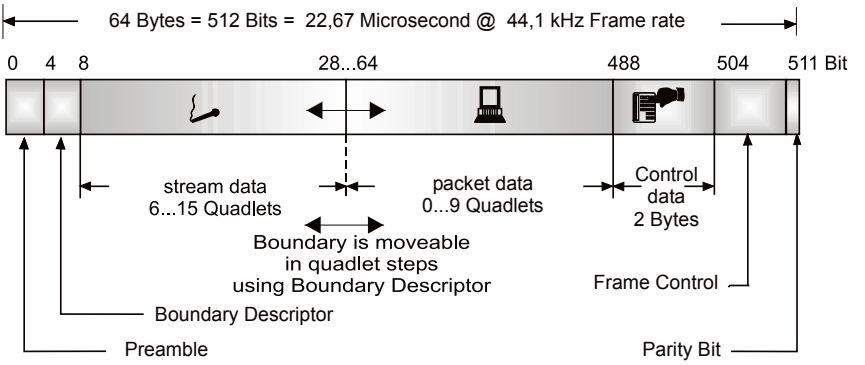


図5.1:MOST25フレーム[MOST 2.4]

プリアンブル

MOST25フレームのプリアンブルは、TimingSlavesをビットストリームに同期するため、およびフレームの最初を特定する目的で使います。スレーブはPLLスイッチを使ってネットワークに同期します。

TimingMasterは、自身のオシレータ周波数またはS/PDIF入力信号に基づいてプリアンブルを生成します。

ビットストリームが全てのMOSTノードを通過してマスタに到着すると、そのビットストリームは各MOSTノード内の信号伝搬遅延の分位相がシフトします。このため、マスタはPLLを利用して受信したフレームに自分自身を同期し、全てのビットをリカバリし、フレームを再生成して位相シフトを補償します。

境界ディスクリプタ

1フレームのうち、合計60バイトをストリーミング データ チャンネルとパケット データ チャンネルで共用します。これら2つのチャンネルの帯域幅は、それぞれの要件に合わせて境界ディスクリプタで調整できます。

これら2つの領域の境界は、4バイト (クワドレット)単位でシフトできます。ストリーミング データのチャンネル幅は24～60バイト(6～15クワドレット)、パケット データのチャンネル幅は0～36バイト(0～9クワドレット)です。

境界ディスクリプタは、6～15の値をとります。TimingMasterのMOSTネットワーク インターフェイス コントローラが境界ディスクリプタを管理します。

TimingMasterが境界ディスクリプタの値を変更すると、全ての同期接続を確立し直す必要があります。

ストリーミング データとパケットデータの各チャンネル帯域幅は、下式で求めます。

パケット帯域幅 = 全体の帯域幅 – (境界 * 4)　　(式5.1)
ストリーミング帯域幅 = (境界 * 4)
パケット帯域幅 + ストリーミング帯域幅 = 60

フレーム制御

フレームの最後のバイトは、フレームの制御に使います。

パリティビット

パリティビットを利用して、フレームのビットエラーを検出します。

5.1.2 MOST50フレーム

MOST50のビットレートは50 Mbit/sで、帯域幅が2倍に拡大しています。MOST50という名称は、このビットレートを表しています。サンプルレートは変わりませんが、フレーム長が1,024ビット(128バイト)に拡張されています。

図5.2と表5.2に、MOST50フレームの構造を示します。制御データ(4バイト)を11バイトのヘッダに格納します。ヘッダには、境界ディスクリプタとシステムロックフラグも含みます。ストリームデータとパケットデータの帯域幅は、現在の要件に合わせて動的に変更でき、同期接続を確立し直す必要はありません。これは、NetBlock.Boundaryプロパティで制御します。ストリームデータの幅は0～29クワドレット+ 1バイト(1～117バイト)、パケットデータの幅は0～29クワドレット(0～116バイト)とする事ができます。

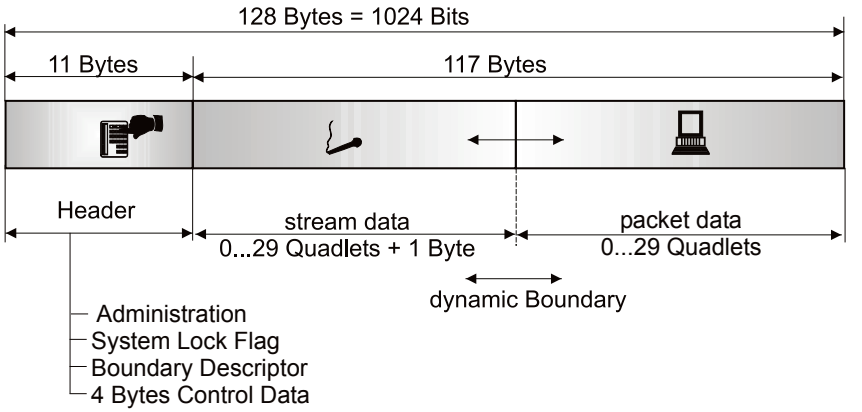


図5.2:MOST50フレーム[MOST 2.5]

ストリーミング データとパケットデータの各チャンネル帯域幅は、下式で求めます。

パケット帯域幅 = 全体の帯域幅 – (境界 * 4) - 1
ストリーミング帯域幅 = (境界 * 4) + 1 (式5.2)
パケット帯域幅 + ストリーミング帯域幅 = 117

バイト	ビット	説明
0-10	0-87	管理用(以下を含む) システムロック フラグ 境界ディスクリプタ 4バイトの制御データ
11-127	88-1023	データバイト

表5.2:MOST50フレームの構造

5.1.3 MOST150フレーム

MOST150フレームは、MOST50と似た構造をしています。サンプルレートは同じでフレーム長を3,072ビット(384バイト)とし、MOST50の3倍のbaudレートを実現しています。

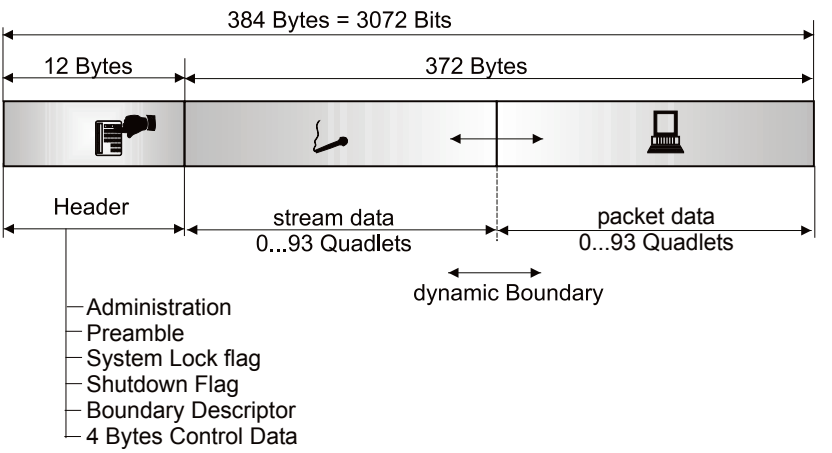


図5.3:MOST150フレーム[MOST 3.0]

図5.3と表5.3に、MOST150フレームの構造を示します。ここでも制御バイトと境界ディスクリプタは、12バイトのヘッダにそれぞれ4バイトずつ格納されます。MOST50同様、帯域幅は動的に変更できます。ストリームデータの幅は0～93クワドレット(0～372バイト)、パケットデータの幅も0～93クワドレット(0～372バイ

ト)とする事ができます。ストリーミング データとパケットデータのどちらも、利用可能な全ての帯域幅を使えます。

バイト	ビット	説明
0-11	0-95	管理用(以下を含む) • プリアンブル • システムロック フラグ • シャットダウン フラグ • 境界ディスクリプタ • 4バイトの制御データ
11-383	96-3071	データバイト

表5.3:MOST150フレームの構造

境界ディスクリプタ

MOST150でも境界ディスクリプタはNetBlock.Boundaryプロパティで制御し、この値を変更しても同期接続を確立し直す必要はありません。この値を変更できるのは、TimingMaster機能を持つノードのアプリケーションのみです。初期値はTimingMasterのコンフィグレーションで設定します。

ストリーミング データとパケットデータのチャンネル帯域幅は、下式で求めます。

パケット帯域幅 = 全体の帯域幅 – (境界 * 4)
ストリーミング帯域幅 = (境界 * 4)
パケット帯域幅 + ストリーミング帯域幅 = 372

表5.4に、全てのバージョンのMOSTにおける境界ディスクリプタの値と帯域幅の関係をまとめます。

NetBlock. Boundary	MOST25 利用可能な帯域幅 (1フレーム当たりのバイト数) ストリーミングデータ	MOST25 パケットデータ	MOST50 利用可能な帯域幅 (1フレーム当たりのバイト数) ストリーミングデータ	MOST50 パケットデータ	MOST150 利用可能な帯域幅 (1フレーム当たりのバイト数) ストリーミングデータ	MOST150 パケットデータ
0	n/a		1	116	0	372
1			5	112	4	368
2			9	108	8	368
3			13	104	12	360
4			17	100	16	356
5			21	96	20	352
6	24	36	25	92	24	348
…						
14	56	4	57	60	56	316
15	60	0	61	56	60	312
16	n/a		65	52	64	308
…						
29			117	0	116	256
30			n/a		120	252
…						
92					368	4
93					372	0

表5.4:MOST25、50、150の場合における境界ディスクリプタと帯域幅の関係

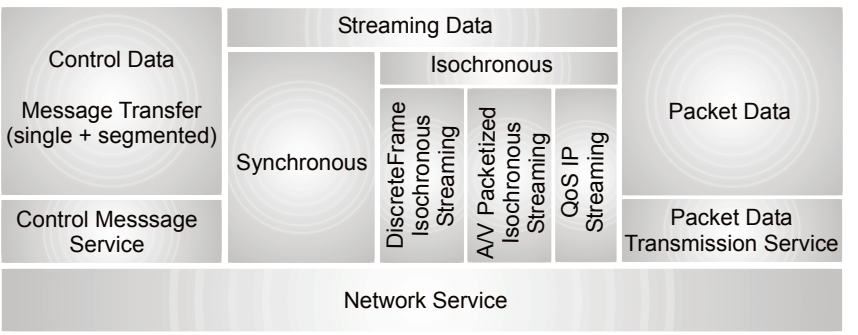


図5.4:MOSTのデータ転送メカニズム

5.2 MOSTのデータ転送メカニズム

MOSTは、制御データ、ストリーミング データ、パケットデータに関するデータ転送メカニズムを定義しています(図5.4参照)。サポートされるメカニズムは、MOST25、MOST50、MOST150で異なります。これらについて、以下のセクションで説明します。

5.3 ストリーミング データ

MOST仕様Rev. 3.0では、同期データとアイソクロナス データを区別します。アイソクロナス データを転送できるのはMOST150のみです。

ストリーミング データ チャンネルの帯域幅は、境界ディスクリプタ (セクション5.1.2参照)とフレームのバージョンで決まります。帯域幅の計算式は以下の通りです。

$$BW = SBC * 4 * 8Bits * Fs \quad \text{(MOST25、MOST150の場合)} \qquad \text{(式5.3)}$$

$$BW = (SBC * 4 + 1) * 8Bits * Fs \quad \text{(MOST50の場合)}$$

BW: 帯域幅
SBC: 境界ディスクリプタ(同期帯域幅制御)
Fs: サンプルレート

まとめ

	MOST25	MOST50	MOST150
サンプルレート(kHz)	44.1	48	48
最小帯域幅(Mbit/s)	8.467	0.384	0
最大帯域幅(Mbit/s)	21.168	44.928	142.848

表5.5:3種類のフレームの帯域幅

5.3.2 同期データ

同期データは、オーディオおよびビデオデータのリアルタイム転送に使います。データを送信する前に、ConnectionMasterはアプリケーション メッセージ サービス(AMS)を利用して接続を確立する必要があります。

この目的のため、NICではいわゆるルーティング エンジン(RE)を使っていました。

INICは、ルーティング エンジンの代わりにソケットを使います。そのためにはソケットを1つ作成し、フレーム(MOSTネットワーク ポート)へのインターフェイスおよびローカルリソース(MediaLBまたはストリーミング ポート)へのインターフェイスに接続する必要があります(セクション10.4.3参照)。

MOST25では1フレームにつき最大15のステレオ接続(各チャンネル2x16ビット)または60個の1バイト接続を同時に確立できます。MOST50では最大29、MOST150では最大93のステレオ接続を転送できます。1つの接続には必ず1つのソースと任意の数のシンクがあります。フレームの内容は、フレームが送信元のノードに戻るまで変わりません。

あるチャンネルで準静的に接続を確立する事を時分割多重(TDM)と呼びます。同じフレーム位置のデータを、特定の時間パターンで周期的に送信します。通信エラーが生じても再送しません。有効な値は次のサイクルで利用可能になります。

5.3.3 アイソクロナス データ

MOST25/50で送信できるのは同期データのみです。MPEGデータストリームを送信する場合は「ビット スタッフィング」を行い、ストリームを固定ビットレートに変換(トランスコード)します。オーディオ アプリケーションでPCMオーディオデータをMOSTフレームレートに変換するには、サンプルレート コンバータが使われる事があります。しかし、オーディオまたはビデオ データストリームによってはこのMOSTタイムベースに簡単に同期できない場合があります。

MOST150ではアイソクロナス転送を導入しました。アイソクロナス チャンネルを、MOSTの同期チャンネルとほぼ同じ方法で処理します。必要な帯域幅を固定的に予約、すなわちアイソクロナス データ用のチャンネルを割り当て、必要に応じてデータをルーティングします。MOST150では、アプリケーションの性格に合わせて3種類のアイソクロナス メカニズムを利用できます。アクセスは、TDMおよび準静的物理チャンネルの割り当てによって行います。

A/Vパケット化アイソクロナス ストリーミング

このメカニズムでは、MOSTタイムフレームへの参照情報なしで到着したビデオデータストリームを転送できます。このデータは既に小さなデータパケットに集約されており、任意でタイムスタンプを含める事もできます。この代表例として、VBR(可変ビットレート)またはCBR(固定ビットレート)で符号化されたMPEGデー

タストリームがあります。MOSTの転送では、必要な最大アイソクロナス帯域幅を最初に予約します。図5.5に、このメカニズムを示します。

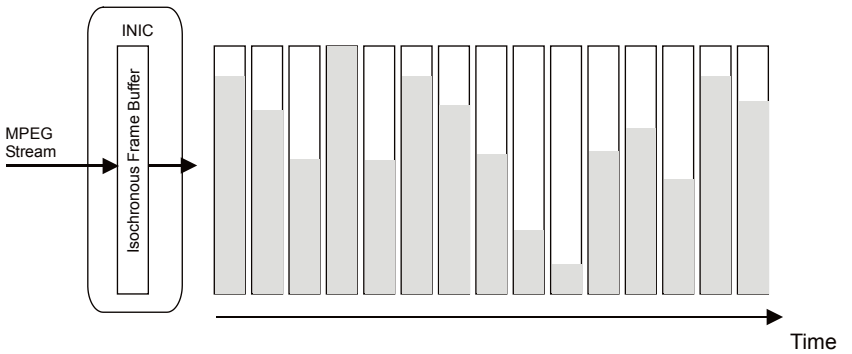


図5.5:A/Vパケット化アイソクロナス ストリーミング
(出典:Microchip Technology Inc.)

MOSTネットワーク インターフェイス コントローラに到着するデータはINICによって自動的に内部メモリへ転送され、その内容はMOST経由で周期的に転送されます。データシンクのネットワーク インターフェイス コントローラは、バスからのアイソクロナス データを内部メモリへコピーし、MPEGデコーダ等の適切なインターフェイスを経由して外部ハードウェアへ渡す事ができます。MPEGデータをMOST150のINICに完全にカプセル化して扱うため、アプリケーションが行うのは必要なアイソクロナス帯域幅の予約と対応する接続の設定だけです。

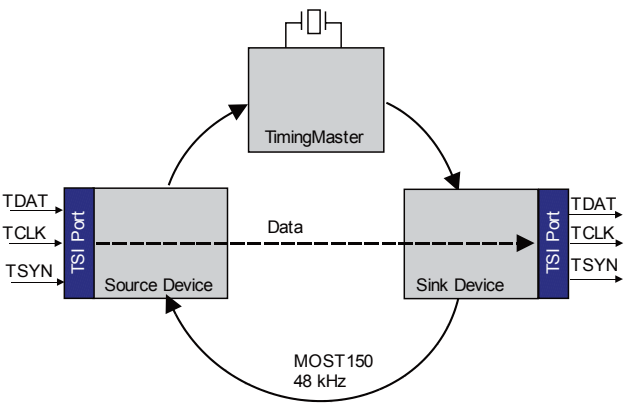


図5.6: アイソクロナス ポート/ソケットとTSIを介したMPEGデータのアイソクロナス データ転送
(出典:Microchip Technology Inc.)

MOST150のINICは、各種の標準インターフェイス以外に、ビデオ チップセットの標準として確立しているトランスポート ストリーム インターフェイス(TSI)を含みます。図5.6に、ノード当たり1つのアイソクロナス ポート/ソケットを使ったMPEGデータのアイソクロナス データ転送を示します。

DiscreteFrameアイソクロナス ストリーミング

オーディオデータ ストリームは同期データとしての性質(単位時間当たりに一定量のデータを高精度クロック信号と一緒に送信する)を持ちますが、MOSTと同じタイムベースには同期しない事がしばしばです。1つの例が、サンプリング周波数44.1(または96) kHzのPCMオーディオデータをMOSTフレームレート48 kHzのMOST150ネットワークで転送するケースです。オーディオCDで使うサンプルレートは44.1 kHzです。もう1つの例は、S/PDIF (Sony Philips Digital InterfaceまたはSony Philips Digital Interconnect Format)信号をMOSTネットワークで転送する場合です。S/PDIFはオーディオデータをデコーダ チップセットに接続するためにホームオーディオの分野で広く採用されているインターフェイス規格で、端子には光TOSLINK(光)または同軸RCAジャック(電気)を使います。通常、ホームオーディオ機器には1つまたは複数のデジタルS/PDIF入力または出力を備えたコンポーネントがあります。このインターフェイスは、IEC 60958に従ってPCMオーディオデータ、またはIEC 61937に従ってDTS、Dolby Digital (AC-3)、THX等の圧縮オーディオデータを一方向に転送するために使います。アイソクロナス転送を使えば、これらのタイプのデータストリームをMOST経由で「トンネリング」できるため、MOSTフレームレートに同期する必要も、サンプルレート コンバータを使ってデータを変換する必要もありません。

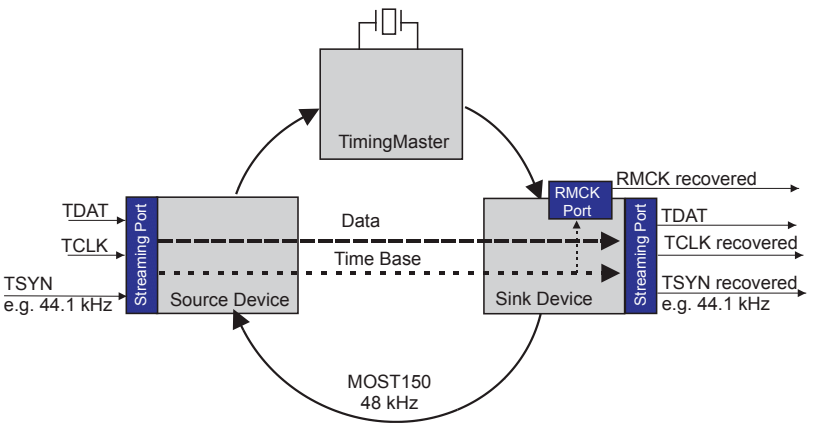


図5.7:2つのアイソクロナス ポート/ソケットを介したオーディオデータのアイソクロナス データ転送(出典:Microchip Technology Inc.)

オーディオ信号をアイソクロナス転送するには、MOST経由でデータをトンネリングするだけでは不十分です。タイムベースを維持する事も必要です。つまりタイムベースを転送し、シンクが高い精度でクロックに再同期する必要があります。

MOST150のINICには、MOST以外のクロック信号を入力する機能があります。この入力信号をサンプリングし、タイムベースをデータと一緒に転送する必要があります。クロックを表すタイムベースをネットワーク上で転送後、シンク内のINICの

専用ユニットがタイムベースを復元し、このクロックレートでデータを再び出力します。

図5.7に、2つのアイソクロナス ポート/ソケット(1つはデータ用、もう1つはタイムベース用)を使ったオーディオデータのアイソクロナス データ転送を示します。

QoSアイソクロナス モード

QoSアイソクロナス モードはQoS IP(ストリーミング)とも呼ばれます。このモードについては、セクション5.8.3で説明します。

アイソクロナス データについては、『MOST Stream Transmission Specification』[MOST Stream]で詳しく説明しています。

5.4 パケットデータ チャンネル

パケットデータ チャンネルでは、長いデータパケットと制御データを送信できます。パケットデータ チャンネルの帯域幅も境界ディスクリプタで決まります。パケットデータ チャンネルはデータリンク層プロトコルを使い、TCP/IPプロトコルやMOST Highプロトコル(MHP)等、周期的には送信されないパケットデータを転送します(セクション5.7参照)。

MOST仕様では、合計4つのパケットデータ プロトコルを定義しています。図5.6にMOST25とMOST50のプロトコル、図5.7にMOST150のプロトコルを示します。

5.4.1 MOST25とMOST50のパケットデータ プロトコル

MOST25とMOST50のプロトコルで使うデータフィールド長には2種類あります(図5.8参照)。48バイトのデータリンク層プロトコルは最大48データバイト伝送し、もう1つの層は最大1,014データバイトを伝送します。

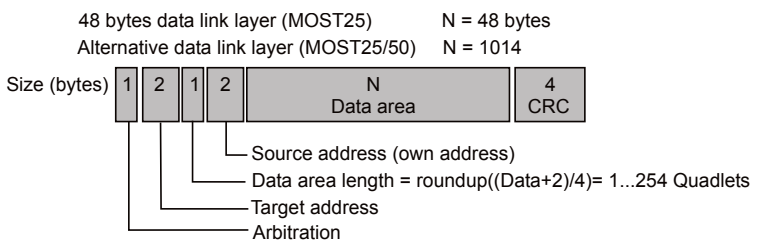


図5.8:MOST25とMOST50のパケットデータ プロトコルの構造

NICはどちらのデータフィールド長もサポートしますが、NICの内部キャッシュは48バイトのプロトコルまでしか対応できません。INICは、I²Cバスで制御される場合は48バイトのプロトコルを使います。1,014バイトのデータリンク層プロトコルを使うにはMediaLBが必要です(第10章参照)。

プロトコルヘッダは、トークンを格納する調停フィールド(1バイト)、ターゲットアドレス(2バイト)、データフィールド長(1バイト)、ソースアドレス(2バイト)から成ります。MOSTネットワーク インターフェイス コントローラが自動的に生成するCRCサム(4バイト)が、このプロトコルを保護します。CRCサムエラーの場合、自動再送は行われません。このエラーは、より高位の階層で対処する必要があります。

5.4.2 MOST150のパケットデータ プロトコル

MOST150のパケットデータ プロトコルはMOST25/50とは構造が異なっており、以下の2つのアドレッシング メカニズムがあります。

- 16ビット ターゲット アドレス(従来のMOSTアドレッシング)
- 48ビット ターゲット アドレス(MACアドレッシング)

CRCサムの位置はデータフィールドの前で、データ長の情報はヘッダの一部です(図5.9参照)。

16ビット アドレッシング モードを使う場合、最初の6データバイトがMOST Highプロトコルのヘッダを格納します。

48ビット アドレッシングは、Ethernet over MOSTで使います。この場合、CRCサムはEthernet FCSアルゴリズムで計算します。詳細は、5.8.2を参照してください。

どちらのアドレッシング モードもローレベル リトライをサポートします。ローレベル リトライの回数とリトライ間の伝搬遅延時間はシステム インテグレータが定義します。

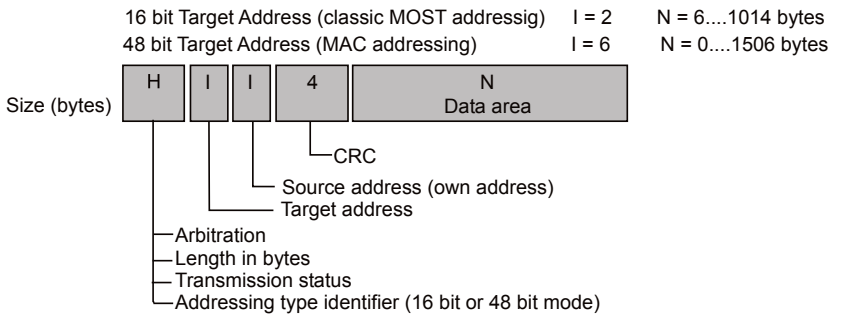


図5.9:MOST150のデータリンク層プロトコルの構造

5.4.3 調停アルゴリズム

パケットデータ チャンネルでは、トークンによるアクセス制御を行います(図5.10参照)。データを送信しようとするノードがない場合、トークンをノードからノードへと渡します。

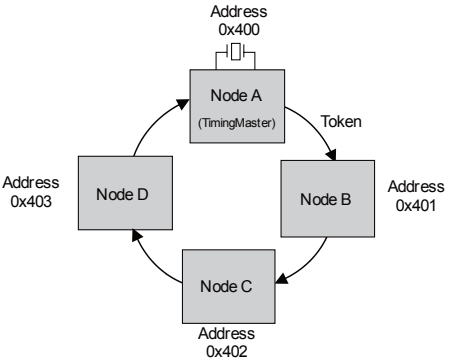


図5.10:4ノードのリングバス構造

ノードがデータを送信しようとする場合、トークンを待つ必要があります。ノードは、バスからトークンを受け取るとパケットデータ チャンネルに対する排他的アクセスの承認を得ます。ノードは1パケットを送信後、トークンをバスに戻します。さらにパケットを送信する場合、もう一度トークンを待つ必要があります。

パケットデータ転送の優先制御は、トークンへのアクセスを放棄する事によって実現します。つまり、優先度の低いノードはデータ送信準備が完了していてもトークンを数回通過させた後にバスからトークンを取得します。こうすると、他のノードの方がトークン取得の優先度が高くなります。

5.5 制御チャンネル

制御チャンネルは、ネットワーク管理に使う制御メッセージ サービス(CMS)を提供します。このチャンネルは、主に低データレートと短いパケット長でのイベント指向の送信向けです。制御チャンネルはCRCで保護され、自動リトライを行うACK/NAKメカニズムを備えています。制御メッセージは、アプリケーション メッセージ サービス(AMS)を1回で、または複数のセグメントに分割して送信します。

以下のセクションで、3つのバージョンのそれぞれ異なる制御メッセージについて説明します。

5.5.1 MOST25のCMS

図5.11に、MOST25の制御メッセージの構造を示します。

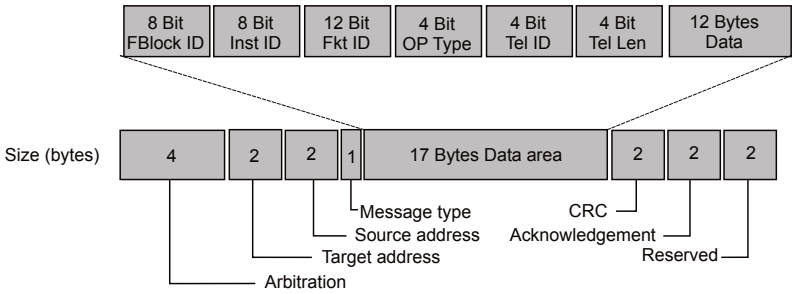


図5.11:制御メッセージの構造(出典:[MOST 2.4])

制御チャンネルが1フレーム当たりの帯域幅をあまり多く占有しないよう、制御チャンネルを16フレームに分散し、これらを結合して1つのブロックとします(図5.12参照)。1フレームで2バイト分の制御チャンネルを伝送します。1ブロックの最初のフレームのプリアンブルには、ブロックを識別するための特定のビットパターンを格納します。

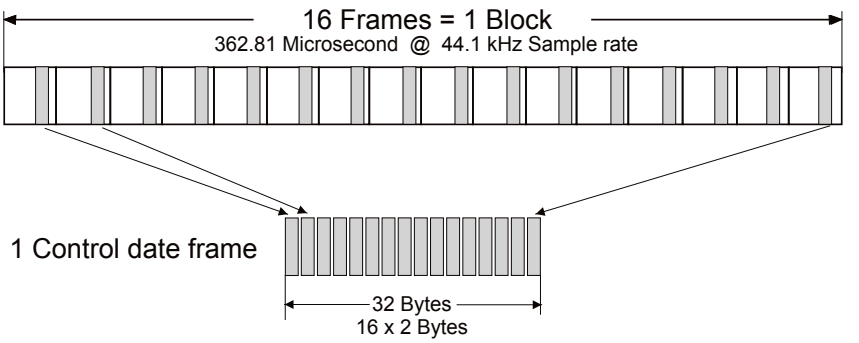


図5.12:制御チャンネルのMOST25フレームへのマッピング

制御チャンネルのプロトコルの長さは32バイトで一定です。

調停

調停は、最初の2フレーム(4バイト)に基づき、CSMA(搬送波感知多重アクセス)方式に従って行います。メッセージの優先度はリング内のノード位置に依存しないため、バス負荷が高い場合でもこのアクセス方式によってバスの公平な割り当てを保証します。調停は、MOSTネットワーク インターフェイス コントローラが自動的に実行します。

アドレス

バイト4とバイト5にはターゲット アドレス、バイト6とバイト7にはソースアドレスを格納します。

メッセージタイプ

バイト8は、以下のいずれか1つのメッセージタイプを伝送します(図5.11参照)。

- 通常メッセージ(0x00)
- システム メッセージ
  - Resource Allocate (0x03)
  - Resource DeAllocate (0x04)
  - Remote GetSource (0x05)
  - リモート メッセージ
    - Remote Read (0x01)
    - Remote Write (0x02)

通常メッセージ

通常メッセージ (メッセージタイプ: 0x00)には、制御メッセージ サービス(CMS)のデータパケットを格納します。つまり、通常メッセージはプロパティの変更およびファンクション ブロックのファンクション開始のためのデータを伝送すると共に、その結果の応答メッセージを伝送します。例えば、CDプレーヤはファンクション ブロックを利用して制御できます。

実際のデータは17データバイトで伝送します。通常メッセージは1つのレシーバに送信する事も(シングルキャスト)、定義した複数のレシーバグループに送信する事も(グループキャスト)、全てのノードに送信する事も(ブロードキャスト)できます。

システム メッセージ

システム メッセージには、0x03～0x05のコードがあります。これらはMOSTネットワーク インターフェイス コントローラのみが使い、アプリケーションからは透過的です。Resource Allocate(メッセージタイプ: 0x03)とResource DeAllocate(メッセージタイプ: 0x04)等のリソース管理に関するシステム メッセージは全てのストリーミング データ ソースが発行できます。ただし、これらを受信して評価できるのはTimingMasterのみです。

システム メッセージには、Remote Read(メッセージタイプ: 0x01)とRemote Write(メッセージタイプ: 0x02)等のリモート メッセージもあります。リモートメッセージを利用して、NICは別のインターフェイス コントローラに対して1～8データバイトの書き込みまたは読み出しを実行します。リモート メッセージはデ

バッグ専用であり、通常の動作モードでは使わないでください。INICにはレジスタウォールがないため、リモート メッセージはサポートしていません(第10章参照)。

メッセージタイプ0x05は、Remote GetSourceメッセージを表します。このメッセージを利用して、ストリーミング チャンネルのソースを確認できます。これは通常ブロードキャスト メッセージとして送信され、ストリーミング ソースの物理、論理、グループアドレスを確認します。何らかの誤りで1つのチャンネルに複数のソースがある場合、ソースを確認して競合を解決するためにメッセージをシングルキャスト メッセージとして送信できます。

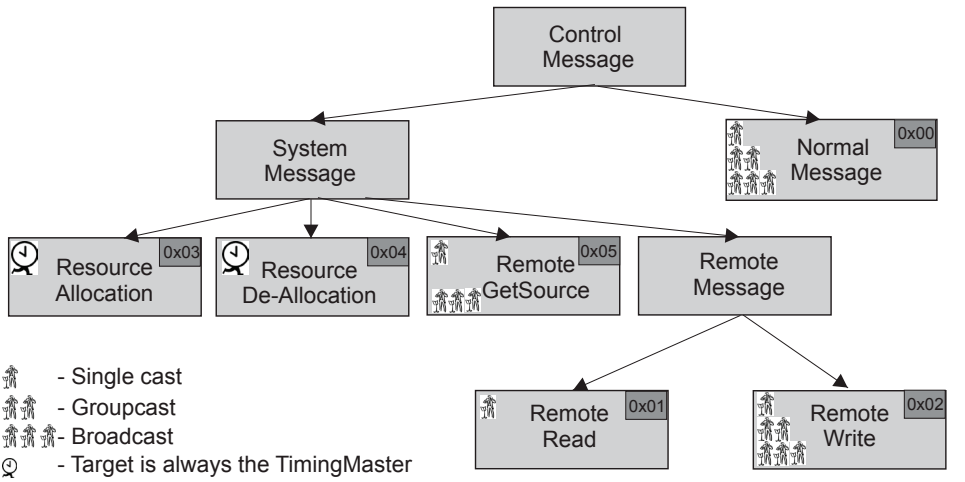


図5.13:メッセージタイプ(出典:[DS 8104])

CRC

レシーバは、このCRCサムを使って送信エラーを検出します。

ACKフラグ

レシーバは、制御メッセージをエラーなしで受信できた事をACKフラグ (バイト28、29)を使ってセンダに通知します。以下の状態を通知できます。

- メッセージを正常に受信した
- キャッシュがブロックされた
- CRCエラー

レシーバはこのフラグに書き込みができるだけで、リセットはできません。このフラグにより、グループキャストおよびブロードキャスト メッセージの場合であってもセンダは少なくとも1つのレシーバがメッセージを正しく受信できなかった事を認識できます。しかし、センダはどのレシーバが正しく受信できなかったのか識別できません。

既定値では、通信エラーの場合に5回の再送が実行されます。この回数は、NIC (OS8104)のbXRTYレジスタで変更できます。INICの場合、RetryParametersファンクションを使います。MOSTネットワーク インターフェイス コントローラ(NIC)は、決まったブロック数の間待機してから再送します。このブロック数は設定可能で、既定値は11です。NIC (OS8104)の場合、この数はbXTIM (Transmit Retry Time)レジスタで設定します。INICの場合、前述のRetryParametersファンクションを使います。

最後の再送でもテレグラムを正しく送信できない場合、センダはホスト コントローラに通知します。

データフィールド

この17データバイトでアプリケーション メッセージを伝送します。このフィールドはDeviceID、ファンクション ブロックID (FBlockID)、インスタンスID、ファンクションID、オペレーション コード、関連パラメータから成り、その構造はファンクション ブロックと密接に関連しています。アプリケーション レベルでは、以下の構文を使ってファンクションのアドレスを指定します。

```
DeviceID.FBlockID.InstID.FktID.OPType (Parameter)
```

詳細は、セクション4.2を参照してください。

サービスが必要とするパラメータが12バイトを超える場合、メッセージを複数のセグメントに分割します(TelID > 0)。また、最初のデータバイトはテレグラム カウンタとして使います。つまり、パラメータに使えるのは11データバイトのみです。最初のテレグラムはTelID = 1で、カウンタが0に設定されます。それ以降のテレグラムは全てTelID = 2です。カウンタは0x01から0xFFまでカウントすると、0x00から再開します。最後のテレグラムはTelID = 3です。TelLenは1テレグラムのバイト数を示します。つまり、TelID = 1または2の場合、TelLenは常に12です。TelID = 0または3の場合、12より少ない事があります。

データレート

1秒に送信できる制御メッセージの数は、サンプルレートで決まります。64の送信ブロックのうち、制御メッセージに使えるのは62ブロックのみです。2つのブロックは、チャンネル リソース割り当てテーブルの送信等、システム内のネットワーク管理に使います。1つの制御メッセージを送信するには16フレームが必要なため、1秒当たりのメッセージ数は下式で求めます[Found 04]。

$$CM = \frac{62}{64} * \frac{Fs}{16} = 2{,}670 Msg/s \,@\, 44.1 kHz \qquad \text{(式5.4)}$$

CM: 1秒当たりの制御メッセージ数

Fs: サンプルレート

各メッセージは19バイト(17バイトのデータ + 2バイトのDeviceID)を伝送するため、総データレート(DR_{gross})は下式で求めます。

$$DR_{gross} = CM * 19 * 8Bits = 405.84kbits/s@44.1kHz \quad (式5.5)$$

制御メッセージの送信が正しく完了すると、各ノードは2メッセージの間停止する必要があります。このため、より優先度の高いノードがデータを送信していない場合の実質データレート(DR_{net})は総データレートの1/3となります。

$$DR_{net} = \frac{DR_{gross}}{3} = 135.28kbits/s@44.1kHz \quad (式5.6)$$

実際のデータレートは、NICで使うインターフェイスの影響も受けます。

5.5.2 MOST50のCMS

MOST50のフレームは、1フレームにつき4バイトの制御データをヘッダ内に格納します。従って、1つのCMSメッセージを4バイトずつに分割してフレームに格納します。MOST25のフレームとは対照的に、制御メッセージの長さは個々のメッセージにより異なります。このため、CMSで使うフレーム数は可変です。この結果、帯域幅の利用率が向上します。ペイロードのないコマンドの場合にフレーム数は最小の6となり(TelLen = 0)、最大フレーム数は9 (TelLen = 12)です。図5.14に、制御メッセージの構造を示します。ペイロードが12バイトを超えるCMSメッセージは複数のセグメントに分割して伝送し、最大長は65535バイトです。MOST25と同様、これにはAMSも含みます。メッセージタイプはMOST50にはありません。

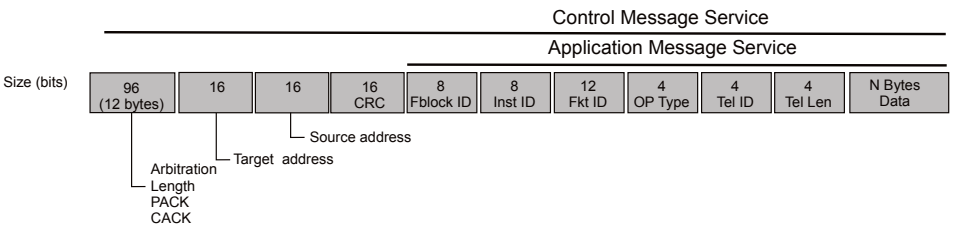


図5.14: MOST50の制御メッセージの構造

ヘッダは12バイトで、調停、メッセージ長、プリエンプティブACK (PACK)、コンプリートACK (CACK)から成ります。

プリエンプティブACK (PACK)

PACKバイトはフロー制御に使います。レシーバがこれを駆動し、レシーバの準備が完了しているか、あるいは受信バッファフルまたはターゲット誤りによってメッセージが中止されたかを示します。

コンプリートACK (CACK)

メッセージの最後に送信されるコンプリートACK (CACK)もレシーバが駆動します。これは、CRCエラーまたは送信の成功を示します。

1フレームのデータレート

1秒当たりのメッセージ数は、下式で求めます。

$$CM = \frac{Fs}{number\_of\_frames} \qquad \text{(式5.7)}$$

$$CM_{\max} = \frac{48000}{6} = 8{,}000@48kHz; TelLen = 0$$

$$CM_{\min} = \frac{48000}{9} = 5{,}333@48kHz; TelLen = 12$$

どのメッセージにもペイロードがない場合(TelLen = 0)、1秒当たりの制御メッセージ数は最大値の8,000です。全てのメッセージのペイロードが最大サイズ(TelLen = 12)の場合、5,333メッセージ/秒です。

ペイロードの最小値 ($Payload_{min}$) は7バイト (DeviceID、FBlockID、InstID、OPType、TelID、TelLen) で、ペイロードの最大値 ($Payload_{max}$) は19バイト ($Payload_{min}$ + 12バイトのデータ)です。従って、実質レートは下式で求めます。

$$\begin{aligned} DR_{gross\_\min} &= CM_{\max} * payload_{\min} * 8Bits \\ &= 8{,}000 * 7 * 8Bits \\ &= 448kbit/s@48kHz; TelLen = 0 \end{aligned}$$

$$DR_{net\_\min} = \frac{DR_{gross\_\min}}{3} = 149.33kbit/s@48kHz; TelLen = 0$$

$$\begin{aligned} DR_{gross\_\max} &= CM_{\min} * payload_{\max} * 8Bits \\ &= 5{,}333 * 19 * 8Bits \\ &= 810.62kbit/s@48kHz; TelLen = 12 \end{aligned}$$

$$DR_{net\_\max} = \frac{DR_{gross\_\max}}{3} = 270.21kbit/s@48kHz; TelLen = 12$$

MOST50の理論上の実質データレートは最小で149.33 kbit/s、最大で270.21 kbit/sです。

5.5.3 MOST150のCMS

MOST150はMOST50に似ています。MOST150のフレームも、1フレームにつき4バイトの制御データをヘッダ内に格納します。また、制御メッセージの長さもMOST50同様に個々のメッセージにより異なります。ペイロードのないコマンドの場合にフレーム数は最小の6となり(TelLen = 0)、最大フレーム数は18 (TelLen = 45)です。図5.15に、制御メッセージの構造を示します。ペイロードが45バイトを超えるCMSメッセージは複数のセグメントに分割して伝送し、最大長は65535バイトです。制御チャンネルでのMOST Highプロトコルは MOST150ではサポートされません。

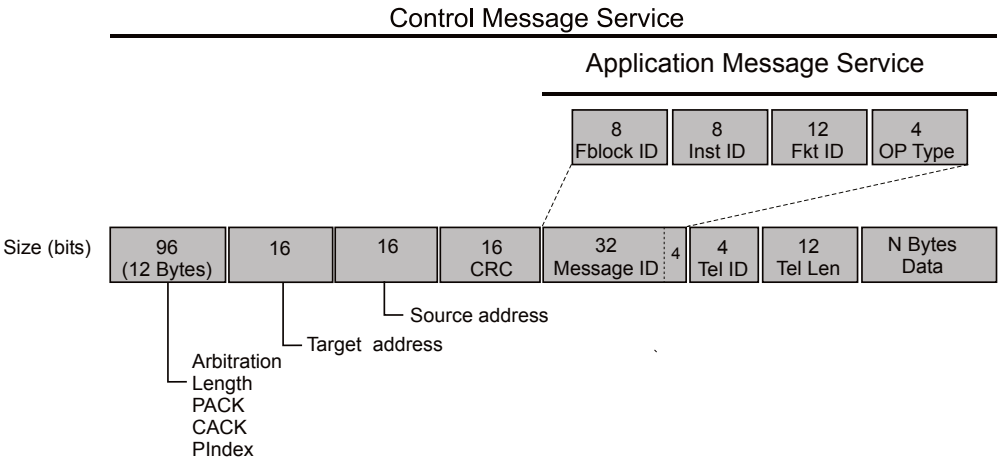


図5.15:MOST150の制御メッセージの構造

MOST150のヘッダは、MOST50のヘッダにPIndexを追加したものです。PIndexは、特定のノードが制御メッセージを1つ送信するたびに増分します。ローレベル再送の場合、この値は変化しません。

メッセージIDは、AMSパラメータのFBlockID、InstID、FktID、OPTypeから成ります。セグメンテーション エラーの場合、AMSはメッセージIDの最後の4ビットを0x0Fに設定します。

1フレームのデータレート

1秒当たりのメッセージ数は、下式で求めます。

$$CM = \frac{Fs}{number\_of\_frames} \qquad \text{(式5.8)}$$

$$CM_{\max} = \frac{48000}{6} = 8{,}000@48kHz; TelLen = 0$$

$$CM_{\min} = \frac{48000}{18} = 2{,}666@48kHz; TelLen = 45$$

どのメッセージにもペイロードがない場合(TelLen = 0)、1秒当たりの制御メッセージ数は最大値の8,000です。全てのメッセージのペイロードが最大サイズ(TelLen = 45)の場合、2,666メッセージ/秒です。

ペイロードの最小値($Payload_{min}$)は8バイト (メッセージID、TelID、TelLenの6バイトとターゲット アドレスの2バイト)で、ペイロードの最大値($Payload_{max}$)は53バイト($Payload_{min}$ + 45バイトのデータ)です。従って、実質レートは下式で求めます。

$$\begin{aligned} DR_{gross\_\min} &= CM_{\max} * payload_{\min} * 8Bits \\ &= 8{,}000 * 8 * 8Bits \\ &= 512kbit/s@48kHz; TelLen = 0 \end{aligned}$$

$$\begin{aligned} DR_{gross\_\max} &= CM_{\min} * payload_{\max} * 8Bits \\ &= 2{,}666 * 53 * 8Bits \\ &= 1{,}130kbit/s@48kHz; TelLen = 45 \end{aligned}$$

MOST150の理論上の実質データレートは最小で512 kbit/s、最大で1,130 kbit/sです。

MOSTデバイスはMOSTネットワークで利用可能な制御メッセージ帯域幅の全てを利用する事はできません。これは通常、I/Oインターフェイス、デバイスドライバ、ネットワーク スタック、アプリケーションによる制限に起因します。

5.6 まとめ

	MOST25	MOST50	MOST150
フレームサイズ (ビット)	512	1024	3072
サンプルレート(kHz)	44.1	48	48
ストリーミング データ			
最小値(バイト)	24	0	0
最大値(バイト)	60	117	372
最小帯域幅 (Mbit/s)	8.467	0.384	0
最大帯域幅 (Mbit/s)	21.168	44.928	142.848
パケットデータ			
最小値(バイト)	0	0	0
最大値(バイト)	36	116	372
最小帯域幅 (Mbit/s)	0	0	0
最大帯域幅 (Mbit/s)	10.841[1]	44.544	142.848
制御データ			
1フレーム当たりのバイト数	2	4	4
最小フレーム数	16	6	6
最大フレーム数	16	9	18
最小データバイト数	19	7	8
最大データバイト数	19	19	53
最小総帯域幅(kbit/s)	405.84	448	512
最大総帯域幅(kbit/s)	405.84	810.62	1130

表5.6:MOST25、MOST50、MOST150のフレーム パラメータのまとめ

[1] 計算式は補遺B参照。

5.7 アドレッシング

ISO参照モデルの第2層で、MOSTネットワークはデータリンク層プロトコルと制御チャンネルに関する以下の5つのアドレッシング モードをサポートしています。

- 内部ノード通信アドレス (ノードの内部通信用に予約済み)
- ノード位置アドレス(RxTxPos)
- 論理ノードアドレス(RxTxLog)
- グループアドレス
- ブロードキャスト アドレス
- Ethernet MACアドレス(MOST150のみ)

Ethernet MACアドレスは長さが6バイトで、それ以外は全て16ビットアドレスを使います。表5.7に、アドレスレンジの一覧を示します。アドレスの上位4ビットは現時点では未使用で、将来のアプリケーション用に予約済みです。

アドレスレンジ	モード
0x0000～0x000F	内部通信
0x0010～0x00FF	静的アドレスレンジ
0x0100～0x013F	動的に求められた(0x0100 + POS)アドレスレンジ
0x0140～0x02FF	論理アドレスレンジ(静的に割り当て)
0x0300～0x03C7	グループアドレス
0x03C8	ブロッキング ブロードキャスト アドレス
0x03C9～0x03FE	グループアドレス
0x03FF	ノンブロッキング ブロードキャスト アドレス
0x0400～0x043F	ノード位置(0x0400 + POS)アドレスレンジ
0x0440～0x04FF	予約済み
0x0500～0x0FEF	論理アドレスレンジ(静的に割り当て)
0x0FF0	任意選択のデバッグアドレス
0x0FF1～0x0FFD	予約済み
0x0FFE	ネットワーク サービスのInitアドレス
0x0FFF	MOSTネットワーク インターフェイス コントローラのInitアドレス
0x1000～0xFFFE	将来用に予約済み
0xFFFF	初期化されていない論理ノードアドレス

表5.7:MOSTネットワークのアドレスレンジ[MOST 3.0]

ノード位置アドレス(RxTxPos)

MOSTネットワークの各ノードには、リング内の位置に基づいた一意のIDがあり、リンク層で動的に確認されます。このIDをノード位置と呼びます。TimingMasterの

ノード位置は常に0x00で、以降のノードには0x01から順にノード位置が割り当てられます(図5.16参照)。1つのリングの最大ノード数は64です。

信号ソースから信号シンクまでの遅延は、ノード位置を利用して簡単に求める事ができます。これは、ソースが自分のノード位置をシンクに通知するので、シンクは信号が通過してきたノード数を判断できるためです。

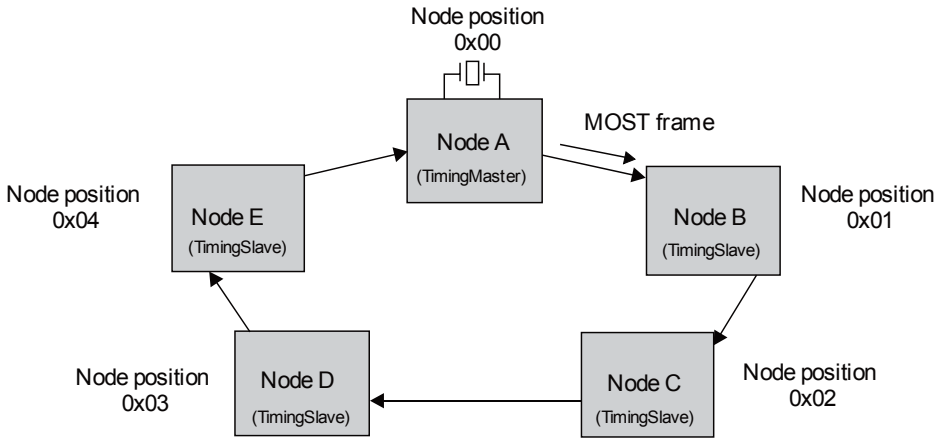


図5.16:MOSTリングとノードアドレス

ノード位置アドレスはノード位置から生成され、ネットワーク内で一意に決まります。システムを起動し、MOSTネットワーク インターフェイス コントローラがロック状態になり、バイパスを開くとただちにTimingMasterがノード位置アドレスを定義します。ノード位置アドレスのレンジはアドレス0x400から開始し、TimingMasterは常にアドレス0x400を使います。ノード位置アドレス(RxTxPos)は、下式で求めます。

RxTxPos = 0x0400 + Pos

1つのリングの最大ノード数は64であるため、最後のアドレスは0x043Fです。残りのアドレスは現時点では未使用です。

論理ノードアドレス(RxTxLog)

論理ノードアドレス(RxTxLog)は、ネットワーク内で一意のアドレスです。電源投入後、全てのノードは初期化前のアドレス0xFFFFとなります。ネットワークの初期化後、ノード位置アドレスと同様に下式に基づいて論理ノードアドレスが動的に割り当てられます。

RxTxLog = 0x100+Pos

TimingMasterの論理ノードアドレスは0x100で、以降のノードには0x101から順にアドレスが割り当てられます。

あるいは、アドレスレンジ0x140～0x2FFと0x500～0xFEFの中からシステム設計者が静的に論理アドレスを割り当てる事ができます。このアドレスは、ノードの機能(例: 最初のビデオモニタが0x200、2番目のビデオモニタが0x201)を指し示す事も、自動車内の位置(例: 車室左が0x100、車室右が0x200、トランクが0x500)を指し示す事もできます。

NICでは、論理アドレスを2つのレジスタ(bNAH、bNAL)に格納します。INICはNodeAddressファンクションを使います。NICは、このアドレスを受け入れる前に、割り当て済みアドレスが一意のアドレスかどうかをstart-address-initializationファンクションを利用して自動的にチェックします。

グループアドレス(GA)

グループアドレスは、アンプ、オーディオCDプレーヤ、デジタルサウンド プロセッサ等のシステムクラスへの割り当てを表します。

グループアドレスのアドレスレンジは0x300～0x3C7と0x3C9～0x3FEです。最初に割り当てられるグループアドレスは、下式に基づいてファンクション ブロックIDから求めます。

GroupAddress = 0x300+FBlockID

ここでは、そのノードの最も特徴的な(すなわち最も頻繁に要求されると予想される)FBlockのFBlockIDを使います。

NICでは、グループアドレスをbGA (Group Address)レジスタに格納します。INICはGroupAddressファンクションを使ってグループアドレスを変更します。

ブロードキャスト

ブロッキング ブロードキャスト アドレスは、リング内の全てのノードをアドレス指定する一意のアドレス(0x3C8)です。全てのノードがブロードキャスト受信に対するACKを返すまで、その他の全ての制御メッセージをブロックします。このため、ブロッキング ブロードキャスト アドレスを多用すると帯域幅を圧迫します。

ノンブロッキング ブロードキャスト メッセージ(0x03FF)は、他の制御メッセージの送信をブロックしません。この種の送信は、必ずしも全てのデバイスが受信する必要がない重要性のないデータ送信に使います。

ブロードキャスト メッセージはシステムリソースを大量に必要とするため、初期化の際に全てのノードから対応するファンクション ブロックを受信する場合等、管理機能にのみ使ってください。車両で使っているコンポーネントの全てのハードウェアおよびソフトウェア バージョンを問い合わせる場合にもブロードキャストメッセージを使います。

グループアドレスを使う場合、ブロードキャスト メッセージの最大データ長はチェックサム用の1バイト分少なくなります。

Ethernet MACアドレス

Ethernet MACアドレスはEUI-48形式の一意のアドレスで、長さは48ビット(6バイト)です。Ethernet MACアドレスをサポートするのはMOST150ネットワークのみです。各メーカーは、EUI-48アドレスとMACアドレスをビルドするために、独自のOUIを取得する必要があります。

このアドレスはEthernetに特有のアドレス比較モードをサポートしています。

- ユニキャスト(完全フィルタ処理、48ビット一致)
- マルチキャスト (ハッシュフィルタ処理)
- ブロードキャスト(0xFFFF FFFF FFFF)
- プロミスキャス (フィルタ処理なし)

モードはMOSTネットワーク インターフェイス コントローラで設定します。

5.8 上位プロトコル

MOST仕様は、OSI参照モデルのトランスポート層レベルでコネクション型通信によってパケットデータを送信するためのMOST HighプロトコルとMAMAC (MOST Asynchronous Medium Access Control)アダプテーション層を定義しています。これにより、TCP/IPスタックをデータリンク層プロトコルに接続できます。これらのプロトコルを利用すると、ナビゲーション データ等のパケットデータを個々のフレームで送信する事も、セグメントとして送信する事もできます。MOST150はEthernet通信に完全に対応しているため、MAMACはサポートしていません。

5.8.1 MOST Highプロトコル(MHP)

MOST Highプロトコルはコネクション型プロトコルで、TCPプロトコルのメカニズムの一部を使います。ただし、データフロー制御のオーバーヘッドは最小限に抑えられています。図5.17に示すように、このプロトコルはネットワーク サービス層Iに接続されています。MHPはパケットデータ チャンネルに実装でき、単方向、すなわちトランスミッタ モジュールとレシーバ モジュールが1つずつあります。双方向でデータを送信する場合、2つの接続が必要です。

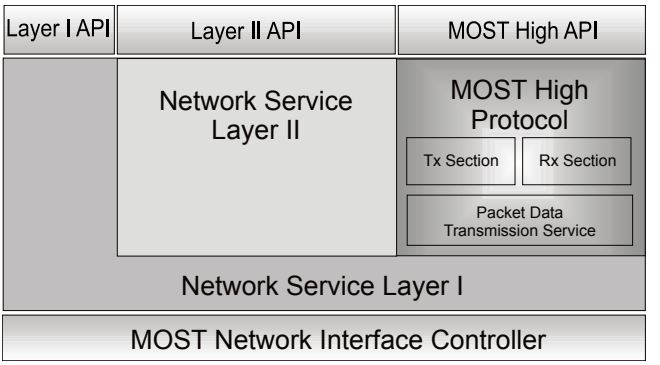


図5.17:MOST Highプロトコルの代表的な階層構造(出典:[PFLNet])

アプリケーションからの視点

アプリケーションから見ると、MOST Highプロトコルはネットワーク サービスとMOSTデバイスのファンクション ブロックの間に位置します(図5.18参照)。コントローラは、レシーバが持つファンクション ブロック内の1つのファンクションにアプリケーション データを送信します。コントローラとMOST Highプロトコルの間のインターフェイスは、ファンクション ブロックの制御メッセージのインターフェイスに対応しています。

```
DeviceID.FBlockID.InstID.FktID.OPType(Data)
```

このプロトコルはコネクション型であるため、通信は常に以下の3つから成ります。

- 接続確立
- データ送信
- 接続終了

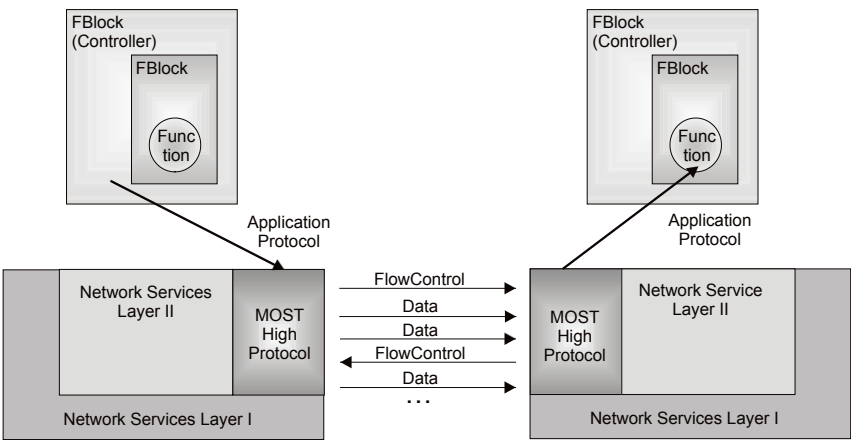


図5.18:アプリケーションから見たMOST Highプロトコル(出典: [MHP])

また、レシーバはデータを正しく受信するとACKを送信してトランスミッタに通知します。データの再送を要求する場合はNAKを送信します。これをフロー制御(FlowControl)と呼びます(図5.18参照)。

	制御チャンネル上のMHP	パケットデータ チャンネル上のMHP		
	(レガシーのみ)	48バイトのデータリンク層プロトコル	1,014バイトのデータリンク層プロトコル	1,512バイトのデータリンク層プロトコル
使用可能なデータバイト	17	48	1014	1524
ヘッダ長(バイト)	5	6	6	6
フロー制御(バイト)	1	2	2	2
1フレーム当たりの最大データフィールド長(バイト)	11	40	1006	1518
1ブロック当たりの最大ユーザフレーム数	15	255	255	255
1ブロック当たりのバイト数	1～165	1～10200	1～65535	1～65535

表5.8:制御チャンネルで送信した場合およびパケットデータ チャンネルで送信した場合のMHPのパラメータ

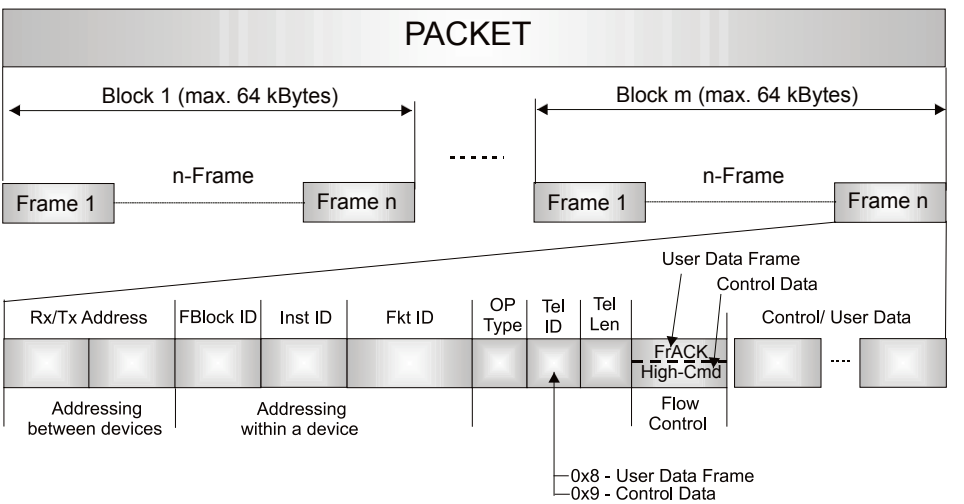


図5.19:MOST Highプロトコルの構造(出典:[MHP])

プロトコル

MHPはパケット、ブロック、フレームを区別します(図5.19参照)。パケットは多数のブロックから成ります。ブロックは最大256フレームから成ります。1ブロックを構成するフレームの数およびフレームの最大データフィールド長は、制御チャンネルに実装する(レガシーのみ)かパケットデータ チャンネルに実装するかにより異なります(表5.8参照)。ブロックのサイズは常に64 KBまでに制限されます。

High-Cmd	コマンド名	説明
CA	接続要求	データの存在、接続の優先度、フレーム当たりの利用可能な最大データバイト数、MHPのリビジョン番号をセンダから通知
F2	接続開始	接続を確立するためのコマンド
F3	センダからの接続終了	センダからの接続終了要求
FC	センダからの接続終了	レシーバからの接続終了要求
FA	フレームACK	フレームまたはブロックの受信に関するレシーバからのACK
FB	要求	特定の1フレームまたはブロック全体の送信要求
FF	複数フレームの要求	現在のブロックの複数のフレームの送信要求
F0	レート調整	送信レートの増減に対するレシーバからの要求
FD	データ受信準備完了	接続の確立に対するACK
F1	センダからの接続保留	送信バッファがエンプティの場合等、センダから通信の中断/継続を示す
FE	センダからの接続保留	受信バッファがフルの場合等、レシーバから通信の中断/継続を示す

表5.9:MHPフロー制御のコマンド(出典:[MHIGH])

フレームは、制御メッセージのヘッダと同じ構造のデータリンク層ヘッダ、およびトランスポート層ヘッダから成ります。フロー制御メッセージ(制御データ)の場合、トランスポート層ヘッダはコマンドトークン(High-Cmd)から成ります。データフレーム (ユーザ データフレーム)の場合、トランスポート層ヘッダはフレームACK (FrACK)から成ります。図5.19に、フレームの構造を示します。

表5.9に、フロー制御コマンドの一覧を示します。

ACKは各ブロックの1フレームごとに送信できます。図5.20に、毎ブロック後ACKを送信した場合のMHPのシーケンス図を示します。個々の手順の内容は、表5.9を参照してください。0-FRAMEはブロックの最初のフレームです。

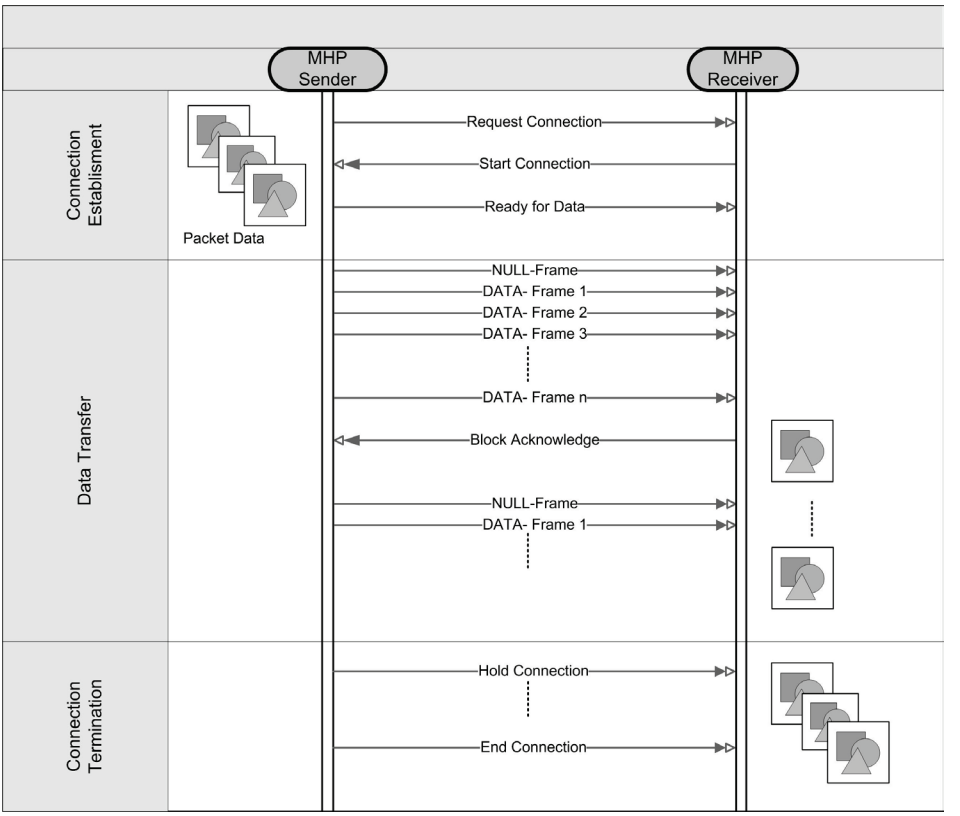


図5.20:MHPの制御およびデータフロー

その他の情報および個々のコマンドに関するMSCについては、[MHIGH]を参照してください。

5.8.2 Ethernet over MOST

MOST Ethernetパケット

MOSTパケットデータ チャンネル(PDC)は、非同期パケットデータの送信に使います。PDCは以下の2つのアドレッシング形式をサポートしています。

- 16ビット アドレッシング: MOSTデータパケット (MDP)
- 48ビット アドレッシング: MOST Ethernetパケット (MEP)

MDPおよびMEPパケットは互いにパケットデータ チャンネルの帯域幅を共有します。このため、MOSTネットワーク内の全てのデバイスがこれらのパケットを同時に送信できます。全てのデバイスがチャンネルに公平にアクセスできるよう、調停メカニズムで保証しています。PDCの帯域幅は、境界ディスクリプタを調整して動的に変更できます。このプロトコルはCRCで保護され、レシーバがCRCエラーを検出した場合、または受信バッファがフルの場合はパケットが自動的に再送されます。

MEPのフォーマットは48ビットMACアドレッシングをサポートしており、IEEE Ethernet規格に準拠しています(表5.10参照)。

MACデスティネーション アドレス	MACソース アドレス	データ	CRC
48ビット	48ビット	最大1,506バイト	32ビット

表5.10:48ビットMACアドレスの構造

データフィールドには、任意で32ビットのVLAN (IEEE 802.1Q)フィールドを含める事ができます。16ビットの長さ/タイプフィールドは必須です。

INICはMACアドレスを格納し、アドレスのフィルタ処理を実行します。つまり、EHC上で動作しているアプリケーションに関係のあるパケットのみをEHCへ渡すようにフィルタを設定できます。48ビット アドレッシング フォーマットは、標準のIEEE Ethernet MACアドレスフィルタ モードとして以下のものをサポートします。

- ユニキャスト(完全フィルタ処理、48ビット一致)
- マルチキャスト (ハッシュフィルタ処理)
- ブロードキャスト(0xFFFF FFFF FFFF)
- プロミスキャス (フィルタ処理なし)

MEPパケットは、パケットデータ チャンネルで標準のIEEE Ethernetフレームを送信します。従って、標準のTCP/IPスタックを一切変更なしにMOST Ethernetドライバ上で使えます。

Note:MEPデータをMOST Ethernetドライバへ移動し、MDPデータを標準のMOSTネットワーク サービス フレームワークへ移動するには、ローレベルのフィルタメカニズムを実装する必要があります。このフィルタが必要なのは、MDPパケット

とMEPパケットが同じデータチャンネルを共有するためです。図5.21に、詳細を示します。

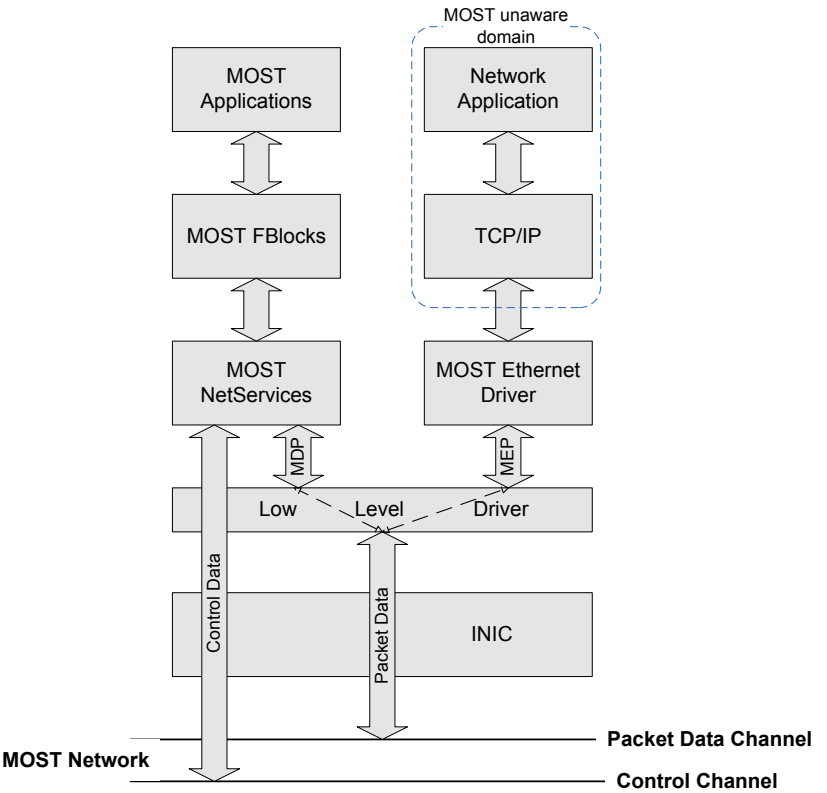


図5.21:MOST Ethernetドライバの概念図

5.8.3 MOSTアイソクロナスQoS IP(ストリーミング)

MOSTアイソクロナス転送のサブクラスであるQoS IP(ストリーミング)は、帯域幅を保証する必要があるパケットデータ(例: カメラまたはビデオサーバ アプリケーション)を送信する場合に使います。

MOSTパケットデータ チャンネルは、MOSTネットワーク上の全てのデバイスで帯域幅を共有します。従って、個々のアプリケーションに特定の帯域幅が保証される事はありません。QoS IPチャンネルは、1つのソースデバイスに対してMOSTネットワークの特定の帯域幅への排他的アクセスを保証します。このチャンネルは、一対一と一対多両方の接続をサポートします。また、このチャンネルは単方向でコネクション型です。接続は通常、接続マネージャがセットアップします。接続マネージャ

はソースに対してMOSTネットワーク上での帯域幅を割り当てるよう指示し、1つまたは複数のシンクに対してこの特定のチャンネルに接続するよう指示します。

MOSTに対応したネイティブ アプリケーションはQoS IPチャンネルを認識できるため、QoS IPチャンネルを直接使います。

5.8.4 QoS IPを利用したMOST Ethernet

MOST Ethernetドライバは、パケットデータ チャンネルだけでなくQoS IPチャンネルでもMPEパケットを処理できます。このドライバは1つまたは複数のQoS IPチャンネルをセットアップし、QoSタグの付いたパケットを全てのターゲットに送信します。QoSタグの付いていないその他全てのパケットは、共有パケットデータ チャンネルで送信されます。この方法を使うと、標準のTCP/IPスタック、IPベースの通信プロトコル、ネットワーク アプリケーションは一切変更なしにMOST Ethernetドライバ上で動作できます。

QoS IPが必要な場合、アプリケーションはIPヘッダのTOSフィールドにタグを付けてデータを送信するか、予約済みポートを使うだけでQoS IPチャンネルにアクセスできます。QoSタグの付いていないパケットは常に共有パケットデータ チャンネルで送信されます。図5.22に、このメカニズムを示します。

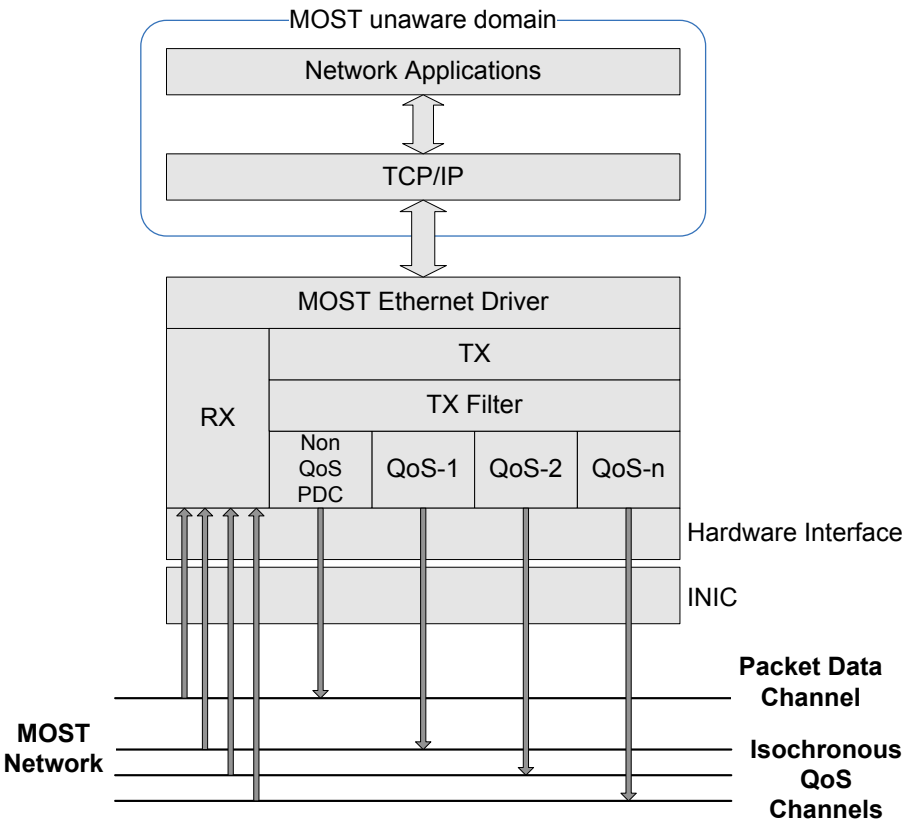


図5.22:QoSを利用したMOST Ethernet

5.8.5 MAMAC

MAMAC (MOST Asynchronous Medium Access Control)は、TCP/IPプロトコルをパケットデータ チャンネルで簡単に送信できるようにするためのアダプテーション層です。MAMACはMHPと同時に使う事ができ、TelID(図5.19参照)で区別します。MAMACは、Ethernetバージョン1と2、SNAP、LLCを直接パケットデータ チャンネルに割り当てます。MOST25とMOST50がMAMACをサポートします。詳細は、[MAMAC]を参照してください。

6 物理層

MOST物理層仕様では、光物理層と電気物理層の両方を定義しています。車載分野では何年も前から光物理層による伝送媒体が使われており、現在では広く普及しています。現在は、ポリメタクリル酸メチル(PMMA)を材料とするコア直径1 mmの光ファイバ (ポリマ光ファイバ、POF)を伝送媒体に使い、これに赤色波長レンジの発光ダイオード(LED)を用いたトランスミッタと、シリコン フォトダイオードを用いたレシーバを組み合わせています。MOSTでは、まずデータレート22.6 Mbit/sの光物理層が[MOST BaPhy]で導入されました。これを一般にMOST25と呼びます。本書の初版は、このMOST25の詳細を取り上げていました。その後、システム帯域幅に対する要求が高まり、新しい光物理層の開発が必要となりました。この結果、帯域幅を147.5 Mbit/sに拡大した追加仕様として『MOST Optical Physical Layer Sub-Specification』[MOST Phy150]がリリースされました。以下のセクションでは、MOST150と呼ばれるこの技術について説明します。信号要件に加え、ポリマ光ファイバの伝送特性と現在使われているトランシーバ技術について説明します。本章の最後では、電気および光バス物理層の最近の開発動向についての概要を示します。

6.1 概要

OSI参照モデルでは、ネットワークの一番下の階層を物理層と呼びます。つまり、物理層は2つのMOST制御デバイス(MOSTネットワーク インターフェイス コントローラ)間の物理的な接続を意味します。『MOST Physical Layer Basic Specification』[MOST BaPhy]では、このポイント ツー ポイント接続で4つの仕様点(SP1、SP2、SP3、SP4)を定義し(図6.1参照)、物理層技術の種類別に基本的な計測手法とパラメータを説明しています。

仕様点SP1とSP4は、MOST NICとコンバータの間の電気信号に関する要件(例: 電気信号レベル、電気信号の立ち上がり応答、接続タイムアウト、ジッタ要件)を定義しています。MOSTデバイスプラグとワイヤハーネス間のインターフェイス特性は、仕様点SP2とSP3で定義しています。光物理層を使ったシステムの場合、SP2とSP3におけるパラメータには、波長、光パワー、光パルスの立ち上がりおよび立ち下がり時間、MOSTデバイスプラグの物理寸法等があります。

各仕様点における個々のインターフェイス要件を詳しく見ていく前に、光ファイバと光トランシーバの基本的な特性について以下のセクションで説明します。光エレクトロニクス伝送技術の詳細は、[Gus 98]、[Vog 02]を参照してください。

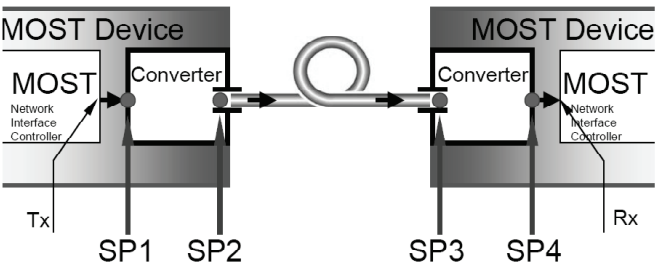


図6.1:ポイント ツー ポイント接続における仕様点SP1、SP2、SP3、SP4の定義

6.2 プラスチック光ファイバ(POF)

EMC(電磁両立性)要件の厳格化に伴い、高データレートが要求されるアプリケーションでは光ファイバが使われる機会が増えています。光ファイバを使った伝送線路は他の機器に干渉する電磁波を放射せず、他の機器からの電磁干渉(EMI)の影響を受けません。また、光ファイバはシールド電線に比べ軽量かつ柔軟であるため、取り扱いが容易です。以下のセクションでは、MOSTアプリケーションで使う光ファイバ(POF)の重要な基本原理についていくつか説明します。

6.2.1 基本原理

光ファイバは、光伝導性のあるコアを屈折率の低いクラッドで取り囲んだ構造をしています。図6.2に、光ファイバの構造を示します。光線が最大受光角以内の角度で光ファイバの端面に入射すると、光線は全反射によってコアとクラッドの界面を進んでいきます。この時、入射した全ての光線が光ファイバ内部をジグザグに進んで伝達されます。

光ファイバの端面に入射できる最大角度を最大受光角または開口数と呼びます。最大受光角φ_aと開口数NAは、下式で求めます。

$$NA = \sin\varphi_a = \sqrt{n_{core}^2 - n_{cladding}^2} \qquad \text{(式6.1)}$$

n_{core}は光ファイバのコアの屈折率、$n_{cladding}$は光ファイバのクラッドの屈折率です。コアとクラッドの材料の屈折率の差が大きいほど、最大受光角と開口数が大きくなります。

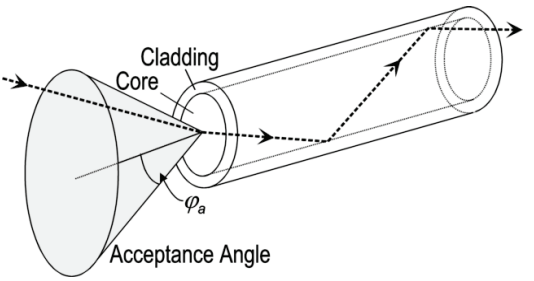


図6.2:光ファイバ内での光の伝搬

MOSTアプリケーションで使うn_{core} = 1.49、$n_{cladding}$ = 1.40の標準的なポリマ光ファイバを例にとると、NA = 0.5となり、最大受光角はφ_a = 30°です。開口数が大きいと、光の結合が容易です。これは、拡散性の高い放射特性を持つLEDを光源に使う場合には有利です。

しかし開口数が大きいとモード分散が大きくなるという短所があります。モード分散は、入射角φの異なる光線は伝搬遅延が異なるために発生します。入射角φの小さい光線の方が、入射角φの大きい光線よりも光ファイバ内を高速に進みます。モード分散の影響により、矩形パルスは光ファイバが長くなるほど歪みが大きくなります。図6.3に、モード分散の影響を示します。L_0で理想的な矩形パルスが、長さL_1の後では$\Delta_{\tau 1}$だけパルスが広がり、長さL_2の後では$\Delta_{\tau 2}$だけパルスが広がっています。光ファイバが長くなるほど、モード分散によるパルス広がりが大きくなります。

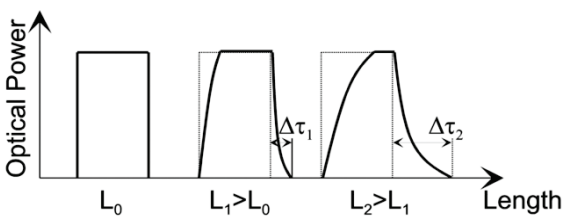


図6.3:パルス広がり

ステップ インデックス(SI)型光ファイバの場合、パルス広がりΔτは下式で求めます。

$$\Delta\tau = \frac{n_{core}(n_{core} - n_{cladding})}{n_{cladding} \cdot c_0} \cdot L \qquad \text{(式6.2)}$$

c_0は光の速度、Lは光ファイバの長さです。導波路の最大ビットレートは、パルス広がりによって制限されます。最大ビットレートは、下式で近似的に求める事ができます。

$$B = \frac{1}{\Delta\tau} \qquad \text{(式6.3)}$$

MOSTアプリケーションで使うn_{core} = 1.49、$n_{cladding}$ = 1.40の標準的なポリマ光ファイバを例にとると、長さ1 mにつき約320 psのパルス広がりが発生します。通常、自動車に用いるファイバの長さは最大20 mで、この場合の最大許容ビットレートは上の式より150 Mbit/sとなります。実際には、ポリマ光ファイバではこれよりも高いビットレートで信号を転送できます。これは、光源に放射指向性がある場合、発射角の異なる全ての光線を励起する必要がないためです。ベストケースでは角度φが小さい光線のみが存在し、モード分散が小さくなります。このため、送信可能な最大ビットレートは容易には決定できません。励起特性が指定されている場合、長さ約20 mの光ファイバで500 Mbit/sを超えるビットレートを達成できます[Dau 01]。

6.2.2 ポリマ光ファイバの特性と利点

ポリマ光ファイバのコアとクラッドについては、各種アプリケーションに適した材料特性の研究が1960年代後半から進められてきました[Wei 99]。ポリメタクリル酸メチル(PMMA)のコアの上に屈折率の低いフッ化アクリレートを押出成形したファイバは、照明用の導光体と長さ100 m未満の短距離伝送システムで使われます。短距離伝送の場合、直径1 mmのポリマ光ファイバにはガラス光ファイバに比べ以下の長所があります。

- 柔軟性が高い:
  直径が同じなら、ガラス光ファイバよりプラスチック光ファイバの方がはるかに柔軟性があります。コア直径が大きい(例: 1 mm)光ファイバは、ポリマ材料の柔軟性が高いため良好な曲げ特性を有します。
- ファイバ直径が大きい:
  ファイバ直径が大きく開口数が約0.5と大きいため、トランスミッタおよびレシーバ素子の調整誤差を比較的大きくとる事ができます。このように調整誤差の要件がそれほど厳しくないため、トランスミッタまたはレシーバ素子とプラスチック光ファイバの位置合わせが容易です。このためトランシーバ (トランスミッタとレシーバをモジュール化したもの)とプラグイン コネクタを射出成形法で低コストに製造でき、車載用途に大きなメリットがあります。

- 加工がしやすい:
  プラスチック光ファイバは材料が軟らかいためため、容易に加工できます。表面は、切断、研磨、溶融による仕上げが可能です。簡単な工具を使って短時間で加工できます。
- 汚染に強い:
  ファイバ直径が大きいため、埃や湿気の結露等でファイバ端面がわずかに汚染されても減衰量はほとんど増加しません。このため、プラスチック光ファイバは過酷な条件で利用できます。
- コストが低い:
  上記の利点により、ガラス光ファイバに比べてコストを抑えられます。システムコストは基本的にコネクタとトランシーバの価格、および設置と保守の労力で決まります。20 m未満の短距離接続、特に自動車ではポリマ光ファイバを使うと非常に高いコスト パフォーマンスが得られます。

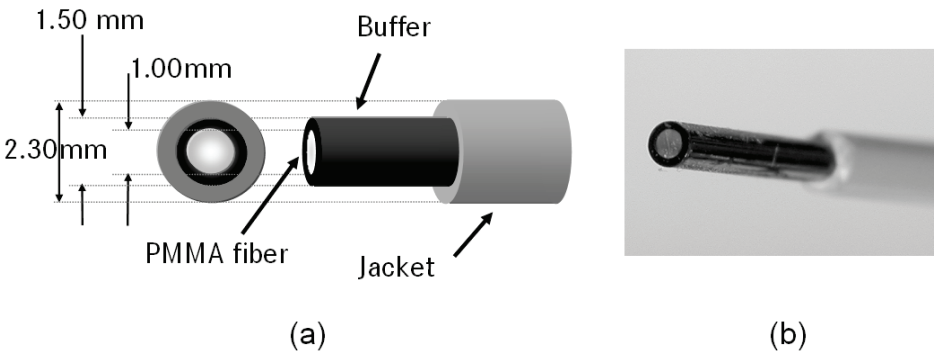


図6.4:(a)ポリマ光ファイバの構造(バッファとケーブル コーティングを含む)。
(b) MOST POFの拡大写真

図6.4に、車載要件を満たしたポリマ光ファイバの構造(保護被覆を含む)を示します。PMMAファイバは、直径980 μmのコアを厚さ10 μmのクラッド材料で覆っています。これにより、PMMAファイバの直径は全体で1 mmとなります。この光ファイバの周囲には黒いバッファ(保護被覆)があります。バッファは光ファイバに固着しており、剥離できません。光ファイバ先端には、レーザ溶着法または圧着法でフェルールがバッファに取り付けられているため(セクション6.3参照)、大きな剥離抵抗が必要です。バッファの周囲はケーブル コーティングで覆われており、物理的損傷およびその他の影響(例: 湿度、車両の作動液)から保護します。

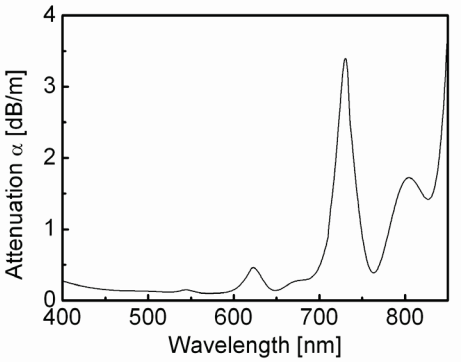


図6.5:PMMAファイバの代表的な減衰スペクトル

光ファイバの重要なパラメータの1つに、光減衰があります。通常、光ファイバの減衰量は対数表示の減衰係数α(単位: dB/m)で表します。図6.5に、波長に対するPMMAファイバの減衰を示します。標準的なステップ インデックス(SI)型PMMAファイバの減衰スペクトルには、可視波長レンジに520 nm、570 nm、650 nmの3つの極小点があります。MOSTアプリケーションは、低コストのトランスミッタとレシーバを利用できる650 nmの赤色波長レンジを使います。650 nmにおける減衰の最小値は0.14 dB/mです。

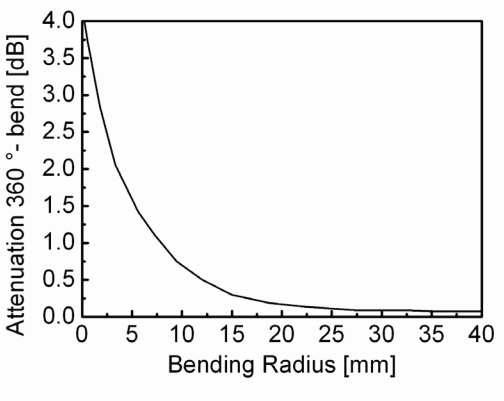


図6.6:全モード励振PMMAファイバの曲げ半径に対する代表的な減衰

標準的なPMMAハーネスファイバは85 °C対応です。このため、自動車での使用は通常車室内に限られます。MOSTへの最初の光ファイバ導入後、温度レンジの上限を105 °Cまで引き上げる事も可能でしたが、高温になる場所では熱対策を施す必要があります。さらに高温に対応するため、最大135 °Cまでの温度耐性を持つポリカーボネート光ファイバが開発されています。しかしポリカーボネート光ファイバは減衰が大きく、最小でも0.4 dB/mです[Wei 99]。

経年劣化以外で、運用中の光ファイバの減衰の要因として大きいのは、曲げによる損失の増大です。図6.6に、PMMAファイバの曲げ半径と減衰の関係を示します。曲げ半径が10 mm未満の場合、減衰は通常の状態に比べ0.5 dB以上大きくなります。曲げ半径が25 mmより大きい場合、屈曲による減衰の増大はほとんどありません。

Note:車載環境で光ファイバを使う場合、損失の増大を防ぐために通常曲げ半径を25 mm以上とします。光ファイバを製造して車両に取り付ける際も、この曲げ半径を守る事で光ファイバの損傷を防ぐ事ができます。

6.3 PMMAファイバの接合

PMMAファイバの接合技術にはいくつかの種類があります。図6.7に、その例を示します。圧着法は、金属製のフェルール(通常は真鍮製)をバッファに圧着します。超音波溶着法は、プラスチック製のフェルールを溶融してバッファに結合します。量産環境でのフェルール取り付けには、レーザ溶着法が適しています。レーザビームでバッファ材料とフェルール材料を溶融します。フェルールにはレーザビームを透過する材料を使い、フェルールとバッファの境界面が溶着点となるようにする必要があります。バッファ材料は、レーザビームの吸収性を高めるためにバッファを押出成形する際に炭素粒子を混合しておきます。レーザ溶着処理の信頼性を高めるには、レーザ溶着ビームが安定している事、そしてバッファ材料の吸収性とフェルール材料の透過性が均一である事が重要です。

図6.7:ポリマ光ファイバの接合技術「(出典:Tyco Electronics社)

圧着法による光ファイバ アセンブリ工具は一般に市販されています。図6.8の左は、光ファイバの被覆除去と切断を行う工具です。次に、図6.8の右に示した圧着工具を用いて光ファイバの先端に金属製のフェルールを取り付けます。

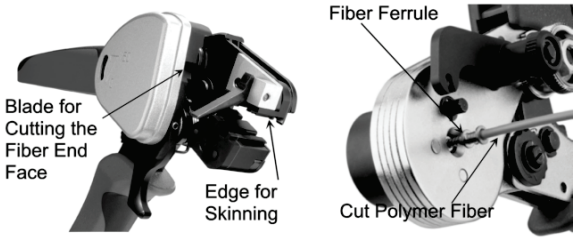


図6.8:光ファイバ アセンブリ工具(出典:Rennsteig Werkzeuge社)

量産環境向けに、レーザ溶着法でフェルールを取り付ける全自動装置が発売されています。図6.9に、このような全自動装置の例を示します。この装置ではPOFファイバを切断し、ファイバ両端の被覆を除去した後、ファイバ端面を処理してレーザ溶着法でプラスチック製フェルールを取り付け、減衰を計測します。

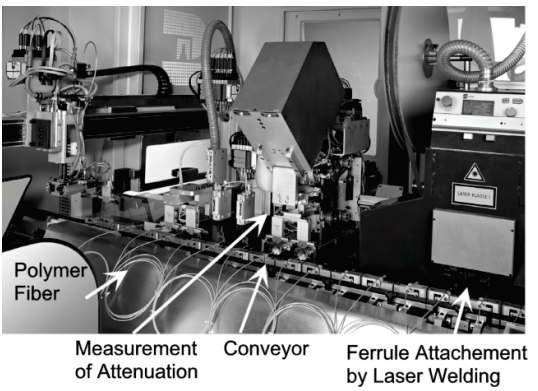


図6.9:ポリマ光ファイバの全自動アセンブリ装置(出典:Schäfer社)

アセンブリ済みの光ファイバケーブルはケーブルハーネス メーカーに納入され、そこでケーブルハーネスとして組み立てられた後、自動車メーカーに納入されます。この工程全体で、温度、湿度、曲げ半径等のストレス限界値を守る必要があります。必要に応じて、最小曲げ半径を維持する保護ホルダとダストカバーで光ファイバを保護する必要があります(第16章参照)。

インライン コネクタを使うと、ケーブルハーネスをモジュール方式で組み立てられます。図6.10にインライン コネクタの例を示します。ガイドスリーブとキャッチ機構により、2つのファイバ フェルールを確実に接続します。なお、この種の不連続点があると2つのファイバ端面の間隙、軸ずれ、フレネル反射によって損失が最大2 dB増加するため、注意が必要です。

Note:インライン コネクタを使う場合、システム設計時に2 dBの減衰を追加で考慮する必要があります(セクション6.5.3参照)。

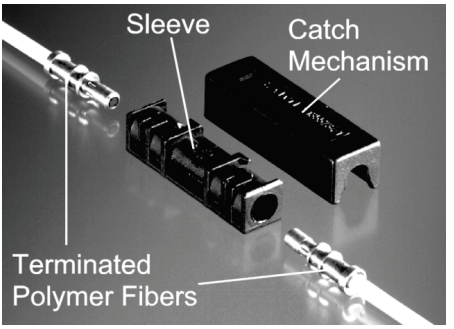


図6.10:光ファイバの不連続点に使うインライン コネクタ(出典:Tyco Electronics社)

6.4 光トランシーバ

MOSTが策定されて以来、電気信号を光信号に変換するトランスミッタには赤色波長レンジの発光ダイオード(LED)を使っています。レシーバにはPINフォトダイオードを使います。トランスミッタとレシーバの両方の機能を持つデバイスをトランシーバ(trans(mitter)+(re)ceiver)と呼びます。以下のセクションでは、LEDとPINフォトダイオードの基本的な特性について詳しく説明します。また、MOSTトランシーバの代表的なパッケージについても説明します。

6.4.1 LEDトランスミッタ

帯域幅100 MHz未満の低コストなPMMAネットワークでは、発光波長が650 nmの発光ダイオード(LED)が魅力的な選択肢です。通常、発光波長650 nmのLEDは材料に化合物半導体のAlInGaPを使います。

図6.11に、代表的なLEDの構造を示します。順方向に電圧を印加すると、キャリアが再結合して光子が発生します。発生した光子の方向は統計的分布をしているため、全ての光子が半導体結晶の表面から取り出される訳ではありません。このため、標準的なLEDの効率はわずか2～4%程度です。構造とパッケージングを工夫する事により、LEDの外部効率を大幅に向上できます。多くの場合、LED側面への発光を有効に利用するため、周囲にリフレクタを取り付けます。LEDの背面にミラー構造を使うと、基板底部に当たる光線を表面に反射できます。LED表面を粗く仕上げるかポリマ製フードで覆うと、光取り出し効率が向上します。必要な対策を講じる事で、現在では20%を超える外部効率が可能です。

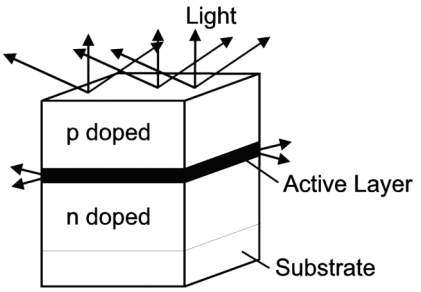


図6.11:LEDの構造

10～40 mAの順方向電流では、データ通信アプリケーション向けLEDの光パワーは通常0.1～1 mWの間です。光パワーは、光パワー1 mWを基準とした対数単位dBmで評価する事が多く、下式で求めます。

$$P_{dBm} = 10 \cdot \log\left(\frac{P_{mW}}{1\text{mW}}\right)\text{dBm} \qquad \text{(式6.4)}$$

すなわち光パワー1 mWは0 dBmに相当し、それより3 dB小さい光パワー0.5 mWは-3 dBmです。

市販されているLEDの変調帯域幅は、最大で約100 MHzに達します。この帯域幅は、活性層の不純物濃度を変える事で拡大できます。しかし不純物濃度を上げると出力パワーがさらに低下します。帯域幅を拡大するもう1つの手段としてピーキングがあります。これは、データパルス開始時の駆動電流を大きくして立ち上がり時間を短くするというものです。しかしこの方法ではオーバーシュートの挙動を制御するために複雑な制御回路が必要です。このため、帯域幅と効率を同時に最適化するのは不可能と考えられていました。

1990年代初め、共振器型LED (RCLED)が登場しました。RCLEDは、発光活性層を短い共振器に埋め込む事で効率と変調帯域幅の改善を図っています。こうして、広い帯域幅と高い出力パワーを両立した部品が実現しました。しかし構造上の理由によりRCLEDの性能は通常のLEDよりも温度の影響を強く受けるため、適切なドライバで補償する必要があります。最近では、MOSTアプリケーション向けに設計された適切なドライバとRCLEDの組み合わせ[DS FCM110]を多くのメーカーが提供するようになっています。

高効率で変調帯域幅が500 MHzを大きく超える赤色レーザダイオードはPMMAネットワークにしています。しかし現時点で入手可能な赤色レーザダイオードの許容周囲温度は約70 °Cまでに制限されており、これらのダイオードを使う場合には複雑な冷却機構が必要です。

6.4.2 PINフォトダイオード

400～1100 nmレンジの波長の光信号を電気信号に変換するレシーバとしては、低コストなシリコンPINフォトダイオードを使います。図6.12に、PINフォトダイオードの構造(左)と波長に対する受光感度R_{Ph}のグラフ(右)を示します。受光感度は、500 nmの0.32 A/Wから900 nmの0.7 A/Wまではほぼ線形に増加します。650 nmでの受光感度は約0.47 A/Wです。コア直径1 mmのPMMA光ファイバの光を効率的に変換するには、受光面積の大きいダイオードが必要です。しかし受光面積を大きくすると静電容量が大きくなり、帯域幅が減少します。最大100 MHzの帯域幅で直径1 mmのフォトダイオードを使った場合、-30 dBm未満のレシーバ感度を達成できます。

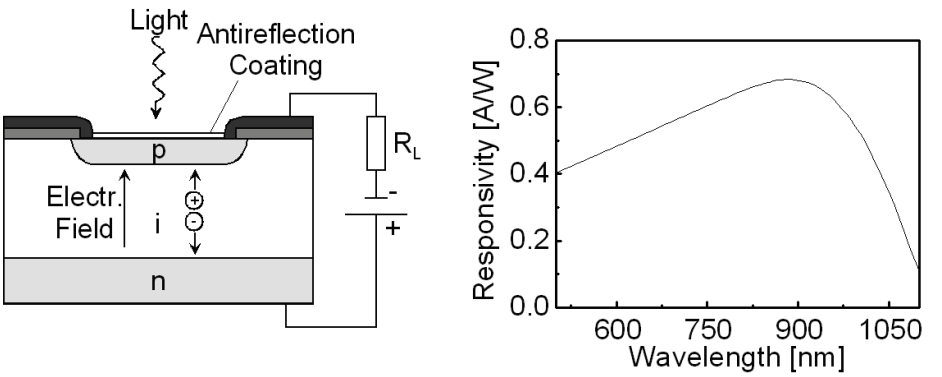


図6.12:PINフォトダイオードの構造とシリコン フォトダイオードの代表受光感度曲線

6.4.3 トランシーバのハウジングとコネクタ

光エレクトロニクス部品を組み込んだMOSTネットワーク用トランシーバ(光ファイバ トランシーバ、FOT)は、数社のメーカーが開発を手がけています。図6.13に、MOSTで使うLEDトランスミッタとPINフォトダイオード レシーバの例を示します。LEDとドライバ部品(トランスミッタの場合)、またはフォトダイオードと受信アンプ (レシーバの場合)をリードフレームに接続し、透明のプラスチック材料に封止しています。プラスチック製ハウジングから引き出された7本の電極ピンに、電源、差動データ信号、その他のステータスまたは制御ラインを接続します。『MOST Physical Layer Sub-Specification』[MOST Phy150]では、光エレクトロニクス部品をプリント基板の穴に挿入して実装する場合のためにTHM (Trough Hole Mount)パッケージを定義しています。THMパッケージの取り扱いはMOST25の頃から良く知られており、部品メーカーが適切な標準プロセスを確立しています。

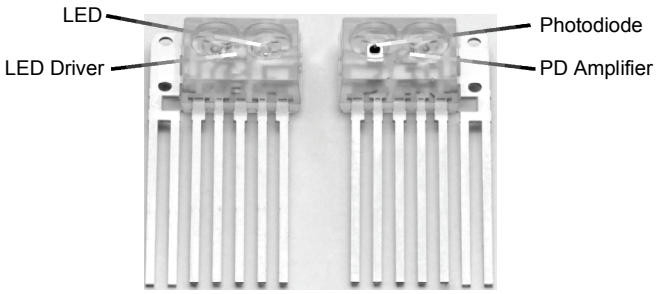


図6.13:透明パッケージに封止されたトランスミッタとレシーバ(出典:浜松ホトニクス株式会社)

電極ピンの配置はMOST仕様で定義しています。図6.14に、ピン配置を示します。トランスミッタとレシーバのどちらにも、電源ピン(V_{cc}、3.3 V)とグランドピン(GND)があります。差動データ入力(Tx+、Tx-)と差動データ出力(Rx+、Rx-)の他に、トランスミッタにはリセット(¥RST)ピンがあり、起動およびシャットダウン中に光出力のONとOFFを切り換え、光グリッチを防ぐ事ができます。レシーバのステータスピンは、レシーバがデータ信号を受信するとすぐにMOSTネットワークインターフェイス コントローラを受信可能な状態にするために使います。これらの部品の電気回路に関する詳細は、各メーカーのデータシートを参照してください。

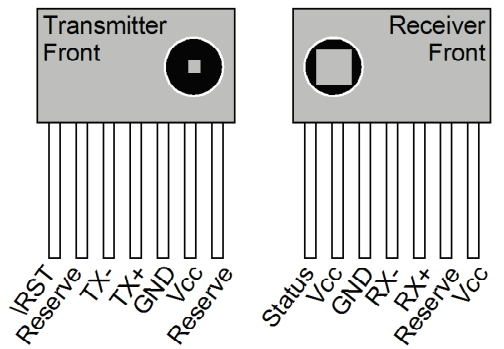


図6.14:THMパッケージのFOTの代表的なピン配置

MOST150テクノロジからは、最先端の組み立ておよびはんだ付けプロセスと互換性のあるSMDパッケージのトランシーバも利用できるようになりました。SMDパッケージはTHMパッケージに比べEMI特性が良好で、ギガビットレンジのデータレートに対応できます。さらに、SMDテクノロジではFOTとコネクタおよびピグテールを別々に購入できるため、サプライチェーン全体の自由度が向上します。光エレクトロニクス部品をFOTベンダから直接購入できるようになるため、コストを大幅に削減できます。

図6.15に、SMDパッケージ トランシーバのピン配置と製品例を示します。このトランシーバ パッケージはSOIC (Small Outline Integrated Circuit)規格に基づいています。ファイバを垂直方向に接触させるため、特別なバネ機構を備えています。ア

センブリを容易に挿入できるように、また十分な力でファイバを安定して保持できるように、2種類のバネの力を定義しています。

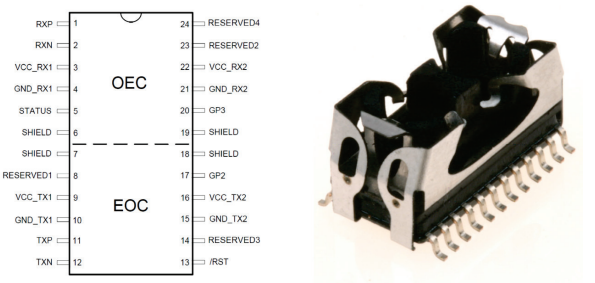


図6.15:SMDトランシーバのピン配置と製品サンプル(出典:Melexis社)

THMパッケージ トランシーバとSMDパッケージ トランシーバのどちらにも、エンドユーザにMOSTの標準コネクタ インターフェイスを提供するデバイスコネクタがあります。図6.16に、2+0および4+40タイプのデバイスコネクタの図面を示します。

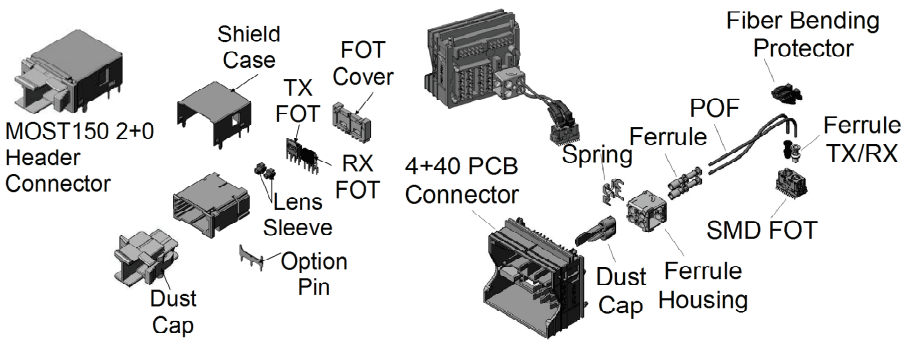


図6.16:MOST150 2+0ヘッダコネクタ(左)とMOST150 4+40ヘッダコネクタ(右)の展開図(出典:矢崎総業株式会社)

2+0、4+40等の表記は、光接点と電気接点の数を表します。図6.16の左に示した2+0コネクタには2つの光接点(トランスミッタ用とレシーバ用に各1)があり、電気接点はありません。2+0ヘッダコネクタは、THMパッケージのトランシーバをコンパクトかつ低コストでワイヤハーネスに接続できます。図6.16の右に示した4+40コネクタには4つの光接点と40の電気接点があります。SMDパッケージのトランシーバとデバイスコネクタの接続には、柔軟なピグテールを使います。柔軟なピグテールを使った場合、FOTを制御デバイスのコネクタ ハウジング外側に置く事ができます。これには、能動部品を制御デバイスの回路基板上の任意の場所に配置できる利点があります。例えばFOTをMOSTネットワーク インターフェイス コントローラの近くに配置しつつEMIの問題を防ぐ事も容易です。ピグテールは、標準の高温対応ポリマ光ファイバにトランシーバ フェルールを接続した構成です。ファイバ

の減衰、ファイバの不整合、表面粗さを含め、最大で1.5 dBまでのインサーション ロスを許容できます。ピグテール ファイバに保護ホルダを付けると、曲げ半径が小さくなり過ぎるのを防ぐ事ができます。

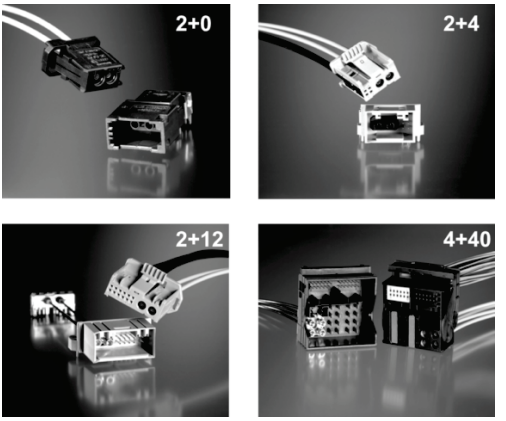


図6.17:MOSTインターフェイス用の各種コネクタ
(出典:Tyco Electronics社)

図6.17に、MOSTで使う代表的な標準インターフェイスを示します。光接点の配置はMOST仕様で定義していますが、電気接点の配置はエンドユーザが必要に応じて定義できます。MOST仕様では、制御デバイス側のコネクタ ハウジングのみを定義しています。ワイヤハーネス側のコネクタ仕様は定義していません。しかしワイヤハーネス コネクタの性質上、光インサーション ロスの最大許容値2.5 dBは守る必要があります。デバイスコネクタ寸法の詳細な定義は、『MOST Physical Layer Sub-Specification』[MOST Phy150]のインターフェイス図面を参照してください。その他、プラグの挿抜力等の要件はエンドユーザによって異なるため、MOST仕様では定義していません。図6.16に示したコネクタ以外に、MOST仕様では2+20コネクタも定義しています。

図6.18に、制御デバイスコネクタとワイヤハーネス コネクタを構成する部品の例を示します。この展開図には2+0制御デバイス コネクタと2+0ワイヤハーネス コネクタを示しており、ファイバ フェルールを内部ハウジングに接続しています。必要に応じて、ポリマ光ファイバ用保護ホルダをワイヤハーネス コネクタに取り付けると、最小曲げ半径25 mmの条件を守る事ができます。これらのインターフェイスの各部品は、ほとんどのコネクタメーカーから別々に購入できます。このため、任意の長さのワイヤハーネス コネクタをユーザ自身で容易に製作できます。修理が必要な場合は、ワイヤハーネス コネクタを簡単に個々の部品に分解できます。

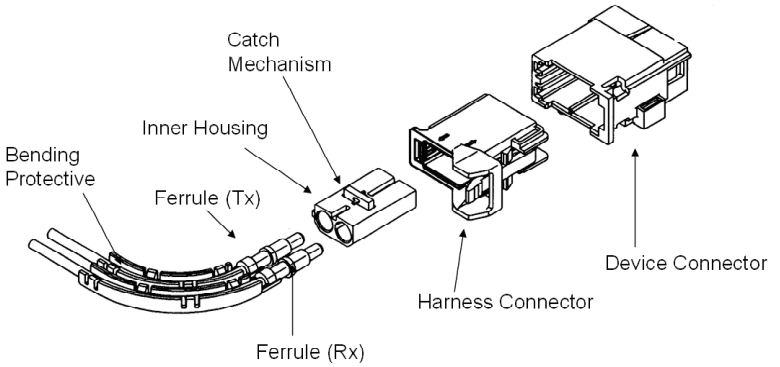


図6.18:2+0コネクタ ハウジングとワイヤハーネス コネクタの展開図(出典:矢崎総業株式会社)

6.5 MOST物理層のシステムレベルの注意事項

このセクションでは、バス物理層の電気的および光学的パラメータと要件について説明します。まず、いくつかの基本的なパラメータの定義について説明した後、具体的な仕様パラメータを解説します。最後に、パワーバジェットとタイミングについて考察します。

6.5.1 電気的および光学的パラメータの基本原理

データレートと符号化

MOST150システムのデータレートは、3072ビットのフレームサイズ(BPF = bits per frame)とフレームレート(Fs = frame synchronization)の積で決まります。一般的に使われるサンプリング レート48 kHzの場合、必要なビットレートは147.56 Mbit/sです。パルス長さの基本的な単位として用いるパルス長UI (Unit Interval)はビット伝送時間の半分と定義され、式6.5で求める事ができます。

$$UI = \frac{1}{2 \cdot Fs \cdot BPF} \quad . \qquad \text{(式6.5)}$$

フレームレートFs = 48 kHzで1フレーム当たりのビット数BPF = 3072の場合、1 UIは3.391 nsです。MOST150は、DCA (DC Adaptive)方式に基づいた符号化を行います。DCA符号化は、データに含まれる累積DCオフセットを制限する事により、DCオフセットを補償します。シンボル長は、2～6 UIの5種類を定義しています。データ内容に応じて2、3、4、5、6 UIがありますが、1 UIパルスは存在しません(図6.19参照)。1 UIのパルス長を避ける事により、必要な転送帯域幅を抑えています。6 UI

のパルスは新しいデータフレームの先頭でプリアンブルとしてのみ出現し、同期の役割を果たします。

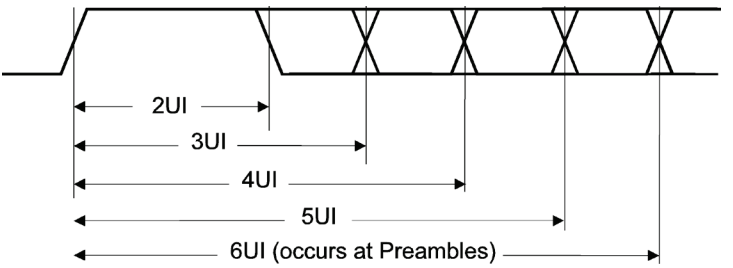


図6.19:MOST150で使うパルス長の種類

図6.20は、各種効果の影響により、理想的な矩形パルスがどのように変形するかを図で示したものです。パルス幅は、2、3、4、5、6 UIの幅からわずかにずれます。温度、ノイズ、ドライバの非直線性等により、デューティ サイクルの変動(パルス幅のばらつき)とジッタ(位相歪み)が発生します。送信経路の全ての部品の帯域幅によりパルスの立ち上がり時間と立ち下がり時間が制限され、ドライバおよびアンプ部品内の制御によって信号にオーバーシュートとアンダーシュートが生じます。

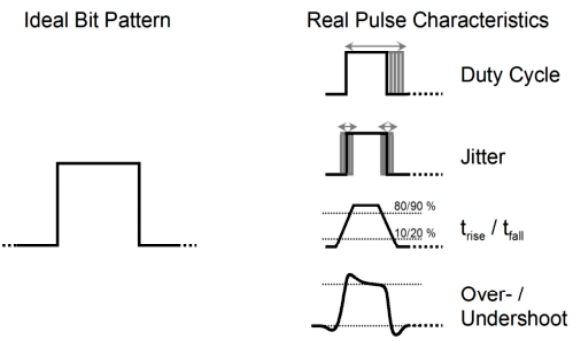


図6.20:理想のパルス形状と実際の波形

MOST仕様では、エラーレートを10^{-9}未満に抑えるため、SP1、SP2、SP3、SP4の各インターフェイスにおける理想信号からの逸脱の最大値を定義しています。タイミングが重要な信号パラメータは、多くの要因から影響を受けます。ここからは、これらの影響について詳しく説明します。

タイミング歪み

MOST仕様では、タイミングが重要なパラメータをタイミング歪みと総称しています(図6.21参照)。タイミング歪みのうち、アラインメント ジッタAjとデューティ サイクルは個々のリンクに見られる現象です。これらは、仕様点SP1、SP2、SP4におけるシグナル インテグリティをアイパターンで計測する場合に関係します。アラインメント ジッタは、リカバリ後のクロックを基準とした相対的ジッタで

す。MOST150システムでは、アラインメント ジッタは125 kHzを超える帯域幅で観察されます。これは、MOSTネットワーク インターフェイス コントローラ内の位相ロックループ(PLL)によって125 kHzまでは位相ばらつきを追跡して除去できるためです。アラインメント ジッタの計測に関して、MOST仕様ではカットオフ周波数125 kHzのローパス伝達関数の形で「ゴールデンPLL」モデルを定義しています。タイミング歪みのうち、システムクロック偏差、ワンダリング、伝送ジッタTjは、1つまたは複数のアクティブノードを伝搬するタイミング歪みに関する定義です。これらのタイミング歪みの累積がマスタ遅延許容範囲のシステムマージン以内に収まるようにする必要があり(セクション6.5.4参照)、これによってネットワーク内の最大ノード数が制限されます。NICのローパス特性により、各ノードにおけるジッタの伝送は帯域幅によって制限されます。このためNICには「ジッタフィルタ」としての効果があり、周波数200 kHzまでのジッタを除去します。

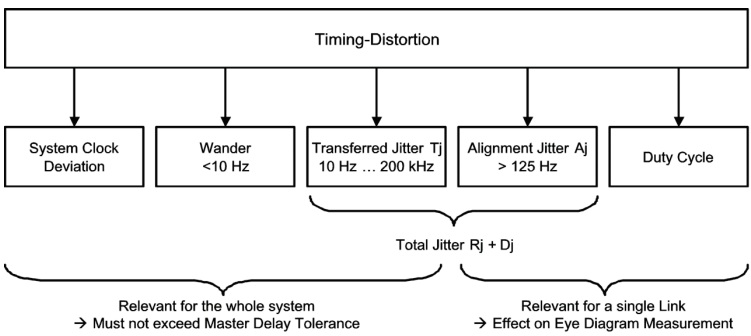


図6.21:タイミング歪みの概要

MOST25物理層仕様では、各タイミング歪みの最大値をパラメータごとに定義していました。つまり、この仕様では全ての種類のタイミング歪みに対して1つのシステムマージンが与えられている訳ではありませんでした。しかも、基本的にPLLの追跡によって除去できる低帯域幅の歪みによって計測結果がさらに悪化していました。MOST150システムでは、アイパターン計測とPLLトリガを使う事でこれらの問題を解決しています。

アイパターンとPLLトリガ

アイパターンは、1ビット期間(UI)のデジタル信号のパルス軌跡を全て重ねて表示したものです。ビット期間は、アイパターンの計測に必要なデータクロックで定義します。通常、アイパターンは参照クロックをトリガとして使って取得します。参照クロックがディスクリート信号として利用できない場合、シリアルデータ アナライザのPLLトリガ法を使って実質上の参照クロックをリカバリできます。バーチャルトリガで毎回信号をサンプリングして、パーシスタンス マップに保存します。

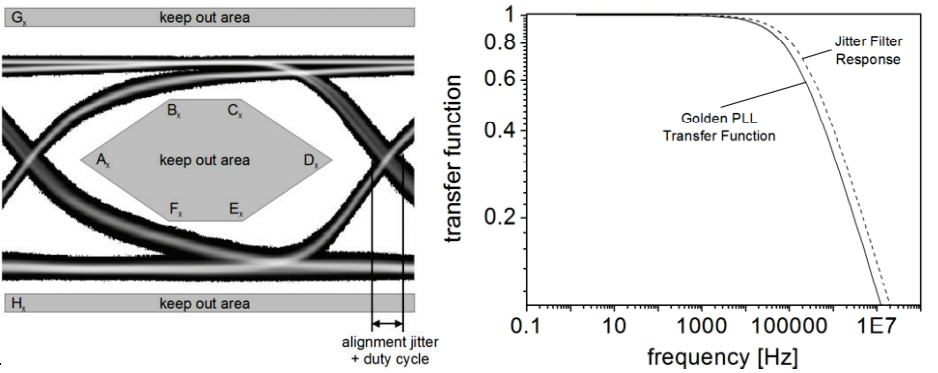


図6.22:アイパターン計測の例: アイマスク(左)とゴールデンPLLおよびジッタフィルタの伝達関数の定義(右)

図6.22 (左)は、MOST150の「ゴールデンPLL」(右の線図参照)に準拠したPLLモデルを使って計測したアイパターンの例を示したものです。アイパターン計測の利点は、全ての種類のタイミング歪みに対して1つのタイミング許容範囲を設定できる点にあります。つまり、デューティ サイクルの条件が良好ならアラインメント ジッタに対する許容量が大きくなり、ジッタの値が小さければデューティ サイクルに対する許容量が大きくなります。信号品質は、取得したアイパターンとアイマスクを比較して評価します。マスクとは、信号トレースが接してはならない違反ゾーンの事です。アイパターンの計測については、SP1、SP2、SP4にそれぞれ異なるマスクを定義しています。図6.22は、リンク品質の計測に用いる仕様パラメータのA_x、B_x、… F_xを示しています。添え字xには、計測対象の仕様点(SP)の番号(1、2、4)が入ります。SP1とSP4については、最大電圧レベルを超えないようにパラメータG_xとH_xを含めたマスク定義を考慮する必要があります。詳細は、『Optical Physical Layer Sub-Specification』を参照してください。

複数ノードで構成されるMOSTシステムでは、伝送ジッタが大きな意味を持ちます。スレーブからスレーブへ伝達されるジッタが余りにも大きくなると、信号品質の劣化によりNetworkMasterがMOST信号に正しく同期できなくなる事があります。従って、システム全体で全てのジッタおよび遅延の合計がマスタ遅延許容範囲を超えないようにする事が重要です。これについてはセクション6.5.4で詳しく説明します。図6.23に、伝送ジッタの計測手順を示します。まず波形を取得し、この波形とそれに最も近い計算で求めたクロック信号との位相偏差から時間間隔誤差(TIE)を求めます。次に、TIEのグラフをローパス ジッタフィルタ特性(図6.22の右のグラフ)で処理します。最後に、フィルタ処理したTIEグラフの2乗平均平方根(RMS)の和として伝送ジッタJ_{tr}のパラメータ値を求めます。ジッタフィルタのエッジ周波数は、予測されるPLLの最高周波数に設定し、MOSTノード間で伝達される可能性のある周波数レンジ全体を含めるようにします。波形を取得する際、低帯域幅のジッタも計測するにはメモリ容量が10Mサンプル以上のオシロスコープが必要です。例えば、メモリ容量が10Mサンプルでサンプリング レートが10 GS/sの場合、最長記録時間は1 msです。この記録時間があれば、計測する周波数を1 kHzまで下

げられます。伝送ジッタはSP1、SP2、SP4に関してRMS値として定義されています。

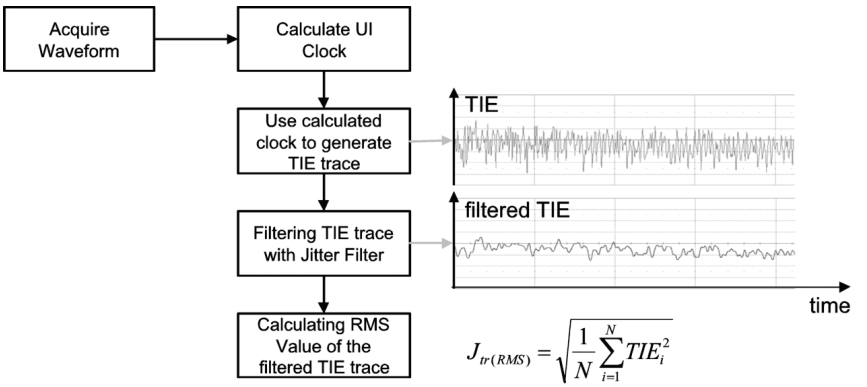


図6.23:伝送ジッタJ_{tr}の計測手順

光パワー

MOSTの部品を特性評価する際の重要なパラメータに、MOSTトランスミッタ(SP2)の光パワー、またはレシーバ(SP3)に入射する光パワーがあります。図6.24に、光パルスの波形と光パワーレベルの定義を示します。平均光パワーは、パラメータP_{AV} (Average Power)で定義します。光Highレベルb_1と光Lowレベルb_0は、それぞれ1と0の状態を表します。レベルb_0は「消灯」と同じではありません。消光比r_e = 10 log (b_1/b_0) dBにより、この仕様値は2つのレベルの間の最小比を示します。MOST150では、5または6UIのパルスの、2.5 UIと4 UIの間を計測領域としてb_0とb_1のレベルを計測するように指定しています(図6.24参照)。

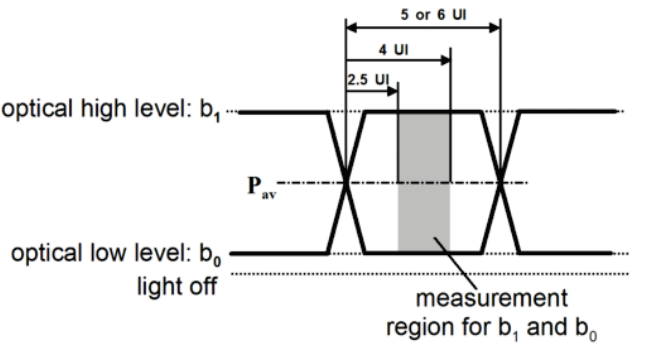


図6.24:光レベルb_1、b_0、P_{AV}の定義

計測ツール

SP1とSP4における電気信号のパラメータを計測するには、被計測信号に歪みを生じないように、帯域幅が広く(>1.5 GHz)、入力容量がなるべく小さく、入力抵抗がなるべく大きいアクティブ プローブが必要です。光信号の仕様点SP2とSP3の評価には、1 mmのPMMAファイバから十分な光を取り込めるように帯域幅750 MHz超の高速OECと受光面積の大きいフォトダイオードが必要です。また、OECは信号/ノイズ比(SNR)が8:1を超えるものを使うようにします。通常、フィールドでの計測ではシンプルな光パワーメータを使って平均光パワーP_{AV}を計測します。このような用途に向けて、多くのメーカーがポータブル型の校正済み機器を販売しています。ここでは、SP2とSP3で直径1 mmで放射される光パワーのみを開口数0.5以内で計測する必要があります。詳細は、『MOST150 oPhy Compliance Measurement Guideline』[ComPhyL 150]を参照してください。

6.5.2 光学および電気的要件

データ転送の信頼性を確保するため、『MOST Physical Layer Sub-Specification』[MOST Phy150]では全てのインターフェイス仕様点(SP1、SP2、SP3、SP4)に関してパラメータ要件を定義しています。車載分野では、最悪の条件下でも全てのパラメータ要件を満たす必要があります。どのような環境条件を考慮すべきかは、エンドユーザによって異なります。環境条件を統一するため、一部の自動車メーカーが共同で推奨事項をまとめた『Automotive Application Recommendation for optical MOST Components』 [MOST AARTHM]、[MOST AARSMD]と呼ばれる文書を作成しています。代表的な環境条件は、温度レンジ-40～+95 °C、湿度耐性最大85% (95 °C)、動作期間中の機械的耐久性約11,250時間、寿命期間15年です。MOST製品の認証に必要なテスト(温度サイクル、湿度サイクル、物理衝撃、EMC/EMI)は、[MOST AARTHM]、[MOST AARSMD]で厳密に定義されています。

SP1の電気的仕様

SP1における信号レベルは、LVDS規格で定義しています。EMIを防ぐため、信号は差動方式で伝送されます。この信号は、アラインメント ジッタ、伝送ジッタ、振幅値等、全ての必要なパラメータ条件を満たす必要があります。また、MOST150では起動およびシャットダウン処理中の安定性を高めるために新しい回路を定義しています。

SP2の光学的仕様

ピーク波長650 nmは、PMMAファイバの減衰が最も小さい赤色波長レンジに該当します。この650 nmから離れるほどPMMAファイバの減衰が大きくなるため、許容波長レンジは630～685 nmに制限されます。同じ理由により、スペクトルの幅は30 nmに制限されます(FWHM)。

正規分布の光スペクトルは、ピーク波長と幅FWHMで定義します。この定義は、明確なピーク波長を定義できない非対称スペクトルには適しません。図6.25に、LEDの対称波長スペクトルと非対称スペクトルを比較します。非対称スペクトルは、中心波長λ_{c2} とスペクトル幅(RMS) $\sigma_{\lambda 2}$を用いた別の定義で評価します。中心波長とスペクトル幅は、計測したスペクトルから求める事ができます。中心波長は、下式で求めます。

$$\lambda_{c2} = \frac{\sum_i P_i \lambda_i}{\sum_i P_i} \qquad \text{(i=500 nm…800 nm、}\Delta i \leq 1\text{ nm)} \qquad \text{(式6.6)}$$

P_iは波長λ_iにおける光パワーです。スペクトル幅は、下式で求めます。

$$\sigma_{\lambda 2} = \sqrt{\frac{\sum_i P_i (\lambda_i - \lambda_{c2})^2}{\sum_i P_i}} \qquad \text{(i=500 nm…800 nm、}\Delta i \leq 1\text{ nm) (式6.7)}$$

十分な計測精度を得るため、分光器はS/N比が200:1以上、分解能が1 nm未満のものを使う必要があります。また、中心波長とスペクトル幅を正確に決定するには、500～800 nmの波長レンジ全体のスペクトルを評価する必要があります。

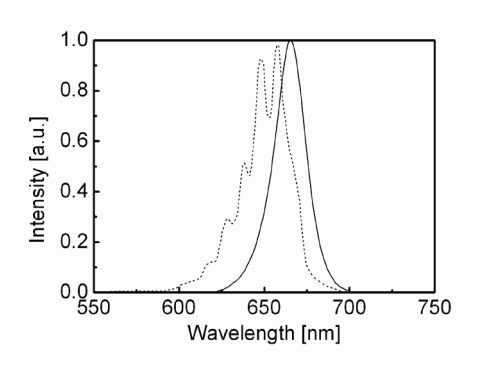


図6.25:発光ダイオードの正規分布スペクトル(実線)と非対称スペクトル(破線)

SP2における光パワーの仕様値は、-8.5～-1.5 dBmです。レシーバ側でデータを確実に受信するには、この下限値を下回らないようにする必要があります。また、レシーバの過負荷または飽和を防ぐために上限値を定義しています。パラメータP_{OFF2} = -50 dBmは、消灯状態の最大許容パワーレベルを定義します。後続のレシーバを確実にOFFにするには、このレベルを守る必要があります。レベルの切り換わる光信号が送信されていなくても、一部のレシーバ部品は最小限のD.C.光に反応し、MOSTバスがスイッチOFF手順(スリープ)を実行できない事があります。レシーバの受光感度を低下させないためには10 dB以上の消光比が必要です。

その他のタイミング パラメータ(遷移、アラインメント ジッタ、伝送ジッタ)はSP1の仕様値に比べ若干緩和されており、EOCにおける歪みに対する若干のマージンがあります。この他、SP2に定義されているパラメータにはオーバーシュート、アンダーシュート、Highレベル信号リップルがあります。これらのパラメータで、光信号のオーバーシュート量を制限します。[MOST Phy150]は、これらのパラメータをグラフでまとめた図を記載しています。

SP3の光学的仕様

SP3の光スペクトルに関する定義は、SP2と同じです。リンクのダイナミック レンジを示すパラメータP_{opt3}は、-22～-2 dBmです。

データレート約150 Mbit/sの光データバス システムでは、セクション6.2.1「基本原理」で説明したポリマ光ファイバの帯域幅の制約が非常に大きくなります。この帯域幅の制約により、光信号の垂直方向(振幅)と水平方向(パルス幅)でばらつきが生じます。MOST物理層ワーキング グループは、ポリマ光ファイバの伝達関数を決定するための調査を行っています。詳細な解析の結果、最大リンク長15 mで全モード励振ファイバの場合、-3 dBカットオフ周波数は約90 MHzと報告されています[MOST AppTF]。MOST150データパターンの最大周波数は75 MHzです。従って、システムはほぼ帯域幅の限界ですが、リンク長が15～20 m以内ならシグナルインテグリティの条件を満たす事ができます。SP3におけるワーストケースの入力信号は、SP2におけるワーストケースの信号と特定のリンク長およびモード励振条件におけるPOFの伝達関数の畳み込みで求める事ができます。

SP4の電気的仕様

仕様点SP1と同様、SP4の電気レベルもLVDS規格に対応します。計測した電気信号のアイパターンが、ジッタとLVDS電圧レベルの許容範囲を定義したマスクの禁止ゾーンに接触しないようにする必要があります。これらの条件で評価できるのは、個々のリンクの性能指標となるリンク品質であり、システム全体の特性とは関係ありません。システム全体を特性評価するには、SP4のレシーバ許容範囲を考慮する必要があります。レシーバ許容範囲の定義に適用するアイマスクは、ネットワークの各ノードにおけるジッタの累積を考慮に入れるため、リンク品質のアイマスクよりもはるかに小さくなります。

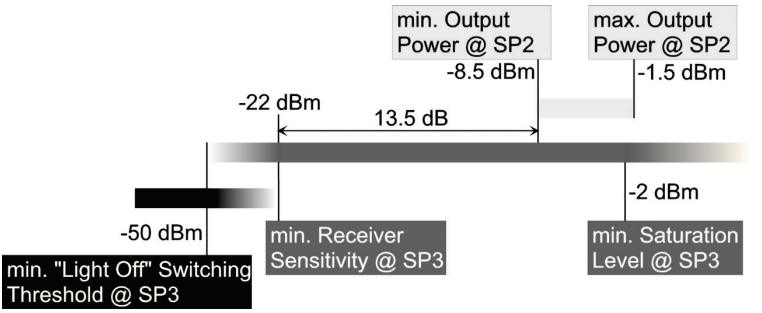


図6.26:SP2とSP3の間の光MOSTシステムのパワーバジェット

6.5.3 パワーバジェット

SP2における最小許容送信パワーとSP3におけるレシーバ受光感度で、利用可能な光パワーバジェットが決まります。図6.26に、インターフェイスSP2とSP3の間のパワーバジェットを示します。この図から、13.5 dBのパワーマージンを求める事ができます。制御デバイスのインターフェイスSP2とSP3におけるMOSTコネクタのインサーション ロスはそれぞれ2.5 dBであるため、ワイヤハーネスに与えられるマージンは8.5 dBまで減少します。

2つの制御デバイス間の光接続全体での減衰が、ワイヤハーネスのマージンである8.5 dBを超えないようにする必要があります。伝送リンクの減衰は、ファイバ自体の減衰とインライン コネクタ、曲げ、経年劣化による損失が原因です。

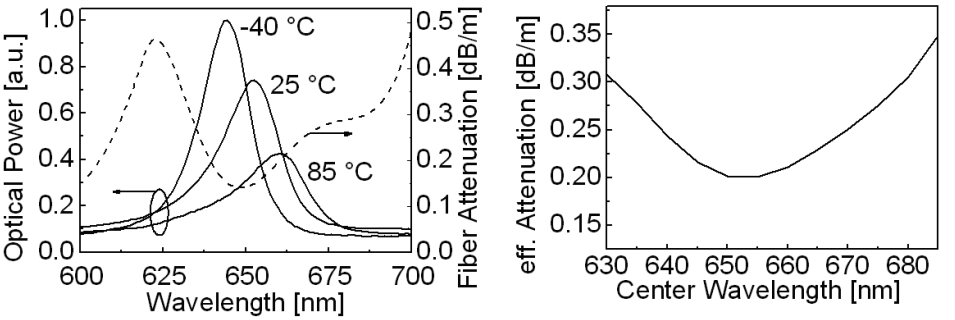


図6.27:(a):PMMA POFの減衰スペクトルと、各種温度における赤色LEDの出力パワースペクトル
(b):中心波長に対する実効ファイバ減衰。送信源のスペクトル幅はl = 30 nm

セクション6.2.2で説明したように、PMMAファイバのファイバ減衰は波長650 nmで0.14 dB/mです。しかし、温度依存性のある出力パワーとLEDの発光スペクトルはどちらも送信システムのパワーバジェットに大きく影響します。図6.27の左のグラフは、600～700 nmの間のPMMAファイバの減衰スペクトルと、各種温度におけるLEDの発光スペクトルの詳細を示したものです。この図からも明らかなように、波長650 nmにおける最小値よりも大きい減衰がLEDの大部分に適用されます。それ以外にも、光パワーは温度の上昇と共に減少し、中心波長は0.16 nm/Kずつ高波長へドリフトし、スペクトル幅が大きくなります。この波長ドリフトにより、実効ファイバ減衰α_{eff}は大きくなります。システム全体を考える際は、下式で求められる実効ファイバ減衰を考慮する必要があります。

$$\alpha_{eff} = \frac{\int \alpha(\lambda) \cdot P(\lambda)}{\int P(\lambda)} d\lambda \qquad \text{(式6.8)}$$

図6.27の右側のグラフは、上の式で求めたPMMAファイバの実効ファイバ減衰を示しています。横軸は、LEDのスペクトル幅を30 nmとして求めた中心波長です。実効ファイバ減衰は、最大で0.35 dB/mに達します。

Note:経年劣化による減衰の増大を考慮すると、PMMAファイバの実効減衰は最大0.4 dB/mとなります。パワーバジェットを考える際は、この値を使う必要があります。

実効ファイバ減衰が0.4 dB/mでインライン ファイバ コネクタ当たりのインサーション ロスが最大2 dBとすると、各種ワイヤハーネスのコンセプト、実現可能性、動作信頼性(例: アセンブリ、曲げ、経年劣化による損失)を求める事ができます。

6.5.4 タイミング要件

セクション6.5.1と6.5.2では、MOSTシステムの各ノードのタイミング要件について説明しました。SP1、SP2、SP3、SP4の各仕様点でリンク品質(LQ)の仕様値を満たしていれば、個々のポイント ツー ポイント接続は正常に動作します。表6.1に、ビットレート147.456 Mbit/s (1 UI = 3.39 ns)の場合のタイミング パラメータの仕様値を示します。位相ばらつきとジッタの発生要因として最も大きいのは、SP3とSP4の間のレシーバ インターフェイスにあるOECです。リンクレベルのテストでは、信号源としてパターン ジェネレータを使います。このパターン ジェネレータは、Optical Physical Layer Sub-Specificationの関連ファイルとしてダウンロードできるMOST150テストパターンに準拠したテスト信号を生成できるものが必要です。テストパターンの変更を防ぐため、ネットワーク インターフェイス コントローラを再タイミング バイパスモードに設定してスクランブリング メカニズムを無効にしておく必要があります。

パラメータ	SP1	SP2	SP3	SP4 (LQ)	SP4 (RT)
アラインメント ジッタ	0.15 UI	0.3 UI	-	0.55 UI	0.6 UI
伝送ジッタ	50 psRMS	112 psRMS	-	230 psRMS	-

表6.1:SP1、SP2、SP3、SP4のリンク品質(LQ)とSP4のレシーバ許容範囲(RT)に関するタイミング パラメータ (ビットレート147.456 Mbit/sの場合)

リンク品質(LQ)仕様のパラメータに加え、SP4はレシーバ許容範囲(RT)に関する上限値を定義しています。SP4のRTは、NICの最小アラインメント ジッタの許容範囲と、ネットワークの任意の場所で発生し得るアラインメント ジッタの最大許容範囲を表します。SP4のRTは、完全なリングを構成したMOST150リングから実際のデータを使ったシステムレベル テストで計測します。セクション6.5.1で説明したのと同じアイパターンの手法を使って、リング内の任意のノードのSP4で計測を行います。このようにして計測を行う事により、リング内のあらゆる位置における全ての累積ジッタを定量化できます。この計測は、ネットワーク内の全てのノードのSP4で実行できます。

実際の部品のタイミング パラメータは、製造ばらつきだけでなく、SP3で受信した光パワーと光トランシーバの周囲温度によって大きな影響を受けます。図6.28に、室温におけるSP3の光パワーに対するアラインメント ジッタの代表曲線の例を示します。この例では、ミッドレンジの光パワーでアラインメント ジッタが非常に小さくなっています。仕様限界値に近付くと、ジッタが大きくなりアイが閉じています。

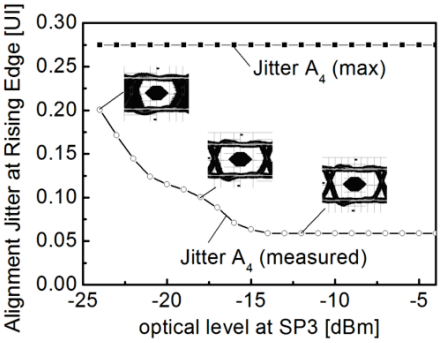


図6.28:SP3の光パワーに対するSP4のアラインメント ジッタの代表曲線

複数のノードで構成されたシステム全体をエラーなく動作させるには、全てのジッタおよび遅延の影響の合計を考慮する必要があります。MOST仕様では、これらの影響をシステム遅延として説明しています。リング内のスレーブノードで累積したシステム遅延が余りにも大きくなると、TimingMasterが入力信号に正しく同期できなくなる事があります。TimingMasterが入力信号を正しく処理できるレンジをマスタ遅延許容範囲と呼びます。マスタ遅延許容範囲は、TimingMasterモードのMOST ネットワーク インターフェイス コントローラに関する属性で、データシートに記載されています。最大値は『MOST Physical Layer Sub-Specification』で0.5/Fsと定義されています。TimingMasterとして使うMOSTネットワーク インターフェイス コントローラのマスタ遅延許容範囲によって、MOSTシステムに接続可能な最大ノード数が決まります。『MOST Physical Layer Sub-Specification』[MOST Phy150]では最大ノード数をN = 20と定義しています。これは、車載分野では十分な値です。

マスタ遅延許容範囲の合計(T_{MDT})は、下式で求めます。

$$T_{MDT} \geq t_{D\,Rx}(Master) + t_{D\,Tx}(Master) + \sum_{n=1}^{m-1} t_D(n) + \sum_{n=1}^{m} t_W(n) \ldots$$
$$\ldots + t_{D\,Medium} + \alpha \cdot \sqrt{\sum_{n=1}^{m} t_{TJ}{}^2(n)} \qquad \text{(式6.9)}$$

値mはMOSTシステム内のノード数、$t_D(n)$はノード当たりの遅延(INIC、トランスミッタ、レシーバ、媒体によって発生)、$t_W(n)$はノードおよびリンク当たりのワンダリング(位相ドリフト)、$t_{TJ}(n)$はノード当たりの伝送ジッタを示します。個々の遅延値の正確な範囲はシステムによって異なり、配線長、各種トランシーバの特性、作動温度の影響を受けます。従って、最大遅延を評価するには現実的条件下でシステムを正確に解析する必要があります。

図6.29に、3つのノードで構成されるシステムの例を挙げ、個々のジッタの定義点の位置を示します。仕様によると、各ポイント ツー ポイント接続は各ノードのSP4で最大230 psの伝送ジッタと最大0.6 UIのアラインメント ジッタを生じさせます。最初のスレーブ内にあるTimingMasterの信号がPLLを経由してSP1インターフェイスに送信されると、PLLのフィルタ特性によって伝送ジッタが減少します(<230 ps)。2番目のスレーブのSP1におけるジッタは、最初のスレーブと2番目のスレーブのそれぞれのジッタの合計です。TimingMasterのインターフェイスSP4では最後のリンクのジッタが加算されて総ジッタとなり、これがシステム遅延となります。

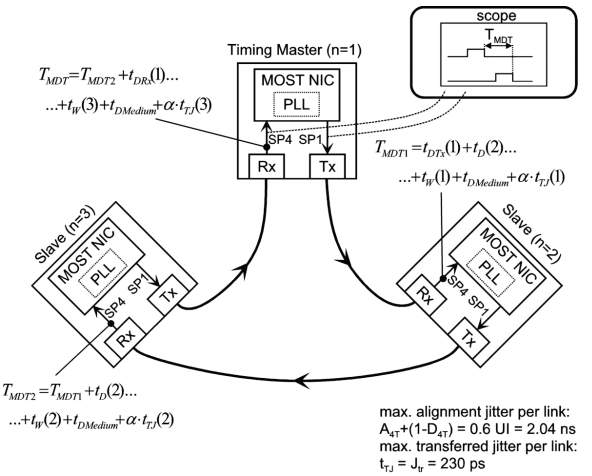


図6.29:マスタ遅延許容範囲T_{MDT}の計測点の位置

マスタ遅延許容範囲を計測するには、リング内のマスタデバイスのSP1とSP4にオシロスコープを接続します(図6.29参照)。MOST150のデータストリームは、各フレームの先頭に10 UIの期間を1つ含みます。この長い期間を、ネットワーク内の任意の2点間の遅延を計測するためのマーカとして利用できます。少なくとも1フレームをキャプチャする事により、マスタ遅延許容範囲を計測できます。サンプルレートが48 kHzの場合、この計測値が仕様限界値の10 µsを超えないようにする必要があります。

6.6 その他の伝送媒体

最初のMOSTシステムは伝送媒体にPMMAファイバを使っていましたが、物理層の改良は続けられています。この取り組みは、MOSTシステムでより高いデータレートをサポートできる電気および光伝送媒体を新たに開発する事を主な目的としています。

現時点で、車室内のポイント ツー ポイント接続として最も対費用効果が高いのは、1 mmのプラスチック光ファイバと赤色波長レンジのLEDを使った光データバスです。しかし、車載ネットワークに対する要求の高まりと共に、次世代ネットワークに向けた新しい光および電気伝送技術の開発が進んでいます。これらの技術に関する仕様は、現在MOSTのワーキング グループaoPhyが策定を進めています。以下のセクションでは、これらの技術について説明します。

レーザダイオードとPCSファイバを用いた光バス物理層

PMMA/LEDベース接続は、システムマージンが13 dB、温度レンジ上限が85 °C、帯域幅の限界が約100～200 Mbit/sという制約によって動作の限界が決まっています。この制約を解消するには、直径200 µmのガラスコアとポリマクラッドを組み合わせたポリマクラッド シリカ(PCS)光ファイバを使います。この光ファイバはファイバ減衰が小さく、温度耐性が最大125 °Cと高いのが特長です。コア直径200 µmのこのファイバは柔軟性にも優れ、車載環境で20年間の耐用年数を持ち、10 mm未満の最小曲げ半径を許容できます。垂直共振器面発光レーザダイオード(VCSEL: Vertical Cavity Surface Emitting Laser)と組み合わせると、システムマージンと帯域幅を拡大できます。

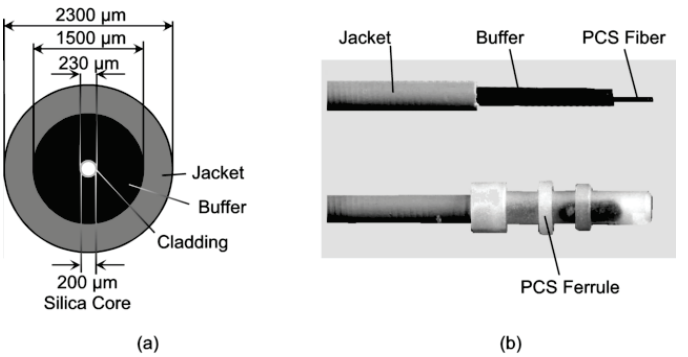


図6.30:(a):PCSファイバケーブルの構造と
(b):フェルールを取り付けたPCSファイバ

図6.30 (左)に、車載アプリケーション向けのPCSファイバのケーブル構造を示します。このファイバは、直径200 µmの溶融石英コアを厚さ15 µmの光ポリマクラッドで包んだ構造をしています。このPCSファイバはコア屈折率1.453、クラッド屈折率1.405で、開口数は0.37です。PMMAファイバと同様、この光ファイバもレーザ溶着でアセンブリできるように黒いバッファで覆われています。図6.30 (右)に、保護被覆を除去したPCSファイバとアセンブリ済みのPCSファイバを示します。PCSファイバはコア直径が小さいため、ファイバの接続点での許容誤差は小さくなります。しかし、一般的な射出成形法で適切なフェルールを製造できます。ファイバ端面の処理には、張力を与えたファイバに側面から傷を加えて切断する従来のクリー

ブ法が使えます。この方法で切断すると、平坦で非常に高品質な端面が得られます。または、CO_2レーザを使ってファイバ端面を処理できます。

図6.31に、PCSファイバの代表的な減衰スペクトルを示します。波長レンジ530～1100 nmにおける減衰は20 dB/km未満で、最小減衰が140 dB/km(波長650 nm時)のPOFに比べ、1桁以上小さくなっています。

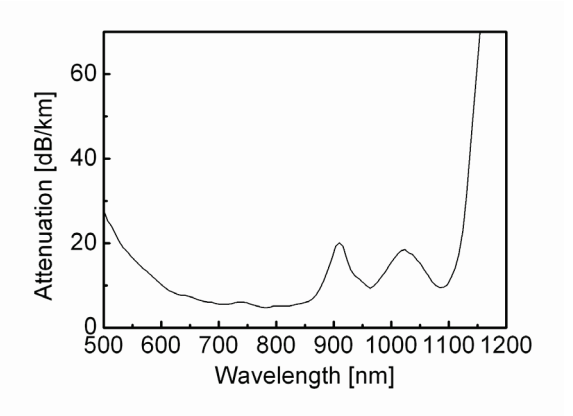


図6.31:コア直径200 µmのPCSファイバの代表的な減衰スペクトル

PCSファイバの減衰は850 nmで最小になるため、トランスミッタ素子にはGaAsを用いたVCSEL(垂直共振器面発光レーザ)が適しています。VCSELは最大でGHzレンジでの変調が可能です。VCSELは短距離のデータ通信分野で広く使われる他、マルチモード光ファイバを用いた200 mを超える距離のGigabit Ethernet通信等にも使われます。VCSELはLEDと端面発光レーザダイオードの利点を兼ね備えています。LEDと共通の特長として、量産コストが低い、ウェハーレベルでのテストが容易、構造が単純といった点があります。また、ビーム プロファイルが円対称で、ビーム幅が狭く、発散角が小さいため、PCSファイバと組み合わせた場合に高い結合効率が得られます。低しきい値電流、高効率、1 Gbit/sをはるかに超える高い変調帯域幅といった特長を持つVCSELは、簡単なドライバ回路で制御できます。VCSEL部品の信頼性はいくつかの研究で実証済みです。最大125 °Cの温度で10,000時間の寿命を達成できます。

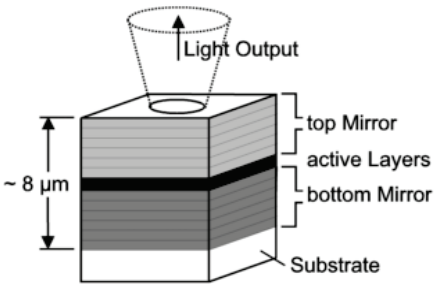


図6.32:VCSELダイオードの構造(左)

PCSファイバベースの接続を使うと、PMMAファイバベースのネットワークの場合で13.5 dBのパワーバジェットを18 dBまで増やす事ができます。表6.2に、これを示します。温度補償したVCSELを用いると、レーザカテゴリ1最小出力-9 dBmと最大出力-1.5 dBmが同時に得られます。このため、パワーバジェットは現行のPMMA接続に比べ1 dB増加します。受光表面の直径約200 µmのレシーバの感度は少なくとも-27 dBmで、これによりパワーバジェットがさらに4 dB増加します。PCSファイバのファイバ減衰は、一般的に使われるファイバ長ではほとんど問題になりません。

パラメータ	PMMA/LEDシステム	PCS/VCSELシステム
最大出力	-1.5 dBm	-1.5 dBm
最小出力	-8.5 dBm	-9 dBm
レシーバ感度	-22 dB	-27 dBm
パワーバジェット	13.5 dB	18 dB
ファイバ減衰	0.4 dB/m	0.02 dB/m
コネクタ インターフェイス損失	2.5 dB	2.5 dB
インライン コネクタ インターフェイス損失	2 dB	2 dB

表6.2:POF-LEDシステムとPCS-VCSELシステムのパワーバジェットの比較

パワーバジェットと温度耐性はPCS-VCSELシステムの方がPOF-LEDシステムより高いため、これまでのケーブル設置時の制約は適用されません。ワイヤハーネスの設置を簡単にするために、2つの制御デバイス間に少なくとも2つのプラグイン コネクタが追加で挿入されます。

PCS-VCSELテクノロジは、データレート25 Mbit/sの物理層仕様が既に完成しています[MOST BaPhy]。この仕様に合わせ、いくつかのメーカーが、VCSEL部品とフォトダイオードを内蔵した200 µmファイバ用のFOTのプロトタイプを開発しています。図6.33に、VCSELとレーザドライバを内蔵したトランスミッタのプロトタ

イプを示します。これらのデバイスをプラスチック製ハウジングに封止して、簡単に扱える低コストのパッケージとして実装しています。この図の右側に、2+0、2+4、2+40タイプのMOSTトランシーバも示しています。

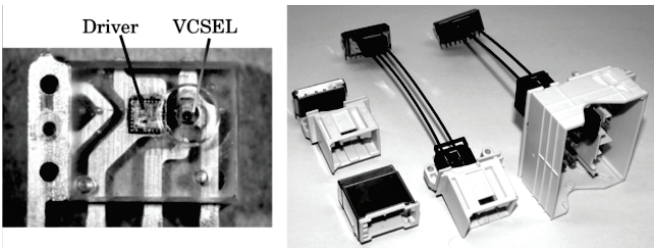


図6.33:左: 射出成形のハウジングに封止したVCSELトランスミッタとドライバ(出典:Finisar社Advanced Optical Components Division)。右: PCSファイバ用にトランスミッタをVCSELに変更したMOSTトランシーバ(出典:矢崎総業株式会社)

車載分野では部品に対する機械的許容誤差の要件が厳しく、これを満たすにはシステムコストが上昇するため、現時点でこの分野ではまだPCS-VCSELシステムは使われていません。PCSとVCSELの組み合わせが一般的に使われるようになるのは、LED-POF技術が物理的な限界に達し、データレートまたはパワーバジェットの要求の高まりに対応できなくなってからと予測されます。

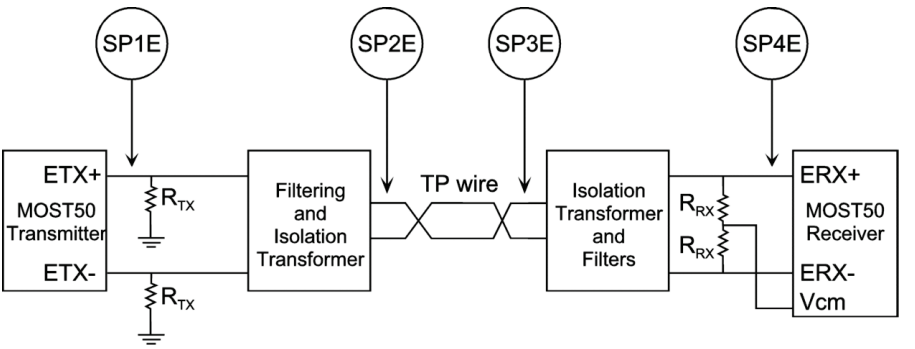


図6.34:電気バス物理層によるMOST接続図

50 Mbit/sの電気バス物理層

光システムに代わるソリューションとして、データレート50 Mbit/sのMOSTに対応した電気伝送技術の策定が現在進められています。これは、既存の電気ケーブル技術を使う事で光接続よりもコストを抑えようという目的があります。しかし電気を利用したデータラインは電磁両立性(EMC)が低いため、シールドが必要です。シールドなしでEMCの要件を満たそうとすると、複雑なドライバ回路が必要です。どちらにしても、システムコストが上昇します。このため、コストを削減できるかどう

かは主にEMCの要件が厳しいかどうか、または最悪条件でのEMC対策のコストで決まります。

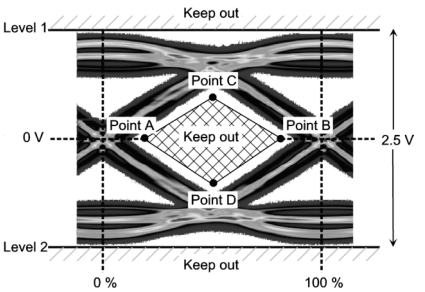


図6.35:MOSTの電気信号のアイパターンとテストマスク

電気伝送のバス物理層は、差動信号の伝送媒体であるツイストペア ケーブルと、その伝送媒体とMOSTネットワーク インターフェイス コントローラ(NIC)間のインターフェイスとなる適切なトランスミッタ(またはトランス)から成ります。光バス物理層の仕様点に相当する電気バス物理層の仕様点を、それぞれSP1E、SP2E、SP3E、SP4Eと呼びます(図6.34参照)。

電気的仕様のパラメータは、基本的にある特定の負荷抵抗と負荷容量における信号振幅と同相電圧であらかじめ決まります。また、光伝送技術の場合と同様、電気信号もタイミング要件(最大パルス幅歪みおよび位相ばらつき)の仕様値を満たしている必要があります。電気信号はアイパターンを使って評価できます。図6.35に、電気信号のアイパターンとテストマスクのパターンの例を示します。電気物理層仕様[MOST ePhy]では、SP1E、SP2E、SP3E、SP4Eの全てのインターフェイスについてそれぞれテストマスクが決まっており、レベル1、レベル2、テストマスクのA～D点を定義しています。

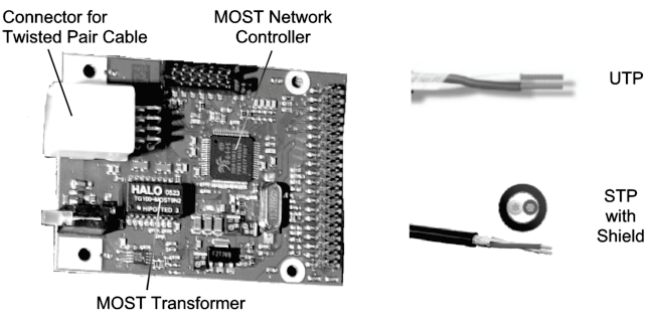


図6.36:左: トランスミッタを実装したテスト用回路基板(出典:SMSC Europe社)。右: シールド付きとシールドなしの電気伝送媒体の比較

図6.36の左側に示したテスト用回路基板は、ネットワーク インターフェイス コントローラ(NIC)、MOSTトランス、トランスミッタ(TX+/TX-)とレシーバ(RX+/RX-)の接続ソケットを実装しています。右側に、2種類の電気伝送媒体を示しています。接続には、シールドなしツイストペア ケーブル(UTP)とシールド付きツイストペア ケーブル(STP)が使えます。どちらの媒体を使うかは、必要なライン長、EMC要件、使うトランスの伝送特性で決まります。

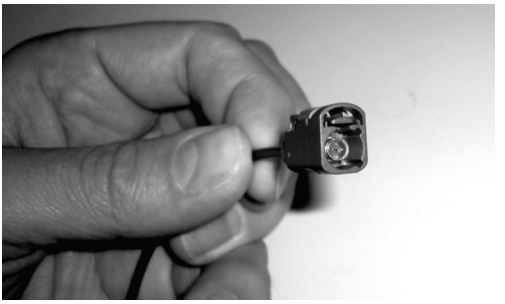


図6.37:同軸ケーブルと同軸コネクタの例

同軸ラインドライバを用いた電気バス物理層

電気物理層の新しいアプローチとして、同軸ケーブルと同軸ラインドライバを組み合わせてデータを送信する方法があります。これは、ラインドライバと車載分野で十分な実績のある標準コネクタと標準同軸ケーブルを使って光コンバータを置き換えようというものです。図6.37に、ケーブルとコネクタの例を示します。

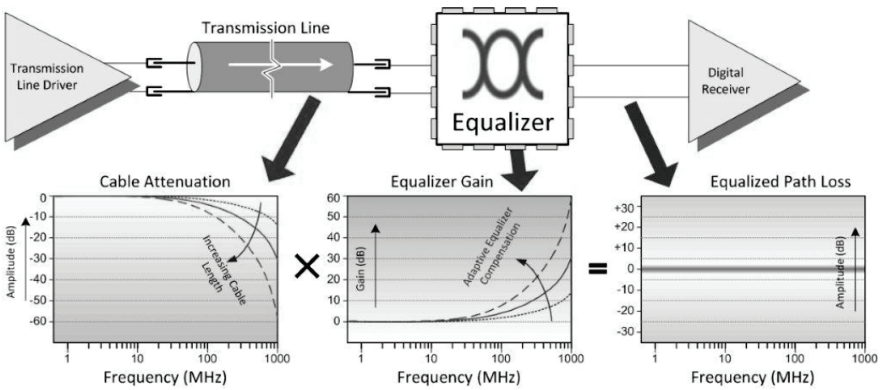


図6.38:電気伝送線路のブロック図とイコライザによるケーブル減衰の補償(出典:Microchip社)

このシステムの最も画期的な点は、1本のケーブルで双方向通信が行え、電源を供給できる事です。これにより、設置スペースの制約により小型コネクタを用いてカメラを組み込む必要のある運転支援システム等、MOSTの応用範囲が広がります。同軸ケーブルの帯域幅の制約を解消するため、レシーバ側でイコライザを使う必要があります(図6.38参照)。イコライザは、ケーブルの高周波領域における減衰を補償します。ケーブルの品質と長さの違いによるケーブル減衰のばらつきを補償するには、適応型イコライザを用いてゲインを調整するのが効果的です。物理層ワーキング グループは現在この技術の仕様策定を進めており、『MOST Physical Layer Basic Specification』の改訂版と新しい『Electrical Physical Layer Sub-Specification』をリリースする予定です。EMIおよび各種グランドレベルに対するイミュニティについてのシステム要件も、この中で検討されています。

7 ネットワークおよびフォルト管理

MOSTシステムにはいくつかのマスタ機能があり、これらはいずれもシステム全体に1つしか存在できません。これらのマスタ機能については、第2章で説明しました。この章では、ネットワーク管理を受け持つNetworkMasterについて詳しく説明します。NetworkMasterは、ファンクション ブロックNetworkMasterを実装したデバイスです。MOSTネットワーク内のその他のデバイスは全てスレーブデバイスまたはスレーブ コンポーネントと呼ばれます。システムの初期化は、起動時にスレーブ コンポーネントがNetworkMasterと情報を交換して行います。

7.1 システムの初期化

システム起動後、NetworkMasterは最初にシステムスキャンを実行し、スレーブ コンポーネントとの通信関係を確立します。システム初期化が正しく完了すると、個々のスレーブ コンポーネント間の通信関係が確立されます。

7.1.1 システムクエリ

安定ロックがあると、NetworkMasterはネットワーク内の全てのノードに対して、それぞれのファンクション ブロックの問い合わせ(クエリ)を開始します。ここでは、各ノードをノード位置アドレスで指定します。NetworkMasterとTimingMasterは同じノードに存在します。TimingMasterは常にリング内のノード0で、ノード位置アドレスは0x400、論理アドレスは0x100です。

問い合わせを受けたノードは、自分自身に格納している論理ノードアドレスと、通信可能なファンクション ブロックの情報をNetworkMasterに通知します。

```
0x0100→0x0401:NetBlock.01.FBlockIDs.Get ( )
0x0101→0x0100:NetBlock.01.FBlockIDs.Status (FBlockIDList)
```

システム内に存在するノードの数についての情報は、NetworkMasterが自分自身のNICの最大位置レジスタ(MPR)から読み出します。クエリに対する応答から、NetworkMasterは問い合わせしたノードに関して以下の情報を抽出します。

- そのノードのノード位置アドレスと論理アドレス(詳細はセクション5.7参照)
- そのノードがサポートしているファンクション ブロック(FBlockIDList)
- 各ファンクション ブロックのInstID

どのMOSTノードにもNetBlock (FBlockID 0x01)が実装されるため、少なくとも1つのFBlockが存在します。通常は、これ以外にアプリケーションFBlockが実装されます。しかしNetBlockはFBlockIDListでは送信されません。

7.1.2 セントラル レジストリ

NetworkMasterは、ノードから取得した情報をセントラル レジストリに登録します。セントラル レジストリに、全てのノードの論理アドレスと対応するファンクション ブロックを登録しますが、ノード位置アドレスは登録しません。つまりセントラル レジストリはシステムの論理マップであり、システムの動作中は常に使います。図7.1に、システムクエリの手順とセントラル レジストリへの情報登録を示します。

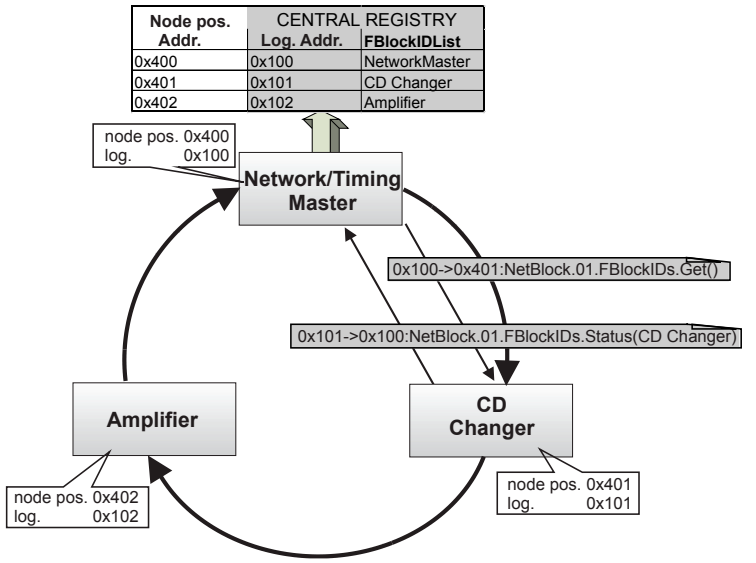


図7.1:システムクエリの手順とセントラル レジストリへの情報登録

システムステートがOKになると、リング内の全てのノードがセントラル レジストリに登録済みの情報を利用できるようになります。これ以降、NetworkMasterは直前の保存済みレジストリとの比較により、現在のネットワーク構成に生じた変化を検出できます。「保存済みレジストリ」には、前回のシステム動作時に保存したセントラル レジストリや、自動車生産ラインの最終工程でのEOL検査で指定したレジストリも含まれます。

ネットワーク ステータス

NetworkMasterは、プロトコルレベルで以下のメッセージを送信してシステム全体にシステムステートを通知します。

```
NetworkMaster.01.Configuration.Status
```

これは任意の時点での現在のシステムステートを表しており、NetworkMasterがMOSTシステム内の全てのノードにブロードキャスト メッセージとして送信します。

システムステートOK

セントラル レジストリの内容が正しい場合、NetworkMaster は Configuration.Status(Ok)メッセージを送信します。このステータス メッセージを受信した場合、全てのノードがセントラル レジストリに対して問い合わせ可能である事を意味します。これ以降、スレーブ コンポーネントは相手と直接通信関係を確立できます。

システムステートNotOK

現在のセントラル レジストリと保存済みレジストリの内容が一致しない場合、またはシステム競合が存在する場合、NetworkMasterからの Configuration.Status(NotOK)メッセージで全ての通信関係を中止し、システムスキャンを開始します(セクション7.1.4参照)。

次のOKメッセージ送信まで、システムステートはNotOKのままです。

Configuration.Status (NewExt)

下層のアプリケーションの通信準備が完了すると、ファンクション ブロックの情報を送信して通知します。場合によっては、通知の混乱を避けるためアプリケーションの起動に時間がかかる事があります。これは、主に複雑なアプリケーションで発生します。

Configuration.Status(OK)の送信後に新しいファンクション ブロックを追加で登録したいノードは、以下のメッセージを送信する必要があります。

```
0x1xx→0x100 NetBlock.xx.FBlockIDs.Status (FBlockIDList)
```

NetworkMasterは、新しく登録されたファンクション ブロックとセントラル レジストリに保存されているファンクション ブロックを比較します。新しいファンクション ブロックが追加されると、Configuration.Status(NewExt, FBlockIDList)をブロードキャスト送信してシステム全体に通知します。

このメッセージは、システムステートがOKの場合のみ送信できます。システムステートはOKのままです。

新しいノードがシステムに追加された場合は、これとは異なります。この場合、NetworkMasterは全てのノードに対して再びファンクション ブロックを問い合わせます。こうして新しいノードを特定し、ノードアドレスを確認します。これが、動的アドレス割り当て(セクション7.1.4参照)の大きな利点です。リングの途中に新しいノードを追加しても、この新しいノードの論理アドレスが未使用の空きアドレスであればシステム競合は発生しません。

Configuration.Status (Invalid)

デバイスはファンクション ブロックを後から登録できる以外に、アプリケーションまたはファンクション ブロックの登録を解除する事ができます。また、デバイ

ス内部で予期しない問題が発生して、ノード全体がシステムで利用不可になるケースもあります。例えば、MOSTデバイスは内部リセットの後、一定期間電気的バイパスを閉じる必要があります。

リセット等、ノード自体が利用できなくなるほどの変化がシステムに発生した場合、NetworkMasterはシステム内の全てのノードに対してファンクション ブロックの問い合わせを開始します。その問い合わせ結果とセントラル レジストリを比較する事により、利用不可になったノードを容易に検出できます。そして、以下のメッセージを用いてこの事をシステム全体にブロードキャスト送信します。
`Configuration.Status (Invalid, FBlockIDList)`

このメッセージは、システムステートがOKの場合のみ送信できます。システムステートはOKのままです。

ファンクション ブロックが利用不可になった後、そのファンクション ブロックの通信相手はノーティフィケーション マトリクス (セクション7.2参照)を適宜更新する必要があります。つまり、無効になったファンクション ブロックをキャンセルする必要があります。また、デセントラル レジストリも同様に更新する必要があります(詳細はセクション7.2参照)。

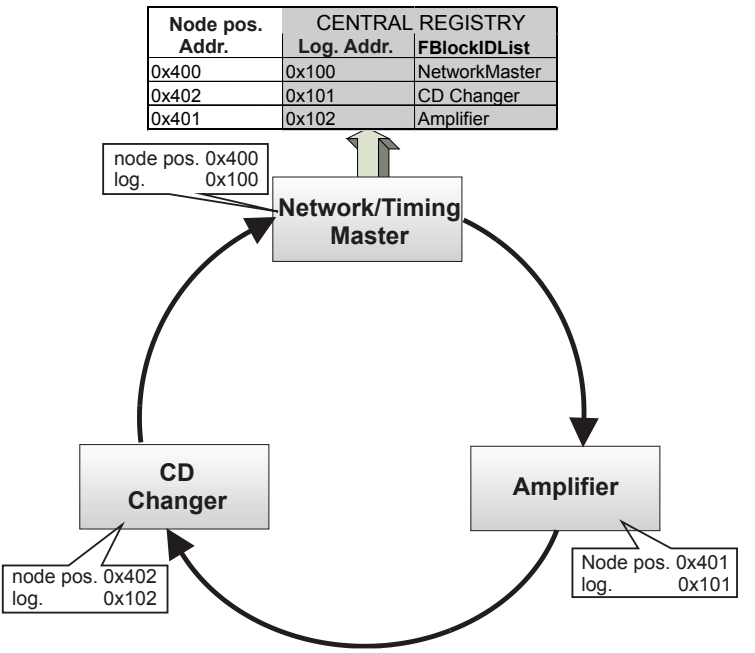


図7.2:論理アドレスを使った通信関係の確立。セントラル レジストリは図7.1から変化なし

7.1.3 動的アドレッシング

セントラル レジストリは論理アドレスのみを格納し、ノード位置アドレスは格納しません。これら2つのアドレスは、互いに分離しています。各ノードは、検出されたファンクション ブロックのアドレスに関する情報をセントラル レジストリから取得し、互いに直接通信関係を確立します。スレーブノード同士はリング内の物理的なノード位置ではなく、論理アドレスを使って互いに直接通信します。2つの通信相手の間でノードが追加または削除されると、それ以降の全てのノードのノード位置アドレスが変化しますが、論理アドレスは変化しません。このため、通信関係はそのまま維持されます。

システム初期化中だけでなく、システム動作中にNetworkMasterがアドレス0xFFFFのスレーブノードから登録を受け取った場合、システム競合が存在します。NetworkMasterはその他全てのアクションを中止し、Configuration.Status(NotOK)をブロードキャスト送信します。これを受けて、全てのスレーブノードが各自に保存してある論理アドレスを破棄し、下式に基づいて論理アドレスを再計算します。

RxTxLog = 0x100 + Pos

7.1.4 システム開始の例

ここでは、以下の構成を例に考えます。

	ノード位置アドレス	論理アドレス	ファンクションブロック	FBlock-ID
ノード0	0x400	0x100	NetworkMaster	0x01
ノード1	0x401	0x101	AudioDiskPlayer	0x31
ノード2	0x402	0x102	AudioAmplifier	0x22

1.セントラル レジストリは前回のシステム実行から変化していません。AudioDiskPlayerは遅れて登録されます

```
0x0100→0x0401:NetBlock.01.FBlockIDs.Get ()
0x0101→0x0100:NetBlock.01.FBlockIDs.Status ()
0x0100→0x0402:NetBlock.02.FBlockIDs.Get ()
0x0102→0x0100:NetBlock.02.FBlockIDs.Status (0x22,0x00)
0x0100→0x03C8:NetworkMaster.01.Configuration.Status (OK)
0x0101→0x0100:NetBlock.01.FBlockIDs.Status (0x31,0x00)
0x0100→0x03C8:NetworkMaster.01.Configuration.Status(NewExt,
                  0x31,0x00)
```

2.AudioAmplifierが長時間にわたって電源から切断されます。これにより、AudioAmplifierが論理アドレスを失います。

```
0x0100→0x0401:NetBlock.01.FBlockIDs.Get ()
```

```
0x0101→0x0100:NetBlock.01.FBlockIDs.Status (0x31,0x00)
0x0100→0x0402:NetBlock.02.FBlockIDs.Get ()
0xFFFF→0x0100:NetBlock.02.FBlockIDs.Status (0x22,0x00)
0x0100→0x03C8:NetworkMaster.01.Configuration.Status (NotOK)
0x0100→0x0401:NetBlock.01.FBlockIDs.Get ()
0x0101→0x0100:NetBlock.01.FBlockIDs.Status (0x31,0x00)
0x0100→0x0402:NetBlock.02.FBlockIDs.Get ()
0x0102→0x0100:NetBlock.02.FBlockIDs.Status (0x22,0x00)
0x0100→0x03C8:NetworkMaster.01.Configuration.Status (OK)
```

7.2 ノーティフィケーション

デセントラル レジストリ

NetworkMasterがConfiguration.Status(OK)を送信すると、スレーブデバイス間の通信の確立が開始します。これは通常、NetworkMasterのコンポーネントから通信相手のファンクション ブロックの論理アドレスを取得する事で開始します。

```
0x0101→0x0100:NetworkMaster.01.CentralRegistry.Get
                  (FBlockID, InstID)
0x0100→0x0101:NetworkMaster.01.CentralRegistry.Status
                  (FBlockInfoList)
```

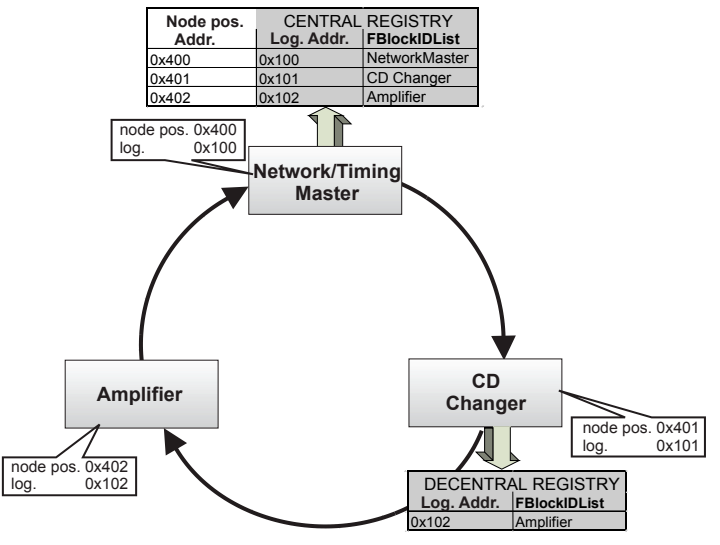


図7.3:セントラル レジストリの情報からデセントラル レジストリを設定

この情報を、各スレーブのデセントラル レジストリに格納します。サーチ結果を毎回ローカルのデセントラル レジストリに格納する事で、通信相手の検出が容易になります。同じ相手と再び通信する場合、セントラル レジストリにもう一度アクセスする必要はありません。各スレーブは通信相手を記憶しています。デセントラル レジストリとセントラル レジストリは同じ構文を使います。デセントラル レジストリには、相手デバイスとの通信に必要な情報のみを格納します。図7.3に、デセントラル レジストリの設定手順を示します。デセントラル レジストリを利用して、デバイスは通信相手と直接通信できます。

ノーティフィケーション メカニズム

あるデバイスに属するファンクション ブロックのプロパティが変化した場合、別のデバイスに通知が必要となるケースがよくあります。これをポーリングで実現しようとすると、バスに高い負荷がかかります。これを避けるためにノーティフィケーション メカニズムがあります。プロパティが変化すると、ノーティフィケーション マトリクスに登録済みのデバイスにイベントを送信します。ノーティフィケーション マトリクスは、ネットワーク サービス層II (セクション9.3.3参照)に実装されているテーブルです。

```
0x0101→0x0100:FBlock.InstID.Notification.Set (Control,
              DeviceID, FktID1, FktID2,…)
```

	FktID1	FktID2	FktID3
DeviceID1	x	x	
DeviceID2		x	x
DeviceID3	x		x
空き			
空き			

例

ディスプレイ ユニット(DeviceID 0x100)に常にAudioDiskPlayerの現在のトラック番号を表示するようにします。そこで、AudioDiskPlayerのTrackPositionファンクション(FktID 0x202)にノーティフィケーションを設定し、トラック番号が変わるたびに自動的に新しいトラック番号の通知を受けるようにします。以下に、必要な制御メッセージを示します。詳細は、MOST仕様[MOST 3.0]を参照してください。

```
0x0100→0x0101:AudioDiskPlayer.01.Notification.Set (0x1,
0x100, 0x202)
0x0101→0x0100:AudioDiskPlayer.01.TrackPosition.Status (01)
0x0101→0x0100:AudioDiskPlayer.01.TrackPosition.Status (02)
:
0x0101→0x0100:AudioDiskPlayer.01.TrackPosition.Status (05)
:
```

7.3 パワーダウン

MOSTシステムのシャットダウンは、大きく2段階に分けられます。1つは、システム内の各ノードにシャットダウンを通知する準備段階です。2段階目で、システムを実際にシャットダウンしてスリープに移行します。これらの段階をそれぞれシャットダウンの「開始(イニシエーション)」と「実行(エグゼキューション)」と呼びます。起動の場合と同様、パワーダウンの制御機能もPowerMasterが実行します。

シャットダウンの開始(イニシエーション)

通常、シャットダウンの開始は自動車のドアロックの施錠等、外部イベントでトリガされます。このトリガイベントは、MOSTメッセージでPowerMasterに送信されます。するとPowerMasterは以下のブロードキャスト メッセージを用いてシャットダウンを開始します。

```
0x0100→0x03C8:NetBlock.00.ShutDown.Start (Query)
```

このメッセージを受信すると、システム内の各ノードはスリープへの移行準備をします。まだ通信が終了しておらず、スリープへの移行準備ができていないデバイスは、以下のメッセージでPowerMasterに通知します。

```
0x0101→0x0100:NetBlock.01.ShutDown.Result(Suspend)
```

PowerMasterは一定の期間($t_{WaitSuspend}$)待機してから再びシャットダウン開始の問い合わせを送信します。

シャットダウンの実行(エグゼキューション)

PowerMasterは、$t_{WaitSuspend}$の期間内にResult(Suspend)のメッセージを受信しなければ、以下のメッセージを用いてシャットダウンを実行します。

```
0x0100→0x03C8:NetBlock.00.ShutDown.Start (Execute)
```

このメッセージを送信してからt ShutdownWait(通常は$t_{ShutdownWait}$ = 2 sec)以内に、PowerMasterは変調信号をOFFにします。それ以降の全てのノードも、RX入力からの光受信が途絶えるとただちに変調信号をOFFにします。

7.4 フォルトの種類

このセクションでは、フォルトの種類としてネットワーク フォルトとデバイス フォルトの2つを区別します。

7.4.1 ネットワーク フォルト

リングブレーク

通常、MOSTシステム内のデバイスはRX入力で変調信号を受信すると復帰します。この結果、デバイスはTX出力から信号を送信します。全てのTimingSlaveがリングトポロジで接続されていれば、TimingMasterがTX出力から信号を送信した後、一定期間内にTimingMasterはRX入力で信号を受信します。しかし、部品の接続が不良、部品の不良、ケーブルの破損のいずれかの理由によりリングがロックできない事があります。この場合、通常の動作状態に起動する事はできません。

突発的信号OFF

シャットダウン開始の手順を踏まずに突然デバイスのRX入力信号がなくなった場合、そのデバイスは自動的にTimingMasterモードに切り換わります。次に、このノードはMOSTフレームの管理領域にシャットダウン フラグをセットし、シャットダウンの開始を他のノードに通知します。$t_{SSO_ShutDown}$の期間内にこの信号を送信したら、このノードは送信をOFFにし、フォルトの原因を保存します。他のノードはシャットダウン フラグを検出しているため、フォルトイベントを格納せずにシャットダウンします。このようにして、突発的信号OFFが発生するとMOSTネットワーク全体がシャットダウンします。

アンロックとクリティカル アンロック

TimingMasterのシステムクロックとTimingSlaveデバイスの間で同期が失われる事をアンロックと呼びます。アンロックの理由としては、RXまたはTXコンポーネントの不良、信号強度の不足によるシステムクロックの検出失敗等、各種の要因があります。アンロックには、ショート アンロックとクリティカル アンロックの区別があります。ショート アンロックは、同期が失われた時間がt_{Unlock} (通常は100 ms未満、代表値は70 ms)未満のものを言います。ショート アンロックは、TimingMasterが信号を再生成して補償します。

t_{Unlock}より長い期間同期が失われた場合をクリティカル アンロックと呼びます。「突発的信号OFF」の場合と同様、影響を受けたデバイスは自動的にTimingMasterモードに切り換わり、適切なシャットダウン フラグをセットしたMOSTフレームを送信し、フォルトの原因を保存した後、シャットダウンします。

符号エラー

符号エラーは、ケーブルの変形による信号減衰の増加等、符号化の違反により発生します。ノードは受信データを復号できないため、再送が多発して通信が遅れるか、データが失われます。符号エラーが大量に発生すると、アンロック、クリティカル アンロック、突発的信号OFFにつながる事があります。

7.4.2 温度および電圧違反によるデバイスフォルト

このセクションでは、温度および電圧違反による誤動作または恒久的損傷からMOSTデバイスを保護するためにシャットダウンが実行された場合のフォルトについて説明します。

当然、システムではこれ以外のデバイスフォルトも発生する事があります。しかし、MOSTデバイスの突然のシャットダウンは突発的信号OFFイベントにつながるため、温度および電圧違反によるフォルトはMOSTネットワークにとって特に大きな意味を持ちます。MOSTには、過熱、低電圧、過電圧によるフォルトを区別するメカニズムと、それぞれに異なる対処法が用意されています。

過熱

車載デバイスでは、温度レンジに関して厳しい要求があります。物理的な環境要因により、動作温度の要件を守る事ができない場合があります。従って、全ての部品を保護できるようにリング内の各ノードには温度に基づいてシャットダウンを開始する機能があります。デバイスは、温度が仕様値$\vartheta_{Shutdown}$を超えると、NetBlock.ShutDown.Result (Temperature Shutdown)メッセージをブロードキャスト送信します。臨界温度$\vartheta_{Critical}$に達すると、デバイスは変調信号をOFFにしてただちにシャットダウンします。PowerMasterは温度シャットダウンの要求を既に受信しているため、一定の冷却期間待機した後、システムを再び復帰させます。

温度シャットダウンにはもう1種類あります。温度がϑ_{AppOff}に達すると、アプリケーションの一部(例: 過熱したドライブ)のみを任意でシャットダウンできます。その他の機能は引き続き使用できます。シャットダウンするFBlockは、セントラル レジストリから登録解除する必要があります。

低電圧

通常、車載電力系の低電圧が全てのデバイスに同じ程度の影響を与える訳ではありません。このため、低電圧しきい値には以下の2つがあります。

- 臨界電圧$U_{Critical}$:これは、ネットワーク通信は十分可能でもアプリケーションの一部機能に支障が出る低電圧レベルと定義されます。ネットワーク サービスはNormal Operationモードにとどまります。ストリーミング通信でのデータ送信は行われません。つまり、ソースはゼロを送り、シンクは出力信号を保護する必要があります。Muteプロパティは変わりません。通常の電圧レベルに達すると、データを再び送信できます。
- 低電圧U_{Low}:これは、ネットワーク インターフェイス間の通信が維持できない低電圧レベルと定義されます。U_{Low}に達すると、デバイスは変調信号をOFFに切り換え、DevicePowerOffモードに切り換わります。電源電圧が回復した場合でも、デバイスはDevicePowerOffモードにとどまります。デバイスは、変調信号が入力された場合、または自身のアプリケーションからの通信要求で復

帰します。デバイスは標準の初期化プロセスを通じてDeviceNormalOperationモードに移行します。

過電圧または超高圧

電圧が超高圧(U_{Super})を超えると、デバイスは安全動作ステートに移行する必要があります。U_{Super}の値は、デバイスごとに定義する必要があります。U_{Super}の代表値は16 Vです。

8 ネットワーク診断

この章では、MOSTシステムの診断処理ついて説明します。まず、ネットワークフォルトを検出し、ネットワーク内でのフォルトの発生場所を特定するメカニズムについて解説します。これらのメカニズムはシステムの製造および保守時のコスト削減につながるため、車載環境では特に重要な意味を持ちます。この章の最後では、MOSTデバイスで発生するフォルトについて説明します。診断テストツールは、特別な通信プロトコル(DAP: Diagnostics Adaptation Protocol)を用いてMOSTデバイスのフォルトメモリを読み出します。

8.1 ネットワーク フォルト検出

MOSTには、いくつかの種類のネットワーク フォルトに対処する機能があります。

8.1.1 リングブレーク診断

リングブレーク診断(RBD)を実行すると、全てのデバイスがリングブレーク位置を起点とした自分自身のノード位置を保存します。リングブレーク位置の直後にあるデバイスがTimingMasterになる事で、これを可能にしています。このデバイスは、自分のノード位置を含む信号を送信します。各TimingSlaveは、受信したノード位置の値をインクリメントします。リングブレークが検出されない場合、元のTimingMasterが通常動作を開始します。

最初に、デバイスに外部トリガが与えられるとRBDが開始します。何をトリガイベントとするかは、システムによって異なります。例えば、ECL (Electrical Control Line) (セクション8.1.4参照)上でのシステムテストをトリガにしてRBDを実行できます。

図8.1に、RBDの手順の例を示します。

この例では、ネットワークのノード0が最初のTimingMasterです。ノード1、ノード2、ノード3は、最初はTimingSlaveとして動作します。ノード1とノード2の間でリングブレークが発生します。まず、各TimingSlaveはそれぞれのRX入力でのアクティビティを待ちます。t_{Diag_Signal}の期間アクティビティを検出しなければ、ノードはTimingSlaveモードからTimingMasterモードに切り換わります。従って、ノード2とノード3はt_{Diag_Signal}の経過後、TimingMasterとして動作します。その後、ノード3はノード2から信号を受信します。従って、ノード3はただちにTimingSlaveモードに戻ります。T1の期間が経過後、最初のTimingMaster(ノード0)が自分自身のRX入力を評価します。ノード0は信号を検出しますが、安定ロックを検出しないため、自分の前に別のTimingMasterが存在する事を認識します。従って、ノード0はTimingMasterモードからTimingSlaveモードへ移行します。

この手順により、TimingMasterのノード2を先頭に、後続の全てのノード (ノード3、ノード0、ノード1) がTimingSlaveとなる開いたチェーンとなります。TimingMasterのノード2は、ノード位置0を含む信号を送信します。ノード3はこの信号を受信すると、ノード位置をインクリメントします。ノード3はノード位置1を保存し、この新しいノード位置の値を含む信号を送信します。以降のTimingSlaveも、同じ方法でノード位置を処理します。

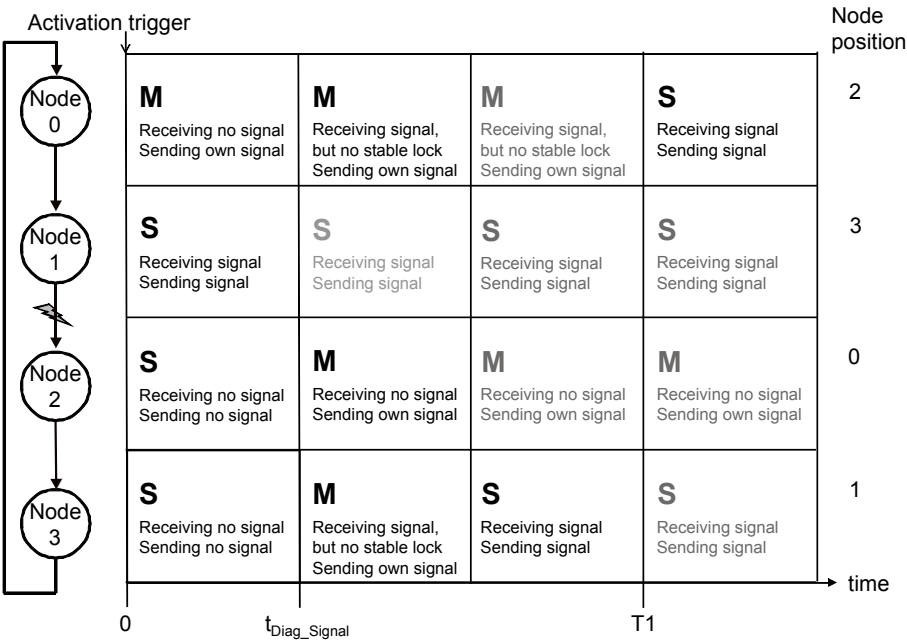


図8.1:リングブレーク位置の特定

最後に、RBDの結果を専用のデバイスに送る事ができます。最後のこの手順は任意で、システム ソリューションにより異なります。

8.1.2 突発的信号OFF/クリティカル アンロックの検出

この機能はMOST仕様Rev. 3.0で新たに導入されました。NetInterface Normal Operationモードでノードに突発的信号OFF (SSO)またはクリティカル アンロック (CU)イベントが発生すると、後続のノードは有効な信号を受信できなくなります。この結果、後続のノードはTimingMasterモードに切り換わり、MOSTフレームの管理領域にシャットダウン フラグをセットしてフレームを送信し、フォルトの原因を保存してからセクション7.1で説明した方法でシャットダウンします。

元のTimingMasterはシャットダウン フラグを検出できません。従って、SSOまたはCUフォルトを検出して保存します。フォルトを正しく評価するには、この点を考慮する必要があります。

評価する側のデバイスは、MOSTのNetBlock.ShutDownReasonプロパティを使ってローカルに保存されたフォルト情報を収集できます。全てのデバイスから情報を収集すると、NetBlock.ShutDownReason.Set(0x00)をブロードキャスト送信します。その後、全てのノードが保存済みのシャットダウン原因を削除します。

複数のノードからフォルトが報告される事があるため、フォルトの発生場所を特定するには、評価する側のデバイスで評価アルゴリズムを適用する必要があります。

- TimingSlaveがSSOを報告している場合、そのTimingSlaveの前で必ずSSOフォルトが発生しています。フォルトは伝搬するため、他のノードがCUを報告する可能性があります。評価アルゴリズムはこれらを無視する必要があります。
- TimingMasterがSSOを報告しており、どのTimingSlaveもSSOまたはCUを報告していない場合、TimingMasterの前で必ずSSOフォルトが発生しています。
- MOSTネットワークではフォルトが伝搬するため、原則として複数のCUが報告される可能性があります。これ以外にSSOが報告されていない場合、CUを報告した最初のTimingSlaveの前で最初のCUが発生しています。
- 少なくとも1つのTimingSlaveがSSOまたはCUを報告している場合、TimingMasterの前の状況は不明です。従って、評価アルゴリズムはTimingMasterの報告を無視する必要があります。

8.1.3 符号エラーの発生場所の特定

各MOSTノードのネットワーク サービスには、符号エラーをカウントする機能があります。通常、ネットワークの起動またはシャットダウン中はカウントせず、システムが通常動作中にカウントします。また、アンロック、CU、SSOのいずれかが発生した場合、ノードは符号エラーをカウントしません。1つのノードがカウントする符号エラーは、システム固有の長さの期間中、最大1つまでです。

各ノードの符号エラーカウンタは、専用のノードがMOSTのDiagnosis.CodingErrorプロパティを使って読み出す事ができます。評価ノードは、システム固有のしきい値を超える数の符号エラーを報告したノードのうち、ノード位置が最も小さいノードの直前に符号エラーフォルトの発生源を検出します。フォルト伝搬を補償するため、それ以降のノードが報告する符号エラーは無視する必要があります。

8.1.4 ECL (Electrical Control Line)

MOSTネットワークの障害で通信が全く行えない場合でもネットワーク フォルトを検出できるように、一部の自動車メーカーはMOSTネットワークに加えECL (Electrical Control Line)を採用しています。ECLはMOST仕様Rev. 3.0に基づく任意実装の機能で、別の文書[ECL]で定義されています。ECLは、専用のケーブルでMOSTデバイスを接続します。これらの接続デバイスは、ECLを利用して電気的なデジタルシリアル通信で情報を交換します。

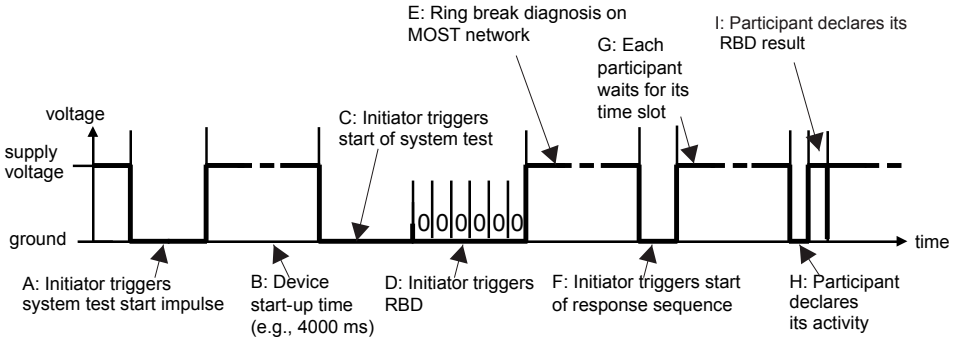


図8.2:システムテストの順序

専用のデバイスがECLを利用していわゆるシステムテストの実行を開始するというのが基本的な考え方です。システムテストには、リングブレーク診断、SSO/CUフォルト発生源検出、符号エラーのカウントを含みます。システムテストが完了したら、テスト対象の各デバイスはECLを利用してテスト結果をイニシエータに返します。

図8.2に、システムテスト全体の手順の例を示します。

ステップAでは、イニシエータ デバイスがECLの電圧を200 msの間グランドにプルダウンして、全てのテスト対象デバイスにシステムテスト開始のパルスを送信します。ステップBの期間、ECLにはアクティビティはありません。全てのテスト対象デバイスがこの期間内に起動します。ステップCでは、イニシエータがシステムテストの開始を示すパルスを送信し、全てのテスト対象デバイスを同期します。システムテストの開始を示すパルスは、イニシエータが電圧を250 msの間グランドにプルダウンして送信します。ステップDで、イニシエータが特定のパラメータ シーケンスを送信してシステムテストをトリガします。送信するパラメータ シーケンスは、以下に示した通りです。ここで、値「0」は電圧を50 msの間グランドにプルダウンする事、値「1」はECLを50 msの間電源電圧のままとする事と定義されます。

- 「0-0-0-0-0-0」はRBDテストです。テスト対象デバイスが安定ロックを検出すると、ステップIのタイムスロットで結果「0」を送信し、それ以外の場合は「1」を送信します。
- 「0-1-0-0-0-0」は符号エラーテストです。符号エラーの検出回数がシステム固有のしきい値を超えた場合、テスト対象デバイスはステップIで「0」を送信し、それ以外の場合は「1」を送信します。
- 「1-0-0-0-0-0」は生存確認テストです。図8.2に示すように、ステップHにはテスト対象の各デバイスがそれぞれの全般的なアクティビティを宣言するためのタイムスロットがあります。全てのシステムテストで、アクティブなテスト対象デバイスはこのタイムスロットで「0」を送信します。生存確認テストの場合、テスト対象デバイスはステップIで何も結果を返しません。生存確認テ

ストは、デバイスが利用可能な状態にあるかどうかを全般的に確認するために実行します。

- 「1-1-0-0-0-0」はSSO/CUテストです。このテスト中にテスト対象デバイスがSSOまたはCUを検出しなければステップIで「0」を送信し、それ以外の場合は「1」を送信します。

システムテスト実行中は、全てのデバイスがECLを電源電圧のままにします(ステップE参照)。ステップFで、イニシエータがECLの電圧を100 msの間グランドにプルダウンして応答シーケンスの開始をトリガします。これにより、全てのテスト対象デバイスを同期します。その後、テスト対象の各デバイスは自分のタイムスロットが来るまで待ちます(ステップG)。各デバイスには、それぞれ専用のタイムスロットが静的に割り当てられます。この割り当ては、MOSTのリング位置に依存しません。前述の通り、テスト対象デバイスはステップHとステップIで応答します。

8.2 デバイスフォルトの診断

このセクションでは、Diagnostic Adaptation Protocol (DAP)について説明します。これは、車両診断用の通信プロトコルをMOSTの通信プロトコルに合わせて応用したものです。DAPはMOST仕様Rev. 3.0で新たに導入された任意実装の機能で、別の文書[DPA]で定義されています。

車両診断の基本的な考え方は、以下の通りです。まず、車両内のデバイス内部に診断サービスを常駐させます。診断テストツールはクライアントとして動作し、これらのサービスを要求します。このようにして、工場の診断テストツールから各デバイスのトラブルコード等を読み出す事ができます。デバイス内部で何らかのフォルトが発生するたびに、フォルトメモリに新しいトラブルコードが登録されます。トラブルコードは、修理保守対応に関する指示を含みます。

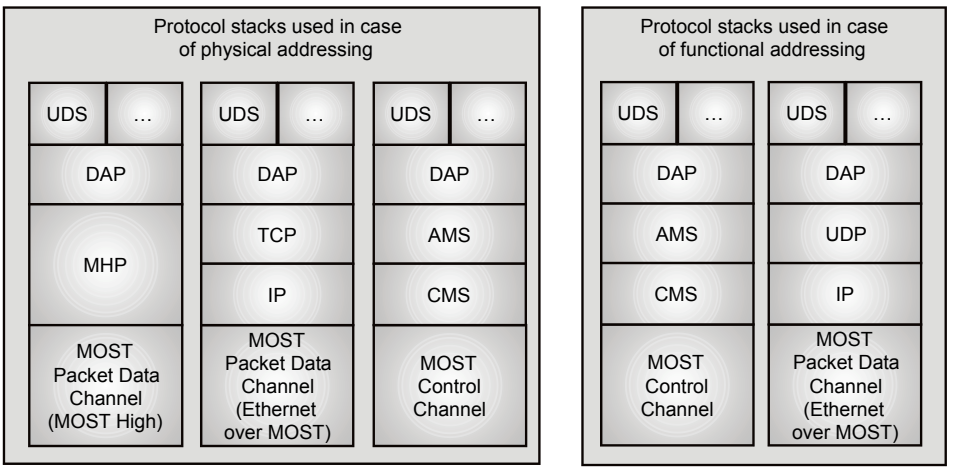


図8.3:プロトコル スタック

クライアント/サーバ通信を実現するには、多くのプロトコルを適用できます。例えば、車両診断にはUnified Diagnostic Services (UDS) [ISO 14229]をよく使います。

図8.3に、DAPを使ってMOST上で診断メッセージを交換するための通信プロトコルスタックを示します。

物理アドレッシングは、クライアントが1つのサーバのみに要求を送信する事を意味します。サービス実行後、サーバは最終応答を返す事ができます。例えば、要求されたデバイスの診断トラブルコードを応答に含める事ができます。さらに、サーバが診断サービス実行中にresponse pending(応答保留)メッセージを返す事ができます。物理アドレスを用いた要求とそれに対する応答は、MHPまたはTCPを使ってMOSTパケットデータ チャンネルで転送するか、AMSを使ってMOST制御チャンネルで転送できます。

ファンクション アドレッシングは、クライアントが多くの分散サーバに要求を送信する事を意味します。これらの要求を送信する自明な手段として、マルチキャストまたはブロードキャスト メカニズムがあります。サーバは、response pending メッセージまたは最終的な応答メッセージを返す事ができます。ファンクションアドレスを用いた要求は、UDPを使ってMOSTパケットデータ チャンネルで転送するか、AMSを使ってMOST制御チャンネルで転送できます。MOST仕様[MOST 3.0]で定義されたブロッキングまたはノンブロッキング ブロードキャストを使う事ができます。

バイト番号	データフィールド	長さ(バイト)	値
1	バージョン	1	DAPヘッダのバージョン:0x01
2 – 5	ペイロード長さ	4	n
6	サービスタイプ	1	システム インテグレータ固有: 0x00 UDS:0x01 KWP2000:0x02 将来用に予約済み:0x03～0xBF システム インテグレータ固有: 0xC0～0xFF
7 – 8	システムインテグレータ用	2	システム インテグレータ固有: 0x0000～0xFFFE 既定値:0xFFFF
9 – 10	送信元アドレス	2	例: 診断テストツールのアドレス 未使用:0xFFFF
11 – (10+n)	ペイロード	n	

表8.1:DAPパケットの構造

最初のデータフィールドは、DAPヘッダのバージョンを表します(現在は0x01)。

次のデータフィールドは、「ペイロード長さ」です。ストリーム指向のTCPでは、このフィールドを使ってバイトストリームから診断メッセージを再構築できます。また、UDSメッセージのサイズがMHPの最大パケットサイズより大きい場合も、セグメンテーションと再構築にこのフィールドが必要です。

次のデータフィールドの「サービスタイプ」でアプリケーション プロトコルを選択します。受信したDAPパケットのペイロードは、UDSまたは別のアプリケーション プロトコル(例: KWP2000)に割り当てる事ができます。

「システム インテグレータ用」データフィールドは、既定値で0xFFFFに設定されます。このフィールドは、シーケンス番号を扱いたい場合等、必要に応じてシステム インテグレータが使えます。

データフィールド「送信元アドレス」は、要求を送信した診断テストツールを特定します。サーバデバイスは、このフィールドのデータを診断要求から対応する診断応答へコピーする必要があります。こうする事で、複数の診断テストツールが存在する場合、および応答を複数のネットワークで転送する必要がある場合でも診断応答を正しい診断テストツールに返す事ができます。

最後のデータフィールドである「ペイロード」は、UDSまたは別のアプリケーション プロトコル(例: KWP2000)で処理される診断メッセージを格納します。

9 ネットワーク サービス

9.1 概要

ネットワーク サービスはMOSTの標準プロトコル スタックです。一般的に使われるMOST NetServicesという用語はMicrochip社の商標であり、MOST NetServicesの名称を利用するにはライセンス許諾が必要です。MOST仕様では、特定の製品に依存しない用語としてネットワーク サービスを使います。

図9.1に示すように、ネットワーク サービスはネットワーク インターフェイス コントローラ(NIC)またはインテリジェント ネットワーク インターフェイス コントローラ(INIC)の上位に位置し、基本的にファンクション ブロックで構成されるアプリケーションに対するプログラミング インターフェイスを提供します。ネットワーク サービスは、パケットデータ チャンネルでパケットデータを転送するモジュールと制御チャンネル経由でネットワークを制御するモジュールで構成されます。つまり、ネットワーク サービスはネットワークの運用と管理を行うためのメカニズムとルーチンを備え、ネットワークの動的挙動を保証します。ネットワーク サービスは外部ホスト コントローラ(EHC)に実装されます。ネットワーク サービスの機能はストリーミング データ チャンネルの制御も含みますが、ストリーミング データの送信は含みません。

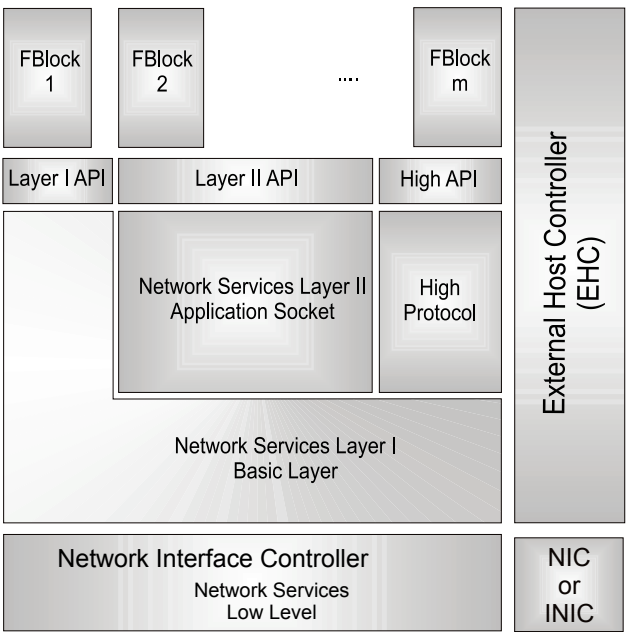


図9.1:MOST NetServicesの基本構造

9.2 MOST NetServices

MOST NetServicesは、Microchip社によるネットワーク サービスの標準実装です。これらはネットワーク インターフェイス コントローラ(NICとINIC)に最適化されています。

MOST NetServicesは基本レイヤ(レイヤI)とアプリケーション ソケット (レイヤII)に分割され、MOST High Protocol(セクション5.8参照)等のモジュールを追加で含める事ができます。

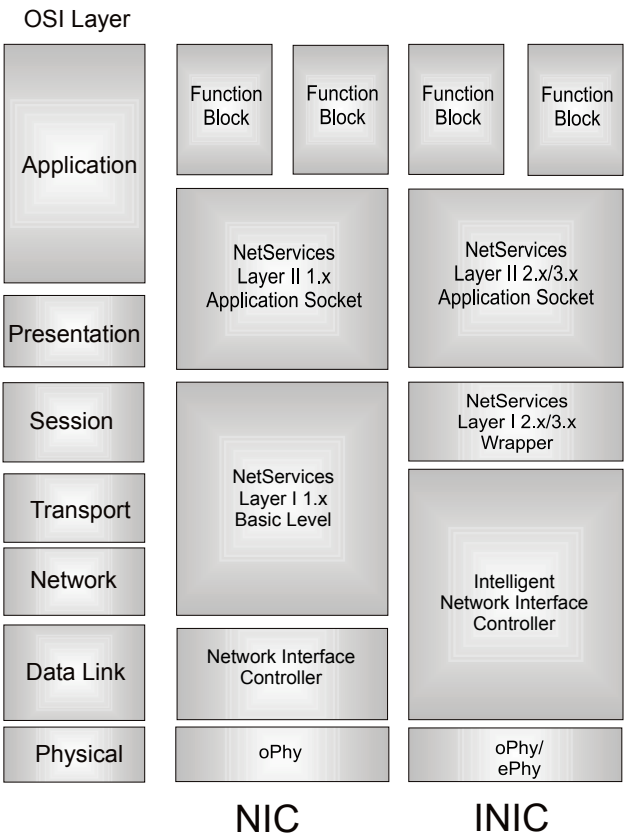


図9.2:MOST NetServicesのモデル

MOST NetServicesはANSI-Cで記述されており、各種プロセッサ プラットフォーム(EHC)に直接移植できます。Microchip社は以下の4種類のライセンスを提供しています。

- 評価用ライセンス
- システム ライセンス
- 製品ライセンス
- ツールライセンス

9.2.1 MOST NetServicesのモデル

MOST NetServicesレイヤIは、NICとINICでバージョンが異なります(NICはバージョン1.x、INICはバージョン2.xと3.x)。これは、INICはレイヤIの基本機能をカバーしており、MOST NetServicesはレイヤIとレイヤIIを接続するインターフェイスのみを提供すればよいためです。MOST NetServicesレイヤII 2.x/3.xでは、モジュールの一部機能をINICが肩代わりしており、これらモジュールのみがレイヤII 1.xと2.x/3.xで違っています。

9.2.2 MOST仕様の各リビジョンとの対応関係

MOST NetServicesバージョン1.xと2.xは、MOST仕様Rev2.xに基づいて設計されています。バージョン1.xはMOST25にのみ対応しており、バージョン2.xはMOST25とMOST50に対応しています。

バージョン3.xはMOST仕様Rev. 3.xに基づいて設計されており、MOST50とMOST150に対応しています。

9.3 MOST NetServicesのモジュール

9.3.1 レイヤIバージョン1.x

MOST NetServicesレイヤIバージョン1.xはNIC用に設計されています。インターフェイスとして、I²Cバス、SPI、パラレル インターフェイスを利用できます。NICのソフトウェア インターフェイスは一連のレジスタで構成され、これらはマイクロコントローラによる読み書きが可能です。これら一連のレジスタをレジスタウォールと呼びます。

以下に、レイヤIのモジュールを紹介します。

図9.3に、レイヤI 1.xのモジュールの概要を示します。

MOST NetServicesカーネル(MNS)

MOST NetServicesカーネル(MNS)は、レイヤIの全てのサービスの中心的なエントリポイントとなるものです。MNSは全てのトリガおよびコールバック関数を初期化します。

MOSTスーパバイザ(MSV)

MOSTスーパバイザ(MSV)は、ネットワーク管理用の有限ステートマシン(FSM)を提供します。以下の機能をカバーします。

- NICの初期化
- デバイスがTimingMasterかTimingSlaveかの決定

- ソースポートの設定
- ネットワークの起動とシャットダウン
- 電源管理
- エラー診断とエラーメッセージ生成
- セルフテスト

各種ソースポート設定をサポートしています。

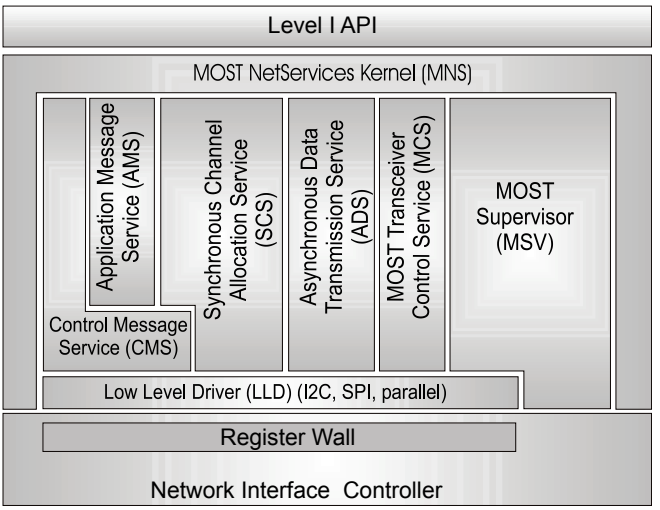


図9.3:MOST NetServicesレイヤIバージョン1.x

制御メッセージ サービス(CMS)

制御メッセージ サービス(CMS)は制御チャンネルで転送される制御メッセージを送受信します。このサービスは以下の機能で構成されます。

- 初期化
- 制御メッセージ受信サービス
- 制御メッセージ送信サービス
- メッセージ バッファリング
- エラー処理とエラーメッセージ

非同期データ転送サービス(ADS)

非同期データ転送サービス(ADS)は、パケットデータ チャンネルでパケットデータを送受信します。このサービスは以下の機能で構成されます。

- 初期化
- バッファ経由でのパケットデータ送信

- バッファ経由でのパケットデータ受信
- エラー処理とエラーメッセージ

アプリケーション メッセージ サービス(AMS)

アプリケーション メッセージ サービス(AMS)は制御メッセージ サービスの上位に位置し、長いアプリケーション メッセージをブロックに分割して制御チャンネル上で転送するための転送プロトコルを提供します。送信側でメッセージを17バイトのブロックに分割し、受信側で復元します。このサービスには以下の機能があります。

- 初期化
- アプリケーション メッセージの送信
- アプリケーション メッセージの受信
- メッセージ バッファリング
- エラー処理とエラーメッセージ

同期チャンネル割り当てサービス(SCS)

同期チャンネル割り当てサービス(SCS)はNICに内蔵されたルーティング エンジン(RE)を制御します。SCSは、ストリーミング データ チャンネルの割り当てと解除、およびストリーミング データ ソースおよびシンクの設定に必要な機能を提供します。具体的には、以下の機能があります。

- チャンネルの割り当て
- 割り当てとルーティング
- 割り当て解除
- 割り当て解除とルーティング切断
- ソース接続
- ソース切断
- シンク接続
- シンク切断
- ラベルによるチャンネル検出

MOSTトランシーバ制御サービス(MCS)

MOSTトランシーバ制御サービス(MCS)は、NICに対してアプリケーション固有の初期化を実行します。このサービスを利用して、ハードウェアに関連したパラメータを設定できます。以下の機能があります。

- NIC内のアドレスレジスタの設定
- 重要なNICレジスタへのアクセス (レジスタアクセスAPIを経由)

- RMCK周波数の設定
- TimingMaster機能(クロック源の選択、バウンダリ ディスクリプタの設定)

詳細は、MOST仕様[MOST 3.0]、『Basic Services (Layer I) Programmer´s Library』[MNS L1 1.x]、『MOST NetServices Layer 1 Example』[MNS L1 Exa]を参照してください。

9.3.2 レイヤIバージョン2.xと3.x

MOST NetServicesレイヤIバージョン2.xと3.xは、INICの上位に位置します。INICは、レイヤIのカーネル機能(ネットワーク ステータスの管理、トランシーバチェック、メッセージ送受信用ドライバ、ルーティング エンジンの制御)を内蔵しています。このため、MOST NetServicesはデータ処理用のバッファとバージョン1.xから知られているAPIのみを提供します。これらのNetServicesのモジュールはラッパとも呼びます。INICへのアクセスにはレジスタウォールを使わず、メッセージを利用して内蔵のFBlock経由で行います。しかし、このアクセス方法の違いはラッパモジュールで抽象化されます。レイヤI APIにはほとんど変更がありません。

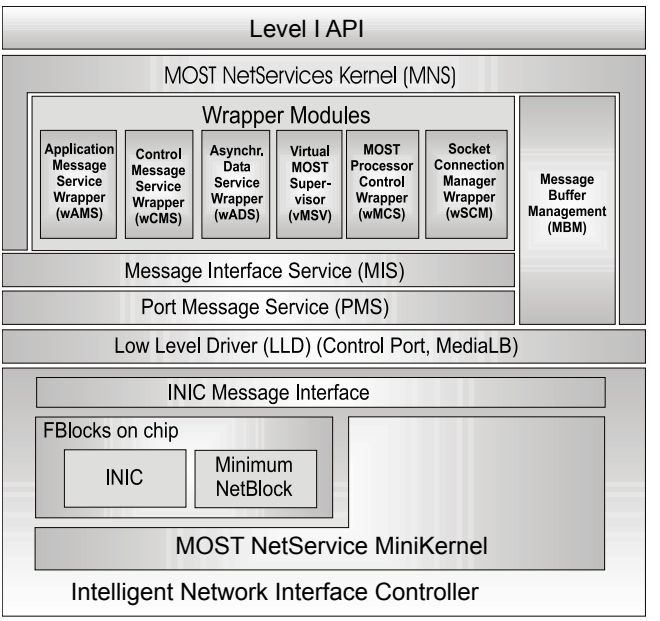


図9.4:MOST NetServicesレイヤIバージョン2.x

NetServicesとINICの間でのMOST制御メッセージと管理通信のやりとりには、制御ポート(I²Cにより物理的にアドレス指定)またはMediaLBポートをインターフェイスとして使います。

パケットデータ チャンネルへのアクセスには、別のインターフェイスを使います。このインターフェイスは、I²C、MediaLB、SPI で物理的にアドレス指定できます(SPIはバージョン3.xのみ)。

図9.4に、MOST NetServicesバージョン2.xの構造を示します。

図9.5に、バージョン3.xの構造を示します。新たにSPIインターフェイスを追加した他、制御メッセージ サービス ラッパ(wCMS)を廃止し、MOSTデバッグ メッセージ モジュール(MDM)を新しく導入しました。

MOST NetServicesカーネル(MNS)

バージョン1.x同様、MOST NetServicesカーネル(MNS)はレイヤIの全てのサービスの中心的なエントリポイントです。MNSは全てのトリガおよびコールバック関数を初期化します。

バーチャルMOSTスーパバイザ(vMSV)

バーチャルMOSTスーパバイザ(vMSV)は、INICに実装された実際のスーパバイザへの割り当てをユーザに提供します。機能はバージョン1.xと同様です。バージョン2.xに比べ、バージョン3.xではネットワーク診断機能を強化しています。

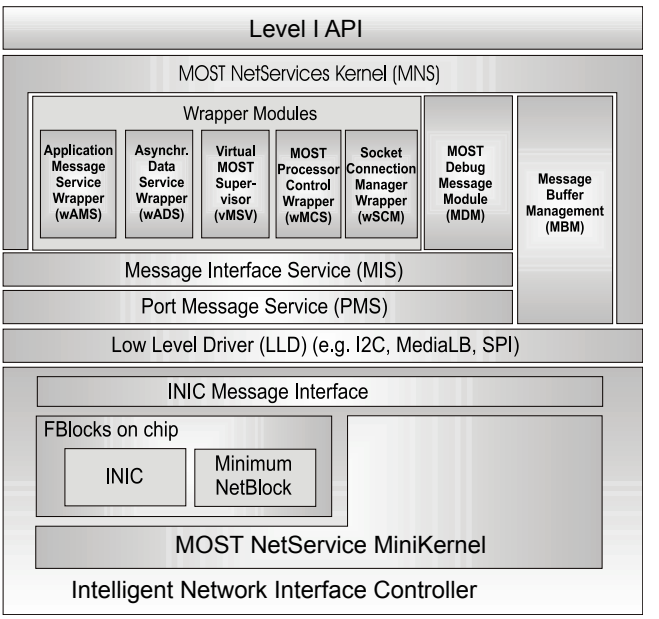


図9.5:MOST NetServicesレイヤIバージョン3.x

ラッパモジュール

ラッパモジュールの制御メッセージ サービス ラッパ(wCMS)、アプリケーション メッセージ サービス ラッパ(wAMS)、MOSTプロセッサ制御サービス ラッパ(wMCS)、非同期データサービス ラッパ(wADS)は基本的にバージョン1.xのAPIをINICの機能に割り当てます。ソケット接続マネージャ ラッパ(wSCM)のAPIのみ、SCS-APIとは大きく異なっています。これは、ストリーミング チャンネルの管理がルーティング エンジンを直接プログラミングして行うのではなく、ソケット経由で直接実行できるようになったためです。

制御メッセージ サービス ラッパ(wCMS)は、バージョン2.xにしかありません。バージョン3.xではアプリケーション メッセージ サービス ラッパ(wAMS)に置き換えられています。

バージョン2.xに比べ、バージョン3.xのソケット接続マネージャ ラッパ(wSCM)には、アイソクロナス ストリーミング データ チャンネルおよびビデオ アプリケーション用のTSI(トランスポート ストリーム インターフェイス)等の新しいインターフェイスにアクセスするためのインターフェイスと設定もあります。

INICに対するデータ処理

INICとMOST NetServicesの間のインターフェイスにはポートメッセージ プロトコルを使います。メッセージ インターフェイス サービス(MIS)、メッセージ バッファ マネジメント(MBM)、ポートメッセージ サービス(PMS)の各モジュールはこのプロトコルを制御し、メッセージをバッファリングし、INIC内の各種FIFOを監視します。

詳細は、『MOST NetServices Layer I –Wrapper for INIC - V2.1.x/V3.x - User Manual/Specification』[MNS L1 2.x]または[MNS L1 3.x]を参照してください。

MOSTデバッグ メッセージ モジュール(MDM)

MOSTデバッグ メッセージはMOST制御チャンネルで報告されるメッセージで、デバイスまたはシステムに関するデバッグ情報を通知するのに適しています。これらのメッセージはシステム解析ツールで記録できますが、他のデバイスは受信しません。MOSTデバッグ メッセージ モジュール(MDM)はこのプロトコルを制御し、これらイベントに関する転送サービスを提供します。このモジュールは、バージョン3.xにしかありません。

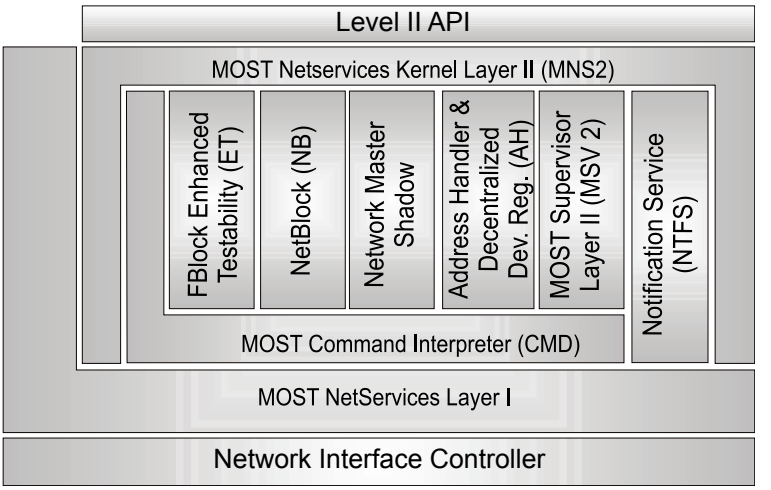


図9.6:MOST NetServicesレイヤII

9.3.3 レイヤIIバージョン1.x、2.x、3.x

MOST NetServicesレイヤIIは、MOST NetServicesレイヤIの上位に位置し、アプリケーション固有ファンクション ブロックに対するAPIを提供します。NetBlock等のシステム ファンクション ブロックはMOST NetServicesに含まれます。ファンクション ブロックEnhancedTestability (ET)の実装はアプリケーションによって異なるため、MOST NetServicesは「外枠」のみを含みます。

バージョン1.xと2.xで異なるモジュールは、各モジュールに関するセクションで説明します。図9.6に、レイヤIIの構造を示します。

MOST NetServicesカーネルレイヤII (MNS2)

レイヤI同様、MOST NetServices Kernel (MNS2)もレイヤIIの全てのサービスの中心的なエントリポイントです。MNS2は、全てのトリガおよびコールバック関数を初期化します。

MOSTスーパバイザ レイヤII (MSV2)

MOSTスーパバイザ レイヤII (MSV2)も、有限ステートマシンを用いてアプリケーション レベルのネットワーク ステート(例: 通信の許可/禁止)を管理します。以下の機能が実装されています。

- アドレスの初期化
- Init Readyイベント後のアプリケーション レベルでの設定
- NetworkSlaveの場合:デセントラル レジストリの初期化
- NetworkMasterの場合:ネットワーク変更イベント(NCE)の監視

FBlockID	InstIDのポインタ(RAMに保存)	ファンクション テーブルのポインタ(ROMに保存)
NetBlock	0	&Func_NetBlock
FBlock 1	&NetBlock.pFBlockIDs.InstID[0]	&Func_FBlock1
FBlock 2	&NetBlock.pFBlockIDs.InstID[1]	&Func_FBlock2
…		
FBlock n	&NetBlock.pFBlockIDs.InstID[n-1]	&Func_FBlockn
FBlockシャドウ1	&InstIDShadow[0]	&Func_FBlockShadow1
FBlockシャドウ2	&InstIDShadow[1]	&Func_FBlockShadow2
…		
FBlockシャドウm	&InstIDShadow[m-1]	&Func_FBlockShadowm

表9.1:利用可能なFBlockとFBlockシャドウのテーブル
n – FBlockの数
m – FBlockシャドウの数(出典:[MNS L2])

コマンド インタプリタ(CMD)

コマンド インタプリタ モジュールは、受信した制御メッセージをテーブル制御で復号化し、ユーザ定義のFBlockハンドラ ファンクションを自動的に呼び出すために使います。

復号用のテーブルは、実装されている全てのFBlockとFBlockシャドウを含みます。このテーブルは表9.1に示した基本構造をしており、容易に拡張できます。

利用可能なFBlockのテーブルとファンクション テーブルはROMに保存できます。インスタンスIDは動作中に変わる事があるため、これらのIDを含むテーブル(表9.1の2列目)はRAMに保存する必要があります。初期化中に、このテーブルの全ての値が既定値に設定されます。3列目は、関連するファンクションのエントリアドレスを含みます。

ノーティフィケーション サービス(NTFS)

ノーティフィケーション サービス(NTFS)を使って、実装されているFBlockのプロパティが変化した時にステータス メッセージを他のデバイスに自動的に送信します。このメカニズムはセクション4.2.2で説明しました。

表9.2に、FBlockのノーティフィケーション マトリクスを示します。このマトリクスは、ノーティフィケーション サービスをサポートしている全てのFBlockに必要です。

FBlock x				
プロパティ0	プロパティ1	プロパティ2		プロパティ(n-1)
Flagfield 0	Flagfield 1	Flagfield 2	…	Flagfield (n-1)
Entry 0.1	Entry 1.1	Entry 2.1	…	Entry (n-1).1
Entry 0.2	Entry 1.2	Entry 2.2	…	Entry (n-1).2
Entry 0.3	Entry 1.3	Entry 2.3	…	Entry (n-1).3
Entry 0.4	Entry 1.4	Entry 2.4	…	Entry (n-1).4
…	…	…	…	
Entry 0.m	Entry 1.m	Entry 2.m	…	Entry (n-1).m

表9.2:FBlock xのノーティフィケーション マトリクス(出典:[MNS L2])
n – ノーティフィケーション サービスをサポートしているFBlockのプロパティの数
m – プロパティのノーティフィケーション サービスに登録できるデバイスの最大数

この表中のEntryという用語は、プロパティのノーティフィケーション サービスに登録したデバイスの論理またはグループアドレスを格納したデバイステーブルのインデックスです。

アドレスハンドラ(AH)

アドレスハンドラ(AH)はステートマシンに基づいて動作します。AHはシンボリック アドレスを実際のメモリアドレスに変換し、スレーブ内のデセントラル レジストリを管理します。レイヤIIの他のモジュール同様、AHもアプリケーション メッセージ サービス(AMS)と緊密に連携します。

コントローラは、どのデバイスが特定のFBlockを実装しているか知っている必要はありません。FBlockIDとInstIDが分かっていれば十分です。制御メッセージのターゲット アドレス フィールドは、ワイルドカード0xFFFFを格納します。アドレスハンドラ(AH)は、このワイルドカードを持つメッセージを受信すると、通常はまずデセントラル レジストリで該当するInstIDを持つFBlockIDを検索します(図9.7参照)。FBlockが見つかると、ターゲット アドレス フィールドを実際のターゲットアドレスに置き換えます。FBlockが見つからない場合、NetworkMasterシャドウ モジュールを用いてNetworkMaster内のセントラル レジストリでそのFBlockを問い合わせます。MOST NetServicesが目的のインスタンスをリング内の各デバイスに対して直接問い合わせるようなモードはMOST仕様では定義されておらず、バージョン3.xでは使用もサポートもされません。

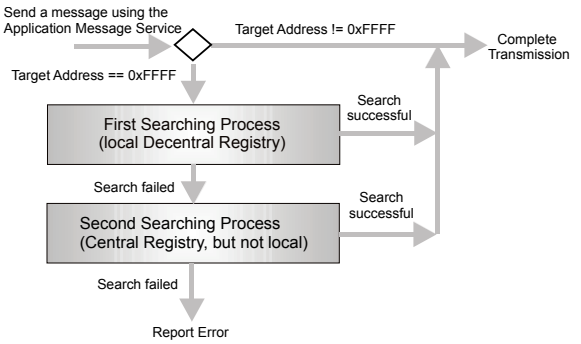


図9.7:アドレスハンドラの検索手順(出典:[MNS L2])

図9.8に、アドレスハンドラの処理の例を示します。HMIでCDチェンジャ(CDC)のトラック10を選択します。CDチェンジャ アプリケーションは、FBlock CD (InstID: 1)を持つデバイスにあります.トラックを変更するため、HMIは制御メッセージ`CD.1.Track.Set(10)`を送信します。HMIアプリケーションはアドレスハンドラにコマンド`0xFFFF.CD.1.Track.Set(10)`を送信し、アドレスハンドラが前述の手順に従ってワイルドカード0xFFFFをCDチェンジャのターゲット アドレスで置き換えます。MOSTネットワーク インターフェイス コントローラは自動的にソースアドレスを追加して、メッセージをCDチェンジャに送信します。CDチェンジャのMOSTネットワーク インターフェイス コントローラは、このメッセージをMOST NetServicesに送信します。CMDモジュールが、FBlock CDのプロパティTrackを変更するファンクションを呼び出します。

図9.8に、制御メッセージがセンダからレシーバへ届くまでの構文の概要を示します。

NetworkMasterシャドウ モジュール

NetworkMasterシャドウ モジュールは、FBlock NetworkMasterのプロパティをコピーしたものです。これはステータスおよび結果メッセージに応答し、通常はNetworkMaster以外の全てのデバイスに実装されます。基本機能は以下の通りです。

1. 初期化手順実行中の NetworkMaster.Configuration プロパティの管理
2. アドレスハンドラによるアドレス検索実行手順中の NetworkMaster.CentralRegistry プロパティの管理

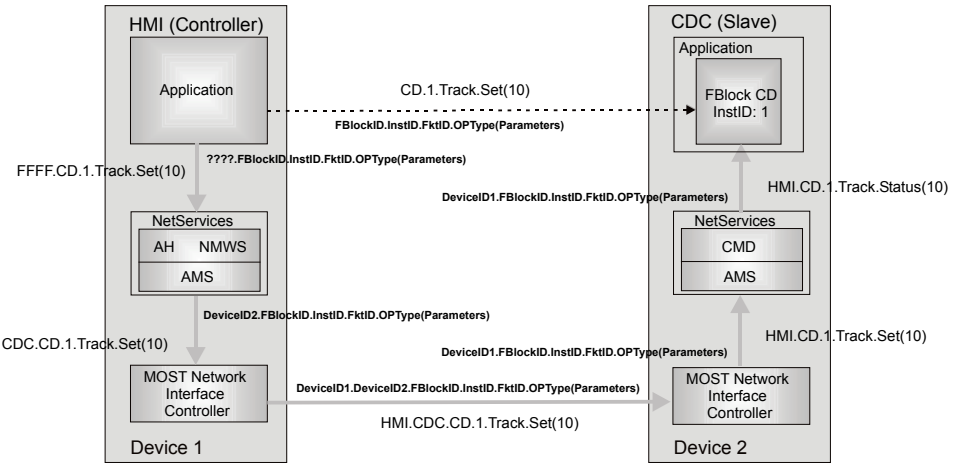


図9.8:アドレスハンドラの処理の例(出典: [MOST 2.5]、[MOST 3.0])

NetBlock (NB)

NetBlock (NB)はシステムFBlockの1つで、全てのデバイスへの実装が必要です。従って、NetBlockはMOST NetServicesレイヤIIに含まれます。NetBlockについてはセクション4.3.4で説明しました。

INICは最小限の機能を持つNetBlock (NBMIN)を内蔵しているため(セクション10.4.2参照)、レイヤIIバージョン2.xと3.xはNetBlockのNetBlock EHC (NBEHC)のみを実装します。

FBlock EnhancedTestability (ET)

FBlock EnhancedTestability (ET)もシステムFBlockで、セクション4.3.4で説明しました。

9.4 MOST NetServicesのターゲット プラットフォームへの実装

MOST NetServicesは標準ライブラリを一切使わずANSI-Cで書かれているため、ターゲット プラットフォームへの移植が非常に容易です。使用するハードウェアおよびソフトウェア プラットフォームに合わせた設定と変更は、ヘッダファイルadjust*.h内のコンパイラ スイッチで行います。これにより、幅広いプロセッサアーキテクチャに容易に対応できます。MOST NetServicesは、内蔵RAM容量が数KBの8ビット マイクロコントローラにメインループ アーキテクチャとして実装できる他、QNX、WinCE、Linux等の適切なマルチタスク ソフトウェア システムを利用した32ビット マイクロコントローラにも実装できます。

図9.3と9.4に示したローレベル ドライバ(LLD)は、ユーザが実装する必要があります。LLDは、MOST NetServicesに対してハードウェアを抽象化するために使います。OS8805とOS88558の2つのSoCコントローラは例外です。これらのSoCでは、MOST NICとホスト コントローラがワンチップに統合されています。ユーザはMOST NetServicesを設定するだけで済みます。

9.4.1 コールバック関数

MOST NetServicesのほとんどはステータスおよびエラーメッセージに対するコールバック関数をサポートしています。これらのコールバック関数はプログラマが利用する事ができ、特定のイベントまたはエラーが発生した場合にMOST NetServicesで呼び出されます。

MOST NetServicesの文書では、コールバック関数はキーワード[callback]で示されています。

```
[Callback]
declaration of function;
```

MOST NetServicesのコールバック関数は、バージョンによって種類が異なります。

MOST NetServicesレイヤI 1.x

MOST NetServicesレイヤI 1.xでは、タイプAとタイプBのコールバック関数を定義しています。

- タイプAのコールバック関数はプログラマが定義する必要がありますが、コールバック関数を呼び出すイベントがそのアプリケーションに関係ない場合はコールバック関数のボディは空のままでかまいません。
- これに対し、タイプBのコールバック関数はPowerOn等のシステム関連イベントを司るため、常に完全な実装が必要です。

MOST NetServicesレイヤI 2.x、3.x

MOST NetServicesレイヤI 2.xと3.xでは、未使用イベントに対するコールバック関数を宣言する必要がありません。MOST NetServicesが定義された挙動を保証します。コールバック関数には、モジュール固有の標準コールバック関数と汎用の標準コールバック関数の区別があります。

- モジュール固有の標準コールバック関数は、ある特定のイベントが発生した場合に1つのモジュールでのみ呼び出されます。
- これに対し、汎用の標準コールバック関数は複数のモジュールから呼び出す事ができます。これらは通常、ネットワーク インターフェイスとのやりとりの結果を示します。

MOST NetServicesレイヤII 1.x、2.x、3.x

MOST NetServicesレイヤI同様、レイヤIIでもタイプAとタイプBのコールバック関数を定義しています。また、バージョン1.xと2.xはタイプCのコールバック関数も定義しています。タイプCのコールバック関数は、ヘッダファイル(adjust*.h)でプログラマが明示的に有効にする必要があります。

9.4.2 メインループ アーキテクチャ

通常、マルチタスク システム ソフトウェアを持たない小規模なプロセッサでは、割り込みサービスルーチン(ISR)を除く全てのアプリケーションとサービスがソフトウェアのmain関数内のループによって周期的に呼び出されます。全てのアプリケーションとサービスについて、中心的なエントリポイントのファンクションが次に呼び出されるまでの最大応答時間、および次にメインループに戻るまでの最大実行時間を守るようプログラマが注意する必要があります。

MOST NetServices 1.xでは、サービスの周期的呼び出しは15 ms以内と決まっています。この時間を超えると、MSVの中心的なファンクション(例: ロックステータスのチェック)がMOST仕様に違反する可能性が出てきます。最大戻り時間は通常1 ms未満ですが、MOST NetServices自身がアプリケーションのコールバック関数を呼び出す場合は注意が必要です。この場合、最大戻り時間内にMOST NetServicesに戻ってメインループに戻る事ができるよう、アプリケーション プログラマが配慮する必要があります。

MOST NetServices 2.xと3.xでは、処理時間に制約のあるファンクションは全てINICが実行するようになっており、ホスト上で動作するMOST NetServicesは基本的にアプリケーション ソフトウェアとのインターフェイスのみを形成するため、この問題は比較的容易に対処できます。この場合でも、INICがウォッチドッグ タイマ(通常は0.5秒に設定)でホストとの通信を監視しているため、最大応答時間を守る必要があります。この期間内にホストから応答がない場合、INICはエラーと見なしてホストをネットワークから切断します。この結果、ネットワーク内でそのデバイスが提供する全てのアプリケーション(FBlock)の登録が解除されます。

メインループのエントリポイントとなるMOST NetServicesのファンクション以外に、MOST NetServicesにタイミングに関する情報を提供する他のファンクションを呼び出す必要があります。これらは通常、タイマ制御による割り込みサービスルーチン(MOSTTimerInt)でトリガされます。バージョン1.xでは、1、10、25 msの呼び出し間隔を選択できます。MOST NetServices 2.xと3.xは先進のタイマを実装しており、INICが提供するタイミング情報を使って、より柔軟な呼び出し間隔をサポートします。

9.4.3 マルチタスク アーキテクチャ

マルチタスク アーキテクチャの場合も、上記の条件が適用されます。ただし、メインループの代わりにタスクループを使います。通常、MOST NetServicesが動作

するループ内では、上記のエントリポイントのファンクションに加え、他のタスクまたはドライバとの通信に必要なファンクションのみを呼び出します。通常、他のサービスとアプリケーションにサービスを提供するファンクションは、別のタスクに実装します。

MOST NetServicesのタスクについては、MOST NetServices 1.xの最小呼び出し間隔を保証できるように優先度を選択する必要があります。一般に、これより高い優先度はリアルタイム機能(例: オーディオ/ビデオ信号処理、タイマ)を伴うタスク、またはタイミング要件の厳しいネットワーク ドライバ(例: CANネットワーク ドライバ スタック)のみに割り当てます。通常、実際のアプリケーションは優先度の低いタスクで実行します。

MOST NetServicesはANSI-Cで作成されているため、スレッドセーフではありません。つまり、APIファンクションは1つのコンテクストからしか呼び出せません。つまり、CMDモジュールのハンドラ ファンクションを呼び出してデータにアクセスするのは、各アプリケーションのコンテクストではなく、MOST NetServicesタスクのコンテクストで実行する必要があります。プログラマは、セマフォ等の適切なシステムソフトウェア同期メカニズムを使ってこの問題を解決する必要があります。この目的のため、MOST NetServices 2.xと3.xのコンフィグレーション ファイルには、プラットフォーム固有のMutex(相互排他)メカニズムに割り当て可能なマクロが用意されています。

複数のアプリケーションで構成される大規模なシステムでは、セントラル コマンド インタプリタの代わりに、AMSとMOST High Protocolから受信したメッセージをタスクを越えて渡すアーキテクチャを使う事が考えられます。また、個々のアプリケーション タスクのコンテクストで(CMDモジュールのファンクションでもある)メッセージの評価を実行する事も考えられます。このようなアーキテクチャはサードパーティのプロバイダも提供しています。

9.4.4 システム管理モジュール

NetworkMaster 、 PowerMaster 、 ConnectionMaster 等 の モ ジ ュ ー ル は MOST NetServicesに含まれません。これらは、MOST NetServicesと同じようにMicrochip社から参照実装としてライセンス供与を受ける事ができます。しかしMOST NetServicesとは異なり、これらはデバイスメーカーによる実装も広く使われています。

10 MOSTインターフェイス コントローラ

10.1 MOSTネットワーク インターフェイス コントローラとMOSTデバイス

MOSTデバイス内部では、MOSTインターフェイス コントローラがMOSTネットワークとデバイス内部を結ぶ双方向ブリッジとしての役割を果たします。図10.1に、代表的なシングル プロセッサ構成のMOSTデバイスの構造を示します。アプリケーション ハードウェアは、MOSTデバイスの特徴的機能の基盤となるものです。この図の例では、D/Aコンバータ(DAC)の後にアンプ段があり、このMOSTデバイスがアンプである事が分かります。

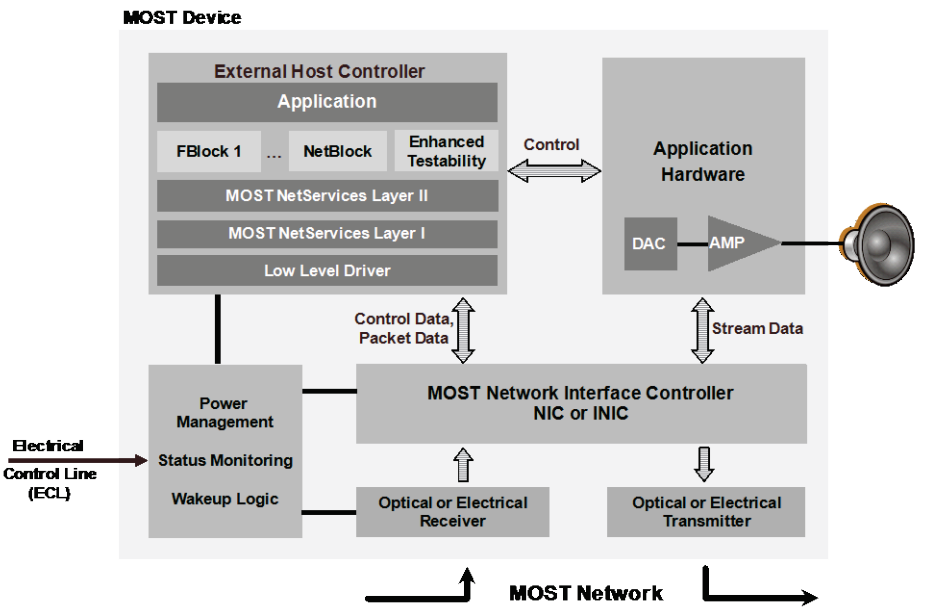


図10.:1:MOSTデバイスの構造

アプリケーション ハードウェアの制御やMOST関連ルーチンの実行等、ソフトウェア タスクは全て外部ホストコントローラ(EHC)が処理します。

MOSTネットワークに対しては、送信経路と受信経路が1つずつあります。受信経路では、MOSTネットワークから受信した信号をレシーバがMOSTインターフェイ

ス コントローラで処理できる形式に変換します。送信経路では、MOSTインターフェイス コントローラの出力を変換します。この変換は外部回路を使って行い、選択した物理層の種類に応じて下位の物理層仕様に準拠した信号を生成します。次に、光ドメイン(光物理層の場合)または電気ドメイン(電気物理層の場合)で信号を送信します。

ネットワーク上のアクティビティでMOSTデバイスを復帰できるようにするため、レシーバは復帰ロジックに接続されます。

MOSTインターフェイス コントローラは、アプリケーションとMOSTネットワークに接続するために必要な標準インターフェイスを提供します。これにより、ストリーミング データ、制御データ、パケットデータ、アイソクロナス データを効率よくMOSTネットワークで送受信できます。MOST側では、MOSTインターフェイス コントローラが物理層からビットストリームを受信し、この信号をリフレッシュした後、必要に応じてデータを変更して送信経路から信号を送出します。リフレッシュは物理層での信号劣化を補償する処理で、シグナル インテグリティを回復して最大限の転送結果を得るために行います。

MOST側の信号処理で特に重要なのが同期です。MOSTインターフェイス コントローラは、TimingMasterとして設定されている場合システム全体で使うクロックを生成し、それ以外の場合は受信したビットストリームに同期します。こうして、MOSTネットワーク内の全てのMOSTデバイスが同期します。これにより、オーバーヘッドを最小限に抑えながら効率よく高品質の信号を転送できます。

デバイスの復帰、電源管理、ステータス監視は、起動、スリープ中の消費電力、電源の例外処理に関してMOSTデバイスがMOST仕様に準拠する事を保証します。

10.2 MOSTネットワーク インターフェイス コントローラ ファミリ

MOSTインターフェイス コントローラには、インテリジェント ネットワーク インターフェイス コントローラ(INIC)とネットワーク インターフェイス コントローラ(NIC)の2種類があります。NICはインターフェイス コントローラの第1世代の実装です。いくつかの理由により、最近のMOSTシステムとデバイスの開発ではNICの代わりにINICが使われます。その最も大きな理由の1つは、INICを使った方がシステムレベルで堅牢性が向上し、簡略化が可能である事です。

10.2.1 INIC

INICは、MOSTネットワークとその管理に対してそれ自体がシンプルなMOSTデバイスとして動作します。システムの起動時間を短縮し、MOSTシステムをなるべく早く安定させるように、INICはEHCから独立して自律的に動作します。この事は、特にEHCのアプリケーション起動に時間がかかる場合、または通常の動作中にアプリケーションでエラーが発生した場合に大きなメリットです。

INICでは、ネットワーク全体の動作を最優先しています。INICは、ネットワークを制御された方法で起動し、ソフトウェア エラー (リセットまたはハングアップ)が発生した場合は悪影響からネットワークを保護します。この時の動作モードを保護モードと呼びます。INICのハードウェア リセット後と起動中も、このモードが適用されます。

INICを使ったMOSTデバイスの基本的な性能は、INICで決まります。EHCとその上で動作するアプリケーションはINICの起動とは独立して起動し、初期化が完了するとINICにログオンします。

INICの制御には、共通のAPIを使います。EHCとINICの間の通信は、ポートメッセージ プロトコルに基づいて行います。INICの機能は、一般的なMOSTデバイス同様にファンクション ブロックとしてモデル化されています。ストリーミングデータのルーティングは、ソケットと接続で容易に制御できます。アプリケーション自体は直接INIC APIを使う必要はなく、MOST API (MOST NetServices)に基づいて動作します。MOST NetServicesは、ポートメッセージ プロトコルを用いてINIC API経由でINICを制御します。MOSTの設計思想に従い、INICの制御はFBlock INICと呼ばれるファンクション ブロック(FBlock)を使って体系化されています。

INICには、MediaLBと呼ばれるストリーミング、制御、パケットデータ用の高速シリアル同期インターフェイスがあります。このインターフェイスは、例えばプリント基板上で複数のオーディオ部品を接続し、ストリーミング データ等を供給する場合に使います。INICは、MOSTネットワークに同期したMediaLBの基本クロックを駆動します。このインターフェイスは全てのINICに実装されます。

これ以外に、I²S等の標準インターフェイスを使うと、アプリケーション ハードウェアとINICを接続できます。

これらの理由により、最近のMOSTデバイスとMOSTベースのマルチメディア システムの実装ではNICよりもINICを使う方が主流です。

10.2.2 NIC

ネットワーク インターフェイス コントローラ(NIC)は、MOSTの第1世代の通信ICです。その後、より堅牢性の高いシステム設計と短期間での製品化が可能なINICが登場しました。NICはMOSTデバイス内でデータリンク層の機能を受け持ちます。NICは、レジスタウォールと呼ばれるレジスタ構造の個々のストレージ位置に読み書きを実行して制御します。

NICを実装したMOSTデバイスの基本的な挙動は、NICと協調してEHCが完全に司ります。実装の際は、アプリケーションの起動時間を十分に短くし、MOST仕様に違反しないように注意する必要があります。NICベースのデバイスの場合、アプリケーション ドメインでエラーが発生するとシステム全体に影響が及ぶ可能性があります。この点を考慮して開発されたのがINICです。原則として、INICはこうしたアプリケーション エラーをローカルに封じ込めます。

ストリーミング データのルーティングは、MOSTルーティング テーブル(MRT)のレジスタで制御します。代表的なNICに、MOST25のOS8104Aがあります。

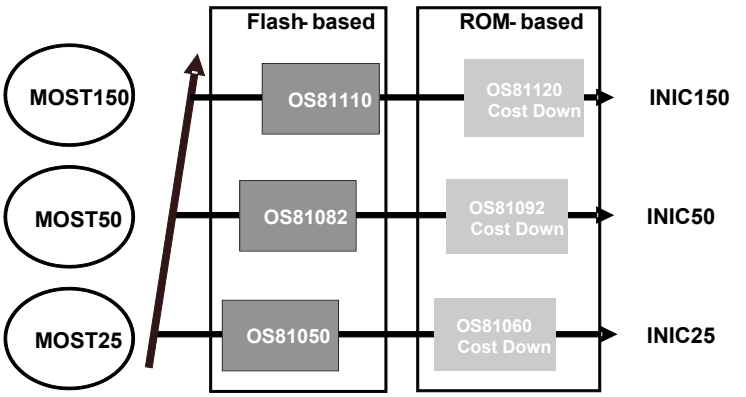


図10.2:INICファミリの概要とロードマップ

10.3 INICファミリ

10.3.1 メンバー

MOST25、MOST50、MOST150の3つのスピードグレードのそれぞれに、2種類のINICを提供しています(図10.2参照)。

1つはフラッシュメモリを使ったもので、もう1つはフラッシュメモリより低コストなROMを使ったものです。フラッシュメモリ ベースのINICでは、起動時の設定データ (コンフィグレーション ストリング)とINICのファームウェア全体を書き換える事ができます。ROMベースのINICでは、ファームウェアをROMに格納し、コンフィグレーション情報をOTPメモリに保存します。OS81092はOTPメモリ空間が2つの部分で構成されているため、コンフィグレーション ストリングを2回書き込めます。

ただし、MOST150は現時点でフラッシュメモリ ベースのOS81110のみを提供しています。低コスト版のOS81120は計画中の段階です。

10.3.2 特長の概要

表10.1に、INICファミリの各メンバーの主な特長をまとめます。表10.1に示した全てのインターフェイスを同時に利用できるわけではない事に注意してください。ピン数を抑えるためにインターフェイスの一部は共用ピンとなっており、どのインターフェイスを利用できるかは構成で決まります。

機能	データタイプとインターフェイス	INIC25 OS81050	INIC25 OS81060	INIC50 OS81082	INIC50 OS81092	INIC150 OS81110
サポートされるMOSTデータタイプ	制御、同期、非同期(レガシー)パケット	X	X	X	X	X
	アイソクロナス、Ethernet					X
インターフェイス(本文参照)	I²C、I²S、MediaLB 3ピン	X	X	X	X	X
	MediaLB 5ピン(レガシー)	X	X	X	X	
	MediaLB 6ピン、TSI、SPI					X
電源		3.3 V / 2.5 V	3.3 V / 1.8 V	3.3 V / 2.5 V	3.3 V / 1.8 V	3.3 V / 1.8 V
パッケージ		ETQFP 44 QFP 44	QFN 40	ETQFP 64	QFN 48	QFN 48

表10.1:INICファミリの特長

10.4 INICへのインターフェイス

10.4.1 利用可能なインターフェイス

MOSTネットワークのスピードグレードに関わらず、INICファミリの全体的な構造は全てのメンバーでほぼ共通です。図10.3の中央に示したINICプロセッサは、MOSTネットワークとの間でデータを送受信し、周辺インターフェイスとの間で双方向のルーティングを実行します。

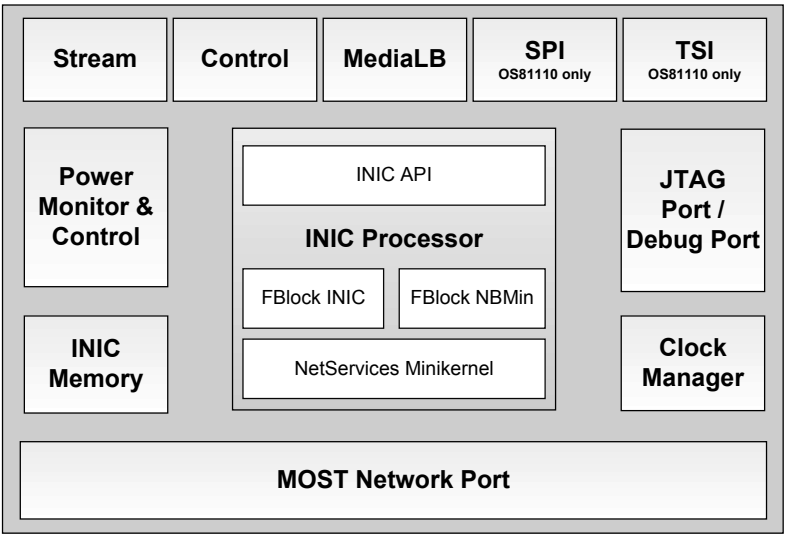


図10.3:INICのブロック図

クロックマネージャは、チップ全体、アプリケーション ハードウェア、そしてINICがTimingMasterとして設定されている場合はネットワーク全体にクロックを供給します。これにより、全体の同期を確保します。

電源監視および制御モジュールは、外部の電源管理回路との接続を確立します。これにより、INICは電源の変化に対してMOST仕様に準拠した対処をします。

JTAG/デバッグポートを利用すると、INICをバウンダリ スキャンテストに含める事ができます。このインターフェイスは、INICへのデバッグレベルでのアクセスをサポートします。このポートにINIC Explorerを接続すると、起動時の設定データとプロパティ値、ストリーミング データのルーティング情報、インターフェイスの設定情報等の内部データを取得できます。フラッシュメモリ ベースおよびROMベースのINICにコンフィグレーション ストリングを書き込む事ができるのも、非常に便利です。JTAG/デバッグポート用のコネクタを量産品に実装しない場合でも、基板パターンのみは作成しておく事を推奨します。

図10.3に示したINICメモリブロックで実際に使うメモリには、いくつかの種類があります。このメモリには、常時変化するデータの他、起動時の設定データ(コンフィグレーション ストリング)、そしてシステムをシャットダウンしてから次回起動するまでのスリープ中に保存しておく必要のある特別なデータを格納します。

A/Dコンバータ(ADC)、D/Aコンバータ(DAC)、DSP等の外付けデバイスは、ストリーミング ポートに接続します。ストリーミング ポートは、業界標準の各種フォーマットでストリーミング データをソースまたはシンクできます。

全てのINICには、ストリーミング、制御、パケット、アイソクロナス データ用のMediaLBインターフェイスが1つあります。このインターフェイスを使うと、プリ

ント基板上の各種オーディオ部品またはビデオ部品を、INICをブリッジとしてMOSTネットワークに接続できます。MediaLBのクロッキングはMOSTに同期しているため、MOSTの同期ネットワーク アプローチがPCBレベルでも維持されます。MediaLBで利用可能なデータの種類と利用可能なMediaLBインターフェイスタイプ(MediaLB 3ピン、MediaLB 5ピン、MediaLB 6ピン)は、使用するINICとその設定で決まります。

通常、衛星チューナ、レシーバ、ディスプレイ、ビデオデコーダ等の産業用デバイスはTSI(トランスポート ストリーム インターフェイス) データストリームを扱います。MOST150用のINIC OS81110には、これらデバイスを接続してMOSTネットワークでTSIストリームを転送するための専用インターフェイスとしてTSIポートがあります。

マイクロコントローラと周辺デバイスには、シリアル ペリフェラル インターフェイス(SPI)を使うものが多くあります。OS81110は、標準4ピン インターフェイスのSPIポートを実装しています。このインターフェイスはフロー制御用に別の割り込みピンを1つ使い、非同期パケットデータまたはQoS IPパケット(アイソクロナス)を転送できます。

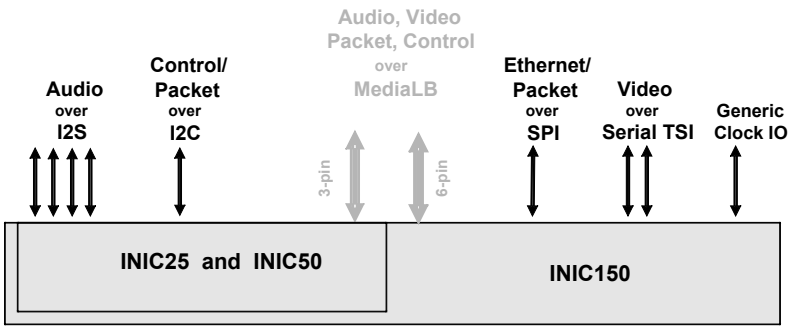


図10.4:INIC接続の概要図

I²Cベースの制御ポートを使うと、EHCとINICの間で直接通信できます。このポートは、メッセージベースのプロトコルでデータを転送します。EHCは制御ポートを介してFBlock INICにアクセスし、レガシーのパケットデータとMOST制御メッセージ(MCM)を送受信できます。

全てのINICは、それぞれの設計されたスピードグレードでMOSTネットワークへの接続を確立します。この役割をになうのがMOSTネットワーク ポートです。従って、MOSTネットワーク ポートは特定のMOSTネットワークへのインターフェイスを表します。このポートは、MOSTネットワークのクロックリカバリ、受信データの復号化と送信データの符号化を実行し、最大限の転送品質を実現するために信号をリフレッシュしてから出力します。

10.4.2 ソフトウェアから見たインターフェイス

全体の構造

INICの全ての内部機能には、ポートメッセージ プロトコルを使って、INIC API経由でアクセスできます。MOSTデバイスの全般的なアプローチに従い、INIC固有の内部機能は1つのファンクション ブロック(FBlock INIC)としてモデル化されています。ファンクション ブロックNetBlockMin (NBMIN)を使うと、アプリケーション ソフトウェアが動作していなくてもINICが自律的にネットワークに対して動作できます。MOST NetServices MiniKernelは特に重要なMOST NetServicesの機能を持ち、EHCに代わってリアルタイム方式の監視タスクを実行します。これも、INICがEHCから独立して動作できる要因の1つです。アプリケーション自体は直接INIC APIを使わず、MOST API (MOST NetServices)に基づいて動作します。MOST NetServicesは、INIC APIとポートメッセージ プロトコルでINICを制御します。

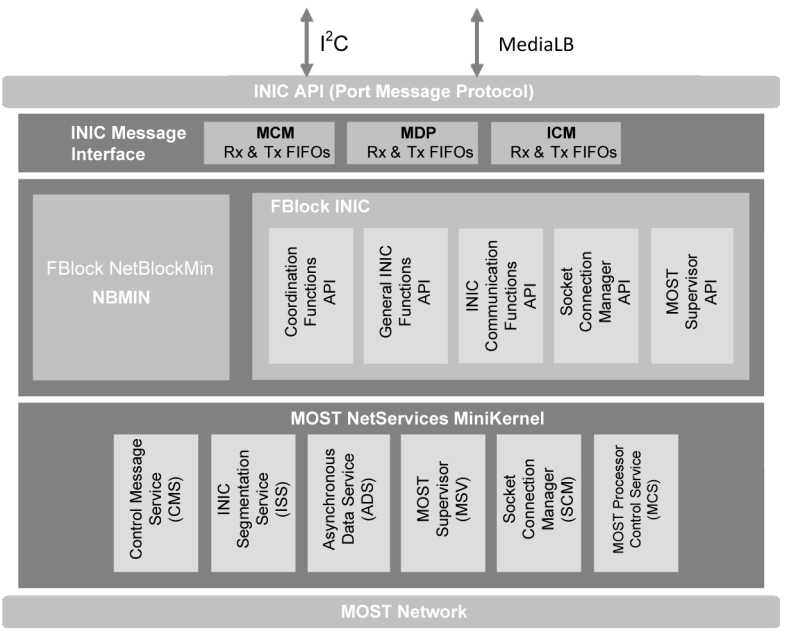


図10.5:INICソフトウェア スタックの概要

INICメッセージ インターフェイス

EHCとの通信に関して、INICの内部構造は3つに分かれています。INICメッセージ インターフェイスは、3種類のメッセージにそれぞれ対応した3つのバッファで構成されています。

MOST制御メッセージ(MCM)は、制御レベルの通信のためにMOSTネットワーク上でやりとりされるMOST制御メッセージに対応します。

MOSTデータパケット(MDP)とMOST Ethernetパケット(MEP)はどちらもパケットデータに対応します。MDPはMOST25の実装以来MOSTネットワークで利用可能なレガシーのパケットデータに属し、MEPはEthernet方式のデータ転送に対応したパケットデータを含みます。

INIC制御メッセージ(ICM)はMCMと同じ構造をしていますが、FBlock INICとの通信にのみ使います。EHCは動作中にICMを使ってINICの制御と設定を行い、INICはICM経由でステータス レポートを返す等の応答をします。

FBlock NBMINとFBlock INIC

この章の冒頭で述べた通り、INICはMOSTネットワークとその管理に対してそれ自体がシンプルなMOSTデバイスとして動作します。全てのMOSTデバイスにファンクション ブロック(FBlock) NetBlockの機能を含める事が必須とされているため、NICも最小限の機能を持つNetBlock、すなわちNetBlockMin (NBMIN)を内蔵しています。NBMINとアプリケーションのNetBlock EHC (NBEHC)を組み合わせると、通常のNetBlockになります(図10.6参照)。

EHC、MOST NetServices、アプリケーションが存在する場合、NBMINはプロキシのような役割を果たします。INICがEHCから独立して動作する必要がある場合はNBMINが全ての役割を引き受け、MOSTデバイスがファンクション ブロックを持たないデバイスとして動作します。

FBlock INICは、INICのハードウェア機能とMOST NetServices MiniKernelを制御するだけのリーン インターフェイスです。FBlock INICを使って、INICのノーティフィケーション機能、バージョン情報等の各種情報、ストリーミング データの制御メカニズム等へアクセスできます。

MOST NetServices MiniKernel

MOST NetServices MiniKernelの各ルーチンは、MOST制御メッセージやデータパケット等を処理します。これらのルーチンは起動時にINICを設定し、ストリームデータのルーティング等を制御します。MOSTスーパバイザ(MSV)は、MOST NetServices MiniKernelで重要な役割を果たします。MSVは動的ネットワーク イベント、ネットワーク監視、電源管理、エラーを処理します。全般に、MSVはMOSTネットワークのコンテクストでINICの挙動を制御し、従ってINICの起動とシャットダウン、TimingMasterまたはTimingSlaveとしての設定等を実行します。

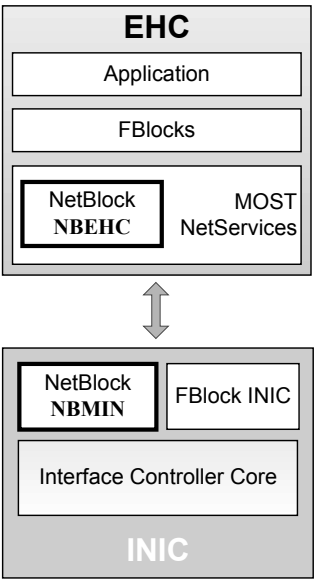


図10.6:NetBlock NBMINとNetBlock NBEHC

10.4.3 ポート、ソケット、接続

INICから見ると、周辺インターフェイスとMOSTネットワークはポートを介して接続されます。起動時の設定の一部はハードウェア ピンで行いますが、ほとんどはINIC APIを経由してFBlock INICのプロパティとメソッドにアクセスして行います。MOSTネットワークへのゲートウェイとなるネットワーク ポートはMOSTで扱う全てのデータタイプを転送できますが、他のポートは特定のデータタイプ(1つまたは複数)の転送に限られます。設定によって、ポートのハードウェアは複数のデータストリームを異なる方向に移動できます。

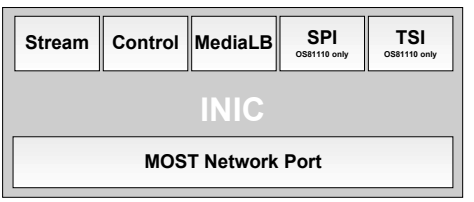


図10.7:INICのポート

ポートは開く必要があります。一部のポートは、INICの設定に基づいて自動的に開きます。それ以外のポートは、INIC APIのOpenPortファンクションにアクセスして開きます。

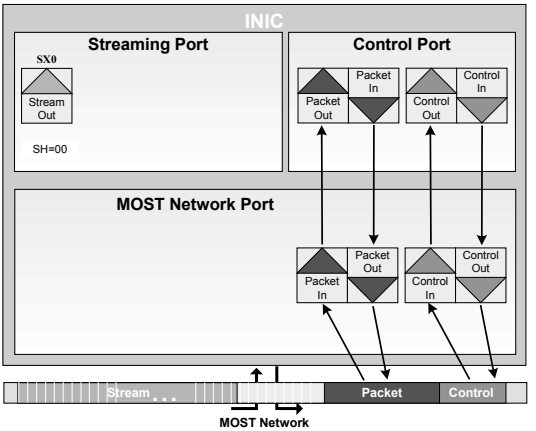


図10.8:MOSTネットワーク ポートのソケット(MOST25)。SH = ソケットハンドル

どのデータタイプにどちらの方向(INまたはOUT)で直接アクセスするかは、ソケットで定義します。

Note:データ方向は常にINICからの視点で記述します。データがINICの内部へ移動する場合の方向をIN、INICから出て行くの場合の方向をOUTと表記します。

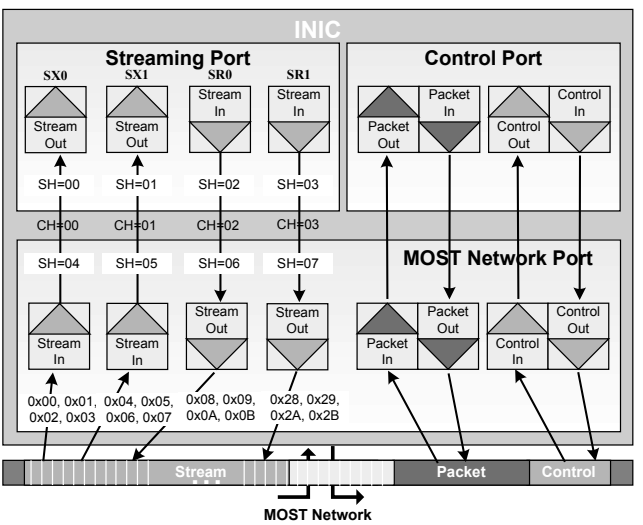


図10.9:ポート、ソケット、接続

ソケットが生成されると、INICは確認メッセージとソケットハンドル(SH)と呼ばれるIDを返します。接続を確立するとデータフローが可能になります。接続を確立するファンクションに必要なのは、INソケットのSHとOUTソケットのSHのみです。

接続を確立すると、接続ハンドルが生成されます。このハンドルは、接続の管理と接続の解除に必要です。

Note:接続できるソケットは、データタイプとパラメータ設定が共通のINソケットとOUTソケットに限られます。

10.4.4 ポートメッセージ プロトコル

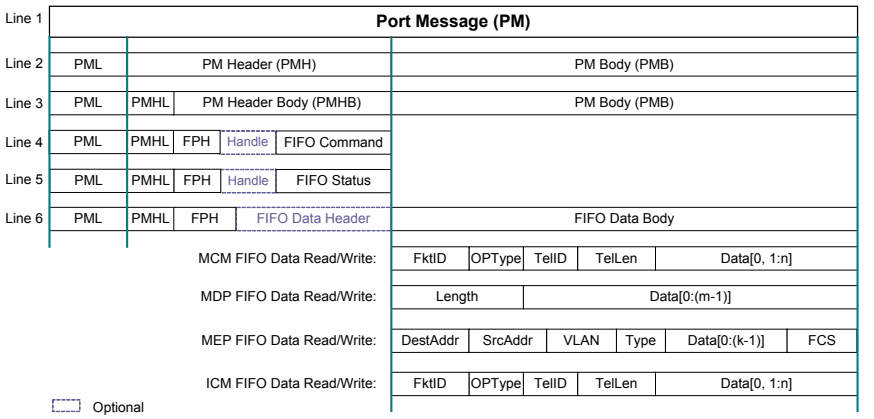


図10.10:ポートメッセージ プロトコルの構造

INICとEHCの間のメッセージ交換は、ポートメッセージ プロトコルに基づいて行います。図10.10に、ポートメッセージ プロトコルの要素を示します(一部要素は任意実装)。ポートメッセージ(PM、図10.10の「Line 1」参照)は、いくつかの部分で構成されています。

ポートメッセージ長(PML、図10.10の「Line 2」参照)は、PMヘッダ(PMH)とPMボディのバイト数を表す16ビットの値です。

PMHは、8ビット値のPMヘッダ長(PMHL)とPMヘッダボディ(PMHB)で構成されています(図10.10の「Line 3」参照)。PMHLはPMHBのバイト数を表します。PMHBは、FIFOコマンド(図10.10の「Line 4」参照)、FIFOステータス情報(図10.10の「Line 5」参照)、後続データに関する情報(図10.10の「Line 6」参照)のいずれかを含みます。

FIFOプロトコルヘッダには、利用可能なメッセージタイプの種類を区別する重要な役割があります。

- FIFOコマンド メッセージ
  INICのインターフェイス再同期または失敗したメッセージの再送に使用
- FIFOステータス メッセージ
  ハンドシェイク、およびINICの初期化完了を示すために使用

- MOST制御メッセージ(MCM)
- MOSTデータパケット(MDP)
- MOST Ethernetパケット(MEP) — OS81110のみ
- INIC制御メッセージ(ICM)

PMにPMボディが含まれる場合、この中にはメッセージデータ本体が特定の構造に従って格納されます。

10.5 保護、封じ込め、EHC

10.5.1 INICとEHCの連携

望ましくない影響がネットワークに波及するのを防ぐため、外部ホスト コントローラ インターフェイス ステートマシン(EHCIステート)と呼ばれるステートマシンでINICとEHCの連携を制御します。

EHCIステートマシンは、以下の3つの状態をとる事ができます(図10.11参照)。

- EHCI-Protected
- EHCI-Semi-Protected
- EHCI-Attached

EHCI-Protected

リセット後、INICの基本ステートはEHCI-Protectedです。これにより、起動中のアプリケーションまたは正しく動作していないアプリケーションがMOSTネットワークにアクセスするのを防ぎます。アプリケーションはEHCI-Semi-Protectedステートを経て、最終的に通常の動作モードであるEHCI-Attachedステートに到達します。

EHCI-Protectedステートでは、EHCはMOSTネットワーク内の他のノードにメッセージを送信できません。また、MOSTバスからEHCがメッセージを受信する事もできません。安全性に関するINICのコンセプトを実現するには、INICでアプリケーションを監視する仕組みが必要です。このため、INICにはいわゆるウォッチドッグメカニズムがあります。EHCI-Protectedステート中、INIC APIを用いてウォッチドッグの設定とEHCIステートマシンの状態変化を実行します。

INICとEHCは、互いにハンドシェイク手順で相手の準備が完了したかどうかを確認します。両方の準備が完了すると、EHC上で動作しているアプリケーションが必要に応じてウォッチドッグ メカニズムを設定し、次のステートへの遷移を開始します。

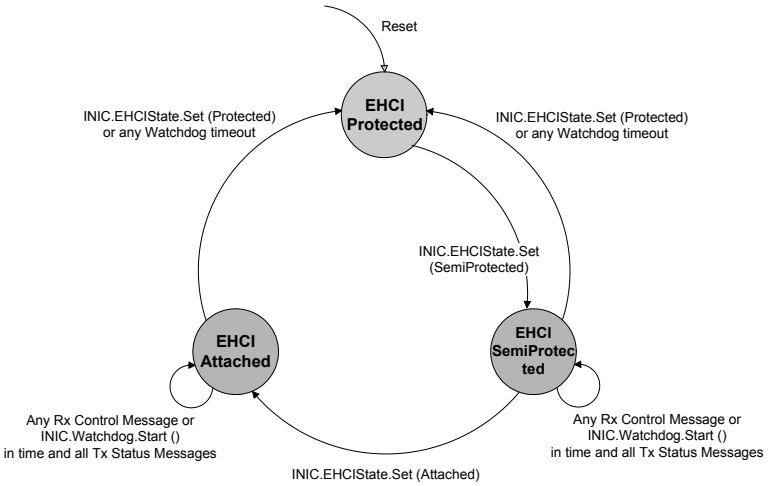


図10.11:EHCIステートマシンのステート

EHCI-Semi-Protected

EHCI-Semi-Protectedステートでも、MOSTネットワークとの通信はできません。しかし、FBlock INICには完全にアクセスできます。これは、MOSTデバイスのハードウェアを完全に初期化するには、連続したクロッキングを必要とする接続の起動も必要になるという考え方が基となっています。

従って、このステートではEHCはストリーミング ポート等のポートを設定して開く事ができます。ポートを開くと、該当するハードウェア インターフェイスが動作を始め、外部同期用のクロック供給も開始します。

全てのハードウェア初期化が完了すると、EHCI-Attachedステートへの遷移を実行できます。

EHCI-Attached

EHCI-Attachedステートでは、EHCはMOSTネットワークの全てのデータタイプに対して完全に双方向でアクセスできます。アプリケーションもネットワークから完全に利用でき、利用可能なファンクション ブロック(FBlock)をNetworkMasterに登録しておく必要があります。

EHCIステートとNetBlockの関係

EHCI-ProtectedステートとEHCI-Semi-Protectedステートでは、ネットワークからの管理クエリ(問い合わせ)に対してNBMINが主に応答します。しかしアプリケーションの動作準備が完了するまでは未知の情報もあるため、一部のクエリには空のリストまたはエラーメッセージを含む確認応答を返します。

EHCI-Attachedステートでも、対応する管理クエリに対してはNBMINが応答します。しかし、EHCとそのNetBlock (NBEHC)に対するアプリケーション関連のクエリはアプリケーションに渡されます。

10.5.2 ウォッチドッグ機能

一般に、ウォッチドッグ回路はアプリケーションの誤動作を監視する目的で使います。ウォッチドッグを使うには、通常動作時に周期的に発生する信号をアプリケーションに実装しておきます。この信号をウォッチドッグ回路に入力します。信号が発生しない場合、アプリケーションの誤動作と解釈できます。この場合、独立したウォッチドッグ回路でハードウェア リセットをトリガし、アプリケーション再初期化します。

INICに実装されたウォッチドッグ メカニズムでは、INICとアプリケーションの間で交換されるメッセージがこの信号の役割を果たします。監視機能は、送信と受信の各方向に個別に実装されます。ウォッチドッグの間隔は、アプリケーションによって初期化中に設定され、送受信両方に適用されます。

EHCからデータを受信する場合、INICは設定した期間内にアプリケーションから有効なメッセージが定期的に送信されているかどうかを常時監視します。通常、アプリケーションがMOSTネットワークと正常に通信していればウォッチドッグに必要な信号が発生します。ネットワークとの通信が不要な場合でも、アプリケーションはウォッチドッグに必要な信号を生成し続けます。

送信方向の場合、INICからアプリケーションへメッセージを送信した場合のみ監視を実行します。この場合、INICは送信メッセージに対する受信確認が指定した期間内に到着するかどうかを監視します。

期間内に到着しない場合、INICはEHCIステートマシンをEHCI-Protectedステートに遷移させて、アプリケーションの誤動作による影響からネットワークを保護します。また、INICにはアプリケーション専用リセット出力があります。アプリケーションの設定でこのオプションを有効にすると、INICはEHC等のハードウェア リセットをトリガします。

10.6 コンフィグレーション ストリングによる設定

通常、EHCの起動が完了するにはある程度の時間がかかるため、INIC内部にはコンフィグレーション ストリングがあり、この中に格納した値を利用して、MOSTデバイスの起動プロセスのなるべく早い段階でINICのハードウェアおよびファームウェアの初期化を制御できます。

Accessible Parameter	Factory Default
NBMIN.NodeAddress.NodeAddress	0xFFFF
NBMIN.GroupAddress.GroupAddress	0x03C8
NBMIN.PermissionToWake.WakeStatus	Off
NBMIN.RetryParameters.RetryTime	11
NBMIN.RetryParameters.RetryNumbers	6
INIC.VersionInfo.ConfString (Major)	0
INIC.VersionInfo.ConfString (Minor)	0
INIC.VersionInfo.ConfString (Release)	0
INIC.RMCK.Divider	Disabled
INIC.RemoteAccess.AccessMode	False
INIC.WatchdogMode.AutoShutDownDelay	65535
INIC.AddressConfig.Mode	0x00
INIC.PortConfiguration.MediaLBClockCfg	512Fs
INIC.PortConfiguration.I²CSlaveAddress	0x40
INIC.PortConfiguration.DefPort	INT_Select
INIC.PortConfiguration.PortVariantCfg	Configuration 7
INIC.PortConfiguration.SerialInterfaceCfg	InOut
INIC.SCMConfig.SCMCfg	0x00
INIC.Bandwidth.AssignBWInit	80
INIC.DeviceMode.DeviceMode	Slave
INIC.RBDOptions.Options	0x01
INIC.NIWakeUpMode.Mode	0x00
INIC.PMIConfig.Config	0x00
INIC.PMIConfig.TimePwrOff	5 s
INIC.UseStatusPin	True

図10.12:INIC OS81110のコンフィグレーション ストリングの例

コンフィグレーション ストリングはECHまたはINIC Explorerから編集できます。ROMベースのINICでは、コンフィグレーション ストリングへの書き込みを2回実行できます。フラッシュメモリ ベースのINICでは、必要に応じて何度でもコンフィグレーション ストリングを編集できます。

10.7 制御ポート

制御ポートはI²Cバスに基づいており、クロックライン、データライン、割り込みラインを使ってデータをシリアル転送します。INICはI²Cスレーブとして動作し、クロック ストレッチをサポートします。

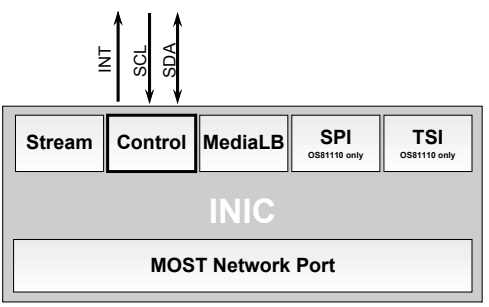


図10.13:INIC制御ポート

データが利用可能になると、INICは割り込みピン(INT)を使って通知します。制御ポートは、INICとEHCの間で制御データとパケットデータを転送します。また、制御ポートを使ってファームウェアとコンフィグレーション データをINICに書き込む事もできます。制御ポートに使うI²Cアドレスは、コンフィグレーション ストリングで指定します。

10.8 ストリーミング ポート

ストリーミング ポートはポイント ツー ポイントのシリアル接続で、INICとマルチメディア ソースまたはシンクと接続します。このポートはMOSTネットワークに同期し、方向、データレート、データ フォーマットを設定可能な最大4本のシリアルデータ ピンを提供します。ストリーミング ポートは、INIC APIで設定します。

このポートは数種類のデータ フォーマットをサポートしており、外部ハードウェアをMOSTクロックに同期させるためのクロックピンが2本あります。最大クロック速度は512 Fsです(OS81110のみ)。

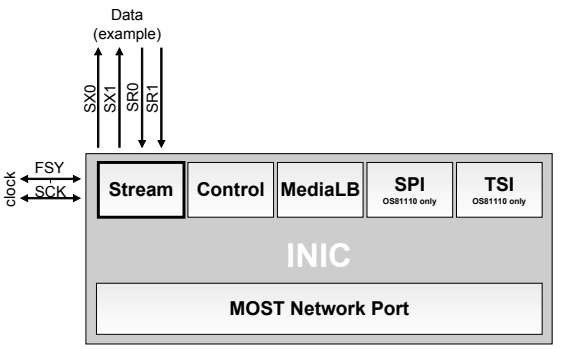


図10.14:INICストリーミング ポート

ストリーミング ポートでは、全てのデータ信号が1本のクロックピン(SCK)と1本の同期ピン(FSY: LeftとRrightの区別に使用)を共用します。FSYとSCKを出力として設定した場合、これらのピンはストリーミング ポートでのシリアルデータ転送を駆動します。この場合、外部に接続した周辺部品をFSYとSCKに同期させる必要があります。FSYとSCKを入力として設定できます。その場合、これらの信号はINICのRMCK (Recovered Master Clock)出力を使って同期させる必要があります。

ストリーミング データをINICから周辺部品へシリアル転送するINIC側の接続をSX0およびSX1と呼びます。アプリケーションから見ると、これらはストリーミング データのソースです。同様に、周辺部品からINICへデータを転送するデータシンクとして、SR0とSR1があります。

10.9 SPIポート

図10.15に、OS81110のSPIポートの実装を示します。このポートは、SPIスレーブとして最大クロックレート25 MHz (SCLKn = 512*Fs @ 48 kHz)で動作します。

INICは、データ転送の方向に関する情報をSPIバスマスタからSDIN経由で受信します。

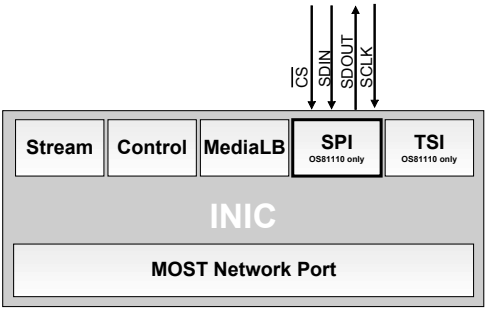


図10.15:INIC SPIポート

10.10 トランスポート ストリーム インターフェイス (TSI)

OS81110には、最大ビットレート50 Mbps (1024*Fs)で動作可能な4ピンのシリアルTSIインターフェイスがあります。このインターフェイスはTSIマスタモード(送信)またはTSIスレーブモード(受信)をサポートしています。

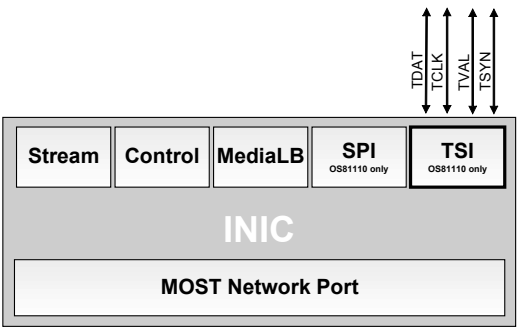


図10.16:INIC TSIポート

TSIポートのパケット構造は188バイトのパケットサイズを使います。OS81110には最大2つのTSIポートがあり、同時に動作させる事ができます。

10.11 MediaLBポート

メディア ローカル バス(MediaLB)は、あらゆるタイプのマルチメディア データに加え、アプリケーション制御用のデータを送信できるように開発されました。詳細

は、最新のMediaLB仕様[MediaLB]を参照してください。このため、よく使われる専用性の高いシリアル インターフェイスとは性格が異なっています。MediaLBはMOSTバスに同期した伝送媒体です。ボード内の占有面積とピン数を抑えてプリント基板上の各種ICを接続するのに使います。

MediaLB信号名	MediaLB物理インターフェイス		
	3ピン (シングルエンド)	5ピン (シングルエンド)	6ピン (差動)
MediaLBクロック (MLBC)	MLBCLK	MLBCLK	MLBCP MLBCN
MediaLB信号 (MLBS)	MLBSIG	MLBSI MLBSO	MLBSP MLBSN
MediaLBデータ (MLBD)	MLBDAT	MLBDI MLBDO	MLBDP MLBDN

表10.2:MediaLBの信号名

MediaLBは3ピン、5ピン、6ピンのいずれかで実装できます。5ピンのMediaLBは、単方向のピン回路しか持たない部品にMediaLBを実装できるように定義されたものです。しかしこの実装では基板面積が増大し、ロジック回路を追加する必要があり、最大速度が低下します。実際には3ピンモデルが最もよく使われ、次に多いのが6ピンMediaLBです。MediaLBは以下の信号を使います。信号名の詳細は表10.2を参照してください。

- MLBCLK (MediaLBクロック、単方向)
- MLBSIG (MediaLB信号、双方向)
- MLBDAT (MediaLBデータ、双方向)

MLBCは、MediaLBシステム全体にタイミングを供給します。これはMediaLBコントローラが生成し、複数のクロックレートをサポートします。MLBCはMOSTネットワークに同期し、全てのMediaLBデバイスに対する入力となります。この原則は、全てのMediaLB実装に共通です。図10.17に、3ピンMediaLBの実装での配線を示します。6ピンMediaLBには、MLBC信号の周波数をPLLを使って内部で逓倍するモードがあります。この場合、実際のデータ転送は内部で復元したクロックに基づいて、高い動作周波数で行われます。このため、外部クロックと実際に使うクロックには差があります。

実際の MLBCLK	MediaLB 3ピン	5ピン	6ピン	1フレーム当たりのクワドレット数	物理チャンネル	物理チャンネル数 (アプリケーションで利用可能な数)
256*Fs	X	X		8	PC0 - PC7	7
512*Fs	X	X		16	PC0 - PC15	15
1,024*Fs	X			32	PC0 - PC31	31
2,048*Fs			X	58	PC0 - PC57	57
3,072*Fs			X	87	PC0 - PC86	86
4096*Fs			X	117	PC0 - PC116	116
6144*Fs			X	161	PC0 - PC160	160
8192*Fs			X	215	PC0 - PC214	214

表10.3:MLBCのクロックレートとチャンネル数の関係

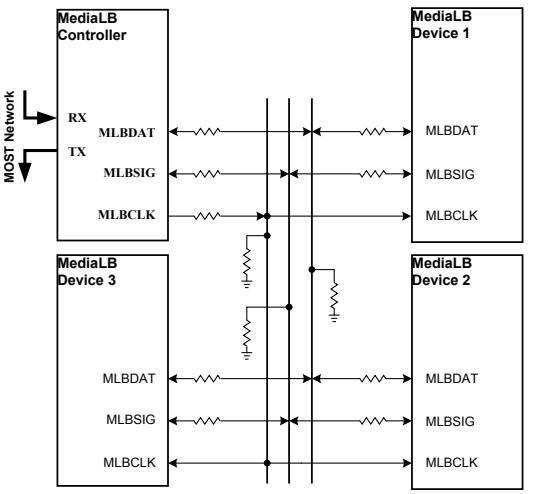


図10.17:3ピンMediaLB実装の配線図

MediaLBバスには、クロックを生成するMediaLBコントローラが1つあります。MediaLBデバイスは、あらかじめ決められたこのクロックに従ってそれぞれのMediaLBインターフェイスを動作させます。

MediaLBの転送容量はクロックレートで決まります(表10.3参照)。4つの連続するバイト(4バイト = 1クワドレット)で1つの物理チャンネルを形成します。MediaLBフレームの最初の物理チャンネル(PC0)は、システム チャンネル用に予約済みです。

複数の物理チャンネルを組み合わせて、指定された一意のChannelAddressを持つ論理チャンネルを形成できます。論理チャンネルの動作は単方向で、1種類のデータタイプ(選択可能)のみを転送します。

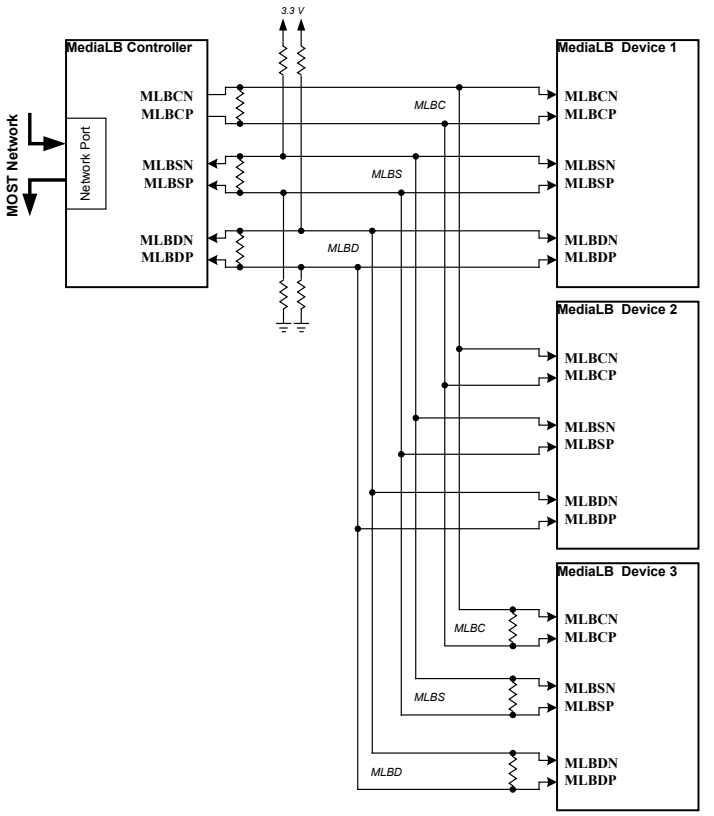


図10.18:6ピンMediaLB実装の配線図

MediaLBは、ChannelAddress、Command、RxStatusのシグナリングに時間多重の双方向信号ラインを使います。このラインをMLBSと呼びます。MLBCとは対照的に、MLBSは全てのMediaLBデバイスによる読み出しと駆動が可能です。ChannelAddressは、特定の論理チャンネルでどのデバイスがデータを送信でき、いつ、どのデバイスがデータを受信できるかを示します。

MLBD信号は時間多重の双方向データラインで、これも全てのデバイスによる駆動または読み出しが可能です。このラインを同時に複数のデバイスが駆動する事はできません。複数のデバイスがMLBDをリッスンできます。

MediaLBはMOSTネットワークに同期して動作するため、MediaLBの通信もフレームに基づいて行われます。

バスアクセスの制御は、チャンネル アドレスを使って論理チャンネルレベルで行います。MediaLBコントローラが、全てのMediaLBデバイスを対象にデータ転送を調停します。実際のアクセスは、1クワドレットの間隔を置いて実行されます。これを使って、例えばMediaLBデバイスはストリーミング データを送信します。

フレームへの分割は、FRAMESYNCパターンと呼ばれる識別機能を使って行います。MediaLBコントローラが生成するこのパターンは、1クワドレットの後にそのシステム チャンネルの次のフレームが開始する事を通知します(図10.20参照)。

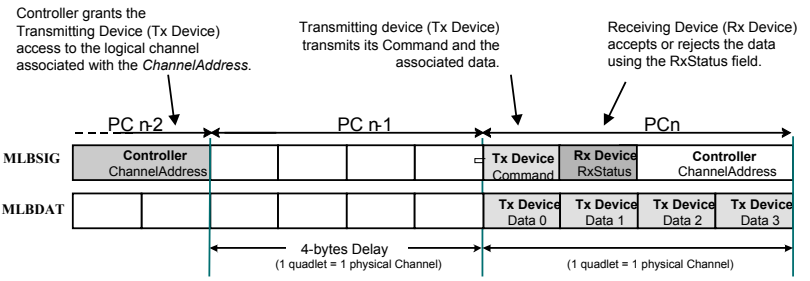


図10.19:MediaLBでのブロックデータ転送

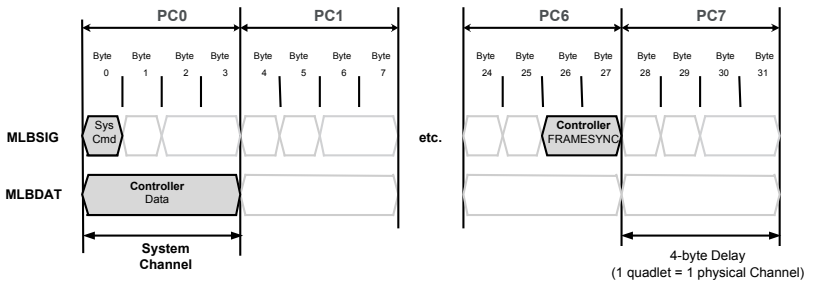


図10.20:MediaLBフレームとFRAMESYNCパターン(256*Fsの場合)

6ピンのMediaLBはクロッキング オプションが拡張され、その信号ラインの設計によってMediaLBのデータ スループットが大幅に向上しています。詳細は、MediaLB仕様を参照してください。

10.12 電源管理

INICは2本の専用入力ピンで外部電源管理回路をサポートします。これらピンの状態に応じて、INICはMOST仕様に基づいた対応をします。

PS1	PS0	ステータス
0	0	U_{Normal}
0	1	Switch-To-Power (STP)
1	0	$U_{Critical}$
1	1	U_{Low}

表10.4:外部電源管理用の入力ピン

10.13 インテリジェントなミュート機能

MOSTネットワークでは、必要に応じてデバイス間でストリーミング接続が確立され、通常は必要な間だけ接続が維持されます。しかし、ネットワークに障害が発生し、結果として音声または映像に障害が生じる事があります。INICはインテリジェントなミュート機能を内蔵しており、シンクとソケットの接続を自動的にミュートし、アプリケーションが破損したストリーミング データを受信するのを防ぎます。インテリジェントなミュート機能はアンロック、またはネットワーク変更イベント(NCE)のどちらかでトリガされます。NCEは、例えばリングからノードが解除された場合にMPRが変化すると発生します。

自動ミュート機能は、ストリーミング ポートとMediaLBポートの両方で同期データとDiscreteFrameアイソクロナス ストリーミング データ(IsocDiscFrameData)に対して使えます。

10.14 ネットワーク インターフェイス コントローラ (NIC)

10.14.1利用可能なインターフェイス

INICとは異なり、ネットワーク インターフェイス コントローラ(NIC)は個々のメモリ位置に直接読み書きを実行して制御します。従って、既存のインターフェイスの役割は、INICの場合とは異なる定義を持ちます。起動やエラー管理等、デバイスの挙動はEHCで動作しているアプリケーションの責任です。

このため、第1世代のMOSTシステムにはいくつかの問題がありました(この点を念頭に開発されたのがINICです)。こうした問題の1つとして、一部のデバイスでアプリケーションの起動に時間がかかるという点がありました。バイパスが長時間閉じるためシステム起動が遅れるリスク、またはデータリカバリが行われない事で不安定性となるリスクがありました。アプリケーションでエラーが発生すると、NICのストレージ位置の記録にエラーが生じる事があります。これはMOSTネットワーク全体に影響します。

こうした弱点は、これらのメカニズムをINIC内部に組み込む事で解決されています。また、INICではNICの8ビット インターフェイスをメディア ローカル バス(MediaLB)で置き換える事で、スペースと実装コストを削減しました。こうした理由により、現在の開発ではINICを使います。

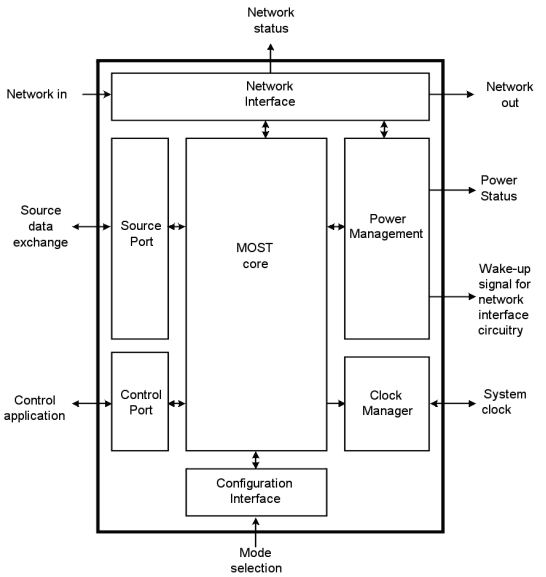


図10.21:OS8104Aのブロック図

このセクションでは、NICの例としてOS8104Aを取り上げます。OS8104Aは第1世代のNIC (OS8104)の置き換えとして既存の設計の更新に使われています。

OS8104Aの基本的な設定は、ハードウェア初期化中にコンフィグレーション インターフェイス(CI)経由で行います。

制御ポート(CP)を使うと、NICの内部メモリに読み書きアクセスできます。このように、CPはNICの設定の微調整に使用できる他、制御メッセージによる通信、および最大48バイトのパケット長までのパケットデータ転送にも使う事ができます。また、ルーティング (ストリーミング データの処理)もCPで制御します。CPはシリアル動作とパラレル動作が可能です。

ソースポート(SP)の主な役割は、ストリーミング データの転送です。一般に、SPのハードウェアは非常に高いデータレートを扱う事ができるため、パラレル動作のSPには以下のモードもあります。

- パラレル非同期モード
- パラレル同期モード
- 物理(パラレル結合)モード

10.14.2シリアル動作

CP

シリアル動作では、CPはI²Cフォーマット(I²Cスレーブ、クロック ストレッチ対応)またはSPIフォーマットのどちらかでデータを転送できます。

SP

INIC同様、ソースポートのデータ信号はクロックピン(SCK)とLeft/Rightを区別するための同期ピン(FSY)を共用します。FSYとSCKを出力として設定した場合、これらのピンはストリーミング ポートでのシリアルデータ転送を駆動します。この場合、外部に接続した周辺部品をFSYとSCKに同期させる必要があります。これが不可能な場合、FSYとSCKを入力として設定できます。

10.14.3CPとSPのパラレル動作

Note:FSYとSCKを入力として設定した場合、NICのRMCK (Recovered Master Clock)出力を使ってアプリケーションを同期させる必要があります。

CP

パラレル動作では、NICのメモリに対してバイト単位でデータを読み書きします。パラレル動作のためには、後述の制御信号を追加する必要があります。

SP

パラレル動作では、SPのスループットが大幅に向上します。性能が非常に高いため、1 MOSTフレームにつき1つのコンポーネントがMOSTネットワークから60バイトのストリーミング データを完全に読み出しながら同時に新しい60バイトを書き戻す事ができます。物理モードでは、1フレーム当たり64バイトをアプリケーションとMOSTネットワークの間で双方向に転送できます。この場合、インターフェイス動作のための制御情報が追加で含まれます。

10.14.4可能な構成(シリアル/パラレル)

NICの動作は、完全にシリアルまたは完全にパラレルにする事ができます。また、CPをシリアルモード、SPをパラレルモードにして、シリアルとパラレルを組み合わせた動作とする事もできます。

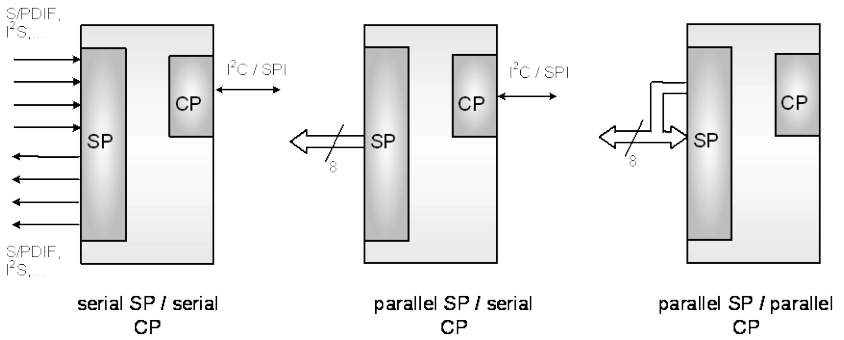


図10.22:CPとSPの基本構成の種類

Note:パラレル動作を選択した場合、SPのシリアルピン(SX0～SX3とSR0～SR3)は使えません。これらは8ビット幅のデータバスに変換されます。

10.14.5NICの基本構成

パラレルまたはシリアル動作の基本構成は、ハードウェア初期化中に行います。これは、コンフィグレーション インターフェイスと呼ばれる一連の接続ピンと、NICのリセットピンを組み合わせて行います。コンフィグレーション インターフェイスは、以下のピンで構成されます。

- PAR_CP (CPパラレル)
- PAR_SRC (SPパラレル)
- ASYNC (SPのパラレル非同期モード)
- LEG (OS8104互換モード)
- SCL (シリアルCPインターフェイスのクロック)

これらのピンをリセットピンの立ち上がりエッジで読み出します。これらピンの状態に応じて、NICが内部設定を実行します。設定が終了して動作準備が完了すると、NICは最初の割り込み(パワーオン割り込み)をトリガして通知します。

Note:アプリケーションは、最初にCPにアクセスする前に必ずパワーオン割り込みを待つ必要があります。

パワーオン割り込みの後は、SCLを除く全てのピンがそれぞれのモードを維持する必要があります。SCLはシリアルCPインターフェイスで使うクロックピンです。選択したバスに応じて、このピンはアイドルではロジック「1」(I²Cの場合)またはロジック「0」(SPIの場合)です。リセットの場合、CPへのデータ転送がないものと考えられます。このため必ずアイドルフェイズが存在し、正しく検出が行われます。

10.14.6NICの制御データ用インターフェイス

CPの動作モードがパラレルでもシリアルでも、制御データは常にCPからNICに入力され、CPから読み出されます。制御メッセージの送受信には、NICの内部バッファ(一連の関連レジスタ)を使います。

10.14.7NICのストリーミング データ用インターフェイス

SPは、シリアルモードとパラレルモードの両方で多くのストリーミング フォーマットとデータレートをサポートします。ストリーミング データのフロー制御は、MOSTルーティング テーブル(MRT)で行います。このテーブルでルーティングエンジンを制御し、ルーティング エンジンが実際のデータ転送を実行します。

シリアルモードでは、最大8本のシリアル インターフェイス ピンを使えます。フォーマットと速度によっては、これら全てを同時に動作させる事ができます。INとOUTの各方向に4本ずつピンがあり、ここにADC、コーデック、DSPのいずれかを接続できます。

ストリーミング データのスループットは、SPをパラレル同期モードにした時に最大となります。物理モードも同じインターフェイス モードに基づきます。ストリーミング データに関しては、同じスループットです。

10.14.8NICのパケットデータ用インターフェイス

パケットデータはNICの内部メモリを経由して送受信できます。このデータには、CPまたはパラレル非同期モードのSPからアクセスできます。この場合、最大ペイロードサイズは48バイトです。

ペイロード長がこれより大きく、より高いデータ スループットが必要な場合は物理モード (パラレル結合モード)を利用できます。このインターフェイスを使うには、適切にアプリケーションと接続する必要があります。

11 ツール

11.1 概要

MOSTシステムの開発プロセスには、有名なVモデルを使います。Vモデルの左側は3つのフェイズで構成されます(図11.1参照)。市場には、これら3つのフェイズに対応したシミュレーションおよび解析ツールが存在します。

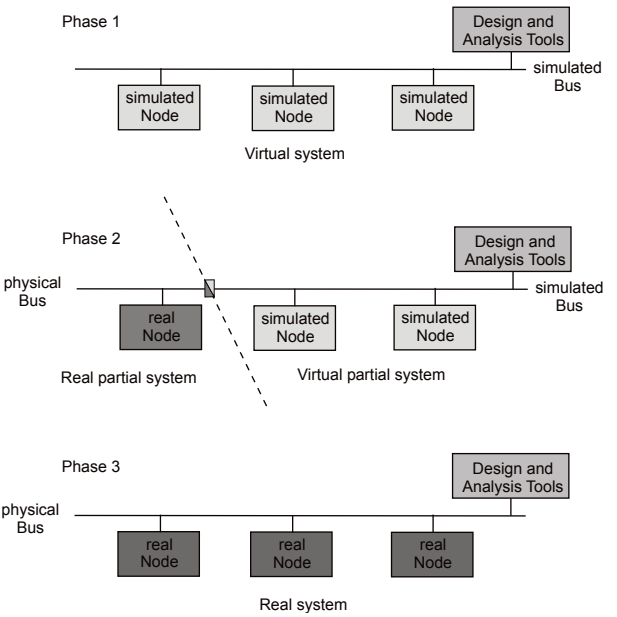


図11.1:ネットワーク シミュレーションから実際のシステム全体までの開発プロセス(出典: [vector])

フェイズ1では、定義された要件に対してシステム全体のシミュレーションを行います。ここでは、MOSTノードの機能と通信に関する挙動を検証します。このフェイズでは、ファンクション ブロックをMOSTエディタで定義し、動的挙動をMSCエディタで記述します(セクション11.2参照)。

フェイズ2では、MOST ECUを開発して段階的にネットワークに組み込みます。実際のECUと仮想のECUが混在するこのフェイズを、リメイニング バス シミュレーションとも呼びます。このプロセスを支援するのが、ラピッド プロトタイピング ツールです(セクション11.4参照)。いくつかのソフトウェア ハウスがネットワーク サービス(第9章参照)用のソフトウェア ライブラリを提供しており、これらを使ってソフトウェア開発を進めます。ハードウェア開発には、INIC Remote ViewerとMediaLB Analyzerを使います(セクション11.5参照)。

フェイズ3では、全てのECUが実際のECUです。ここでは、完全なシステムのテストと検証ができます。

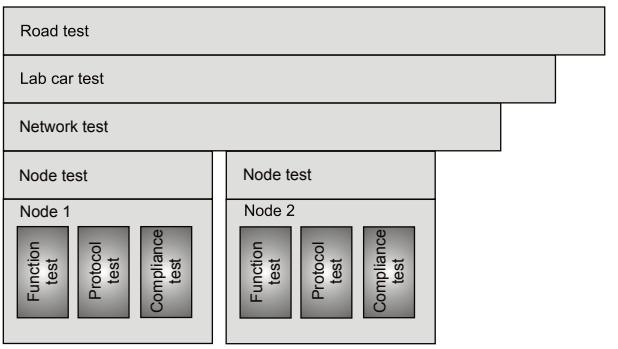


図11.2:MOSTシステムのテストレベルの概要

MOST ECU単体およびネットワーク全体のテストは、フェイズ2と3で行います。図11.2に、必要なテストレベルを示します。最初に、ECU単体の機能テストとコンプライアンス テストを実施します。機能テストはユーザ アプリケーションが正確に実装されている事を確認し、コンプライアンスおよびプロトコルテストは通信挙動がMOST仕様に準拠している事を確認します。次に、MOSTネットワーク全体をラボでテストした後、テストボードでのテストを経て、最後に実車環境でテストします。第12章ではコンプライアンス テストについて、第13章ではMOSTのテストについて説明します。

11.2 仕様定義ツール

11.2.1 MSCエディタ

Message Sequence Chartエディタ(MSCエディタ)は、Message Sequence Chart (MSC)をグラフィカルに編集するためのツールです。より高度なツールとして、通信サービスおよびプロトコルの解析、仕様定義、文書作成、テストが行えるビジュアル モデリング ツールがあります。MSCエディタには、オープンソースのものと商用のものがあります。

MSCの詳細は、補遺Aで説明します。

11.2.2 MOSTエディタ

OEMによっては、MOSTファンクション ブロックを含む車両の全バスシステムの通信パラメータを完全にデータベース化しています。このようなOEMは、独自の編集ツールを使っています。これに対し、MOST Cooperationが独自に開発したのがMOST Editorです。このセクションでは、このエディタについて説明します。

MOST Editorは、MOST Cooperationが提供するXMLフォーマットのファンクション カタログの作成とデータ管理のためのツールです。ファンクション カタログは、ファンクション ブロックとその全ての定義済みメソッドとプロパティを含みます。個々のMOSTデバイスは、このカタログをベースにして記述します。ファンクション ブロックのメソッドとプロパティのうち、実装しないものは削除し、存在するパラメータの値のレンジは必要に応じて変更します。

ユーザ インターフェイスは2つの部分で構成されています(図11.3参照)。左側には、ファンクション ツリーが表示されます。ここには、個々のファンクション ブロックとそのファンクションが表示されます。右側には、ファンクションの詳細が表示されます。

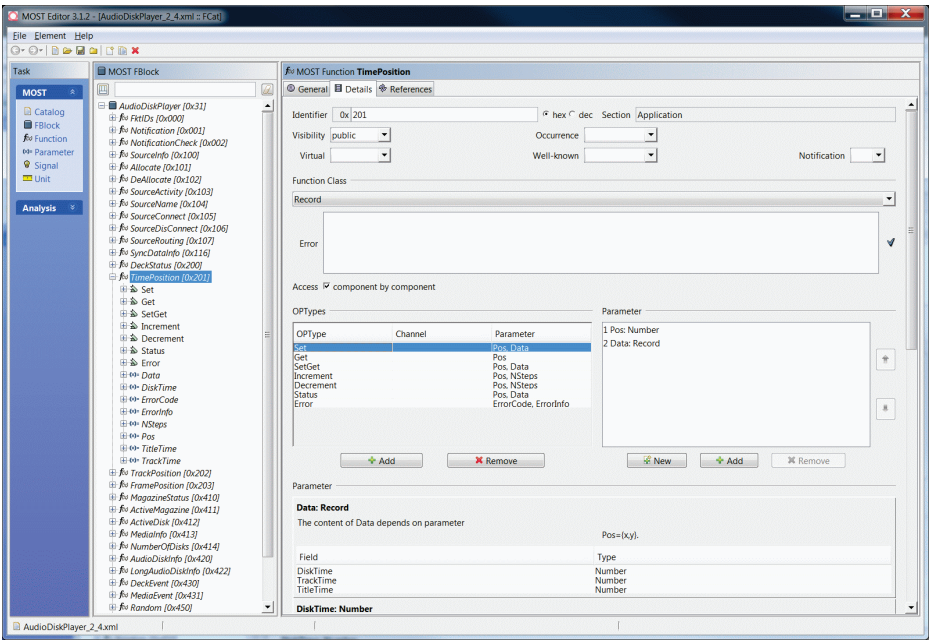


図11.3:MOST Editor

ファンクション ツリーは、各エントリを右クリックして表示されるコンテクストメニューを使って拡張または修正できます。ファンクションを選択すると、そのプロパティが右側のウィンドウに表示されます。

各ファンクションの記述には、以下のエントリが必要です。

- ID
- ファンクション クラス
- セクション
- HTMLフォーマットでのテキストによる説明

- 利用可能なOPType (例: Get、Set)
- バージョン情報

図11.3に、ファンクションのオペレーション タイプを定義するダイアログを示します。ここに示した例では、ファンクション ブロックAudioDiskPlayerのTimePositionファンクションにGet、Set、SetGet、Increment、Decrement、Status、ErrorのOPTypeがあります。次に、各OPTypeのパラメータを定義します。

パラメータは全て、以下のエントリを含むXMLフォーマットのファンクション カタログに保存されます[Cat 02]。

- XMLバージョン
- ファンクション カタログ ヘッダ
- 現在のファンクション ブロックのヘッダ
- ファンクションとそのパラメータのリスト
- ファンクション クラスの定義

以下に、ファンクション カタログの基本構造を示します。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE FunctionCatalog (View Source for full doctype...)>
<FunctionCatalog>
   <CatalogVersion>
   </CatalogVersion>

   <!-- FBlock:AmFmTuner -->
   <FBlock>
       <Function>
       ….
       </Function>

       <Function>
       ….
       </Function>
   </FBlock>

   <Definition>
   …
   </Definition>

</FunctionCatalog>
```

11.3 シミュレーションおよび解析ツール

シミュレーションおよび解析ツールは、MOSTネットワークを一貫性のあるグラフィカル ユーザ インターフェイス(GUI)環境で開発、テスト、解析するためのツール群です。開発プロセスの最初に、これらのツールを使ってMOSTノードの挙動をエミュレートするシミュレーション モデルを作成します。開発中、これらのモデルが解析、テスト、統合の基盤となります。これらのツールは、以下の機能をサポートする必要があります。

- MOSTネットワーク データの**オンライン解析**では、リアルタイムでデータを解析します。これに対し、**オフライン解析**では記録されたデータを評価します。フィルタおよびトリガ設定の調整に基づき、特定のイベントをハイライトまたはスコープ表示できます。ビューア、グラフ、ウォッチ機能をタイムスタンプで同期させる事により、時間軸を揃えてデータを評価できます。
- **ストレスツール**は制御データおよびパケットデータ チャンネルで負荷を生成し、変調信号とロック シーケンスを正確なタイミングで適用します。
- 照明とロック、システムロック、コンフィグレーション ステータス等の**ハードウェアおよびネットワーク ステートにアクセス**するためのツール。ストレスツールと組み合わせると、伝送品質を統計的に観察できます。
- **制御チャンネル ビューア**は、CMSおよびAMSプロトコルをハイレベルで解析します。このビューアは、MOSTリングで発生するMOSTイベントを表示します。
- **FBlockビューア**を使うと、バス上で通信されるアプリケーションのステートを観察できます。このビューアは、バス上で転送される全ての(または選択した)プロパティの値を表示します。値を分解するには、XMLファンクション カタログが必要です。
- **パケットデータ チャンネル ビューア**は、実装したプロトコルに応じてMOST High Protocol、MAMAC、MDPを表示します。MOST150システムでは、Ethernetチャンネルの可視化が必要です。トランスミッタ ファンクションは、パケットデータを他のネットワーク ノードに送信します。
- **ストリーミング チャンネルへのアクセス**を可能にするツールが必要です。44.1または48 kHzで全ての同期およびアイソクロナス チャンネルをPCとリングの間で双方向に転送できます。
- **MOSTスレーブノード**を解析する場合、TimingMaster、NetworkMaster、PowerMasterをエミュレートするツールが必要です。
- セントラル レジストリには、MOSTシステムで利用可能な全てのFBlockをノード位置アドレス、論理アドレス、インスタンスIDと一緒に保存します。FBlockとそのパラメータを表示できるビューアがあれば、システムアクセスの利便性が向上します。

11.4 ラピッド プロトタイピング ツール

ラピッド プロトタイピング ツールは、MOST ECUの開発を支援します。以下の種類から選ぶ事ができます。

- ネットワーク サービス ライブラリ (ソフトウェア開発をサポート、第9章で説明)
- シミュレーションおよび解析ツール用プラグイン
- ラピッド プロトタイピングと評価のためのハードウェア プラットフォーム
- PCツールキット
- システム コンポーネント

11.4.1 シミュレーションおよび解析ツール用プラグイン

これらのツールには、MOSTデバイス用のコントロール パネルを簡単に作成できる機能が必要です。コントロール パネルとは、ユーザ定義のウィンドウにバス信号とグラフィック コントロール類を表示したものです。多くの場合、テキストボックス、スライダ、指針メータ等の各種コントロールを使えます。コントロールパネルには、静的要素(テキストまたはフレーム)、マクロ、背景グラフィックを含める事ができます。スクリプト言語を使うとシミュレーション、テスト、解析を自動で実行できます。

11.4.2 ラピッド プロトタイピングと評価のための ハードウェア プラットフォーム

このハードウェア プラットフォームは、物理層(oPhyまたはePhy)、MOST25/50/150用のINIC、外部ホスト コントローラ(EHC)およびオーディオ/ビデオ入出力へのインターフェイスで構成されています。図11.4に、このプラットフォームの基本的アーキテクチャを示します。基板上のマイコンまたはFPGAでストリーミング データ チャンネル、パケットデータ チャンネル、制御チャンネルを分離します。パケットデータ チャンネルと制御チャンネルはUSB経由でPCへ転送し、ストリーミング データはコーデックへ転送します。PCではネットワークサービスとEHCアプリケーションを動作させます。MediaLBヘッダにMediaLBアナライザを接続する事もできます。

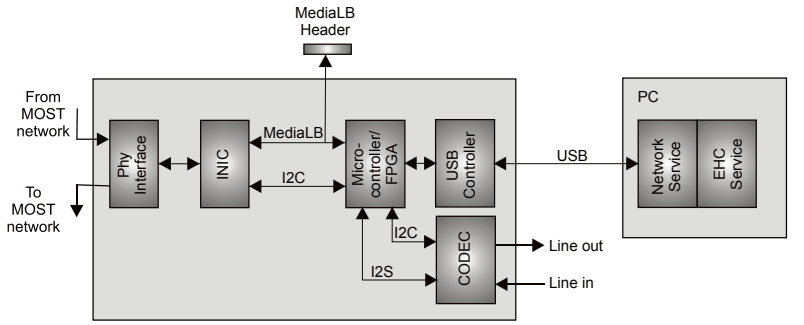


図11.4:ラピッド プロトタイピングと評価用のプラットフォームのアーキテクチャ

11.4.3 PCツールキット

通常、PCツールキットはMOSTインターフェイスにマルチメディア アプリケーションとMOST制御ファンクション群を接続した構成です。図11.5に、このアーキテクチャを示します。MOSTインターフェイスは、PCとMOSTネットワークを高速接続します。このインターフェイスは物理層(ePhyまたはoPhy)、INIC、ファームウェアで構成され、パケットデータ チャンネル、ストリーミング データ チャンネル、制御チャンネルへのアクセスをサポートします。PCへのインターフェイスには、PCI、USB、PXIのいずれかを使います。

PCツールキットを使うと、追加のハードウェアなしで独自のソフトウェアツールを開発することができます。

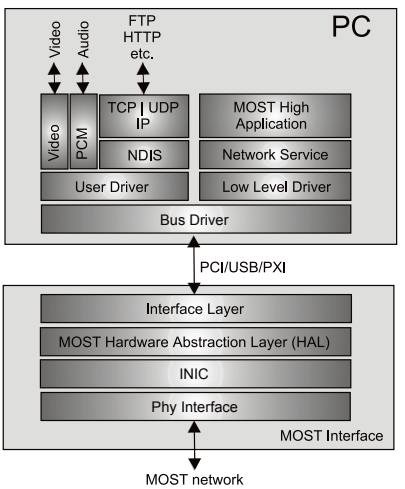


図11.5:PCツールキットのアーキテクチャ

11.4.4 システム コンポーネント

リピータ、ハブ、ゲートウェイ、コンバータを使うと、効率的なシステム開発が可能です。

リピータはネットワーク エクステンダとして動作可能なデバイスで、それ以外にもネットワークの構造に干渉せず監視用の信号を出力する機能があります。ハブは信号の波形を復元し、スター型アーキテクチャでデータストリームを分配します。ゲートウェイは、MOSTネットワークと他のネットワーク(例: CAN)を接続します。

コンバータは、MOST50電気物理層(ePhy)と光物理層(oPhy)等、異なる物理層間でネットワーク信号を変換します。コンバータは、MOST50 ePhyを被試験デバイス(DUT)として、その電磁両立性(EMC)と動作時の静電気放電(ESD)特性を検証するために使います。

11.5 ハードウェア開発ツール

全てのMOST ECUに内蔵されているINIC(旧式のMOST ECUの場合はNIC)は、2つの専用ツールを使って解析できます。これらのツールとは、INIC Remote ViewerとMediaLB Analyzerです。

11.5.1 INIC Remote Viewer

INIC Remote Viewerは、INICベースのMOSTネットワーク ノードの情報をリモートから解析できる分離型ゲートウェイです。INIC Remote Viewerには、以下の機能があります。

- 通常はデバッグヘッダからしかアクセスできないFBlock NetBlockとFBlock INICのプロパティとステートを監視
- INICコンフィグレーション ストリングの読み出し
- 以下の可視化:
  - INICパラメータ
  - NetBlockパラメータ
  - INICのステートとステート変化
  - ポート、ソケット、接続
  - コンフィグレーション ストリングのエントリ

11.5.2 MediaLB Analyzer

MediaLB AnalyzerはMediaLBデータの観測と可視化が可能で、以下の機能を備えています。

- INICポートメッセージの分解
- フィルタとトリガの定義
- MediaLBのロックと速度の検出
- 3ピンおよび6ピンMediaLBに対応したスピードモード
- MediaLBフレームと物理チャンネルのカウンタ
- MediaLBリンク層プロトコル チェック

12 コンプライアンス テスト

12.1 コンプライアンス テストの目的

第11章で説明したように、ネットワーク システムとモジュールの統合は複雑であるため、特に困難です。この課題に対する効果的な手法は、認証テストプロセス(コンプライアンス プロセス)を導入し、デバイスとモジュールのテストが完了してからシステムに統合する事です。これにより品質が大幅に向上し、統合の複雑さを軽減できます。

MOST認証テストプロセス (コンプライアンス プロセス)は、モジュールとデバイスがそれぞれの仕様に準拠している事、すなわち仕様の要件を満たしている事をテストするのが主な目的です[ISO 9646] [ETS 300 406]。ここでは、高い相互接続性の実現という目標に深く関係する品質項目を重視します。

コンプライアンス テストは、MOST Cooperationの法的な側面をカバーします。コンプライアンス テストに合格したデバイスのみが、MOST IP (Intellectual Property)プールへの登録とMOST®商標の使用を認められます(図12.1参照)。

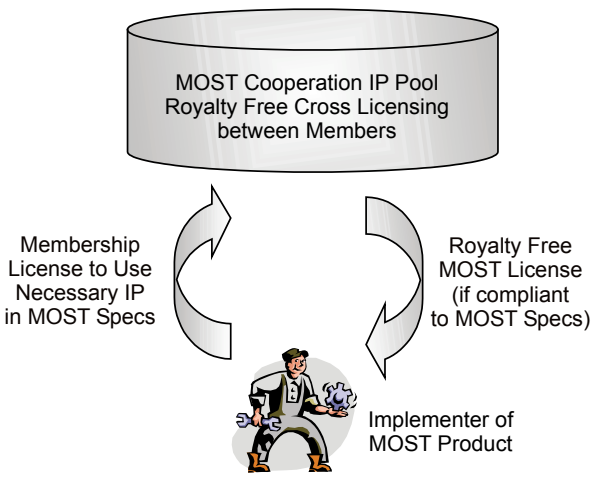


図12.1:MOST IPとコンプライアンス プロセス

MOSTの階層モデルに準じ、認証テストプロセスは以下の領域(コンプライアンスレベル)をカバーします(図12.2参照)。

- 物理層に対するモジュールテスト(例: FOT、ピグテール)
  (完全な物理層コンプライアンス)
- 物理層に対するデバイステスト
  (限定的な物理層コンプライアンス)

- 高次の通信レベルに対するデバイステスト
  (コア コンプライアンス)
- アプリケーション層に対するデバイステスト
  (プロファイル コンプライアンス)

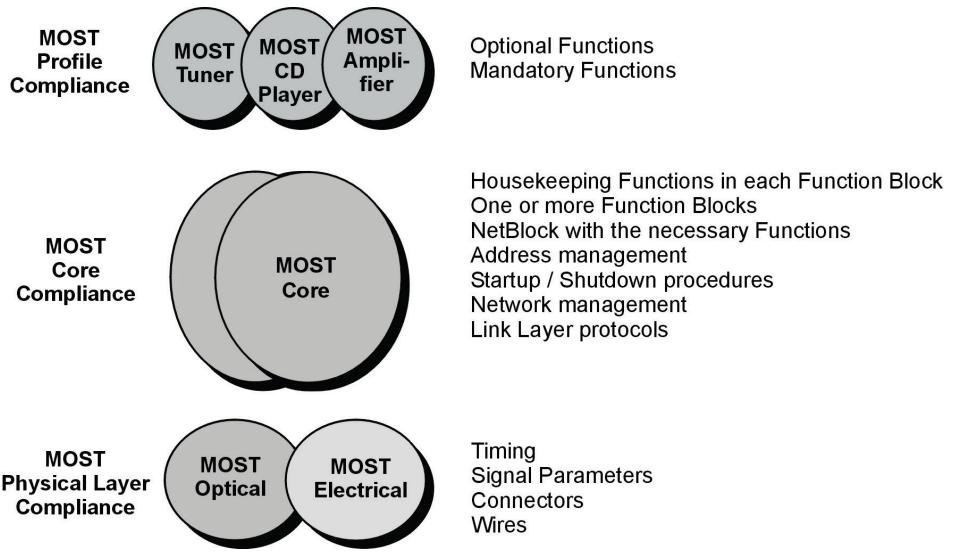


図12.2:コンプライアンス レベル

12.2 コンプライアンス プロセスの構成

12.2.1 コンプライアンス プロセスの所轄部署

MOST Cooperationでは、認証テストプロセスに関していくつかの所轄部署を設置しており、これらが協調して認証テストプロセスを担当しています(図12.3参照)。プロセス全体の統制と監督は、MOSTコンプライアンス監督委員会(MCSB)が担当します。MCSBは、テストプロセスの監督権限を持つMOSTコンプライアンス管理者(MCA)を任命します。MCAはMOSTコンプライアンス テストハウスから受け取ったテスト報告を精査し、最終的に認定証を授与します。

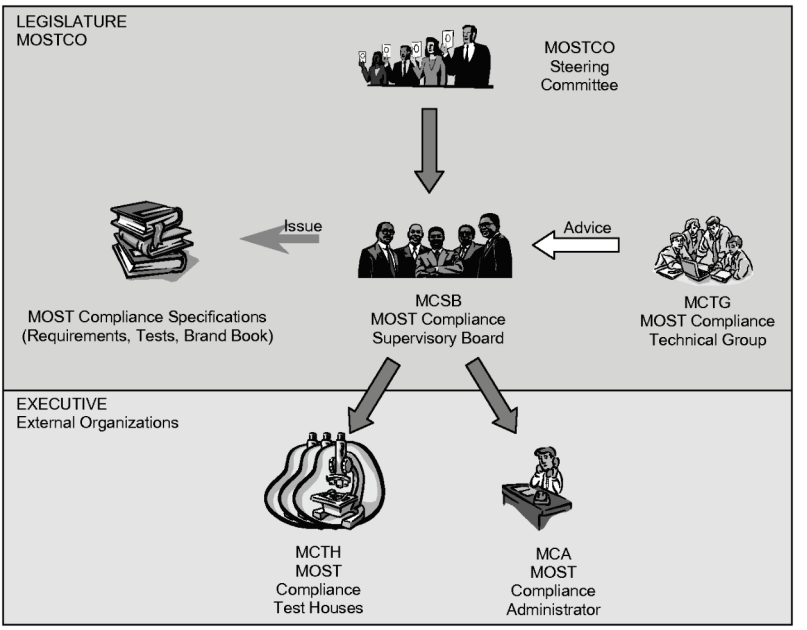


図12.3:コンプライアンス プロセスの組織編成(出典:[ComReq])

コンプライアンス テストの信頼性は、MOST Cooperationの承認を受けたMOSTコンプライアンス テストハウス(MCTH)の信頼性にかかっています。MCTHは、ISO 17025『試験書および校正機関の能力に関する一般要求事項』に従いドイツ技術審査認定機関(DAkkS: Deutsche Akkreditierungsstelle GmbH、ベルリン/フランクフルト)が認定しています。

テスト仕様、エラッタ、技術的助言は、テストハウスの代表者を含む専門家の委員会であるMOSTコンプライアンス テクニカル グループ(MCTG)が提供します。

12.2.2 コンプライアンス プロセスのワークフロー

認証テストのワークフローは、以下の3段階で実行されます。

ステップ1
メーカーが任意のMOSTコンプライアンス テストハウス(MCTH)を選び、テストハウスと共同でテスト計画を作成します。この作業では、メーカーがテストに必要な全ての文書をMCTHに提出する必要があります。

ステップ2
MCTHが必要な全てのテストを実施し、テストレポートを作成します。また、メーカーの宣言書としてコンプライアンス ファイルに添付する適合宣言書(DoC)も作成します。

ステップ3
MOSTコンプライアンス管理者(MCA)がコンプライアンス ファイルを精査し、メーカーからの申請に応じてMOST仕様準拠認証製品としてMOSTコンプライアンス製品リスト(MCPL)に掲載します。こうして、メーカーにMOSTのライセンスが許諾されます。

図12.4に、コンプライアンス プロセスのワークフローを示します。

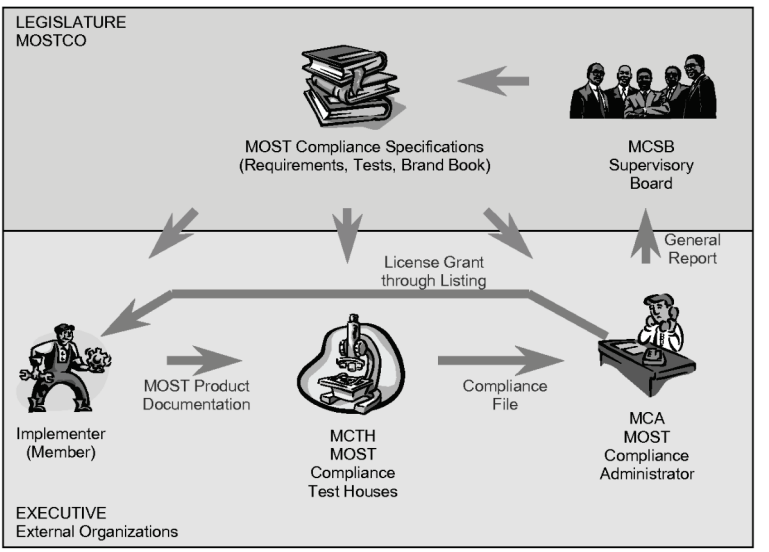


図12.4:コンプライアンス プロセスのワークフロー(出典:[ComReq])

12.3 物理層コンプライアンス テスト

物理層のコンプライアンスに関しては、計測の基本原理と計測手法、そしてテストの前提となる許容誤差について定義しています。MOSTの物理層コンプライアンステストには、テストポイントSP1～SP4が含まれます(図12.5と第6章参照)。

コンプライアンス プロセスの2番目のステップでは、実施すべき手順(テストケース)が定義されています。これらのテストケースには、テストポイントSP1～SP4で以下の項目を計測する際の電圧および温度レンジが示してあります。

- ビットレート
- ロジックレベル
- 立ち上がり時間
- パルス幅のばらつきと歪み

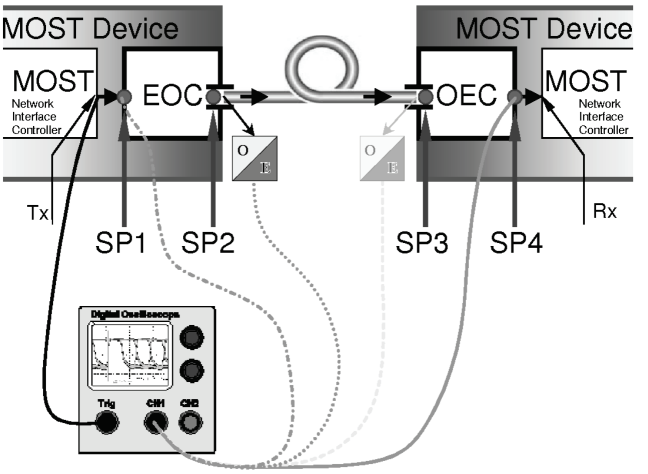


図12.5:物理層コンプライアンス テストの構成例(EOC: 電気/光コンバータ、OEC: 光/電気コンバータ) (出典:[ComPhyLP 1.0])

- データ依存ジッタと非相関ジッタ
- 入力抵抗と入力容量
- 波長
- 光出力パワー
- 消光比
- オーバーシュートおよびアンダーシュートの挙動
- 信号リップル

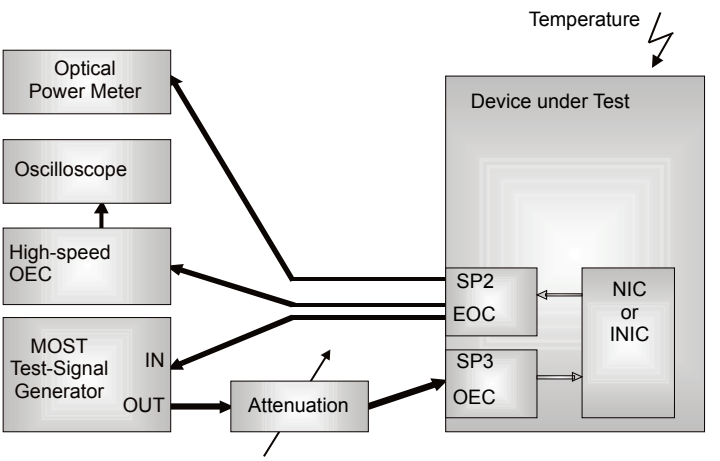


図12.6:限定的な物理層コンプライアンス テストの構成

限定的な物理層コンプライアンス テストは完全なデバイスをテストする事が目的ですが、テストできるのはテストポイントSP2とSP3のみのため、これは一種のブラックボックス テストと見なす事ができます。図12.6に、限定的な物理層コンプライアンス テストの構成を示します。MOST150では、物理層ストレステストツールを適用します。

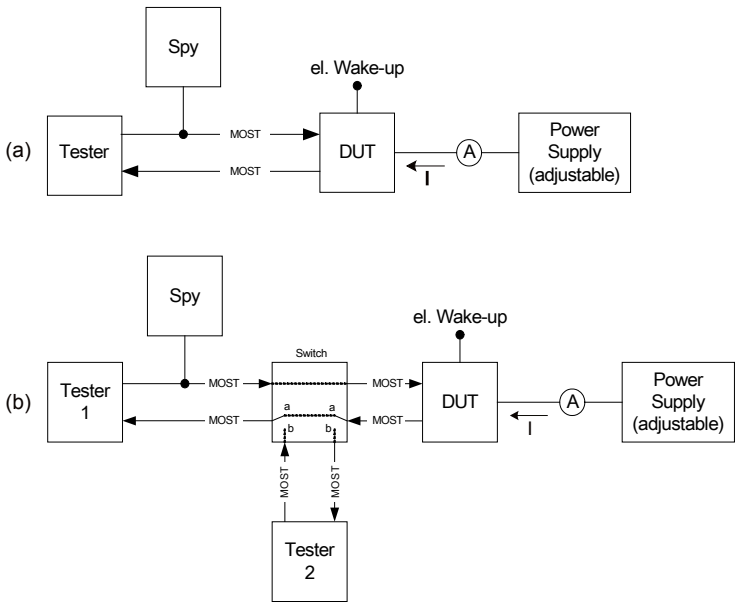


図12.7:コア コンプライアンス テストの構成: テスタを1つ使った場合(a)と2つ使った場合(b)

12.4 コア コンプライアンス テスト

MOST仕様の解析に基づき、コア コンプライアンス テストの要件(仕様点)が決定され、テストケースが作成されています。これらのテストケースは、以下の挙動を検証します。

- 復帰(ウェイクアップ)挙動
  - 復帰 - TimingMaster
  - 復帰 - 復帰を実行される側のTimingSlave
  - 復帰 - 復帰を実行する側のTimingSlave
- 通常の挙動(通常動作)
- 電源管理
- エラー管理
- リングブレーク診断

- システム設定
  - 初期化(NetworkMaster)
  - システム設定(NetworkMaster)
  - システム設定(NetworkSlave)
- ノードのアドレッシング
- 制御メッセージ(ACK/NACKテスト)
- ノーティフィケーション マトリクス
- メッセージのセグメンテーション

図12.7に、テスト構成の概要を示します。図の上側(a)はテスタが1つのシンプルな構成で、下側(b)は2つのテスタを使った構成です。これらの方法でリング内の挙動をシミュレートします。

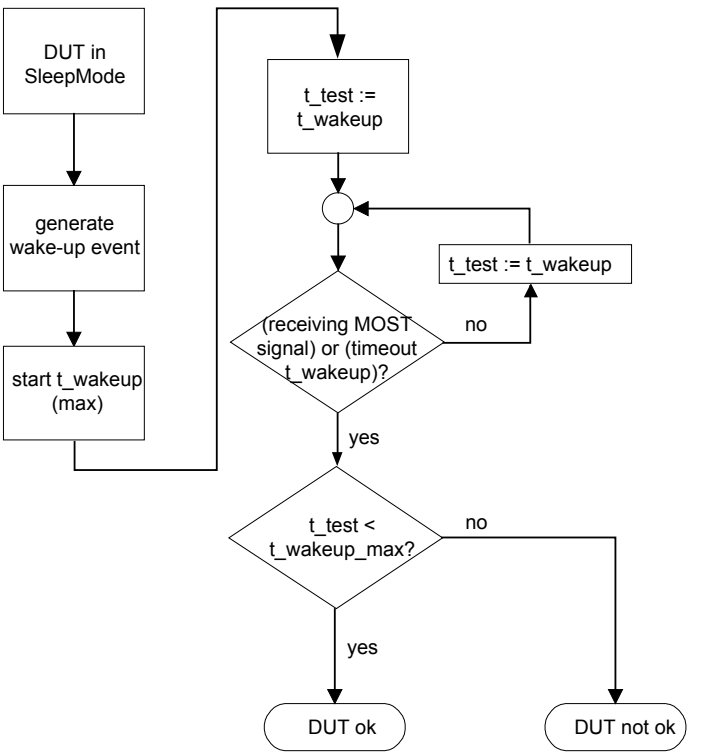


図12.8:テストケース Signal On Test

しばしば、テストのワークフローをファンクション ブロックEnhancedTestability (ET)でサポートする事が必要です。FBlock ETはアプリケーションとMOSTネットワーク インターフェイス コントローラの間の「フィルタ」として動作し、各種イベントでトリガできます。FBlock ETは以下の特徴を備えます。

- FBlock ETはアプリケーションとNetInterfaceの間に位置し、全てのデバイスおよびMOSTノードに必須のFBlockです。FBlock ETはデバイス全体の挙動に影響を与えてはいけません。これはデバイスがテスト中でなくても同じです。
- このFBlockは信号を生成または抑止します。
- このFBlockはNetOnステートの場合のみ利用でき、電源の障害に関する値は一切格納しません。
- FBlock ETの代表的なファンクションはAutoWakeup、NetInterfaceState、SendMessage、EchoMessage、Reset等です。

図12.8に、テストケースSignal On Testのワークフローの例を示します。

12.5 プロファイル コンプライアンス テスト

図12.2で示したように、プロファイル コンプライアンス テストはConnectionMaster等の個々のアプリケーション (プロファイル)をカバーします。図12.9に、プロファイル コンプライアンス テストの構成を示します。テストの構造は、プロファイル (ファンクション ブロック)の全体的な構造と共通しています。各FBlockは汎用のFBlockテンプレートであるGeneralFBlockから派生しているため、各FBlockに対して汎用的に実行するテストもハイレベルのMOSTプロファイル コンプライアンス テスト仕様に記述します。

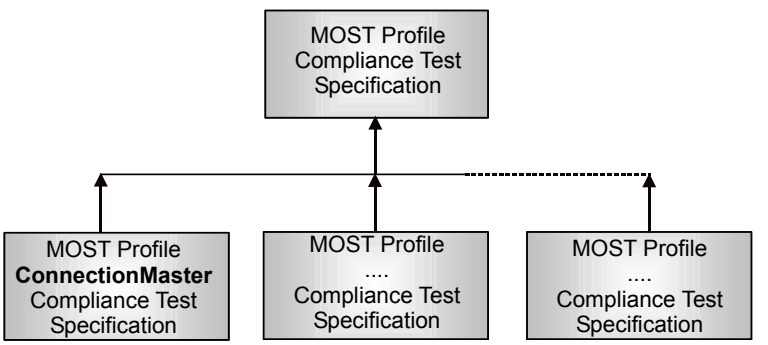


図12.9:プロファイル コンプライアンスの構造

FBlockから派生した各プロファイル固有のテスト仕様では、特にそのプロファイルに固有の機能と動的挙動をテストします。

図 12.10 に、プロファイル コンプライアンス テストのテストケース(“SyncConnectionTable_Get”)をMessage Sequence Chart (MSC)で記述した例を示します。この方法で記述したテストケースは、MOSTコンプライアンス テストハウス(MCTH)とセクション13.7で説明するテストツールで基盤的な役割を果たします。ここで抽象度の高いテスト仕様に基づいて定義したテストも実装する必要があります。

12.6 相互接続性テスト

相互接続性テストは、正常に動作する他のMOST製品に対する機能テストと定義されます。フィールドに存在するMOST製品の相互接続性が向上すれば、MOSTシステムに対するユーザの信用が高まります。このため、MOSTコンプライアンス プロセスでは相互接続性テストの実施を要求しています。

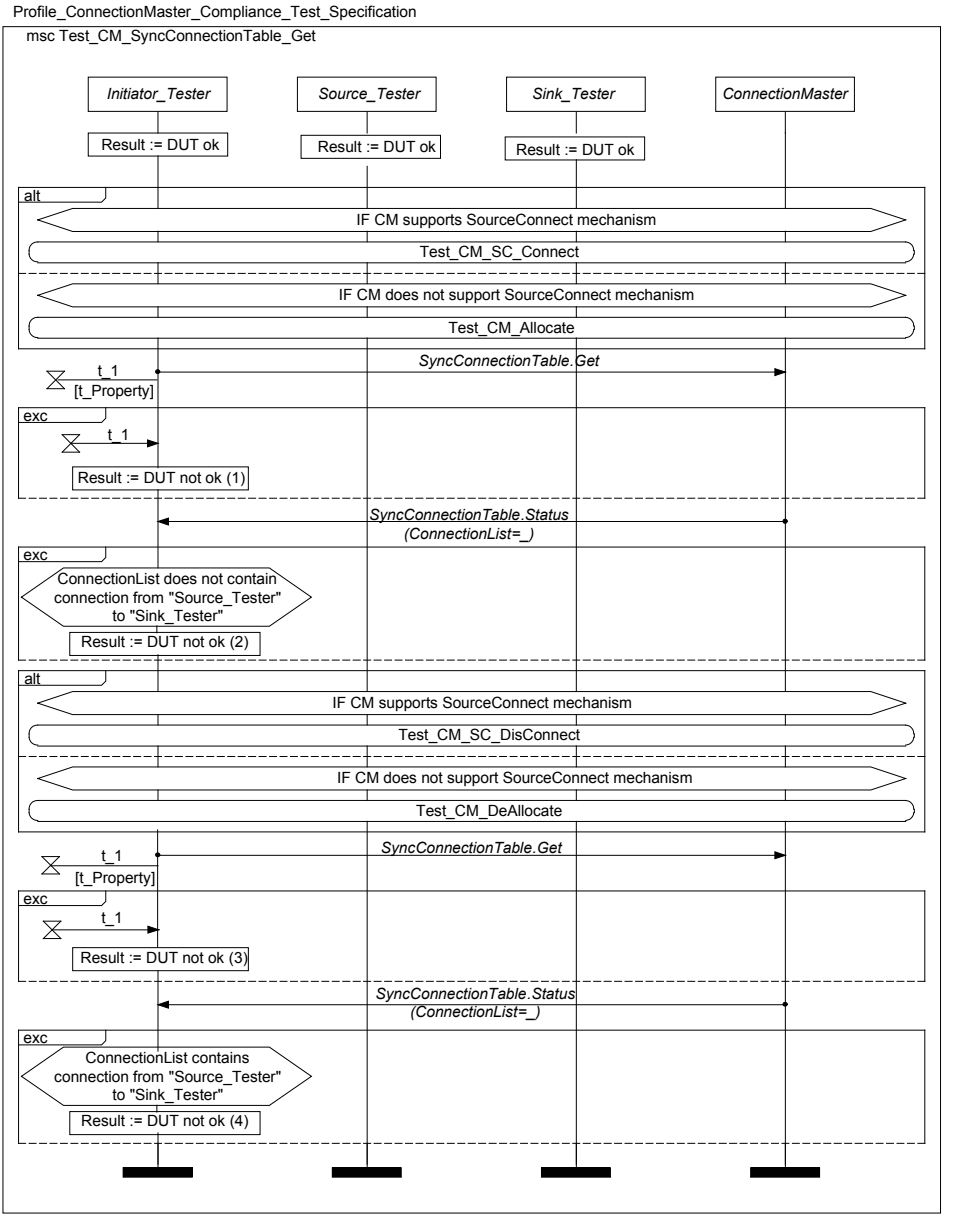


図12.10:プロファイル コンプライアンス テストの例(SyncConnectionTable-Get)

相互接続性テストの再現性を高め、全てのメーカーが一貫性のある形で実施できるようにするため、相互接続性テストで使う製品群が定義されています。これらの製品はゴールデンMOSTインプリメンテーション(GMI)と呼ばれ、対応するリストに登録されています。実際のテストにどのデバイスを使うかは、製品で使う機能で決まります。

13 MOSTベースのインフォテインメント システムのテスト

13.1 MOSTシステムのテストに関する課題

最近の強力なインフォテインメント システムは、その多くがMOSTベースで開発されています。こうしたシステムの統合はOEMにとって非常に困難な作業であり、その事がシステムのテスト方法に影響を与えています。通常、OEMは各種サプライヤから納入された部品を組み合わせて1つの完全なシステムを構築しています。

サプライヤは社内の様々な開発プロセス、技術、規格を使って部品を開発します。これらの部品を統合し、車両システム全体の中でこれらが正しく連携する事を確認するのはOEMの仕事です。重要なシステム機能の中には、多くの部品が協調動作して初めて機能を実現できるものもあります。音楽CDの再生中にハンズフリー通話の着信を処理する機能はその1つです。音声通話の着信があるとただちに音楽CDを一時停止し、オーディオアンプがオーディオ チャンネルの割り当てを変更してサウンドシステムのマイクとスピーカを使ってハンズフリー通話を行えるようにします。

こうしたシステムは非常に複雑であるため、しばしばハードウェアとソフトウェアの開発が完了する前に統合作業を始めます。また、インテグレータがテストの制御と監視に使えるインターフェイスは限られており、通常は部品の外部インターフェイス(例: MOST、CAN)のみです。後の統合段階で問題が見つかってサプライヤに変更を指示しようとしても、問題の原因がどの部品にあるのかを特定するの事は困難です。この章では、OEMの立場から見たテストプロセスについて説明します。セクション13.2では、OEMが実施するテストの概要をVモデルにあてはめて説明します。セクション13.3では、全ての統合フェイズに共通する基本的なテストプロセスについて説明します。以降のセクション13.4～13.6で各統合フェイズについて説明します。統合フェイズは、以下の3つに分類されます。

- 部品テスト
- 統合テスト
- システムテスト

セクション13.7では、テスト用ツールの概要を説明します。

13.2 OEMが実施するテストの概要

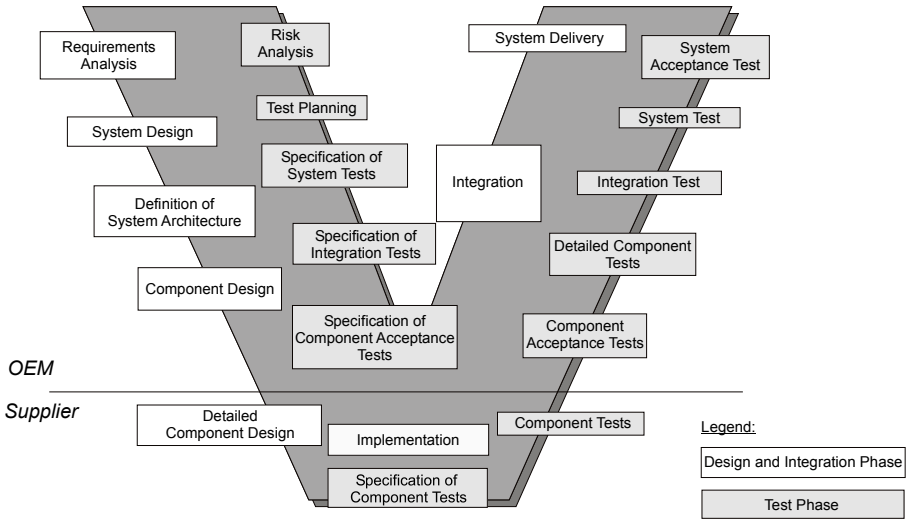


図13.1:Vモデル(出典:Daimler AG社)

図13.1に、OEMとサプライヤがそれぞれ実施する開発工程をVモデルにあてはめたものを示します。最初に、システム全体の要件を定義します。これを受けて、システム設計では主にシステムを構成する各種部品を特定します。個々の部品の要件を、部品要求仕様書としてサプライヤに配布します。サプライヤは、各部品の詳細な設計、実装、テストを実施します。OEMは、納入された部品に対して部品検収テストを実施した後、これらの部品をシステム全体に段階的に統合していきます。統合テストは、部品同士が正しく連携動作するかどうかを確認します。統合テストの基盤となるのが、仕様決定段階で定義される相互接続性仕様です。各テスト段階では、以下の品質指標をチェックします。

- 機能仕様への適合性
- 信頼性と耐障害性
- 性能と安定性

テストの計画と仕様定義は、要件が定義され次第開始できます。

MOSTコンプライアンス プロセス(第12章参照)の導入で、OEMにおける開発プロセスに以下のような影響が現れています。MOST部品の挙動は現在、MOST仕様とOEM固有機能の仕様への参照という形で指定されています。今後OEM間の協調が進めば、要件定義の多くの項目が標準仕様またはOEM定義によるテスト仕様(例: Automotive Application Recommendation)への参照で置き換えられるようになるため、OEM固有機能は少なくなると考えられます。すると、OEMは社内テストの量

を大幅に減らす事ができ、MOSTシステムの統合作業で問題が発生する事が少なくなります。この結果、OEMはOEM固有機能の検証に専念できます。

13.3 テストプロセス

各テスト段階(部品テスト、統合テスト、システムテスト)で実行するタスクは、以下の通り体系化されています。

13.3.1 リスク解析

リスク解析では、開発中のMOSTシステムに関する潜在的なリスクと問題を洗い出し、必要に応じてリスクを最小化する手段をアーキテクチャ レベルで定義します。また、リスク解析は開発プロセスにおける品質保証作業計画と優先順位決定に影響します。その一環として、テスト計画でテスト作業の重点目標を定義します。テスト計画では、テストの対象、テストの目標、目標達成のための方法論、リソース計画、サプライヤ管理を明確化します。その後、テストを実施しながらテスト計画を段階的に改善し、実行可能なテスト仕様へと仕上げていきます。

13.3.2 テスト設計

テスト設計では、テスト計画フェイズで定義したテストを体系的に改善して、多数のテストケースへと仕上げていきます。テスト設計フェイズの目的は、抽象度の高い多数のテストケースを定義する事です。テストケースは、実装の細部に詳しくないアプリケーション専門家が理解できるようにテストを記述します。可能なら、テストケースはテスト対象がリリースを重ねても毎回再利用できるようにすると共に、共通のプロパティを持つ別のテスト対象にもなるべく多く再利用できるようにしておくのが理想です。テストを高い抽象度(例: 要件定義と同じ抽象度)で記述する事で、より簡単に再利用が可能です。テストケースを選択する際は、テスト目標を適切にカバーできるように分類ツリー法[Lehmann 2000]等の体系的アプローチを適用する必要があります。

13.3.3 テスト実装

テスト実装では、抽象度の高いテストケースに基づいて実行可能なテストケースを実装します。テスト入力データと予測結果値を選択します。テストケースを手動で実行する場合、テスタの作業手順を定義済みのフォーマットに従って自然言語で記述します。これによって、テスト手順を明確に定義します。テストを自動で実行する場合(セクション13.7参照)、テストケースをプログラムまたは自動生成生成する必要があります。この場合、テストの実行に必要なシステムも準備する必要があります。

テスト実行の開始は、テスト対象のリリース時期に左右されます。このため、それまでになるべく多くのテスト準備(計画、設計、実装)を完了しておく事が重要です。こうしておけば、テスト対象が利用可能になると同時にテストを開始でき、余分な遅れを防ぐ事ができます。

13.3.4 テスト評価

テスト評価では、テスト結果を評価し、テスト計画で定義した目標を達成したかどうか査定します。テストケースに失敗した場合、またはテスト自体に不備があった場合はテストプロセスの一部または全体を繰り返す必要があります。テスト評価の結果はテストレポートの一部として文書化し、発見されたテスト対象の問題はエラー トラッキング システムに入力します。

13.3.5 テスト管理

テスト管理の役割は、テストプロセスを正しく適用し、そのプロセスの目的を確実に達成する事です。これ以外にも、テスト管理は多くのタスクを実行する必要があります。例えば、社内(テストチーム)および外部の担当窓口(調達、プロジェクト管理、サプライヤ、品質保証)の決定や、テスト成果物のレビューの調整と仲介の他、テスト計画およびシステム受け入れ業務に積極的に関与します。

13.4 部品テスト

部品テストはサプライヤ側で実施済みですが、OEM側でも部品検収テストを実施します。これは、そのデバイスが部品の要求仕様を満たしている事を確実にしてからシステム統合に移るためです。部品テストで何を重視するかは、デバイスの開発ステータスで異なります。

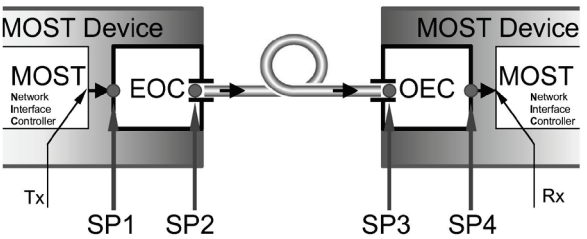


図13.2:MOST物理層のアーキテクチャ(出典:[ComPhyLP 150])

このテスト段階で特に重要なのが、光モジュールのテストです。図13.2でEOC(電気/光コンバータ)およびOEC(光/電気コンバータ)と表記されたFOT/ピグテール モジュールを、第12章で説明したMOSTコンプライアンス プロセスに基づいてテストし、仕様への適合性を確認します。このテストは、特に仕様点SP1とSP2の間、およびSP3とSP4の間で実施します(第6章参照) [MOST 2003]。OEMは、デバイス内部でモジュールが正しく統合されている事をテストします。ここでは、FOTの入力信号が仕様レンジ内にある事の確認に重点が置かれます。よくある問題として、FOTの周辺回路の不良、NICとFOTの間のPCB配線が長すぎて起こる反射、デバイス内のピグテールの曲げ角度が大きすぎて起こる減衰等があり、これらをテストする必要があります。

さらに、他のモジュール(例: GSMモジュール)からのEMC干渉、およびFOT/ピグテール モジュール自体の原因(デバイス ハウジング内のMOSTコネクタのプラスチック開口)による他のモジュールへの干渉についてテストを行います。出力信号を計測するため、一般的には通常動作中のNICで信号を生成し、SP2で計測を実行します。無作為テストでは、計測した信号と事前に決定した許容限界値の間に、量産環境での品質ばらつき、および経年変化と温度ドリフトを補償するのに十分なマージンがある事が期待されます。許容限界値は、物理層コンプライアンス テストで文書化します。

OECは、制御デバイスの入力点でもテストします。MOST仕様に準拠した光入力信号をSP3に印加し、光パワーの仕様レンジ(-2～-23 dBm)全体で変化させます。中距離レンジで光レシーバの増幅が切り換わった時のパルス幅歪み(PWD)の増大といったエラーの発生を検出するには、光入力パワーの全レンジで信号ロックが得られる事が必要です。SP4でのタイミング パラメータをチェックするには、テスト装置を使ってワーストケースの入力信号をSP3に印加し、オシロスコープを使って計測します。

MOST物理層のテストでは、テストの構成に特に注意を要します。入力容量の非常に小さいアクティブ プローブと、広帯域オシロスコープが必要です。また、計測構成によるライン反射等、好ましくない副作用がテスト中に記録されないように注意する必要があります。また、MOSTシステムはリング アーキテクチャであるため、閉じたリング内でデバイスを計測しなければならない点に注意が必要です。

MOSTネットワーク管理機能のテストは、部品テストの重要な項目の1つです。このテストは、テスト対象の部品がNetworkMasterかNetworkSlaveかで内容が異なります。デバイスの起動、デバイス情報とFBlockの問い合わせ、シャットダウンをテストします。特に自動化したテストを連続的に実行する場合、各テストはデバイスが特定のステートにある事が前提となるため、デバイスが正しくシャットダウンする事が次のテストにとって重要です。

また、リングブレーク診断、自動再送(低、中、高レベル)、温度管理等の機能をテストします。温度管理のテストは、正確かつ繰り返し可能な温度変化をテストできる人工気候室で実施します。

部品テスト(特にコア コンプライアンス テスト)では、FBlock ETで温度をシミュレートします。この方法は、人工気候室を使うよりも短時間で簡単です。

車載環境で使うMOSTデバイスの場合、電源電圧の変動への耐性が重要な特性の1つです。静的低電圧テストは、ある一定の電圧しきい値での挙動(例: シャットダウンが実行されるかどうか、機能が制限されるかどうか)をチェックします。動的低電圧テストは、電圧を変化させてデバイスをテストし、どのレンジで変動が補償されるかを確認します。

検収テスト中に見つかったエラーをサプライヤ側で再現できるように、部品検収テストの結果をサプライヤに提供する事がよくあります。サプライヤ側でエラーを再現するには、テストスクリプトだけでなく、テストの実行に必要な全ての関連するツール、データベース、文書も必要です。

ネットワーク管理層のテストで特に課題となるのが、可変電源を使った電源の自動化と、シミュレーションおよび計測の時間精度です。

最近のインフォテインメント システムは機能が複雑化しているため、制御デバイスの相互接続性と機能をなるべく早い段階で確認しておく事の重要性が高まっています。しかし、システム横断的な機能の実装に必要な全ての制御デバイスが同時に揃う事はあまりありません。このため、部品テストではまずリメイニング バス シミュレータとテストシステムを使って相互接続性をテストする事があります。ここでの目標は、デバイスのMOST機能を効果的にテストするのに十分なシステム環境を提供する事です。このようなテスト環境で必要な機能の例として、MOST NetworkMaster、およびデバイスに実装するファンクション ブロックをアドレス指定するFBlockコントローラがあります。

13.5 統合テスト

統合テストの目的は、MOST部品間の相互接続性を確認する事です。多くの場合、統合テストは必要な全てのインターフェイスに簡単にアクセスできるカスタマイズ済みのテスト基板を使って実験室環境で実行します。これには、構成と統合が短時間で完了し、テストおよび監視ツールを柔軟に利用できるという利点があります。

リソースおよび接続管理、診断、ソフトウェア ダウンロードといったネットワーク管理機能は複数のデバイスで機能を実現しており、シミュレートには相当な手間がかかるため、多くの場合、統合テストのフェイズでしかテストできません。汎用FBlockのテストもこのフェイズで実施します。このテストでは、デバイス内の各FBlockに対して問い合わせを実行し、仕様で定義されたファンクションが全て実装されMOSTコア コンプライアンステスト仕様[Core]に従って動作している事を確認します。例えば、誤ったOPTypeをアドレス指定した場合やパラメータの限界値を超えた場合に正しいエラーコードが返されるかどうかをテストします。

ファンクション ブロック間の動的挙動と相互接続性は、Message Sequence Chart [Z.120 99] (補遺A参照)を使って定義します。全ての作業を手作業で行った場合、1つのMessage Sequence Chartのテスト設計、実装、実行、評価だけでも数日を要す事があります。MSCのような形式仕様記述言語でテストを自動生成するとこうした手間が大幅に軽減され、これまで以上に入念なテストが可能です。しかし、MSCから実行可能なテストを得るのは意外と難しく、以下の手順が必要です。

テスト インターフェイスの定義:仕様では、システム内の全てのFBlockの連携を記述します。統合のフェイズによっては、一部のFBlockまたはFBlockコントローラが準備できていない状態でテストを実行する事になります。テストを完全に自動で実行できるようにするには、システム内の少なくとも1つのデバイスをシミュレートする必要があります。例えば、統合テストではユーザによるキー押下をシミュレートする事がよくあります。部品テストでは、少なくともテスト対象の部品と直接通信するデバイスをシミュレートする必要があります。テスト インターフェイスは、どの部品をテストしどの部品をシミュレートするかを定義します。

テストの挙動のモデル化:上記のテスト インターフェイスをテスト対象の全てのMSCで特定したら、その結果得られた複数の仕様を1つの完全なモデルに結合します。複数のMSCに分散している経路を結合し、他のMSCへの参照を解決します。モデル全体を構築するのに必要な個々のMSC間の関係は、通常ハイレベルMSCを使って定義し、ここから個々のテスト シーケンスを推論します。

テストデータの生成:MSC仕様で記述したメッセージは、実行中に変化するパラメータを含む事があります。代表的な例は、通話のステータスを記述するメッセージで使うパラメータtelephone numberです。このようなメッセージをテスト環境でシミュレートする場合、具体値を使う必要があります。これらの値をどのように選択するかが、テストに大きく影響します。例えばアドレス帳に既に登録されている電話番号の場合、発信者の名前を表示する必要があります。関連するテスト シー

ケンスを特定するだけでなく、どのパラメータ値を使ってそのシーケンスを実行するかを特定するには、体系的アプローチが必要です。

これ以外にMOSTシステムの統合テストに関して問題となるのは、仕様で定義されていないシステム挙動(主にエラーのケース)のテストや、システム構成が多岐にわたる場合(例: 追加のオプション機能や国別のバリエーションの組み合わせで構成が増える場合)のテストです。

テストの具体化:テストケースを実行するには、抽象化のために仕様には含まれていない多くのシステム固有情報(例: リング内の個々のデバイスのノード位置アドレス)が必要です。そこで、テスト記述から実行可能なテストスクリプトを生成するための具体化を実行する必要があります。

13.6 システムテスト

システムテストでは、MOST部品を最終的な車両システムとして統合してテストします。システムテストの目的は、全てのMOST部品とそれ以外の車両システム(例: 電装システム、CANバス)の相互接続性、および実車環境下での個々の部品の安定性をテストする事です。ここでは、実験室環境でテストできない状況をなるべく多く体系的に網羅する事が特に重要です。例えば、フィールドでほとんど起こらなくても重大な動作異常につながる可能性のある各種動作条件の組み合わせを網羅しておく必要があります。この場合、同一条件でエラーの原因を繰り返せる事が重要です。この作業では、バス全体の通信を記録、解析できるデータロガーをよく使います(セクション13.7参照)。通常、これらのツールでは大量のデータが生成されるため、必要なデータを取りこぼさないように注意しながらフィルタおよびトリガ条件を設定してデータを絞り込む必要があります。システムテストの実施でもう1つ問題になるのは、GSM携帯電話とのシリアル接続等、異なるバスシステムの時間を同期して、他のバスからのメッセージに対するMOSTバスの応答を再現できるようにする必要があるという点です。エラーの原因を特定するには、MOSTリングの特定のセクションのみ信号パワーを低下させる等、MOST物理層に変化を加える事が必要になる場合もあります。

13.7 テストツール

体系的、効率的、繰り返し可能なテストを可能にするには、各種テストに対応したテスト環境とテストツールが必要です。規格品の標準テストツールは、テストプロセス全体の一部しかサポートしません。このため、商用および独自のツールを組み合わせ、各OEM固有のプロセスの個々の要件に合わせてカスタマイズしたツールスイートが必要です。テスト環境とテストツールを使う際は、これらが技術的に成熟している事、そしてテスト対象に関係ないエラーレポートが生成されないようにする必要があります。このセクションでは、MOSTシステムのテストに関係するテストツールの概要を示します。

TestDirector [Mercury]等のテスト管理ツールは、テスト計画とテスト トレーサビリティをサポートします。要件をインポートして、テストケースの定義中に参照できます。そうするとテスト要件とテスト結果を直接関連付ける事ができます。多くのテスト管理ツールには、不具合管理の機能もあります。これらの機能は、テスト中に見つかった不具合の管理、およびこれら不具合解決の追跡をサポートします。また、テスト管理ツールにはテストプロセスの結果(例: 実行したテスト回数、未解決不具合の数)をチャートまたはレポートとして出力する機能もあり、システム品質をプロジェクト管理者に報告するために使えます。

テスト実施を自動化するツールには、大きく2つの種類があります。1つはMOSTバス上でのMOSTの挙動を監視および記録するツール (トレーシング ツール)で、もう1つは特定のスティミュラスをあらかじめ定義されたスクリプトに基づいてテスト対象に送信し、その応答と予測されるMOST信号を比較するツールです。テスト対象の挙動をなるべく完全に把握するために、通常は両方のツールを組み合わせて使います。

MOSTトレーシング ツールに求められる重要な特徴は、トレースの時間精度、記録したメッセージをXMLベースのファンクション カタログ仕様に基づいて(リアルタイムに)デコードする機能、分割されたメッセージの自動合成、パケットデータ チャンネルと制御チャンネルの監視、複雑なトリガおよびフィルタ条件の定義、ストリーミング データ チャンネルとパケットデータ チャンネルへのアクセス等です。一部のトレーシング ツールは、デスクトップまたはノートブックPCと接続しなくても車両内に設置してデータロガーとして使えます。このようなツールは車両内に容易に組み込んでMOSTバスのログを常時記録できるため、特にシステムテストに適しています。MOSTシステム用のトレーシング ツールは、Microchip社[Microchip]、Vector社[Vector]、Optitas社[Optitas]から提供されています。

テストスクリプト ベースのMOST用テスト自動化ツールは、GADV社[GADV]、Ruetz Technologies社[Ruetz]、Vector Informatik社[Vector]から提供されています。これらのツールはMOSTインターフェイスだけでなく、他の車載バスシステムもサポートしています。これらのツールに求められる重要な特徴として、テスト結果を明確に評価できる事、結果を計測する際のシステムの時間精度、受信メッセージに対する動的な応答機能等があります。

標準化されたテスト技術の1つに、TTCN (Testing and Test Control Notation)があります。最新バージョンのTTCN-3 [ETSI 2003]は、ETSI (ES 201 873シリーズ)とITU-T (Z.140シリーズ)で標準化されています。TTCN-3は非常に抽象度と表現性の高い表出型のテスト仕様記述言語で、MSC-2000の性格を強く受け継いだグラフィカルなプレゼンテーション フォーマットも備えています。TTCN-3を使うと、仕様と同じ抽象度で統合テストを記述でき、仕様と同等の可読性が得られます。従って、元々電気通信プロトコルのコンプライアンス テスト標準化手段として登場したTTCN-3は、今後もその特性を活かし、標準化したテストの記述に適した言語として定着する可能性があります。テスト言語以外に、TTCN-3は言語の動作セマンティクスも標準化しており、ツールが異なっていてもテストスクリプトの挙動の一貫性が保証されます。このため特定のツールの使用を指定しなくても、MOSTテストを異なるOEMおよびサプライヤ間で交換できます。TTCN-3がMOSTシステムのテストに対応できる事は、一部のOEM [Burton 2004]とMOSTテストツール メーカー[Ruetz]が検証済みです。

14 MOST150への移行

14.1 MOST150のアプリケーション

車載インフォテインメント システムの要件に特化したMOST25テクノロジは高い品質に定評があり、各種車両プロジェクトで採用実績があります。長年にわたり、プロセスチェーン(開発、品質保証、製造、カスタマサービス)全体でMOST25に関する専門知識の蓄積が進み、プロセスが確立されています。しかしビデオ転送やEthernet接続等、車載インフォテインメント システムに対する要求がますます高度化する中、MOST25は特に利用可能な帯域幅の面で限界に達しています(図14.1参照)。この結果、第3世代のネットワーク テクノロジであるMOST150移行への下地は整っています。

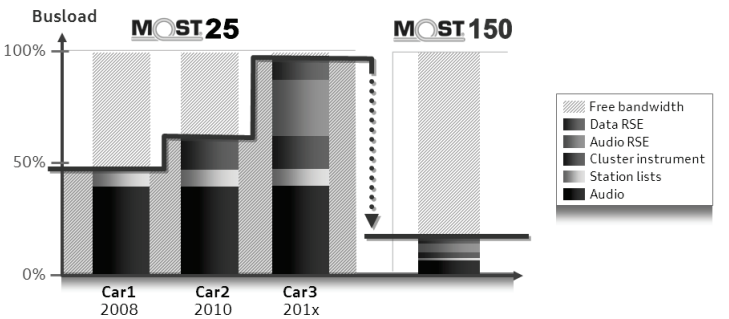


図14.1:Audi社システムにおけるバス負荷の増大

既存のMOST25テクノロジの正統な後継技術として、MOST150は以下の強力な利点を備えています。

- MOST25に比べ帯域幅が6倍に拡大(図14.1参照)
- IPデータをネイティブに転送するEthernetチャンネル
  これにより、IPアプリケーションの柔軟な導入が可能
- アイソクロナス ストリーミング
  ビデオ ストリーム(MPEGトランスポート ストリーム、VideoOverMOST)の転送が可能
- MOST25アプリケーションのリユースが可能
  現在のインフォテインメント システムに使われているMOST25アプリケーションは、ほとんど変更なしにMOST150への移行が可能

MOST150マルチチャンネル ネットワーク

MOST150では、1つのバスシステムで上記全てのサービス (パケットデータ転送と時間同期ストリーミング)を並行して利用できます。システム設計に応じて、個々のサービスに割り当てるリソースを実行時に動的に変更できます。

14.2 MOST150テクノロジの評価

14.2.1 要件と評価の構成

MOST150に移行するには、MOST25と同等の成熟度に達する事、すなわち量産適格性が実証される事が前提条件です。すべてのOEMは、新しいテクノロジを導入する前に徹底的なテストを実施しています。MOST150のように、既存システムの発展的進化の結果を評価する場合、現行の認証方法、システム、プロセスを引き続き利用できるかどうかにも注意を払います。しかし、改善点と新機能が最も重要である事には変わりありません。重要な要件が満たされ、ソフトウェア、ハードウェア、ツール、テストシステムの品質が量産レベルに達して初めて、新しい技術に移行できます。

図14.2に、MOST25からMOST150へ環境を移行する場合に影響を受ける階層を示します。変更の範囲は、物理層からアプリケーション層にまで及びます。このため、影響を受ける全ての要素を検証できるように、評価プロジェクトの構成には十分注意する必要があります。

MOST150の場合、以下の作業パッケージを含めるように評価プロジェクトを構成できます。

- **物理層**
  ここでは、関連する全てのハードウェア(POF、FOT、ピグテール、INIC)とそれぞれの電気的パラメータを検証します。
- **ネットワーク サービス/INICファームウェア**
  これらはMOSTバスに対するソフトウェア インターフェイスであり、性能と信頼性に大きな影響を与えます。
- **ネットワーク管理/電源管理**
  これらはネットワークの基本的な要素となる機能で、管理、復帰、スリープ動作に必要です。

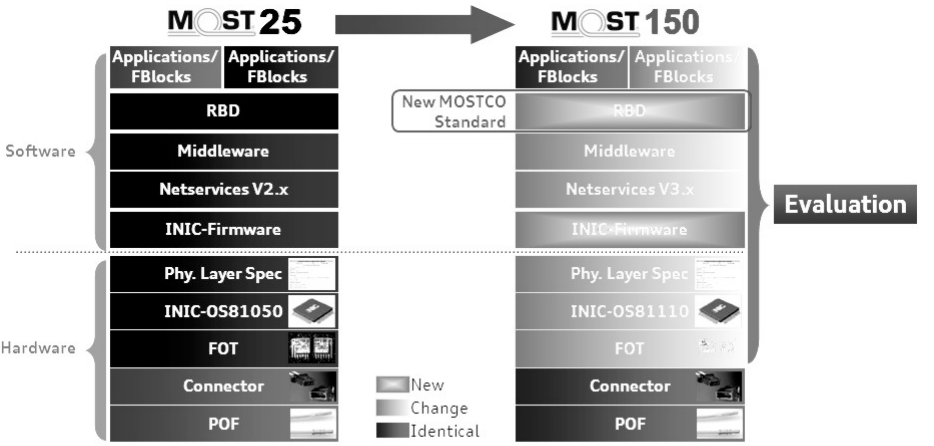


図14.2:移行の影響を受ける階層

- **MOST診断機能**
  MOST診断機能に関しては、リングブレーク診断(RBD)、MOST150で初めて標準化されたECL (Electrical Control Line)、突発的なリングブレークおよびクリティカル アンロックの発生場所を検出する新しい診断機能「突発的信号OFF/クリティカル アンロック」(SSO/CU)を検証します。

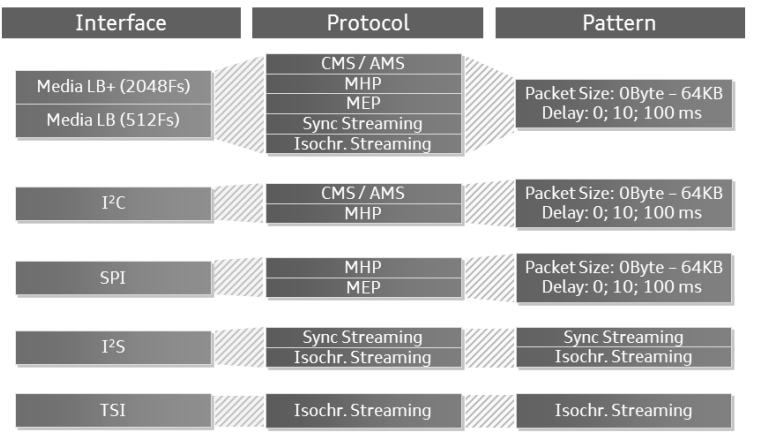


図14.3:評価用ユースケースの定義

- **ユースケース**
  MOST150は各種構成およびECUに導入される事になります。OEMに関係のある全てのMOST150アプリケーションをカバーするため、抽象モデリングでユースケースを生成します。実際のアプリケーション シナリオ(例: メディアプレーヤからヘッドユニットへタイトルリストを送信)に基づき、ネットワー

キングおよびインフォテインメントの担当部門がINICインターフェイス、ネットワーク プロトコル、対応するデータパターンの組み合わせを定義します(図14.3参照)。多数のユースケースを用意する事で、全てのINICインターフェイス(3ピンMediaLB、6ピンMediaLB、I^2C、SPI、I^2S、TSI)、MOSTプロトコル(CMS/AMS、MHP、MEP)、MOSTチャンネル(制御チャンネル、パケットデータ チャンネル、ストリーミング チャンネル)を検証できます。

- **ツール**
  MOST150の構成要素を量産規模で開発するには、ツールの対応が欠かせません。アクティブ型ツールとリッスンオンリー型ツール自体の検証、およびこれらのツールのテスト オートメーションへの統合を検証する必要があります。

前述の通り、テクノロジが量産レベルまで成熟しているかどうかを評価するには、幅広い要件を考慮に入れる必要があります。以下のセクションでは、1つの評価方法の例としてVolkswagenグループが実施した評価プロジェクトを紹介します。

14.2.2 MOST150リファレンス プラットフォーム

MOST150のテクノロジ評価に対する要件は多岐にわたるため、2つの検証プロジェクトを立ち上げる必要がありました。最初のプロジェクトのハードウェアはMOST150 リファレンス プラットフォームと呼ばれ、TimingMaster、NetworkMaster、PowerMasterを含む7ノード(評価用基板)のMOSTリングで構成されます。このシステムの目的は、前セクションで説明した全ての作業パッケージを実験室環境で検証する事です。各評価用基板は、プラグイン方式で容易に交換可能なドータボードにINICとFOTを実装する事により、MOSTハードウェアを切り換えられるように設計されています。最初に重点を置いたのは、復帰/スリープ挙動、信頼性、バス診断(RBD、SSO/CU)のテストです。次に、このシステムが提供する全てのINICインターフェイスへのアクセスを利用して、事前に定義した72のユースケースをこのプラットフォーム上で完全に実行しました。ここまでで、Audi社とVolkswagen社に関連する全てのアプリケーション シナリオを検証できました。3つ目の柱として、全てのMOSTチャンネルと伝送プロトコルの性能を完全に計測しました。

レイアウトの都合上、MOST150リファレンス プラットフォームは車両への設置には適しません。

14.2.3 MOST150_InCar

次に、MOST150_InCarシステムを使ってMOST150のプロトタイプを車両環境へ組み込みました。このシナリオでは、車両環境がMOST150の通信に与える影響を特に重視しました。

以下の分野でテストを実施しました。

- 過電圧、低電圧: オンボードの電装系シミュレータを使って電源電圧を変動
- 温度ストレス: 人工気候室内で各種温度プロファイルを適用
- 復帰/スリープ挙動: 間隔を変えて車両のバスを自動で復帰
- 試験運転: 実際の運転環境で通信を検証(受信パターンにエラーがない事をチェック)

これらのテスト以外に、MOST150_InCarにはリファレンス実装も含まれており、これを使って異なるECLトランシーバの相互接続性の確認、およびリングブレーク診断の検証を実施しました。

14.2.4 テクノロジ評価の結果

以下に、評価を実施および完了して得られた結果を、評価の構成と同じ順番で紹介します。

- **物理層**
  MOST通信に関連する全てのハードウェアの検証に成功しました。量産レベルでの部品の入手性も同時に調査し、問題ない事を確認しました。

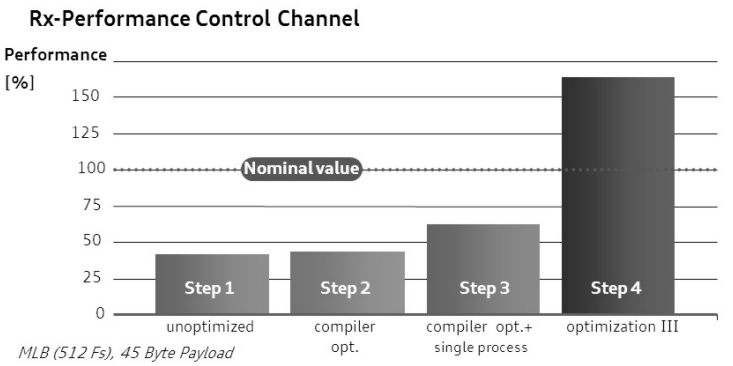


図14.4:制御チャンネルの最適化ステップ

- **ネットワーク サービス/INICファームウェア**
  性能、機能、信頼性の面でバージョンの異なるネットワーク サービスとINICファームウェアをいくつか検証しました。検出された不整合を解決した結果、重要な分野で性能が大幅に向上しました。図14.4に、制御チャンネルのスループットの例を示します。ここでは、異なる最適化手法を試験的に適用した所、最終的にネットワーク サービスの「最適化III」と呼ばれる手法で、それまで社内で定義していた基準値をさらに上回る結果が得られました。

- **ネットワーク管理/電源管理**
  この機能分野を検証するため、数万サイクルの復帰/スリープを実行しました。そして、ネットワーク ストレス、電圧ストレス、温度ストレスを与えた条件下で挙動をテストしました。この幅広い統計データに基づき、この機能分野に問題がない事を検証しました。
- **MOST診断**
  リングブレーク診断の検証に加え、新機能のSSO/CUおよびECL通信の評価結果も良好でした。このため、MOST25でも定義していた静的中断の診断だけでなく、短時間のブレーク検出やエラー発生時の通信といった新しい機能も一律に仕様に定義され、全てのOEMで利用できます。
- **ユースケース**
  テクノロジ評価で最も中心となるのが、関連するユースケースのテストです。このテストでは、パケットサイズとタイミングを変えた異なるパターンを用意し、各パターンを少なくとも6時間適用して実行しました。4,800時間を超えるテスト実行の結果、アプリケーション シナリオの評価を正しく実行できました。これらテストの副産物として、サプライヤから実装、テスト、開発までプロセスチェーン全体の弱点を洗い出し、最適化しました。これにより、エラーが検出された場合、対応する連絡窓口を通じて迅速に対策を提供し、統合し、再テストできました。開発リスクを最小限に抑えるには、このようなプロセスが非常に重要です。
- **ツール**
  MOST150は新しいテクノロジであるため、検証プロジェクトでの計測に使うツールは、新たに開発する必要がありました。このため多くの場合、動作異常が発生した場合にMOST150テクノロジ自体が原因なのか、ツールまたは計測用ソフトウェアの実装に原因があるのか切り分けが難しという問題がありました。この領域で発生したエラーへの対策をとってもらうために、ツールメーカーとの緊密な協業が必要でした。

14.3 まとめ

2つのプラットフォーム(MOST150 リファレンス プラットフォームとMOST150_InCar)を使う事で、MOST25からMOST150への移行で影響を受ける全ての要素を実験室環境と実車環境の両方で検証できました。

評価の過程で検出されたベーステクノロジの問題は全て修正できました。データレート、信頼性、機能、部品の入手性と品質は期待通りの結果が得られました。これらの評価結果に基づき、MOST150の量産への移行を果たしました。それでも、特にMOST High Protocolを使うアプリケーションについては性能とプロセッサ負荷に関する最適化の余地が残ります。これらの最適化は、今後MOST Cooperationの仕様およびサプライヤのソフトウェアが更新される事で量産プロジェクトに導入できます。

この評価プロジェクトに関連して収集した専門的知識は、将来の量産開発に大きなメリットがあります。以下の方法で拡張されたネットワーク サービス統合プロセスを適用する事により品質が向上します。

ネットワーク サービスまたはINICファームウェアの新バージョンがリリースされたら、暫定的な統合を実行した後、MOST150リファレンス プラットフォームに対して回帰テストを実施します。これらのテストが正しく実行された場合のみ、インフォテインメント デバイスへの統合を行います。

これにより、異なるバージョンおよび構成間の相互接続性がMOSTシステムレベルで保証されます。

14.4 今後の展望

帯域幅に十分な余裕があり、民生用電気機器の接続をサポートしたMOST150は、インフォテインメント ネットワーキング向けインフラストラクチャとしての将来性があります。

技術上の優位性により、MOSTテクノロジは今後新しい車載アプリケーション分野でも存在感を増すと考えられます。1つの可能性として、運転支援システムの分野があります。ネットワーキングに対する画像処理とセンサデータ処理の要件は、インフォテインメント システム分野の要件とほぼ同じです。

- システムの複雑さは、画像処理システムとインフォテインメント システムで同等
- 転送されるデータタイプ (パケットデータとストリーミング データ)の類似性が高い
- 遅延を最小限に抑えてオーディオデータとビデオデータを時間同期して転送する事が必要

さらにコンセプト解析を進めていけば、MOSTが回路構成と伝送信頼性の要件にどの程度まで対応できるかを明確にできます。

15 MOST150への移行

15.1 はじめに

この章では、MOST25からMOST150への2つの移行方法を提案します。1つは進化型コンセプトで、特別なハードウェア インターフェイスを使って全てのMOST25デバイスをMOST 150にアップグレードします。もう1つはMOST25とMOST150の間に特別なブリッジ アーキテクチャを開発する方法です。MOST Cooperationは進化型のアプローチを推奨しています。

15.2 MOST150への進化型アプローチ

MOST25のインフォテインメント システムをMOST150に直接移行するアプローチをMOST150進化型コンセプト[Schr 01] [Schr 02]と呼びます。このコンセプトは、個々のECUのハードウェアとシステム ソフトウェアをアップグレードして、デバイス全体をMOST150に対応させるというのが基本的な考え方です。このアプローチでは、MOST25システムの既存のワイヤハーネスには一切変更が必要ないため、ワイヤハーネスのコスト面ではMOST25/MOST150ブリッジ コンセプトよりも有利です。

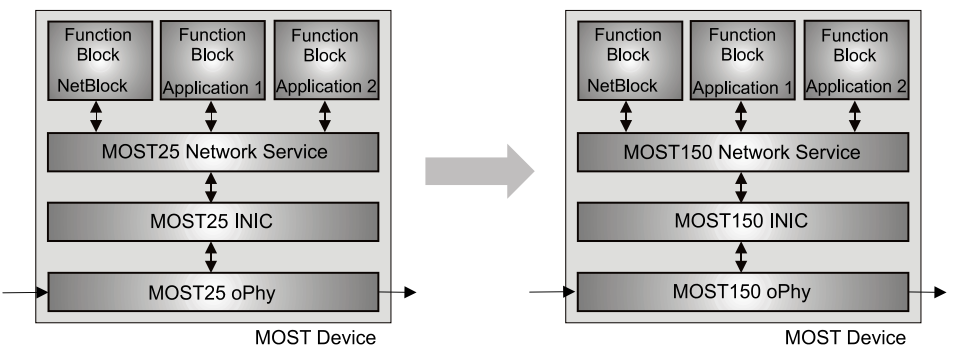


図15.1:MOST150進化型コンセプトの概念図

15.2.1 ハードウェアの進化型移行

進化型コンセプトは、ECUハードウェア全体の再設計なしにECUをMOST150に直接移行させる方法を示す事を目的としています。しかしこの方法の前提条件は、MOST150へ移行するMOST25デバイスがINICアーキテクチャを採用している事です。この条件を満たしていれば、一般的なPCのネットワーク カードを交換するように簡単に移行できます(図15.1参照)。

進化型コンセプトの基本的な作業手順としては、まず既存のネットワーク インターフェイス カードを取り外し、新しいネットワーク インターフェイス カードを取り付けてドライバを更新します。これを車載ネットワークのMOSTにあてはめて言えば、MOST25の部品(INICとFOT)を新しいMOST150の部品に交換し、MOST150ネットワーク サービスを使ってECUソフトウェアをコンパイルし、最後にこのファームウェアをデバイスのフラッシュに書き込むという手順に該当します。

しかし、PCIネットワーク インターフェイス カードが使えるパソコンとは異なり、MOST25とMOST150の部品には物理的なコネクタのピン互換性がありません。このため、MOST150進化型コンセプトを実現するにはMOST150 Phyボードと呼ばれる一種のMOST150ネットワーク インターフェイス カードが必要です。

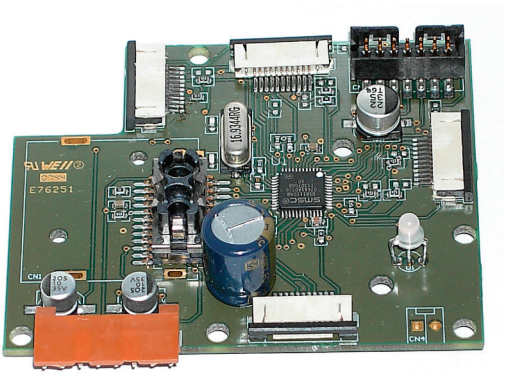


図15.2:MOST150 Phyボード(出典: BMW AG)

Phyボードは、MOST150 FOTとMOST150 INICに加え、抵抗、コンデンサ、水晶振動子等の必要な周辺部品を全て実装しています。新しいMOST150 INIC (OS81110)のポートは、4つのFFC(フレキシブル フラット ケーブル) コネクタに配線されています。これらのコネクタのピン配置は、MOST25 INIC (OS81050)のピン配置に対応するように設計されています。このため、FFCを使ってPhyボードをPCB上のMOST25 INICの基板パターンに1対1で接続できます。このようにして既存のECUをアップグレードする方法を、図15.3に示します。

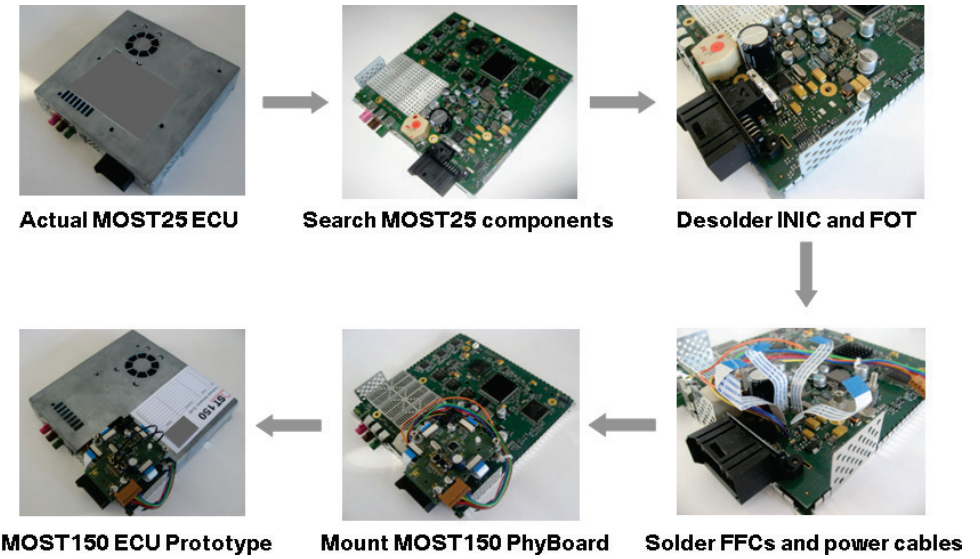


図15.3:MOST150進化型コンセプトの作業手順(出典: BMW AG)

移行対象となるECUの筐体を開け、はんだ付けされているMOST25 INICとMOST25 FOTをPCBから取り外します。次に、FFCをINICの基板パターンにはんだ付けします。Phyボードの電源は、FOTの基板パターンにケーブルをはんだ付けして元のPCBから供給します。次に、MOST150 PhyボードをECUのPCBに取り付けます。これは、スペーサを使って非常に簡単に行えます。次に、FFCと電源ケーブルをPhyボードに接続します。これで、MOST150に完全対応したECUのプロトタイプ ハードウェアが完成します。後は、新しいネットワーク サービスを使ってソフトウェアをアップグレードし、移行済みのECUのフラッシュに書き込むだけです。

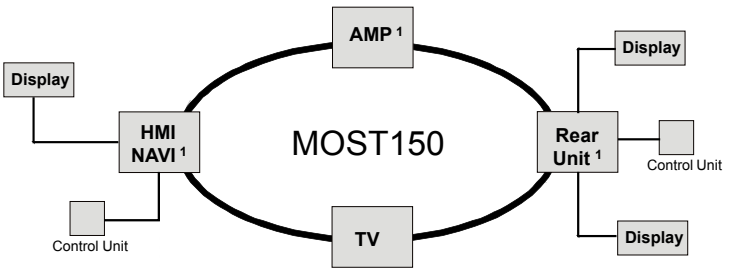
図15.4に、この概念図を示します。いくつかのECUがMOST150にアップグレードされています。

15.2.2 システム ソフトウェアの進化型移行

アップデートが必要なのは、NetworkMaster、NetBlock、ConnectionMasterの各FBlock、およびGeneralFBlockテンプレートの関連するファンクションのみです。アプリケーション自体を変更する必要はありません。しかし、MOST 150の最小限の機能は実装する必要があります。以下に、必須機能の例をいくつか示します。

- MOST150システムは、新しいフレーム構造をサポートする必要があります。
- FBlockバンドル3.0のファンクションで現行システムのファンクションを置き換えます。FBlockバンドル3.0は、GeneralFBlock、NetBlock、NetworkMaster、ConnectionMasterを含みます。

- AMSテレグラムの最大ペイロード サイズが12バイトから45バイトに増えています。このためTelLenフィールドの大きさが12ビットとなり、MOST High ProtocolのTelLenフィールドと同じサイズです。
- サンプリング周波数が30～50 kHzのレンジではなく、44.1 kHzと48 kHzのどちらかになりました。推奨値は48 kHzです。
- ファンクションの「必須」領域と「拡張」領域が統合され、「アプリケーション」領域になりました。
- InstIDを変更できない場合、NetworkSlaveは変更されないリストを含む応答を返すようになりました。



1 MOST150 adapted prototypes based on real MOST25 ECUs

図15.4:MOST150進化型コンセプトの代表的な構成

- Configuration.StatusにはControl=NewExtを使い、DeviceIDも提供します。
- ファンクション クラスが変更されました(例: コンテナ、ストリームの使用に関する制約、ストリーム信号)。
- セントラル レジストリが「フル」の場合、すなわちこれ以上新規FBlockを登録できない場合、NetworkMasterはConfiguration.Status(NewExt, <empty list>)で応答します。全体ではなく一部のFBlockのみを登録できる場合、NetworkMaster.Configuration.Status(NewExt, <partial list>)を送信します。

さらに、実装済みのMOST25/50のコンセプトのうち、MOST150で廃止されたものについては実装を取り除く必要があります。例えば、以下のものがあります。

- MAMACは廃止されました。MAMACを使っていた場合、MOST150ではEthernet over MOSTで置き換えます。
- MOST25に固有のSourceConnect/SourceDisconnectアプローチは廃止されました。
- セカンダリノードのサポートは廃止されました。
- 制御チャンネルでのMOST High Protocolのサポートは廃止されました。
- 「データリンク層48バイトモード」は不要になりました。しかしMOST150でも最大ペイロードはデバイス固有の特性(例: 使用するI/Oインターフェイスまたはバッファサイズ)の制約を受けます。

- SourceHandlesファンクションのサポートは廃止されました。
- NetworkMaster.Configuration.Status(OK)を送信してセントラル レジストリの大規模なアップデートを通知する機能は廃止されました。

これ以外に、条件付き機能とオプション機能も考慮する必要があります。条件付き機能とは、例えばパケットデータ チャンネルを使わないシステムの場合、そのチャンネルでローレベル再送のサポートは不要という意味です。オプション機能はMOST150の新しい機能で、DiscreteFrameアイソクロナス、A/Vパケット化アイソクロナス、QoS IPのユースケースをサポートしたアイソクロナス転送等が導入されています。

Note:MOST150への移行が必要な場合は、MOST Cooperationが発行する最新版の『Application Note MOST150 Migration』[AppN Mig]を参照してください。

15.3 MOST25/MOST150ブリッジ コンセプト

既存のMOST25インフォテインメント システムからMOST150ベースのシステムへのもう1つの移行方法として、いわゆるMOST25/MOST150ブリッジ コンセプトがあります。このアプローチは、まず広い帯域幅が要求される部品のみをMOST150にアップグレードします。それ以外のノードはMOST25ベースのままとします。このコンセプトでは2つの光リングが必要です。MOST25/MOST150ブリッジと呼ばれる新しいデバイスを追加してこれらのリングを連結します。このブリッジはいずれか1つのECU(例: HU)に実装できますが、このコンセプトはコストがかかるため推奨しません。

15.3.1 ブリッジ アーキテクチャ概要

MOST25ネットワークとMOST150ネットワークはブリッジデバイスを使って接続し、ブリッジを経由してストリーミング データ、パケットデータ、制御データをルーティングします。図15.5に、ブリッジの全体的なアーキテクチャを示します。

2つのMOST INIC (INIC_AとINIC_B)は、それぞれのMediaLBインターフェイス[MediaLB]を使ってEHC(外部ホスト コントローラ)に接続します。MediaLB(メディア ローカル バス: MLB)はMOST専用の高速インターフェイスで、MOSTバスの帯域幅を完全に利用できます。同期データのブリッジングの場合、2つのINICをそれぞれのI2Sストリーミング ポートで接続し、EHCの介在なしに同期チャンネルのデータ内容を転送します。

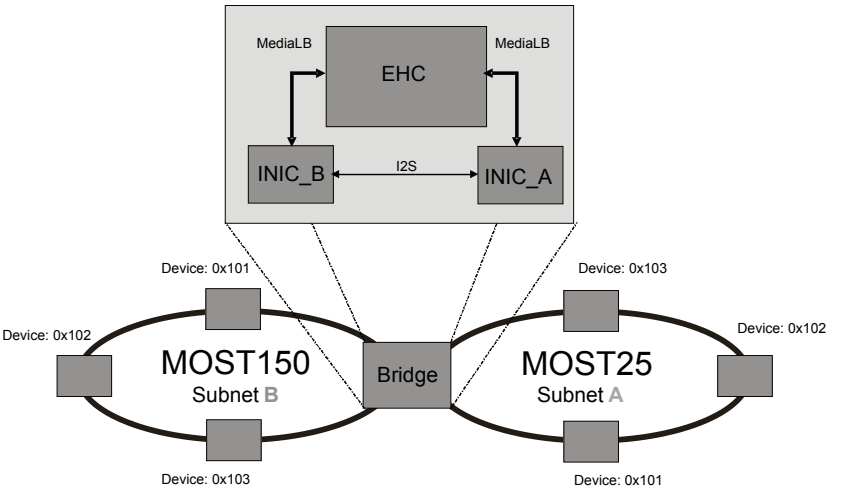


図15.5:MOST25/MOST150ブリッジ アーキテクチャ概要

15.3.2 アドレッシングのコンセプト

MOST150へスムーズに移行するには、2つのサブネットで共通のMOSTデバイスが変更なしにそれぞれのサブネットに統合できる必要があります。このため、通常のアドレッシング方法がブリッジの影響を受けない事、そしてサブネット内部のローカル アドレッシングが現在のMOSTネットワークで使われているアドレッシングと同じである事が要求されます。

ブリッジをまたいだ別のサブネットに属するターゲットのアドレッシング方法を定義する必要があります。解決方法は簡単です。現在、MOSTデバイスはDeviceIDの下位3ニブル(12ビット)のみを使い、上位4ビットは無視しています。そこで、MOSTのDeviceIDの最上位ニブル(4ビット)をSubnetIDとして使ってサブネットのアドレスを指定するようにします。図15.6に、ソースアドレスとターゲット アドレスMOST25/MOST150 ブリッジ経由で割り当てる方法をMessage Sequence Chartで示します。

ブリッジのEHCは、片方のサブネットからのメッセージを受け取るとデバイスアドレスとSubnetIDを変更してから反対側のサブネットへ転送する必要があります。図15.6で、サブネットAのデバイス0x0101からサブネットBのデバイス0x0102へ要求を送信する場合を考えてみます。ターゲット デバイスは反対側のサブネットに属しており、サブネットAではブリッジのNetworkMasterでDeviceID 0xB102として登録されています。そこで、デバイス0x0101はこのDeviceIDのアドレスに要求を送ります。ブリッジは、このメッセージのソースおよびターゲット デバイス アドレスを変更(ソースアドレスにSubnetID Aを追加し、ターゲット アドレスの最上位ニブルをローカル サブネットを表すゼロに設定)してからサブネットBへ転送します。サブネットBのデバイス0x0102からサブネットAのデバイス0x0101への応答も、同様に処理します。

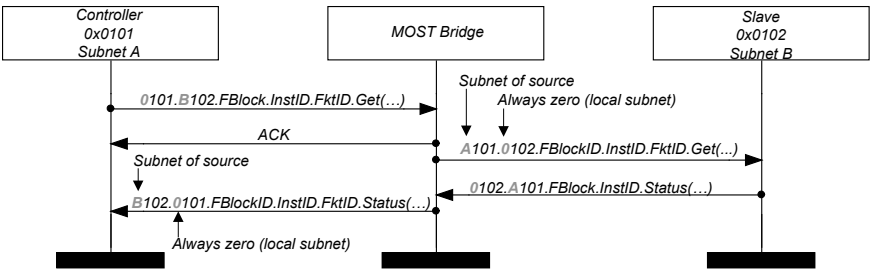


図15.6:サブネットのアドレッシング方法を示したMessage Sequence Chart

15.3.3 ネットワーク管理

各サブネットは、それぞれのNetworkMasterで管理する必要があります。従って、ネットワーク スキャンはローカル ネットワーク内でのみ実行する必要があります。各サブネットの構成は互いに完全に独立しています。構成の変化は、追加または削除されたサービスをブロードキャストすると他のサブネットで認識されます。

各サブネットのセントラル レジストリには、システム全体で検出された全てのMOSTサービスが登録されます。ローカル ネットワーク内のサービスは、通常のDeviceID(最上位ニブルが0)で登録されます。リモート サブネットに属するサービスは、最上位ニブルをそのリモート サブネットのSubnetIDで置き換えたDeviceIDで登録されます。従って、特別なメカニズムを追加しなくても、デバイスはセントラル レジストリに要求を送る事で常に正しいサービスと通信できます。

図15.7に、ネットワーク管理の観点から提案されるブリッジ アーキテクチャを示します。この図では、ブリッジは両方のサブネットでNetworkMasterとして動作します。このコンセプトでブリッジをヘッドユニット以外に統合する場合、ヘッドユニットをスレーブデバイスとして実装する必要があります。レガシーのMOST25システムでは通常ヘッドユニットがマスタとなるため、この事は1つの制約です。

しかしこのような「マスタブリッジ」コンセプトには、以下のような利点があります。

MOSTサービスのインスタンス(FBlockID / InstID)の有効性を、全てのサブネットをまたいで確認できます。従って、ブリッジ内部で全てのアドレス競合を解決できます。

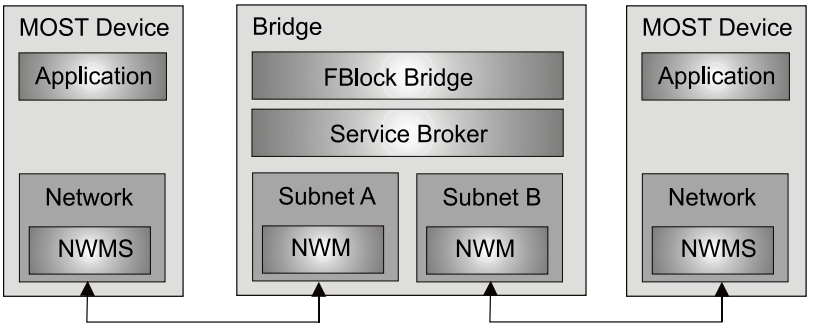


図15.7:ネットワーク管理に関するブリッジ アーキテクチャ
NWMS: NetworkMasterシャドウ
NMW: NetworkMaster

2つのサブネットをMOSTブリッジで結合する場合、両方のNetworkMasterをブリッジ内部に実装する事が1つの大きな利点です。この構成では、両方のサブネットのレジストリの有効性を共通のインスタンスで確認できます。このため、FBlockIDとInstIDのペアを両方のサブネットで一意のものにできます。

「マスタブリッジ」は両方のサブネットのTimingMasterを含むため、システム周波数を完全に同一に維持できます。これによって、2つのINICをI2Sインターフェイスで直接接続し、バッファリングもサンプリング レート変換もなしでサブネット間で同期データを転送できます(セクション15.3.4参照)。

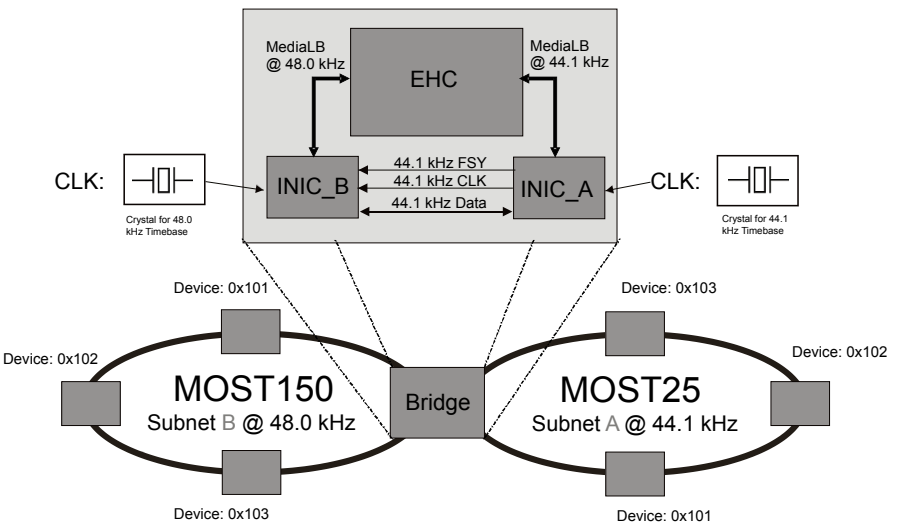


図15.8:非同期サブネットワークを接続するブリッジ アーキテクチャ

15.3.4 PCMデータのブリッジング

システム周波数が異なる場合、サンプリング レート変換回路を実装するか、MOST150ネットワークでアイソクロナス転送を使ってMOST25の同期データを処理する必要があります。図15.8に、レガシーのMOST25リング(44.1 kHz)とMOST150リング(48 kHz)の接続例を示します。MOST150にはアイソクロナス機能があるため、同期/アイソクロナス データもストリーミング ポートを使ってリンクできます。

また、2つのネットワークをブリッジ経由で同期接続する事を考慮する必要があります。

ブリッジ コンセプトの詳細は、MOST Cooperationが発行する最新版の『Application Note MOST150 Migration』[AppN Mig]を参照してください。

16 MOST部品の製造とプロセス

MOSTの光インターフェイスを備えたデバイスの製造プロセスは、光学電子インターフェイスを持たない電子部品の製造プロセスとは大きく異なっており、数多くの変更と修正が発生します。これらの変更および修正点は、大きく2つに分類できます。

1つは、光学電子部品のプロセスとその特性によって影響を受けるパラメータで、これらは従来の電子部品とは異なる特別な対策が要求されます。

もう1つは、通常の電子部品の製造では使われる事がなく、光学電子MOST部品の製造で必要となる全く新しいプロセスと手法で、これらを確立する必要があります。

以降のセクションではこれら2つの面について説明すると共に、製造プロセスに関する要件について詳しく見ていきます。

16.1 MOSTの光学電子部品の構造とテスト

E/Oコンバータ(光ファイバ トランスミッタ、FOX)とO/Eコンバータ(光ファイバレシーバ、FOR)、機械的アダプタ、そして場合によってはデバイス内部のファイバで構成されるアセンブリをピグテールと呼びます。製造プロセスでピグテール モジュールを1つの機械的単位として扱う必要があるかどうかは、個々の製品をどのように実装するかによって決まります。

ピグテールの機械的設計コンセプトは、基本的に2種類あります。1つは、水平または垂直のファイバ出力を持つフレキシブル ピグテールです。これらは、2本の外部ファイバ(以下「ファイバセット」)を使ってコネクタに接続します。コネクタは、ワイヤハーネスを接続するためのMOST光インターフェイスです。ファイバの両端共、端面から光能動素子(PINダイオードまたは光トランスミッタ)までの距離はなるべく小さくしておく必要があります。

もう1つの構造コンセプトは、外部のファイバセットを使わないリジッド ピグテール(例: µPigtail、Minipigtail)です。FOXとFORは1つのコネクタ内に統合され、このコネクタにハーネス側の光プラグイン コネクタを直接挿入します。

この場合、FOXまたはFORとハーネスのプラグイン コネクタの間隔を、短いPOF、スティック状ガラス、レンズ構造のいずれかを使って光学的に接続します。

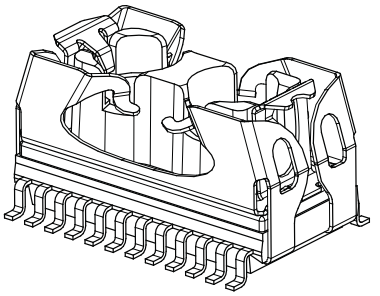


図16.1:MOST150で定義されたピグテール ヘッダ ソリューションとしてのSMD FOT

16.1.1 ヘッダ

ここでのヘッダとは、O/E変換、ライトガイドの位置合わせ、ライトガイドの固定、EMIおよびクロストークのシールドの機能要素で構成される機械的構造です。ヘッダは、これらの要素機能を1つ以上備えた個別部品を組み合わせたアセンブリとして構成される事があります。通常、THM (Through Hole Mount)パッケージ技術のFOTを使ったヘッダ製品は、FOXとFOR、必要なプラスチック アダプタ、シートメタル ケース、ファイバ保持機構で構成されます。

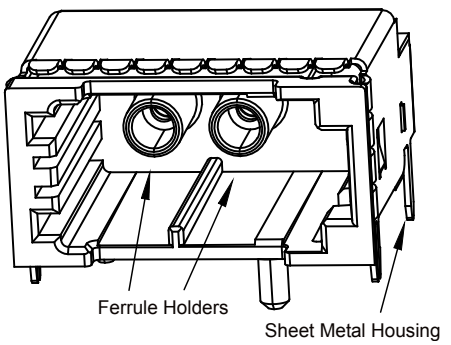


図16.2:電気接点の追加が不要なリジッド ピグテール

リジッド ピグテール ヘッダの場合、FOX/FORとハーネスのプラグイン コネクタの間隔を接続する光インターフェイス部品もあります。その代わり、内部ファイバセットを固定する機構はありません。図16.1に、SOIC24規格に基づいたSMDパッケージ技術のヘッダを示します。MOST150では、このSMDパッケージがMOST150 oPHYサブ仕様で定義されています。この中には、上で定義したデバイス内部ファイバセットに対するインターフェイスおよび保持部品が全て含まれるため、ヘッダに分類されます。

これに対し、リジッド ピグテールのフェルール アダプタは車両のワイヤハーネスに直接接続します。ヘッダとコネクタが一体化しています。図16.2に、リジッドピグテールのフェルール アダプタヘッダの構造例を示します。このヘッダには、内部ファイバセットへの接続はありません。

ヘッダの機能要素

ヘッダ内部に関する技術ソリューションは個々の製品によって異なります。実際の技術的な実装にかかわらず、提供する機能セットは同じです。これらの機能は、専用の部品で実現できます。複数の機能をサブコンポーネントに集約する事もできます。

ヘッダの最重要機能は電気信号と光信号の変換です。従って、光電子半導体部品がリードフレームまたはモールド成形済みリードフレームにボンディングされています。ICダイを接続したリードフレームは、透明のモールド樹脂(または光信号用の窓を開けた不透明なモールド樹脂)を使ってインサート成形でパッケージ化します。モールド成形済みリードフレームの場合、透明のグロブトップで封止できます。

FORとFOXを別々にパッケージ化する場合、FOTを支持し、適切な位置関係に固定するために専用部品を使います。FOTのTHMピン位置の誤差を制限するように配慮したスペーサもあります。

ピグテール ファイバまたはライトガイド部品を十分に位置合わせするには、光軸の基準を機械的に提供するインターフェイス構造が必要です。製品寿命全体にわたってあらゆる条件下で光の良好な結合を得るには、ファイバ フェルールをこの位置合わせ機構に固定する必要があります。この固定方法は、ファイバを繰り返し着脱可能にするか接続したままとするかによって設計が異なります。SMDパッケージでは、一定以上の力で挿入するとフェルールを所定の位置に固定できるクランプ固定法が定義されています。保持力を超える力で引くと、ピグテール ファイバをヘッダから抜く事ができます。

EMI要件への適合を保証するために、適切なシールドが必要になる事があります。シールドケースには、レシーバとトランスミッタ間の電気および光クロストークを防ぐ役割もあります。多くのヘッダ設計では、シールドケースがアセンブリの全体的な固定および安定化の役割を果たします。

構造

部品をシートメタル ケース内に組み立てる際は、適切な組み立てプロセスを用いて、FOXとFORがファイバまたはライトガイドに対して正しいターゲット位置に固定されるようにします。

シートメタル ケースとプラスチック アダプタは誤差が大きいため、これらは実際の光軸の位置合わせには使いません。位置合わせには、FOXまたはFORのケースのキャビティを使います。どのような構造であっても、ライトガイドをFOXまたはFORのキャビティに差し込むのは同じです。このようにして、コンバータとファイバ端面の位置関係を正しく決定できます。位置誤差は、キャビティとライトガイドの誤差範囲より大きくはなりません。

テスト

ヘッダを組み立てたら、取り付ける前に性能と機能をテストします。このテストでは、システム関連のパラメータ(FOXの放射パワー、FORの低入力パワー時のタイミング パラメータ、消費電力)を計測、記録します。これにより、不良品が取り付けられる事を防ぎます。

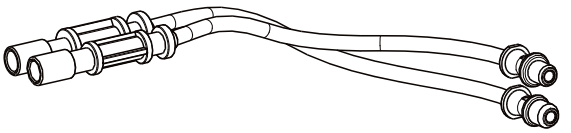


図16.3:MOST150 SMDヘッダとデバイスコネクタを内部で接続するファイバセット

16.1.2 ファイバセット

フレキシブル ピグテールとリジッド ピグテールでは、デバイス内部で使うファイバの種類が異なります。内部で使うファイバは単層コーティングのため、ハーネスのファイバよりもはるかに細い(約1.5 mm)のが特長です。ファイバセットは2本のPOFファイバで構成されます。それぞれのファイバはあらかじめ決められた長さに切断され、両端にフェルールが取り付けられています。ワイヤハーネス側のフェルールの形状寸法は、MOST仕様に適合している必要があります。MOST25とは異なり、MOST150ではSMDヘッダ側の内部フェルールもMOST仕様で定義されています。このため、ファイバセットとSMDヘッダのメーカーが異なっていても互換性があります。MOST25では、ヘッダ側のフェルールはFOXまたはFORのメーカーによって設計が異なる場合があります。この場合、片方のフェルールは標準MOSTインターフェイスに準拠し、もう片方のフェルールはメーカー独自の内部インターフェイスに準拠します。図16.3に、内部ファイバセットの例を示します。

ヘッダ側フェルールは、ワイヤハーネス側フェルールよりもはるかに小型です。POFの長さは、個々のデバイス要件により異なります。

POF

POFは、ソーを使って必要な長さにカットします。図16.4に、POF切断装置の基本構造を示します。図中の矢印は、ファイバの搬送方向を示します。

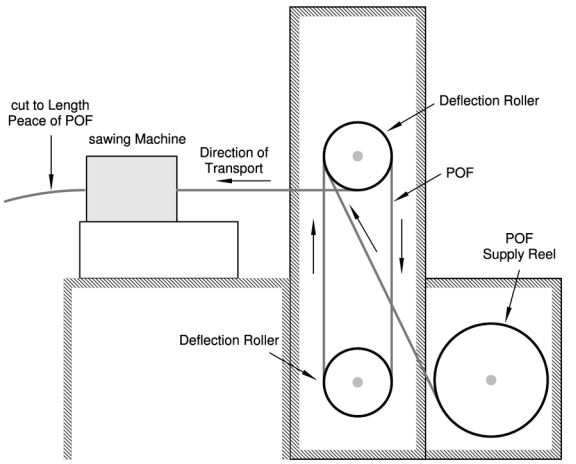


図16.4:POFファイバ切断装置の構造

材料のPOFはロール状で納入されます。このPOFフィードロールから切断装置へファイバを供給します。ファイバは多数の滑車で方向を変え、一定の張力を与えて応力がかからないように送ります。これにより、POFの損傷を確実に防ぎます。

また、切断工程でファイバ端面が損傷または汚れないようにする事も非常に重要です。このため、POF切断装置では専用開発されたソーブレードを使います。ブレードは定期的なクリーニングが必要です。

POFを選ぶ際の重要なパラメータとして、光学特性以外に温度安定性があります。現在、車両のワイヤハーネスで使われているPOFの最大温度仕様は85 ° Cです。これに対し、デバイス内部で使われているファイバは最大105 ° Cまでの温度耐性があります。プロセス中にこの限界値を超える温度にさらされると光学特性が低下する事があるため、このような高温は避ける必要があります。

フェルール

フェルールをライトガイドに取り付ける方法にはいくつかの種類があります。接着剤を使う方法、圧着法、レーザ溶着法です。それぞれフェルールの製造プロセスと関連コストは異なります。標準のMOST製品では、レーザ溶着法を使ってフェルールとPOFを接合します。高出力の短パルスレーザを使って、ファイバの黒いプラスチック コーティングとその周囲のフェルール材料を溶融します。これらの材料が再び硬化すると、フェルールとPOF被覆が高強度のプラスチック接合を形成します。図16.5に、この方法で溶着した接合部(点線の楕円内)の断面図を示します。実際の溶着接合部は、その部分だけ空隙がない事で判別できます。

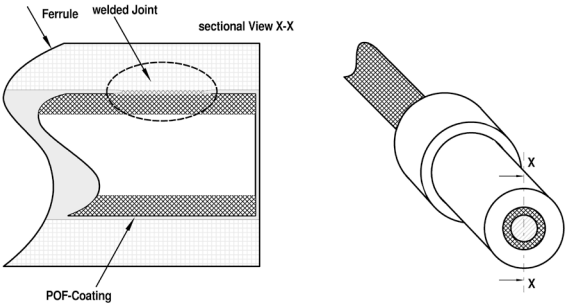


図16.5:レーザ溶着したフェルールの拡大図

引っ張りテストの結果、通常は溶着部よりもPOF被覆の方が先に破損する事が確かめられています。図16.6に、レーザ溶着装置(手動でファイバを送るタイプ)の構造図を示します。

この装置は、作業効率を高めるために2本のファイバを同時に処理します。1回の工程で両方のファイバをフェルールに溶着します。フェルールは、エンドレステープとして2つのロールから溶着ポイントまで供給されます。次に、オペレータがファイバの先端を所定の穴に挿入し、手動で溶着プロセスを開始します。

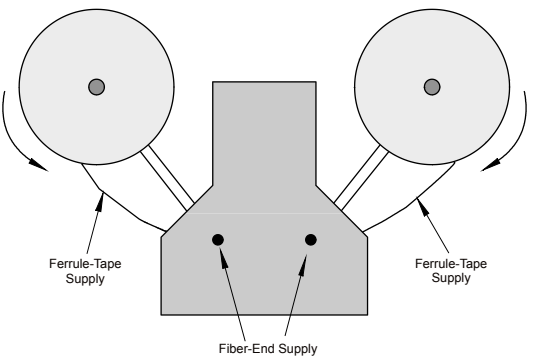


図16.6:半自動レーザ溶着装置の構造

フェルールのプラスチック材料は、レーザ溶着に適したものを使う必要があります。このような材料として知られ、現在市販されているものは1種類のみです。それはポリアミド12で、製品名Grilamidとして何種類かが販売されています。この材料はガラスを含有し、レーザ溶着性に影響を与えず着色も可能です。

製造プロセスでは、ファイバ端面をフェルール端面から0 mm以上後方にずらしておく必要があり、このクリアランスが定義されています。これは、接続時にファイバ同士がぶつからないようにするために必要な措置です。MOST150では、MOST150 FOトランシーバ(SMDヘッダ)の図面でヘッダ側のクリアランスがフェルール端面から0.01～0.1 mm後方と定義しています。MOST25の場合、ヘッダ側のクリアランスは各メーカーの裁量で決めていました。コネクタ側は、MOST仕様で-75 µm +/- 25 µmと定義しています。レーザ溶着装置のフェルール固定機構には1本

のピンがあり、このピンが突出して十分な精度でクリアランスの要件を満たせます。

一般的に、ファイバの減衰はなるべく小さく抑える必要があります。しかし場合によっては、一定の減衰量が必要となる特殊なケースがあります。このような場合、フェルール端面に対するファイバ端面の後方クリアランスを大きくします。この場合も、通常と同じようにレーザ溶着装置のピンを対応する値の分前方に出して、後方クリアランスの条件を満たすようにします。

フェルールは、エンドレステープとしてロール状で供給されます。溶着と切断はどちらもレーザ溶着装置で行います。切断工程では、フェルールをエンドレステープから切断してバリを許容範囲内まで小さくします。

テスト

ファイバセットとヘッダで別々に製造とプロセスを行う場合、減衰値が許容範囲内である事を確認する必要があります。MOST150では、ピグテール ファイバの減衰を最大1.5 dBと定義しています。このため、減衰量が最大許容値のヘッダとファイバセットを組み合わせた場合でも、放射効率と応答性がMOST仕様の最大許容値に準拠している事を確認する必要があります。この条件を満たさない場合、放射効率に関してファイバセットとヘッダの全体的なレイアウトについて最終段階での選択テストを実施する必要があります。

また、無作為抽出による引っ張り強度テストを行い、溶着点の引っ張り強度が十分である事を確認する必要があります。

引っ張り強度の値は非常に安定している事が確認されています。また、応力は非常に小さく、車両のワイヤハーネスとは異なります。

ファイバの光減衰に影響を与える既知の要因には、以下があります。

- フレネル損失(1端面当たり約0.2 dB、1コネクタ当たり約0.4 dB)
- ファイバ同士の軸方向ずれ
- ファイバ同士の半径方向ずれ
- ファイバ同士の角度ずれ
- レーザのパラメータ誤りとファイバ コーティングの厚さのばらつきによるクラッド損傷
- 切断面の光学品質/ソーブレードの状態

フレネル損失は、異なる媒体の界面で発生します。コネクタ1つにつき、減衰が最大で約0.4 dB増大します。

軸方向ずれの最大値は、MOST仕様(コネクタ側)とメーカー仕様(ヘッダ側)で定義されています。

レーザ溶着装置の計測機能を使て、製造したファイバの寸法公差が許容値内である事を確認します。

半径方向のずれは、フェルールとファイバの直径公差、およびPOFコーティングの偏心によって生じます。

ファイバ同士の角度ずれは、コネクタの傾きまたは切断時のファイバ角度ずれによって生じます。後者は、切断装置を適切に準備する事で防げます。前者が原因となる事はほとんどありません。

溶着装置のレーザ出力設定が適切でないと、コーティング厚さには常にばらつきが伴うため、POFクラッドに損傷を与える可能性があります。この場合、減衰が増大します。

経験的に、ソーブレードの状態がファイバの減衰に大きく影響する事が知られており、製造中は十分な監視が必要です。その他のパラメータはそれほど大きく変動しないため、事実上無視できます。前述の通り保守間隔を厳密に定義し、ファイバ減衰を全数テストする事でこの問題を解決できます。

ファイバ減衰を求める方法として、MOST150計測ガイドラインではIEC 61 300-3-4によるインサーション ロス法を推奨しています。このテスト構成は、DUTなしで計測を実行して定期的に校正します。DUTありとDUTなしの場合の光パワーの計測値の差が、DUTの減衰です。

ファイバ減衰を求めるテスト方法としては、IEC 61 300-3-4による置き換え法も使われます。このテスト構成は、参照標準(ゴールデン ファイバ)を使って定期的に校正します。ゴールデン ファイバは、減衰容量と構造が厳密に定義されています。図16.7に、計測の基本構成を示します。

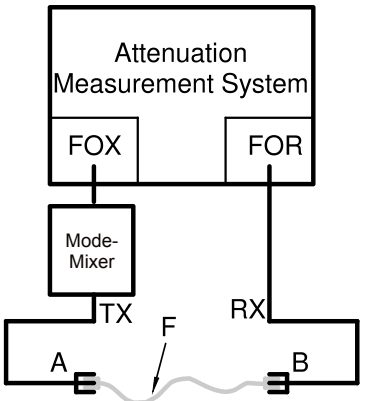


図16.7:ファイバ減衰を計測する基本的な構成

テスト対象のファイバFを校正済み計測装置のテスト接点A、Bにクランプ留めし、計測を始めます。次に、減衰計測装置を使って既知の特性を持つファイバと比較します。こうして、それぞれの減衰を求めます。

以下の内容は、両方の計測方法に共通です。IEC 793-1-1に従って、Fに光を放射します。FOXの出力信号は、モード スクランブラを経由して伝導します。これにより、出力における角度分布と強度密度の配分が「長い」POFを経由した後の端面で予測されるものと同じになるようにします。これは、使う放射源への依存(明点の寸法と形状、角度依存)をなくし、収集したデータを比較可能とするために必要です。

16.1.3 コネクタ

デバイスは、デバイスコネクタを経由して外部のワイヤハーネスに接続します。コネクタの表記(2+N、4+N)は、各デバイスが利用できる光信号と電気信号の数を示しています。互換性を確保するため、MOST仕様ではワイヤハーネスのインターフェイス形状寸法を定義しています。しかしデバイスの内部設計は異なる事があります。例外は、MOST150の2+0コネクタです。このコネクタは、フットプリントと最大寸法が物理層仕様で定義されています。

図16.8に、フレキシブル ピグテールで使う2+4コネクタを示します。右端に4つの電極が見えます。2つのフェルールホールにMOSTフェルールを挿入します。

2+0コネクタはフレキシブル ピグテールとの組み合わせも可能ですが、通常はリジッド ピグテールにのみ使います。これに対し、光接点と電気接点を持つコネクタは主にフレキシブル ピグテールとして実装されます(図16.8参照)。従って、FOXとFORの電気接続、またはMOST部品内で必要な信号線のどちらかの形で、デバイス側の全てのコネクタに電気接点があります。このため、全てのコネクタにはんだ付けが必要です。

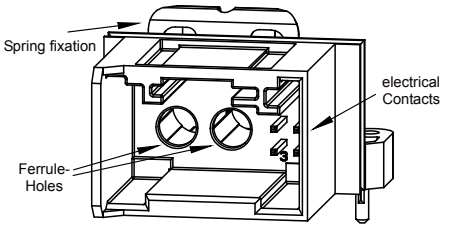


図16.8:2+4デバイスコネクタ

16.2 電子部品の製造に関するプロセス

光学電子部品は、同等の電子部品に比べはんだ付けプロセスの影響を受けやすい特性があります。特にFOXとFORだけでなくPOFもはんだ付けの際の熱に影響されるMOST部品では、この傾向が顕著です。MOST150で定義されたSMDヘッダは、ピグテールをヘッダとピグテール ファイバの2つに分離する事により、ヘッダを自動組み立て可能とすると共に、ピグテール ファイバセットに対する熱負荷を防いでいます。

16.2.1 ヘッダのマウントとはんだ付け

THMベースのヘッダ

通常、THMパッケージのFOR/FOXベースのヘッダは、チップマウンタではマウントできません。手作業でPCBの穴に通してはんだ付けする必要があります。ただし、ヘッダの全てのTHMピンの位置がPCBのピン穴位置および直径の最大公差内に収まり重なり合う事がない事が確実であれば、ヘッダを機械でマウントする事もできます。ピグテール ファイバがマウント済みの状態ではんだ付けを行う場合、損傷を避けるために注意が必要です。ファイバの保護と扱いやすさの観点から、はんだ槽での熱処理の後に内部ファイバセットをマウントする事を推奨します。

ウェーブはんだと部分はんだの場合、通常は赤外線ヒータを使ってアセンブリのはんだ面を最初に予熱します。これは、熱応力を防ぐために必要です。はんだ面の温度を最大100℃まで予熱します。その後、アセンブリが噴流はんだの上を通過します。

このプロセス中、はんだ槽への浸漬時間は数秒ですが、FOXとFORのハウジングとピン、および外部シールドのシートメタル ケースはかなりの高温に達します。はんだ槽内の温度は240～260℃です。鉛フリーのはんだ付けプロセスを使う場合、鉛フリーはんだは融点が高いため、これらの値は約20℃上昇します。FOXとFORは、上記の高温でのプロセスに対応したものを使う必要があります。構造上の理由、および光学的に透明なプラスチック材料を使っているため、高い熱応力でアセンブリに損傷が生じる事があります。プラスチック材料とリードフレームは熱膨張係数が異なります。材料を加熱すると、使っている埋込用樹脂がほぼガラス転移点に達します。この温度では熱膨張係数が大きく変化し、応力が増大します。はんだ

プロファイルが適切でないと、層間剥離(デラミネーション)またはボンディングの剥離が生じる事があります。

はんだ工程前にファイバセットをマウントした場合、ファイバ端面がライトガイドの最大許容温度を超えないように加熱に注意する必要があります。特徴的な損傷パターンとして、ファイバ断面が丸く凹みます。この場合、ファイバ減衰が大幅に増大します。

通常、ここで使う透明の埋込用樹脂は高い吸湿性があります。FOTの代表的な湿度クラスは2Aです。このため未保護の状態での長期間にわたる保管は避ける必要があります。必要に応じて、作業前に部品を乾燥させます。

SMDヘッダ

通常、MOST150物理層仕様で定義されている表面実装用のヘッダパッケージは、テープ&リールで納入されます。これは、電子機器の組み立てラインで広く使われている標準のピックアンドプレース装置で処理できます。SMDヘッダは、DIN EN 60286-3:2008-4に従ってパッケージング テープのポケットに収められており、このテープをリールに巻きつけます。通常、納入されたSMDヘッダの上部にはKapton™ テープが貼ってあり、ピックアンドプレース装置の真空ノズル側はクリーンな平坦面です。従って、SMDヘッダはサイズと重量が同様な通常のSMD部品と同じようにピックアンドプレース装置でPCB上に配置できます。

16.2.2 コネクタのマウントとはんだ付け

ヘッダと同様、はんだ付け前にデバイスコネクタを正しい位置にマウントする必要があります。これは、光インターフェイスを備えた標準コネクタの場合も、µPigtail(電気接点の有無に関わらず)の場合も同じです。形状寸法の理由により、これらの部品は自動ではマウントできません。

標準コネクタのはんだ付けは、それほど注意を要するプロセスではありません。これらは温度安定性の高い受動部品です。ただしヘッダの項で説明したように、リジッド ピグテールは特別な扱いが必要です。この場合、FOXとFORはそれぞれのはんだ付け温度に対応したものを使う必要があります。プラスチック レンズ、およびFOXまたはFORとファイバセットを接続するための短いファイバはどちらも高い処理温度にさらされます。

16.2.3 内部ファイバセットのマウント

ピグテールには、内部ファイバセットとヘッダを別々に処理するもの(SMDヘッダの標準的な考え方)と、ファイバセットとヘッダが一体の状態で納入されるものとがあります。後者のプロセスには特別な扱いが必要です。ただし製造ライン終段のファイバセットの接続は容易です。ファイバセットとヘッダを別々にマウントする場合、最後の熱処理の後でファイバをヘッダ側とコネクタ側にマウントします。コネクタ側のフェルールを固定するバネは、固定だけでなくバネ動作に関するMOST仕様の要件も満たします。ファイバをSMDヘッダ側にマウントする前に、SMDトランシーバの上に貼ってあるKapton™ テープをはがす必要があります。これで、ファイバをそれぞれのキャビティに挿入できます。誤挿入を防ぐため、FOTの穴とファイバ フェルールの直径は送信側と受信側で異なります。この識別には、フェルールの円筒部分とリング状に広がった部分の両方を使います。直径は、送信側フェルールと受信側フェルールで逆です。フェルールのリング状の広がりは、SMDヘッダに内蔵されたクランプ機構に対する座面として機能します。ピグテールファイバはスナップ機構でSMDヘッダに固定され、一定以上の力で挿入すると特定の保持力でフェルールがヘッダのキャビティに固定されます。ファイバを抜くには、この保持力よりも大きい引っ張り力が必要です。

16.2.4 デバイスのテスト

製造工程の最後に、MOSTデバイスの最終テストを実施します。これまで説明したテストとは異なり、このテストは物理レベルと論理レベルで実施します。このテストは、デバイス機能の信頼性を検証し、製造時に生じた不具合を発見する事を目的としています。

スティミュラス信号(放射パワー、信号波形)を適切に選択する事により、入力信号レベルが低い場合のシステム信頼性をテストできます。また、光出力におけるFOXのパワーをテストし、MOST仕様の要件に適合している事を確認します。特徴的なエラーパターンとして、FOXまたはFORの完全な障害、TX出力における伝送レベル不足、信号レベルを絞っていった際の伝送エラーの早期出現があります。

この手順を実行する事で、顧客に納入されるデバイスがMOST通信に関して問題のない事を確認します。

16.3 まとめ

この章では、MOSTの光学電子部品の製造とプロセスについて説明しました。特に従来の電子部品とのプロセスの違いについて説明し、組み立てとテストの手法に対する要件を説明しました。

MOST25は既にフィールドで数年の実績があり、本章で説明した手法に現実的な問題が存在しない事が実証されています。これらの手法は十分に確立され、安定しています。

MOST150によってデータレート向上に対する要求は満たされましたが、それで全ての問題が解決した訳ではありません。製造プロセスにおける取り扱いを簡便化したいという要求に応えて定義されたのが、SMD FOTです。SMD FOTは機械によるマウントが可能で、リフローはんだプロセスを利用でき、選択的ウェーブはんだ付けの必要をなくす事も可能です。完全な表面実装部品であるため、PCB裏面に他の電子部品を実装する事もできます。従って、SMDヘッダを使う事で貴重なPCB面積を増やす事ができます。

17 MOSTへのゲートウェイ

自動車では、複数の通信システムを中央ゲートウェイ経由で接続して使う事がよくあります。信号は、このゲートウェイを介して双方向に伝送されます。

17.1 ゲートウェイを介した信号の分配

図17.1に、車載ネットワークにおける信号分配の例を示します。ABSセンサは車輪の回転数に基づいて車速センサパルスを生成します。ECUはこのパルスを信号WheelSpeedに変換します。この信号は、このCANに接続された全てのECUからと、Powertrain CAN経由で接続されたゲートウェイから利用できます。この種の情報転送は、パブリッシャ サブスクライバ モデルに基づいて行います。センダ (パブリッシャ)は周期的にデータを送信し、レシーバ (サブスクライバ)は必要に応じてバスからデータを読み出します。図に示した例では、中央ゲートウェイがデータを受信し、そのデータをインフォテインメントCANへ送信しています。さらに、そのデータはゲートウェイ機能を実装したHMIでMOSTシステムへ転送されます。例えば、ナビゲーション システムはこの信号WheelSpeedを使ってナビゲーションデータを補正したり、GPS信号を受信できないトンネル内で車両位置を検出したりします(デッドレコニング)。ラジオも、音量調整のためにこの情報を必要とします。MOSTシステム内では、信号WheelSpeedはFBlock Vehicleのプロパティに変換されます。

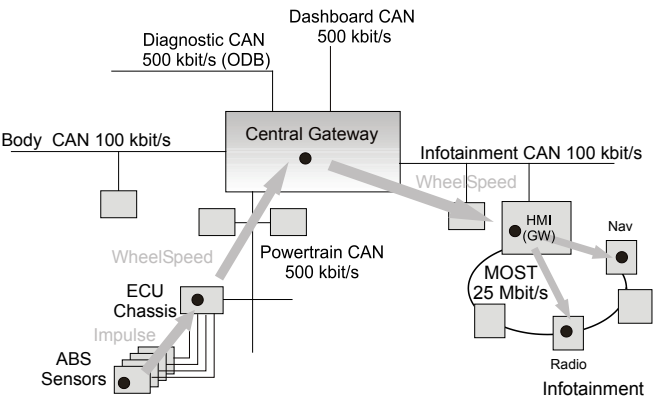


図17.1:ABSセンサからMOSTシステムへの信号転送の例

パワートレインCANのビットレートは500 kbit/sですが、インフォテインメントCANのビットレートは100 kbit/sです。CAN- MOSTゲートウェイは、ヘッドユニット(HMI: ヒューマンマシン インターフェイス)に統合されています。図17.1に示すように、信号WheelSpeedは中央ゲートウェイとHMIの両方で変換される必要

があります。ビットレート変換は両方のゲートウェイで実行する必要があり、HMIではプロトコル変換も必要です。

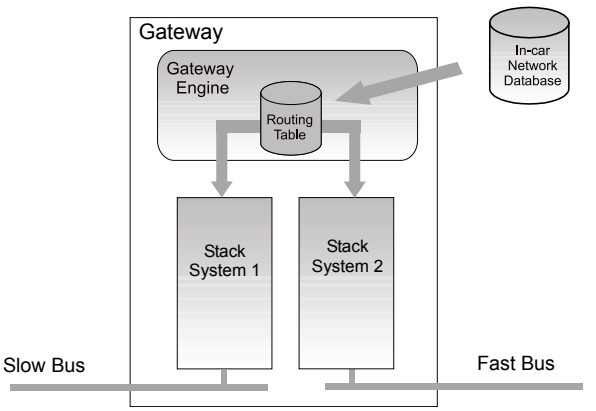


図17.2:ゲートウェイの基本構造

17.2 ゲートウェイのアーキテクチャ

図17.2に、ゲートウェイの基本的アーキテクチャを示します。通常、各バスシステムにはOSI (Open Systems Interconnection)参照モデルの7つの階層全てから成る完全なスタックを実装する必要があります。その上位にゲートウェイ エンジンを配置し、静的なルーティング テーブルに基づいて、1つのシステムの信号をもう1つのシステムへルーティングします。ルーティング テーブルは全ての信号をソースおよびターゲットと共にリストにしたもので、OEMの中央データベース(In-car Network Database)から生成されます。ゲートウェイは、以下のタスクを実行します。

- アドレス変換
- メッセージ変換
- メッセージの中間バッファリング
- パケット確認
- フロー制御とビットレート変換
- 信号のルーティング
- ネットワーク管理
- エラー管理
- 必要に応じたバスのロック/分離(例: 2つのバスを同時にフラッシュ/プログラムする場合)
- メッセージのセンダ/レシーバの認証(例: MOSTのECU xのみがCANのECU yに送信できるようにする)

この一覧からも分かるように、ゲートウェイの実装は通常複雑で、エラーが発生しがちです。ネットワークのビットレートが異なると遅延およびメッセージの損失が生じる事があります。

ここで、MOST‐ CANゲートウェイとMOST‐ Bluetoothゲートウェイを例に、前述の構造を詳しく説明します。

17.3 MOST‐ CANゲートウェイ

MOST‐ CANゲートウェイは、HMIまたは中央ゲートウェイに実装します。図17.3に、このゲートウェイの構造を示します。CANバスとMOSTインターフェイスでは、存在する抽象レベルが異なります。CANではフレーム内の特定の位置に信号が転送されますが、MOSTインターフェイスはファンクション ブロックで構成されており、MOST側の信号はファンクション ブロックのプロパティとして扱われます。MOSTシステムとその他の車載ネットワーク全体との間で交換が必要な重要な信号は、全てを1つのFBlock(例: Vehicle [FBVehicle 1.6])で表現するか、情報のソースドメインも反映する場合は複数のFBlock(例: Powertrain、BodyCompartment、HumanInterfaceController)で表現します。

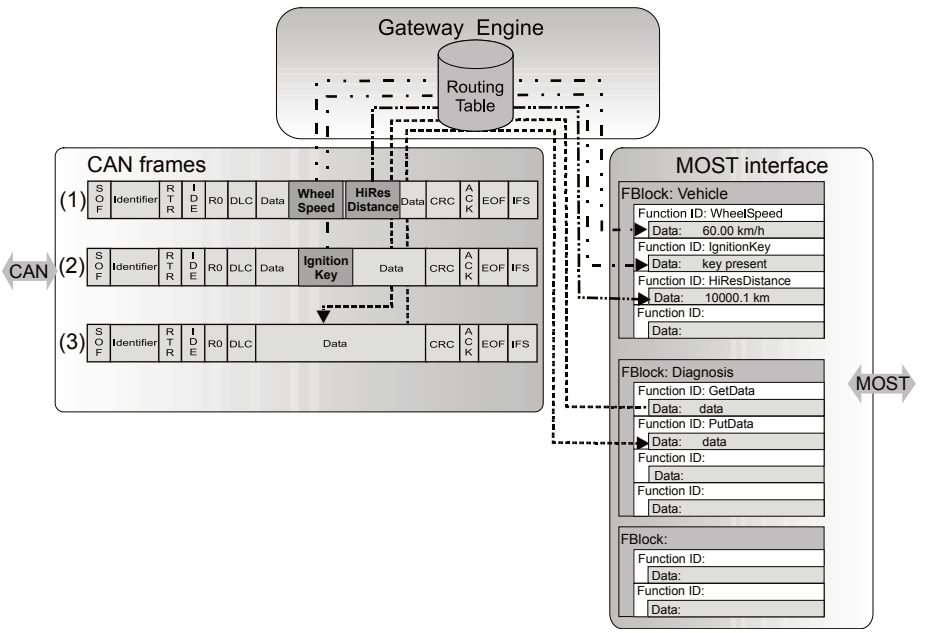


図17.3:MOST‐ CANゲートウェイの構造[出典: Por 2004]

図17.3の例では、CANフレーム1の信号WheelSpeedとHiResDistance(高分解能オドメータの読み取り値)と、CANフレーム2の信号IgnitionKey(イグニッション ロックのステータス)が、FBlock Vehicleの対応するプロパティにコピーされます。これらのプロパティを使うMOSTデバイスは、ゲートウェイのFBlock Vehicleの各プロパティに対するノーティフィケーション サービスに登録できます。

ゲートウェイで診断プロトコルを使う場合、ビットレートが異なるためフロー制御を実装する必要があります。

反対に、MOST側で発生した信号をゲートウェイ経由でCANに転送する場合、ゲートウェイはシャドウをインスタンス化します。これらのシャドウがノーティフィケーション サービスに登録し、関連するFBlockのプロパティのステータスが変化したら通知を受け取るようにします。

従ってプロトコル ゲートウェイは、その他のネイティブなMOST要素(リング内のFBlockまたはシャドウも含む)から見ると、MOST仕様に完全に準拠した方法で全てのMOST定型に従っています。

17.4 MOST‐Bluetoothゲートウェイ

ハンズフリー プロファイル[HFP 1.5]は、携帯電話をヘッドセットまたは車載の電話ユニットに接続するために定義されています。このプロファイルは、主に携帯電話に採用されるAG (Audio Gateway)ロールと、ヘッドセットや自動車のMOST Bluetoothゲートウェイに採用されるHF (Hands Free Unit)ロールで構成されます(図17.4参照)。

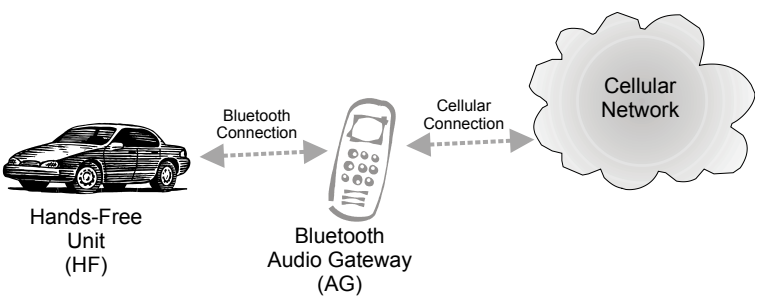


図17.4:ハンズフリー プロファイルのロール[出典: HFP 1.5]

Bluetoothゲートウェイでは、制御チャンネルに加えオーディオ チャンネルのルーティングも必要です。

17.5 制御チャンネルのルーティング

図17.5に、MOST‐ Bluetoothゲートウェイの制御チャンネルのスタック構造を示します。ここからも明らかなように、BluetoothとMOSTはアーキテクチャが似ています。MOSTスタックの上位にはFBlock Bluetoothがある一方、BluetoothはHands Free Controlプロファイルを使ってハンズフリー ユニットを制御します。これは基本的にCall Handlingを標準化したものです。通話を受け取り、オーディオ チャンネルのセットアップと終了を実行します。ゲートウェイ エンジンは、Bluetoothプロファイルのサービスを MOSTのファンクション ブロックBluetoothのプロパティとファンクションに接続します。

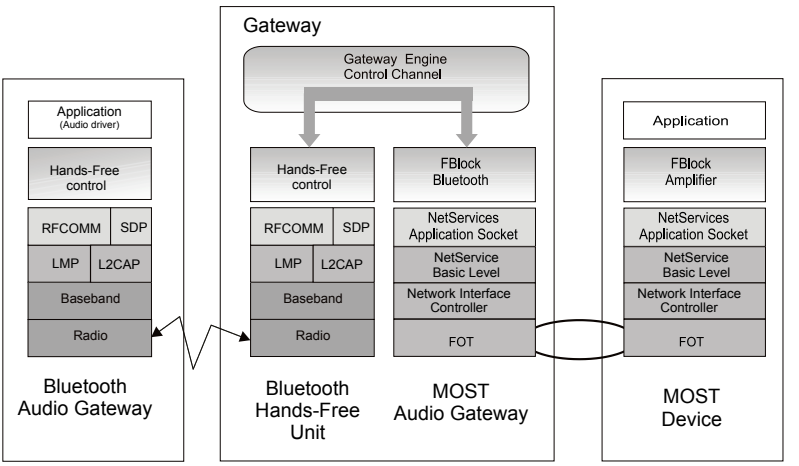


図17.5:制御チャンネルのアーキテクチャ

図17.6は、着信通話の場合のオーディオ接続を図解したものです。オーディオ接続はHFP (AG)で開始します。接続に成功すると、その情報がHFP (HF)に伝わり、ゲートウェイ エンジンに転送されます。この時点で、MOST側のオーディオ接続が開始します。まず、ゲートウェイ エンジンがFBlock ConnectionManager (CM)のBuildSyncConnectionメソッドを呼び出し、次にこのメソッドがSourceConnectメソッドとConnectメソッドを使ってBluetoothゲートウェイとアンプの間でMOST同期接続を行います。接続に成功すると、ConnectionMasterはBuildSyncConnection.ResultAckを使ってBluetoothゲートウェイに通知します。同時に、呼び出し音がMOSTシステムで生成されます。

電話帳の登録データは、ATコマンドまたはSyncML (Synchronization Markup Language)を使って携帯電話とMOSTシステム間で交換できます(SyncMLは実装されている場合のみ)。もう1つの方法として、PBAP (Phone Book Access Profile)が定義されています。この場合、携帯電話とゲートウェイの両方にPBAPが実装されている必要があります[PBAP]。

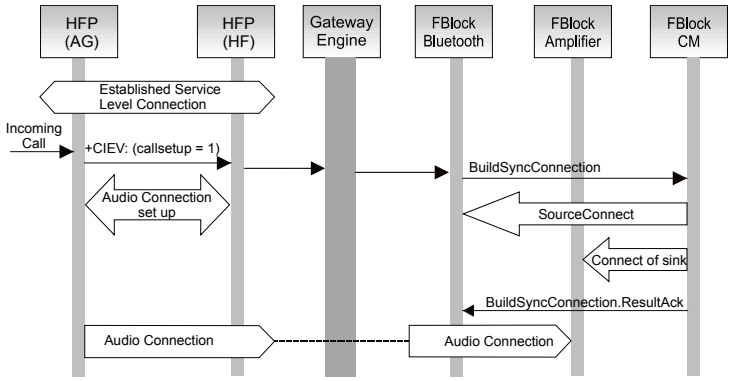


図17.6:同期接続の基本シーケンス

自動車では、SAP (SIM Access Profile) [SAP]もよく使います。これは、携帯電話と自動車の間でSIMデータを交換する目的にのみ使います。Bluetoothで登録されているならば、車載の電話ユニットはSAPを使って携帯電話からSIMデータを読み出し、これらのデータを使えます。これにより、SIMカードを2枚持つ必要がなくなります。

17.5.1 同期オーディオ接続のルーティング

制御チャンネルに加え、同期オーディオ接続もルーティングが必要です。この場合、周波数スペクトル(および必要に応じて振幅)を変換する必要があります。Bluetooth SCOチャンネルはISDNのオーディオ品質(サンプルレート8 kHz、すなわち信号の最大周波数は4 kHz)に変換されます。

Bluetoothは、パルス符号変調(PCM)またはCVSD (Continuous Variable Slope Delta)変調を使ってSCOチャンネルのオーディオ信号をソース符号化します。

PCMフォーマットには、対数特性A法則(A-law)またはμ法則(μ-law)が使えます。標本値の量子化ビット数が8ビットでサンプルレートが8 kHzのため、SCOチャンネルでは64 kbit/sの帯域幅が必要です。

CVSD変調は予測値と実際の標本値の差分を送信します。これはデルタ変調の一種で、予測信号のサンプリング周期を漸次増減する事により、アナログ入力信号に対する信号の適応性を高めるという方法です。変換の際は、1つ前の標本値との差分(正または負)のみを1ビットで符号化します。サンプルレートは64 kHzで、振幅は16ビットです。1ビットずつ送信するため、このチャンネルも64 kbit/sが必要です。

通常、BluetoothにはPCM変調を使います。サンプルレートの変換は、図17.7に示したような双方向の非同期サンプルレート コンバータを使って行います。BluetoothチャンネルからMOSTシステムへオーディオデータを送信する場合、このコンバータが補完法でサンプルレートを44.1 kHzに引き上げます。反対方向の場合、サンプルレートをコンバータで8 kHzに引き下げます。このコンバータはローパスフィルタを内蔵しています。

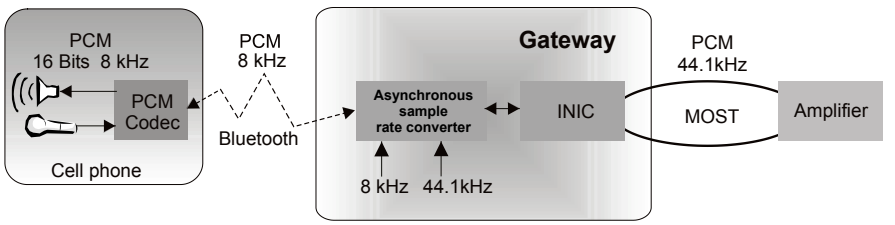


図17.7:オーディオ ゲートウェイのアーキテクチャ

ストリーミング チャンネルと制御チャンネルの制御は、ゲートウェイが調整する必要があります。例えばアンプがソースを電話に切り換える前に、同期接続に有効な信号を印加しておく必要があります。そうしないとホイッスリング等のノイズが発生する事があります。

18 MOSTのアプリケーション

この章では、MOSTベースのアプリケーションの概要をいくつか紹介します。その前に、まずインフォテインメント システムのこれまでの歴史と今後の見通しについて説明します。

18.1 概要

1990年代なかば、車載エンターテインメント市場は**技術および運用上の新しい要件**に直面していました。

1. エンターテインメント コンポーネントの数が増え、車内であらゆる種類の情報を供給および消費するようになった結果、これらコンポーネント間の相互接続と制御は複雑化していきました。
2. 自動車メーカーは、多数の異なるハードウェア プラットフォームを搭載する事によって生じる電磁干渉(EMI)およびワイヤハーネスの重量とコストの問題に取り組みました。各コンポーネントの診断はシステムレベルでは対処できず、コンポーネント単位でしか行えませんでした。
3. エンドユーザ(運転者または同乗者)は全てのコンポーネントを別々のヒューマン マシン インターフェイスで操作したいというニーズは現在も根強く続いています。

システムを構築する際は、レイテンシを最小限に抑えて情報をリアルタイムで利用できるような方法で、各コンポーネント(情報のシンクまたはソース)が他のコンポーネントと通信できるようにする必要があります。さらに、これらの情報に補正処理やポストプロセッシング(同期協調処理)を適用した後、新しい情報として再利用される事もあります。「システム コントローラ」とも呼ばれるインフォテインメント ヘッドユニットは、全てのコンポーネントの全てのイベントを統合的に管理し、ユーザがシステムの複雑さを意識する事なく、システムから生成される結果のみを扱えばよいようにする必要があります。

これらの要件を満たすために開発されたのがMOSTシステムです。MOSTでは、メディア信号もリアルタイムに転送できるような形で情報(データ)と制御コマンドを転送および「ブロードキャスト」できます。

MOSTは最新インフォテインメント システムのバックボーンとして急速に普及し、現在では自動車業界のインフォテインメント ネットワーキングの事実上の標準です。しかし、MOSTはデータ転送だけのものではありません。MOSTが提供する標準API群により、システム コントローラ内の全てのコンポーネント (プロキシ)の論理表現を使ってソフトウェアを設計できる事も重要です。

さらに、MOSTはインフォテインメント システム設計における困難な課題を解決し続けています。ユニークで安定したシステム アーキテクチャを使って各種レベルの機能と性能を実現し、幅広い市場の要件に柔軟かつスケーラブルに対応しています。

第1世代のMOSTベース インフォテインメント システムは、限られた(そして同時に高価な)コンピューティング性能のインフォテインメント機能を数多く組み合わせた、分散型の性格が強いアーキテクチャが特徴です。この結果、専用の機能をもった専用のコンポーネントを集めた、非常にモジュール式の性格が強いアプローチでした。システム コンポーネントはいくつものサプライヤによって設計、製造されており、ハードウェアとソフトウェアの設計と製造に関するスキルはサプライヤごとに異なっていました。そのため、当初はサプライヤ間でコンポーネントの品質と性能に大きな差が見られました。その後、サプライヤ全体の習熟度は向上しました。図18.1に、このようなシステムを示します。

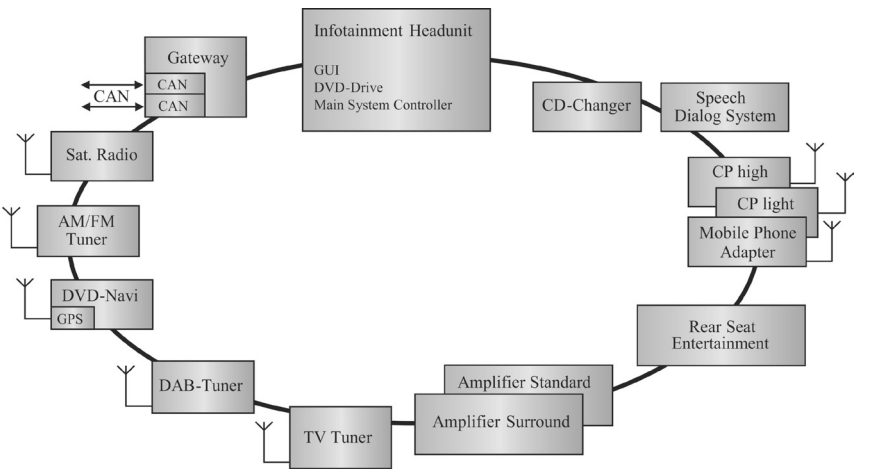


図18.1:第1世代のMOSTベース インフォテインメント システム

第2世代のインフォテインメント システムは、外部コンポーネントをヘッドユニットに物理的に統合する傾向が強まったのが特徴です。コンポーネントの数は、それまで12～15個が一般的でしたが6～8個へと減少しました。代表的な第2世代システムのコスト、重量、消費電力は、第1世代と比べて最大30%低減しました。このような物理的統合の例として、「チューナアンプ」モジュールや、ナビゲーション コンピューティング モジュールのヘッドユニットへの統合があります(図18.2参照)。第1世代のMOSTシステムはBMW 7シリーズ(E65)やメルセデス Eクラス(W211)に採用されましたが、第2世代のシステムはBMW 5シリーズ(E60)やメルセデス Cクラス(W203)に採用されました。

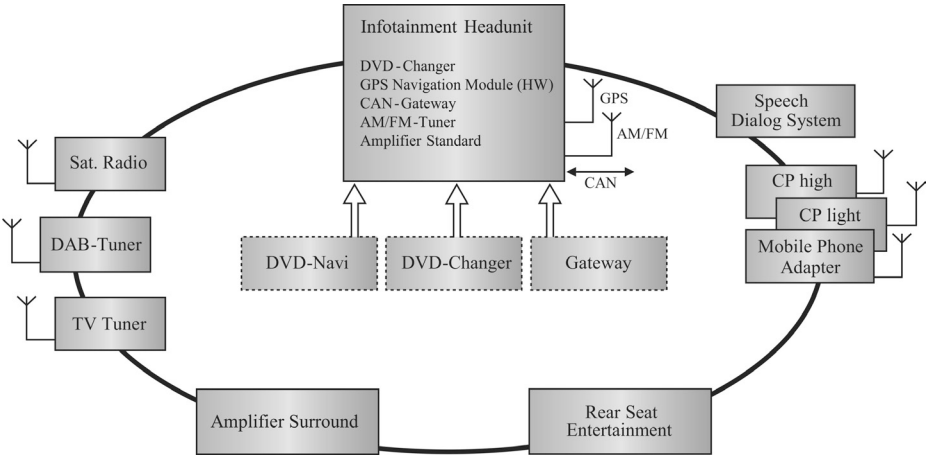


図18.2:第2世代のMOSTベース インフォテインメント システム

第3世代のインフォテインメント システムは、2005年以降に市場に投入されたシステムの基盤となるものです。この世代は、ハードウェア コンポーネントからソフトウェア モジュールへの置き換えが進んだ事が特徴です(図18.3参照)。これにより、コスト、重量、消費電力がさらに低減されました(-30%)。多くのシステム機能がヘッドユニットに統合され、外部システム コンポーネントの数がさらに減少しました。エントリレベルのインフォテインメント システムでは、外部システム コンポーネントが全くない場合もあります。この場合でも、内蔵のMOSTインターフェイスを使って外部オーディオアンプ、メディア チェンジャ、衛星ラジオ等の国別チューナ オプションといった先進のオプションを接続できます。このようにして、ミッドレベルおよびハイエンド システムと同じデバイスを使う事ができます。

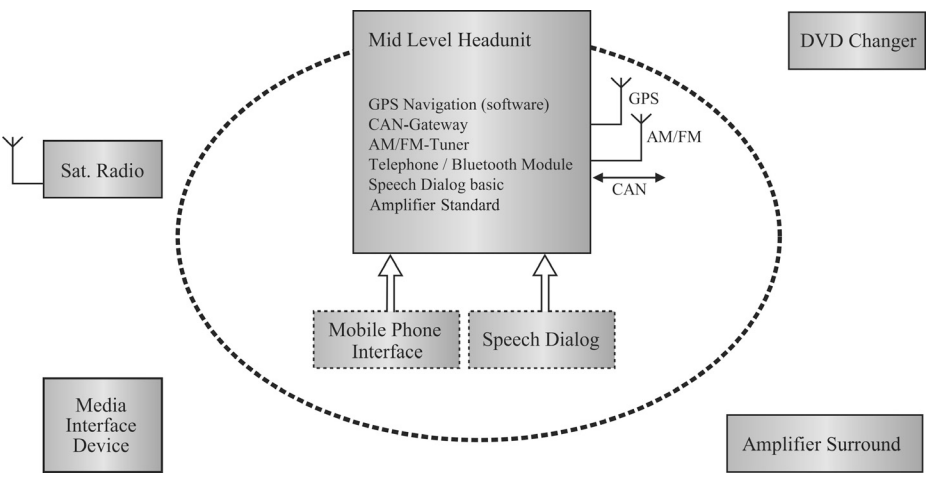


図18.3:第3世代のミッドレベル インフォテインメント システム

第4世代のインフォテインメント システムは、2013年以降に市場に登場します。MOST150の新機能、特に帯域幅の拡大により、インフォテインメントに関する全てのビデオデータをMOSTネットワークで転送できます。ビデオコンテンツは、その他全てのメディアデータ (デジタル放送または無線LAN等の無線媒体、DVDまたはBlu-rayディスク、ポータブル メディアのメモリ)同様、圧縮フォーマットで転送されます。

代表的なシステムは、インフォテインメント ヘッドユニットといくつかの外部コンポーネント、すなわちオーディオアンプ、デジタルチューナ(国/地域別)、メディア インターフェイス、リアシート エンターテインメント ユニット、計器パネルクラスタ(IPC)等で構成されます。LCDパネルが主流になった現在、IPCへのビデオ機能の統合が進んでいます。また、現在は専用のビデオケーブルでヘッドユニットに接続されているリアビューカメラをMOSTシステムに統合する事もできます(図18.4参照)。

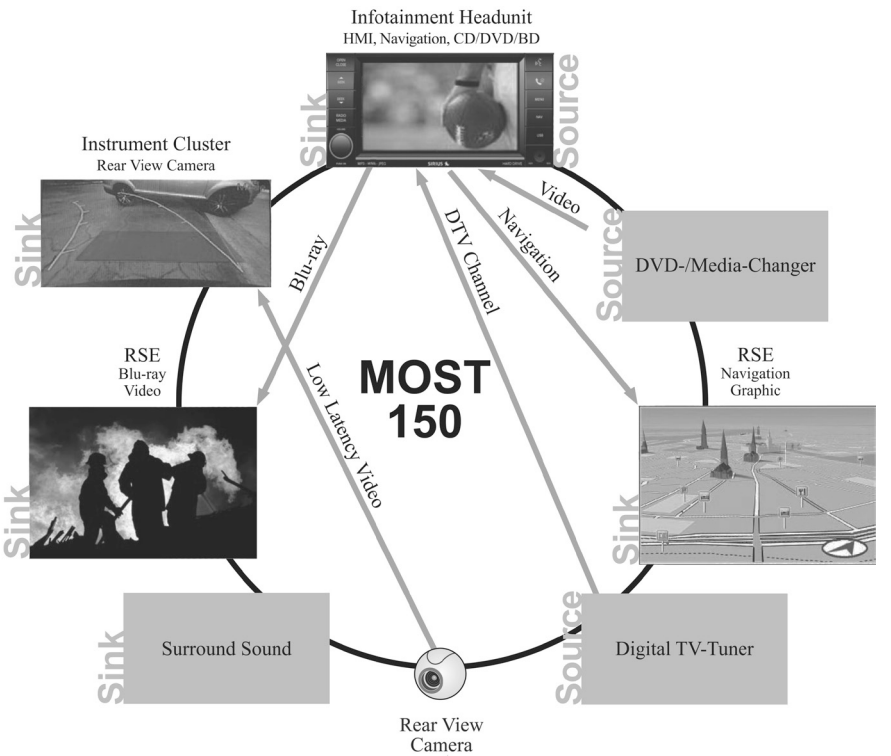


図18.4:第4世代のMOSTベース インフォテインメント システム

以下の3つのセクションでは、MOSTシステムのコンポーネントの例として、インフォテインメント ヘッドユニット、メディア インターフェイス、サウンドシステムの3つを詳しく見ていきます。

18.2 インフォテインメント ヘッドユニット

2010年現在、代表的なインフォテインメント ヘッドユニットは、車載の過酷環境向けに最適化されたスケーラブルなコンピューティング プラットフォームであり、32ビットベースの400 MIPSからデュアルプロセッサ アーキテクチャによる2 GIPSにまで至る演算性能を達成しています。また、オーディオ/チューナおよびグラフィック処理用の専用ハードウェア(DSP)も使われています。オペレーティングシステムは、Windows Automotive、QNX、組み込みLinux、μITRON等、先進の32ビットOSを使っています。

従って、ハードウェアのスケーラビリティに対応するソフトウェア スイートが必要です。起動時間の短縮、リアルタイム処理、ストリーミング データ処理、モジュールのサーバ機能、マルチユーザ アプリケーション用のマルチ グラフィック ドライバ、マルチモードHMI動作等の革新的な機能は、先進のアプリケーション ソフトウェア モジュール、先進のフレームワーク、リアルタイム オペレーティング システム(RTOS)によって実現しています。エントリレベルからハイエンド システムまで、MOSTの設計理念に沿ったソフトウェア アーキテクチャを一貫して実装する事がきわめて有用です。

エンドユーザが素早くシステム機能を使えるようにするため、各システム コンポーネントの起動時間を短縮する事は非常に重要です。この高速化のために、I/Oプロセッサとも呼ばれる16ビット マイクロコントローラを追加し、別のリーンなOSを実行するケースが多く見られます。ヘッドユニットにゲートウェイ機能を内蔵する場合、通常はさらにこのI/Oプロセッサにゲートウェイ機能(MOST-CAN)を実装します。

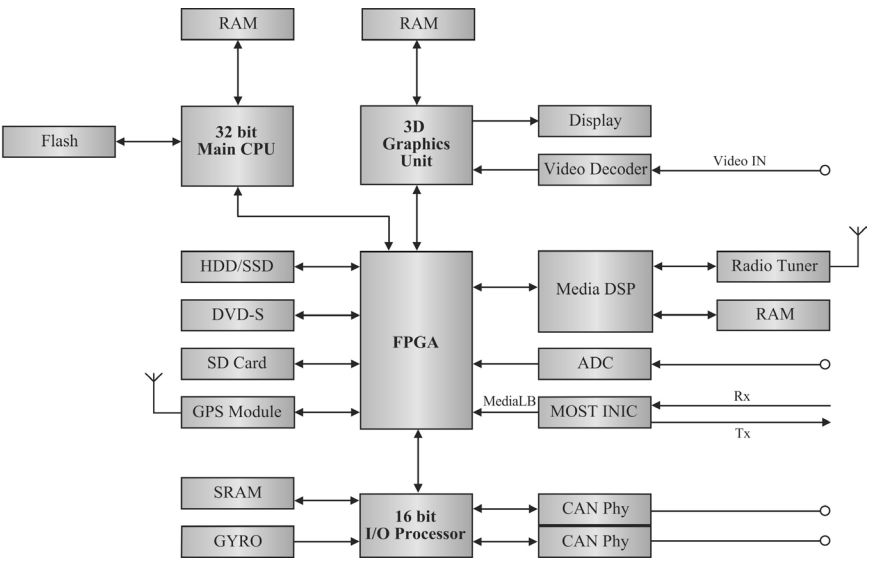


図18.5:FPGAアーキテクチャに基づくインフォテインメント ヘッドユニット

MOSTは「信号ベース」の情報交換を可能にするための包括的なデータタイプを定義すると共に、拡張およびカスタマイズ可能な定義済みAPI群も備えているため、(a)MOST物理ネットワーク上での通信、(b)メインCPUとI/Oプロセッサ間の通信、(c)32ビットエンジン内部の「オンチップ」通信を一元的な通信アーキテクチャでサポートするという非常に画期的なアプローチが可能です。第1および第2世代のMOSTシステムでも、MOSTベースのヘッドユニットの大多数がこのようなソフトウェア アーキテクチャを採用していましたが、そのほとんどは独自規格に基づいたソリューションでした。

現在は、このようなソフトウェア アーキテクチャをサポートした既成のソリューションを提供するソフトウェア フレームワークが広く使われ、また市販されています(例: K2L社(www.K2L.de))。

市場をリードするFPGA (Field-Programmable Gate Array)ベンダ各社は、メインボードの拡張およびハードウェアのインターフェイスを柔軟かつ最適化した方法でサポートする次世代デバイスの開発に注力しています。このコンセプトは、システム ソフトウェアを使ってFPGAコードによる再構成を行う事で、多くのハードウェア レベルをスケーラブルに提供できるアプローチに利点があります(図18.5参照)。

図18.6に、これとは異なるアプローチを示します。こちらは、よりPCに近いアーキテクチャです。このアプローチでは、PCと同様にFPGAを専用のサウスブリッジ チップ(I/Oハブ)で置き換えています。ただしPCで使うサウスブリッジ チップはオーディオ/ビデオポートがないため使えません。この部分は車載アプリケーション(組み込みマルチメディア ユニット)とPCアプリケーションで大きく異なるため、車載専用のI/Oハブを開発する必要があります。

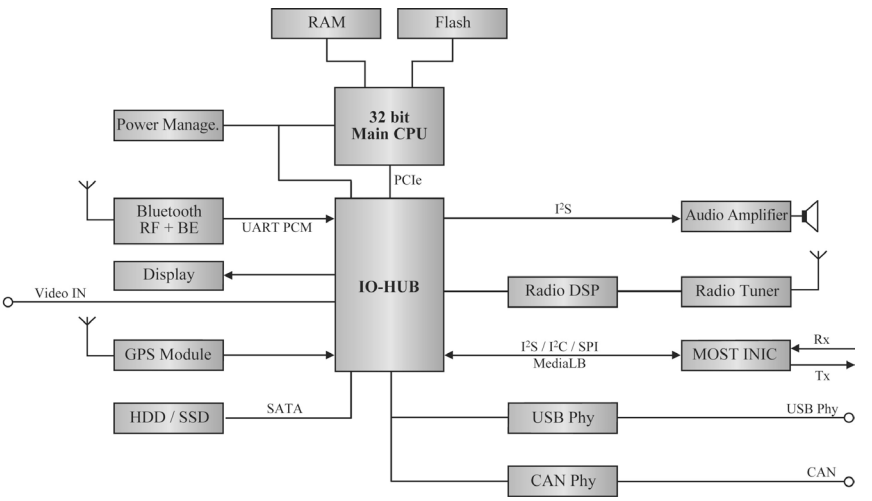


図18.6:PCアーキテクチャに基づくインフォテインメント ヘッドユニット

マルチメディア(A/V処理)機能をメインプロセッサに完全に統合できるかどうか、そしてそれがいつになるかは、まだ見えていません。いずれにせよ、このような統合にはRTOSとアプリケーション フレームワークで構成される包括的なソフトウェア スイートが必要です。このようなフレームワークには、ネットワーク サブシステム(MOST)、グラフィック ライブラリとドライバ、ナビゲーション、音声制御と音声補正、多規格対応メディアデコーダ、ディスクリートかつ強力なサウンド処理、デジタルラジオ処理、汎用システム診断、ネットワーク制御を含むシステム起動およびシャットダウン等のソフトウェア スイートが必要です。

2010年現在、このようなエコシステムの構想がいくつか存在しています。しかしこの移行には、性能、スケーラビリティ、設計/ソフトウェア統合の面から非常に大きな困難が伴います。その主な理由として、スマートフォンやメディアプレーヤ等、量産規模のはるかに大きいコンシューマ向けモバイル/携帯型機器の機能セットが急速な発展を続けているという点があります。

今後のインフォテインメント ヘッドユニットの設計により深く関係すると思われるもう1つのトレンドとして、運転支援機能に対する新しい要件の出現があります。インフォテインメント システムが主要なユーザ インターフェイスを提供する以上、伝統的なインフォテインメント ドメインと今後の運転支援クラスタの間には明らかに重複が生じます。この事は、インフォテインメント ヘッドユニットの設計とアーキテクチャにも影響を与えます。

この場合、安全に関係する機能と関係しない機能の両方を考慮する必要があるため、よりモジュール式のアプローチを採用し、ISO 26262規格のASIL (Automotive Safety Integrity Levels)に応じてヘッドユニットの各部位の開発を別々に進めた方がよいと考えられます。

いずれにせよ、MOSTのように複数レベルの通信方法を提供できる強力な通信バックボーンの重要性がこれまで以上に高まっているのは明らかです。

18.3 メディア インターフェイス デバイス

メディア機器はますます多様化が進み、これら機器の設計は急速に変化しています。iPodに代表されるMP3プレーヤは製品によって使っているAPIが異なり、コピー保護/コンテンツ管理の方式を使っているものもあり、自動車側に頻繁な変更が要求されます。一部メーカーのメディア機器では、機器を認証して通信を実行するために専用ハードウェア(IC)が必要な場合もあります。

これらの変更をマルチメディア ヘッドユニットに実装しようとすると、比較的大規模な開発が発生し、特にテストと再認定に大きな負担がかかります。また、ヘッドユニットとマルチメディア システムの開発段階ではメディア機器が入手できない場合や、自動車の生産段階になって新しい機器が登場する場合さえあります。OEMにとって、このような自動車のライフサイクルとコンシューマ製品のライフサイクルの差は悪夢です。

従って、サイクルの短いコンシューマ製品とサイクルの比較的長い車載製品を分離するのは理にかなっています。これを実現するためにしばしば用いるのがいわゆる「メディア インターフェイス デバイス」であり、「ユニバーサル通信インターフェイス」とも呼びます。通常、これらのデバイスには各種カードリーダと複数のUSBインターフェイスを備え、独自方式の機器との通信用に認証用デバイスを備える場合もあります。図18.7に、メディア インターフェイス デバイスのブロック図を示します。

前のセクションで説明したように、現在のスマートフォンはナビゲーション、音声制御、複数の規格に対応したメディアサーバ、GSM/UMTSおよびWLANを含む無線通信機能等、多機能化が進んでいます。これらの機器を自動車に接続し、機器の内蔵機能を使いたいというニーズは着実に高まっています。
MOSTベースのメディア インターフェイス デバイスを使うと、自動車メーカーは十分に定義され入念に設計とデバッグが行われたインフォテインメント インフラストラクチャを維持しつつ、コンシューマ機器市場の急激な変化に柔軟に対応する事もできます。

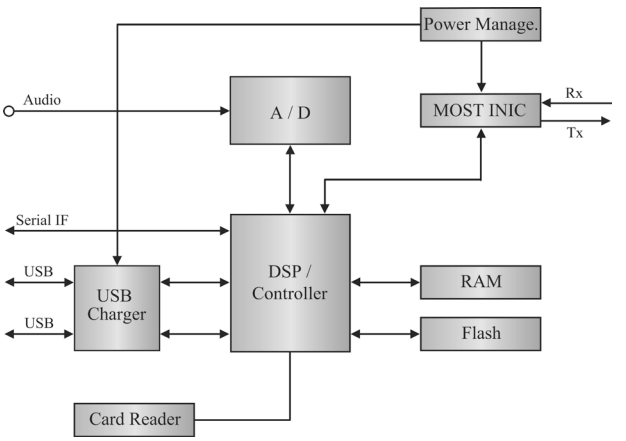


図18.7:MOSTベースのメディア インターフェイス デバイス

18.4 サウンドシステム

多くの場合、自動車メーカーは各車両タイプに複数のオプション機器を用意しています。ベーシック システムからプレミアム サウンドシステム (ブランド品が多い) まで複数のトリムレベルを揃えています。これらは機能だけでなく出力と信号処理機能にも大きな違いがありますが、こうした違いにかかわらず全てのトリムレベルの機器をネットワーク側から一貫した方法で制御できる事がきわめて重要なのは明らかです。MOSTでは統一された制御APIが提供されており、システム アーキテクチャと統合プロセスを明快かつ単純化するのに役立っています。

サウンドシステムを車両のネットワークに接続する必要がある例として、車速連動音量調節があります。これは、車速に応じて大きくなる騒音を補償する機能です。

もう1つの例は、パーク ディスタンス コントロールやシートベルト非装着時に生成する警告音(チャイム)です。運転支援機能が増えるにつれ運転者が受け取る視覚情報が増え続けているため、今後は音響情報の重要性が増すと予測されています。これらの信号をMOSTまたはCANバスシステム経由でヘッドユニットに送信する際は、リアルタイム性がきわめて重要です。

プレミアム システム

プレミアム システムでは、多くの信号がヘッドユニットまたはその他のソースからMOSTを経由し、複数のアンプ出力を持つ専用のDSP (Digital Signal Processor) アンプへ送信されます。このアンプはデジタルサウンド プロセッサとも呼ばれます(図18.8参照)。

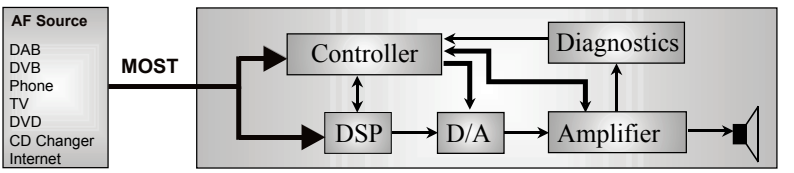


図18.8:DSPサウンドアンプのブロック図

このDSPは車速に応じて広帯域で音量を上げるだけでなく、低周波数帯域の音量をより大きく上げて騒音の影響を補償します。車室内の騒音はほとんどが低周波数帯域であり、周波数が10倍になると約20 dB減少します。このため、プレミアム システムのように全周波数帯域の信号レベルを一様に上げると、低周波帯域のみ騒音に埋もれるため、高域が強調された印象となります。騒音の大きさは車速依存のためスピードメータから推測できます。または、マイクで直接計測できます。この結果、高度な閉ループ制御が可能です。マイクは1つまたは複数を使う事ができ、これらのマイク信号も通常はMOSTバスで転送します。

18.5 オーディオおよびビデオの転送フォーマット

18.5.1 オーディオ ストリーミング

車載マルチメディア システムは、各種ストレージ メディアおよび符号化フォーマットへの対応が求められます。また、デジタル オーディオおよびビデオ放送の各種転送フォーマットも認識する必要があります。機器の種類によっては高い転送レートが必要であり、これをMOSTシステムで処理する必要があります。

各種応用分野から派生した多くのバスシステムとは対照的に、MOSTはその名称(Media Oriented Systems Tranport)からも分かるようにシステム全体でマルチメディア データを転送する事を目的に設計されています。

ホームシステムとは異なり、自動車では複数のオーディオおよびビデオシンク、アンプ、ヘッドホン、複数のディスプレイが同時に動作するマルチソース/マルチシンク環境が必要です。

オーディオ分野には非常に多くの種類の転送フォーマットがあります。自動車では、ステレオ再生に加えてプロフェッショナル システムで採用されているような各種サラウンド フォーマットも使われます。以下に、概要を示します。

ステレオ

非圧縮信号のデータレートは、サンプリング レート44.1 kHz、分解能16ビットの場合で1.4 Mbit/sです。最も広く使われている圧縮レートは、MP3の128 kbit/sです。

アナログDolbyサラウンドPro Logic

このシステムは、フロント右、フロント中央(音声用のセンタースピーカ)、フロント左のフロント3チャンネルと、サラウンド1チャンネル (リア左右)で構成されます。サラウンド チャンネルは2つのリアスピーカにフロント チャンネルよりも遅れて出力され、帯域は100～7,000 Hzに制限されます。センタースピーカは音声を中央に定位させます。このため、車内のどの位置で聴いても明確な音像が得られます。

アナログDolbyサラウンドPro Logic II

DolbyサラウンドPro Logic IIは、Pro Logicをさらに強化し、帯域制限のない2つの独立したサラウンド チャンネルとサブウーファ チャンネルを追加する事により、5.1システムを実現します。

Dolbyデジタル(AC3)

AC3は、帯域制限のない5つの独立したチャンネルと、120 Hzに制限された独立サブウーファ チャンネル(LFE: Low Frequency Effect)で構成されます。サンプリング周波数48 kHzで分解能16ビットの場合、データを10:1に非可逆圧縮して転送レートは384～448 kbit/sとなります。2チャンネル ステレオの場合、データレートは192 kbit/sに低下します。MPEG-2がビデオ分野のDVD標準とすれば、AC3はオーディオ分野のDVD標準です。これ以外にDVDでオーディオ符号化フォーマットとして使えるのはMPEG-2のみです。

MPEG-2

AC3同様、MPEG-2もDVDのオーディオ信号符号化規格として使われます。6チャンネル全体でデータレートは約400 kbit/sです。MPEG-2の半分のデータレートで同等の品質が得られるように改良されたものがMPEG-2 AAC (Advanced Audio Coding)で、最大48オーディオ チャンネル、16サブウーファ チャンネル、16コメンタリ チャンネルを符号化できます。MPEG-2はデジタルTV (DVD衛星、DVDケーブル、DVD地上波)の標準として使われます。日本では、2000年初めから全てのデジタル放送システムのサウンド フォーマットとしてMPEG-2が独占的に使われています。

DTS (Digital Theatre System)

DTSも5.1システムです。AC3よりも音質が良く、ダイナミック レンジも向上しています。DTSはデータ圧縮率が非常に低く、1.536 Mbit/sのデータレートでサラウンド サウンドを符号化します。

Logic 7

Lexicon社が開発した7.1サラウンド システムのLogic 7は、8つの独立したチャンネルをデジタル制御するか、ステレオ信号からマトリクス デコードに基づいて対応する制御信号を生成します。このようにして擬似的にサラウンド感を創出し、非常に良好なアンビエント サウンドを再生できます。5.1ソース信号は7.1に拡張されます。Logic 7は車載向けプロフェッショナル システムで多く使われます。

18.5.2 ビデオ ストリーミング

ビデオ分野では、市場別に様々な企業が開発した様々なフォーマットが使われています。図18.9に、一般的なグラフィック/ビデオ規格の一覧を示します。

Standard	Resolution	Frame / Hz	MPixel/s	RGB 24 (Mb/s)	YUV4:2:0 (Mb/s)
QVGA	320 x 240	60p	4.6	110.6	55.3
NTSC	640 x 480	60i	9.2	221.2	110.6
PAL	720 x 576	50i	10.4	249.8	124.4
VGA	640 x 480	60p	18.4	442.4	221.2
XGA	1024 x 768	60p	47.2	1132.5	566.2
HD Ready	1280 x 720	30p	27.6	663.6	331.8
HD Ready	1280 x 720	60p	55.3	1327.1	663.6
Full HD	1920 x 1080	24p	49.8	1194.4	597.2
Full HD	1920 x 1080	60i	62.2	1493.0	746.5
Full HD	1920 x 1080	60p	124.4	2986.0	1493.0
Full HD 3D	1920 x 2205	24p	101.6	2438.6	1219.3

Graphic standard | Video standard

図18.9:一般的なグラフィック/ビデオ規格の一覧

ビデオCD、DVD、BD等の回転メディア、DVBまたはDMBチューナからのデジタルビデオ ストリーム、あらゆる種類の携帯型機器では、圧縮ビデオが使われます。

通常、コンシューマ ビデオ アプリケーションで扱うデジタルデータは圧縮フォーマットを採用しています。その理由は明らかです。そうしないと、データ伝送チャンネル(無線またはインターネット)またはストレージ容量(ディスクまたはメモリ容量)にコストがかかり過ぎるためです。

記録済みコンテンツの場合、非圧縮ビデオを転送する必要はありません。圧縮ビデオの転送原理には、2つのシナリオが考えられます。どちらのシナリオにも長所と短所があり、対応する座席数、機能に関連するコスト、総コスト、柔軟性、拡張性、帯域幅、その他のハードウェア要件等、個々の要件を考慮する必要があります。

シナリオ1:ソース側でトランスコードする方法

ストリームのナビゲーションとデコードをソース側で直接実行します。ビデオとオーディオはそれぞれ別々にMOSTネットワークに送出します。ビデオは、MOSTで定義された共通のフォーマット(例: H.264、MPEG2/4)にトランスコード(再エンコード)されます。オーディオはPCMにトランスコードされます。オーディオとビデオが分離しているため、ビデオシンクとオーディオシンクを互いに独立させる事ができます(例: ビデオを複数のディスプレイに送信し、オーディオをアンプに送信)。システムはソースからシンクへのプッシュモードを使います。MOSTの同期転送は遅延時間が確定的かつ一定で最小限に抑えられているため、分離したオーディオ ストリームとビデオストリームを最適な形で転送できます。

DVDの場合も(場合によってはDivXも)、サブピクチャ情報の扱いを考慮する必要があります。この情報は、トランスコードしたフレームに直接記録するか、ビデオとは別にシンクに送信できます。

シナリオ2:プログラム ストリームを転送し、シンク側でデコードする方法

ソースから取り出したデータをそのままのフォーマット (オーディオ+ビデオ、トランスポート ストリームまたはプログラム ストリーム)で、MOSTネットワーク経由でディスプレイ装置等のシンクへ転送します。デコードとナビゲーション/制御はシンク側で実行します。

オーディオは、再びMOSTネットワークを経由して分離型のオーディオシンク(例: メインアンプ、分離型のヘッドフォン アンプ)に送信できます。システムは、ソースがメモリ媒体(例: HDD、メディアドライブ)の場合はプルモードでデータパケットを要求し、デジタルチューナ等がソースの場合はプッシュモードを使います。ビデオシンクは、デコードしたオーディオおよびビデオデータをプッシュモードを使ってコピーモードの別のシンクへ送信できます(同じコンテンツを視聴する場合)。

18.6 MOSTを使った運転支援アプリケーション

運転支援システムは、急速に普及しつつあります。最新の調査では、2005～2013年の間に年間900万～6200万台の成長が予測されています。今後の自動車では、運転支援機能が従来のインフォテインメント システムの機能を補完、拡張するでしょう。ナビゲーション システム、道路交通情報、機能警告等の情報機能に加え、カメラシステム、車間制御、車線逸脱警報等の運転支援機能を搭載した自動車が急速に増えていくと考えられます。この応用分野での通信障害は、深刻な物損事故や人身事故につながる可能性があるため、深刻な問題です。そのために必要な改善策の研究が進められており、最近の調査では絶対的安全性の要求される応用分野もMOSTでサポートできる可能性が示されています。

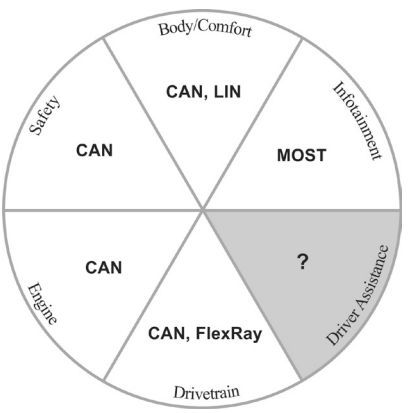


図18.10:車両内の電気/電子ドメインの発展

各種車載電気/電子システムへのインターフェイスを備えた運転支援システムは、自動車の必須機能となりつつあります。アプリケーション自体が複雑で、自動車の各部位との情報交換が必要であるため、効率的なシステムとするには適切なネットワーク インフラストラクチャの構築する事が非常に重要です。このため、自動車の電気/電子ネットワーク アーキテクチャを構成する5つの伝統的なドメインに加え、運転支援システムに特化した新しいドメインが導入されようとしています(図18.10参照)。

既存のMOSTはインフォテインメント用に開発および最適化されてきたため、このままでは運転支援アプリケーションのごく一部にしか使えません。このため、IEC 61508等の現行の規格はもちろん、自動車業界の新しい安全規格ISO 26262の要件を満たすためにどのような対策が必要かという点について、MOST Cooperationの依頼による調査が進められています。

物理層に関しては、リング アーキテクチャだけを見ても運転支援アプリケーションに対する制約が非常に多くあります。このため、リング型に代わりスター型またはデイジーチェーン型等のトポロジの開発が進められています。配線に関しては、光ファイバは必ずしも全てのユースケースに適しておらず、MOST50で導入された十分に実績のあるUTP物理層も帯域幅の拡張性に限界がある他、ハイブリッドまたは電気自動車で予想されるEMCの要件に対して十分なロバスト性がないという問題もあります。このため、MOST仕様では現在、低コストでロバスト性の高い代替方式(例: 同軸ケーブルを使った物理層)の定義が進められています。

プロトコル面でのソリューションとしては、安全関連アプリケーションの基盤となるセーフティ層を追加したアーキテクチャが提案されています。セーフティ層は、安全関連アプリケーションのデータをMOSTネットワーク上で転送できるようにするセーフティコードと高信頼性のサービスを使う事により、MOSTネットワーク上で安全な通信を実現します。

今後、第5世代のMOSTシステムではインフォテインメント システムと運転支援システムを別々のトポロジと物理層でサポートする実装が1つの可能性として考えられます。

補遺

A 記述言語Message Sequence Chart (MSC)

車載分野では、分散型のテレマティクスおよびインフォテインメント システムの動的挙動を記述する言語としてMSCが広く使われます。Message Sequence Chartの構文と意味は国際電気通信連合(ITU)のITU-T Z.120 [ITUZ120]とその正誤表1で標準化されています。これまで、コンポーネント同士の連携をグラフィカルに記述するための記法が数多く提案されてきました。Message Sequence Chartは、通信システム内のインスタンス同士の非同期メッセージ交換を直感的に記述する方法として実績があります。

この半形式的記法は、電気通信分野から発祥したもので、特に分散型電気通信システムの要件工学、仕様定義、設計の分野で使われていました。当時、MSCは常にSDL (Specification Description Language)モデルの一部と考えられてきました。しかしその後MSCはSDLから分離し、ITUによる標準化が進められてきました。

MSCは、システム コンポーネント同士、およびシステム コンポーネントと環境の通信動作をメッセージ交換によって定義、記述するための技術言語を開発する事を目的としています。MSCは通信の挙動を直感的かつ透過的な方法で記述するため、習得、使用、解釈が容易です。MSCと他の言語と組み合わせる事で、システムの仕様定義からシステム設計、シミュレーション、テスト、文書作成までをサポートできます。

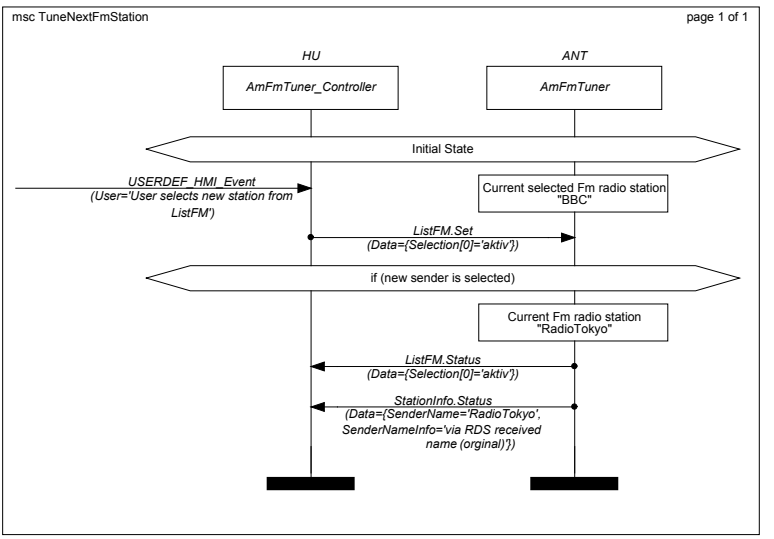
MSCは特定の通信プロトコルに依存しません。バス通信に加え、デバイス内部のソフトウェア コンポーネント間の通信モデルもシステム動作に関係する場合はMSCで定義できます。

MSC規格はモジュール型構造をサポートしています。繰り返し通信手続きは1回記述するだけでよく、必要に応じて何度でもリファレンスとして使えます。状態遷移図とは異なり、MSCは完全性が必須でなく、要件に応じて項目化が可能です。難しい項目および不明確な項目は、より詳細に定義されます。

しかし、MOSTシーケンスを記述するには複雑なデータ型に対する拡張が必要である事が明らかになりました。このために導入されたのが、いわゆる名前付きパラメータです。詳細は、『MOST MSC Cookbook Specification』の「5　補遺A: Extension to MSC2000 (Named Parameters)」で説明しています。

MSCは、ベーシックMSCとHMSC(ハイレベルMSC)の2つのカテゴリを定義しています。図A.1に、ベーシックMSCの例としてTuneNextFmStationを示します。これは、現在選局中のFMラジオ局(BBC)からリストに保存されている次のFMラジオ局(RadioTokyo)へ切り換えるシーケンスを記述したものです。ベーシックMSCはこのシーケンスを抽象化し、メイン コンポーネントのAmFmTuner_ControllerとAmFmTunerの間の情報交換のみを示します。シーケンスの開始時にシステムは初

期状態にあり、ラジオ局BBCが選局されています。HMIは、USERDEF_HMI_Event メッセージにパラメータUser selects new Station in ListFMを指定してAmFmTuner_Controllerへ送信します。AmFmTuner_ControllerはListFM.Setメッセージに適切なパラメータを指定してAmFmTunerへ送信し、AmFmTunerがアクションを実行します。新しいラジオ局(RadioTokyo)が選局されると、AmFmTunerは新しいリストステータス(ListFM.Status)とRDSから受信したラジオ局名(StationInfo.Status)をAmFmTuner_Controllerへ送信します。



図A.1:ベーシックMSCの例

インスタンス

インスタンスは、メッセージと合わせてベーシックMSCの言語を構成する最も重要な要素です。インスタンスは、インスタンス同士またはシステム環境との間でメッセージを交換するコンポーネントです。インスタンスの例として、CANノードやMOST FBlockがあります。インスタンスは、下に垂直線が付加されたインスタンスヘッドとして表記します。インスタンス ヘッドの中にインスタンス名を記述します。MOSTドメインで認められるインスタンス名としては、NetBlock、AmFmTuner、PowerMaster等があります。また、インスタンスのタイプをインスタンス ヘッドの上に示す事ができます(図A.1参照)。インスタンスの終了は、インスタンス終了記号(−)で示します。ただし必ずしもインスタンスがここで完了するとは限りません。このMSCの範囲内では、このインスタンスに関してそれ以上記述すべきイベントが存在しないという意味に過ぎません。

MOSTで使うMOSTインスタンスには、汎用インスタンスとユーザ定義インスタンスがあります。MOSTインスタンスにはファンクション ブロック名(FBlockID)とオプションのインスタンスID (InstID)があります。汎用インスタンスは自由に定義でき、MOSTファンクション ブロックで記述できないインスタンスに使います。

ユーザ定義インスタンスは、ヒューマンマシン インターフェイス(HMI)等のMOSTインスタンスのユーザ インターフェイスです。抽象化層がヘッドユニットでユーザの動作とユーザ定義インスタンスを分離します。この層は例えば、プッシュボタンまたはスイッチの押下を、ユーザ定義インスタンスがトリガまたは受信できるイベントに変換します。このため、メニュー ナビゲーションの複雑さには関係なく、HMIインスタンスがリストに保存された次のラジオ局を選択する(図A.1のTuneNextFmStation)には、1つのイベントさえあればシステムをモデル化できます。

メッセージ

MSCでは、MOSTメッセージは常にMOSTファンクション カタログに対応している必要があります。メッセージは、時間の経過を示す矢印で表示されます。矢印の起点がメッセージの送信、矢印の先端がメッセージの処理を示します。矢印の上側にメッセージ名を記述し、矢印の下側にオプションのパラメータを( )で囲んで記述します。メッセージは、発生時間の順にインスタンス軸上に配置されます。シーケンスの順番は厳密に定義されているため、この配置は全順序とも呼ばれます。

MOST、CAN、HMIのメッセージを区別するために接頭辞が導入されています。HMIインスタンスが送信するメッセージには接頭辞HMI_Eventが付き、HMIインスタンスが受信するメッセージには接頭辞HMI_Updateが付きます。

MOSTシステムでOPType StartResultまたはStartResultAckによってメソッドがトリガされた場合、ProcessingまたはProcessingAckメッセージが発生する事があります。呼び出されたインスタンスがあらかじめ決められた期間内に返答できない場合、そのインスタンスはProcessingメッセージを送信し、問い合わせがまだ処理中である事をセンダに通知します。Processingメッセージは、MSC内でメソッドに対して問い合わせを実行した後はいつでもどこでも発生する可能性があります。このため、ProcessingメッセージをMSCに記述するとMSCの可読性が損なわれる事になり、必ずしも合理的ではありません。

環境

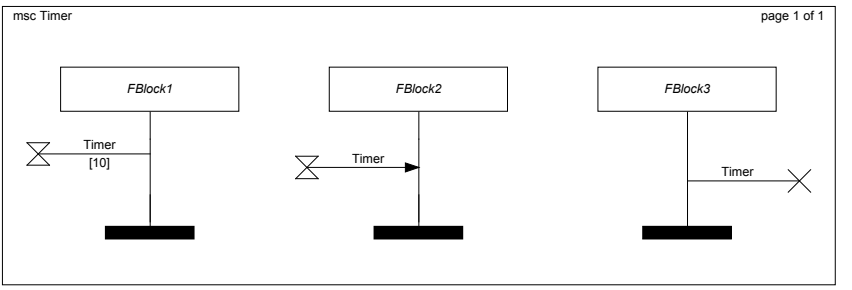
MSCのチャート領域は四角形のフレームで囲まれ、このフレームを環境と呼びます。これは、MSCのシステム環境を定義します。システム環境との間で交換されるメッセージは、環境で開始および終了します(図A.1のUSERDEF_HMI_Eventメッセージ参照)。インスタンス軸に配置された全順序のメッセージとは異なり、環境に対するイベントの順番は定義されません。

アクション

メッセージに加え、インスタンスによるアクションを定義できます。MSCではアクションは四角形のシンボルで表し、この中に任意の文字列を記入できます(図A.1の「Current selected Fm radio station "BBC"」参照)。

タイマ

MSCには、タイマのラベル付けに使う言語要素としてTimer-Start、Timeout、Timer-Stopがあります。Timer-Startはタイマの開始、Timeoutはタイマの経過時間、Timer-Stopはタイマのリセットを定義します。タイマは常に1つのインスタンスに割り当てられます。このため、タイマのシンボル(図A.2参照)は常に対応するインスタンスに接続されます。タイマのシンボルと一緒に、タイマの名前とオプションのパラメータを記述します。砂時計の記号がTimer-Startを表し、砂時計から伸びる矢印がTimeoutを表し、×印がTimer-Stopを表します。



図A.2:MSCタイマのシンボル

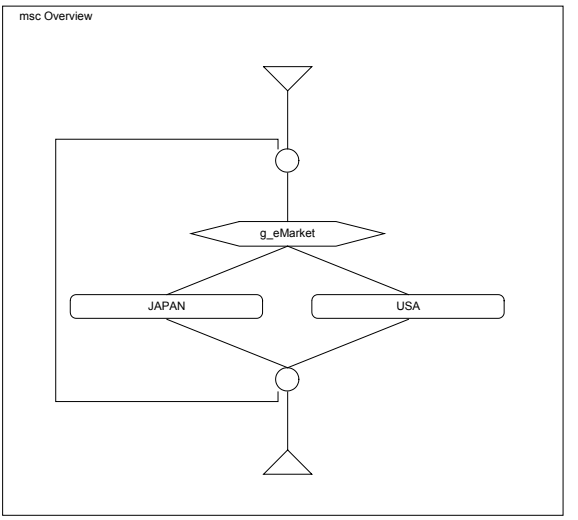
条件

条件は、MSCの複数のインスタンスに関係します。MSCでは、条件を六角形で表します。MSCでは、重要なシステムステートを記述するために条件を使います(例: 図A.1の「Initial State」)。

メッセージ シーケンスを記述するベーシックMSC以外に、さらに上位の記述のための言語構成要素もあります。これらの言語要素を使うと、MSCに別のMSCの一部を組み合わせて複合シーケンスを作成する(インライン表現とハイレベルMSC)、MSCチャートを別のMSCチャートで再利用する(参照)、MSCインスタンスを細分化する(分解)、インスタンスの汎用的なイベント構造を定義する(コリージョン、汎用順序)事ができます。ここではハイレベルMSC (HMSC)についてのみ説明します。MSC言語のこれ以外の構成要素については、MSC仕様([Z.120 99], [Z.120AB 98], [Z.120C1 01])および『MOST MSC Cookbook』[MSC]で詳しく説明しています。

ハイレベルMSC

ベーシックMSCはコンポーネント間のメッセージレベルの通信を記述しますが、ハイレベルMSC (HMSC)は個々のMSCの関係を図で記述するためのものです(図A.3参照)。



図A.3:ハイレベルMSCの例[出典: MSC]

HMSCはインスタンスとメッセージ シーケンスを抽象化します。MSCの組み合わせを記述する目的で設計されたのがHMSCです。このため、この種のグラフはしばしばロードマップと呼ばれます。HMSCを使うと、MSCの並列、順序、代替組み合わせを簡単に記述できます。図A.3に示したHMSCは、販売市場(すなわちグローバルenum定数のg_eMarketがJAPANかUSAか)に応じてMOSTインフォテインメントシステムの機能を決定するシーケンスを記述しています。

この章では、MSC言語の最も重要な要素についてのみ取り上げました。
ここで説明した言語要素以外にも、MSCには多くのコンセプトが定義されており、MSCを完全な仕様記述言語としています。MSCの詳細と完全な説明、およびMOST固有の変更点については、SDL Forum (http://www.sdl-forum.org)およびMOST Cooperationのサイト(http://www.mostcooperation.com)の[MSC]を参照してください。

B MOST25のパケットデータ チャンネルのデータレート

MOST25のパケットデータ チャンネルの実効データレートは、基本的にチャンネル帯域幅とテレグラム長で決まります。テレグラムのセグメント数は下式で求めます[DS 8104]。

$$Tp = roundup\left(\frac{Pd + Ph}{Af}\right)$$
$$= roundup\left(\frac{Pd + Ph}{(15 - SBC) * 4}\right) \quad \text{(式B.1)}$$
$$= roundup\left(\frac{Pd + 10}{(15 - SBC) * 4}\right) Frames$$

1テレグラムの転送時間は、サンプルレート(1秒当たりのフレーム数)と遅延時間(MOSTネットワーク インターフェイス コントローラが次のテレグラムを送信するまでに必要とする間隔)で決まります。

$$Tt = \frac{Tp + Ta}{Fs} s \quad \text{(式B.2)}$$

TtとPdに基づき、実効データレートを下式で求めます。

$$R = \frac{Pd * 8}{Tt} Bit / s \quad \text{(式B.3)}$$

Tp: テレグラムのセグメント数
Pd: 1テレグラムに含まれるデータバイト数
Ph: 1テレグラムに含まれる制御バイト数

調停	1バイト
ターゲット アドレス	2バイト
データ領域の長さ	1バイト
ソースアドレス	2バイト
CRC	4バイト
合計	10バイト

Af: パケットデータ チャンネルの帯域幅
Tf: 1テレグラムの転送時間
Ta: 同一ノードの2つのテレグラムの間隔(フレーム数、NICの場合は常に4フレーム)
SBC: 同期帯域幅制御レジスタの値(6～15)
Fs: サンプルレート(通常は44.1 kHz)
R: 実効データレート

表B.1に、代表的なテレグラム長と想定されるパケットデータ帯域幅における実効データレートをまとめます。

SBC	Pd						単位
	1014	512	256	128	64	48	
6	10.841	9.507	7.526	5.645	3.226	2.822	Mbit/s
7	9.937	8.602	6.947	5.018	3.226	2.822	Mbit/s
8	8.725	7.854	6.451	5.018	3.226	2.419	Mbit/s
9	7.611	6.947	5.645	4.516	2.822	2.419	Mbit/s
10	6.388	5.827	5.018	4.105	2.822	2.419	Mbit/s
11	5.261	4.882	4.301	3.474	2.509	2.117	Mbit/s
12	3.975	3.763	3.345	2.822	2.053	1.882	Mbit/s
13	2.710	2.580	2.377	2.053	1.613	1.411	Mbit/s
14	1.376	1.338	1.272	1.158	0.982	0.891	Mbit/s
15	0	0	0	0	0	0	Mbit/s

表B.1:Ta=4、Fs=44.1 kHzの場合のパケットデータの実効ビットレート

Note:MOST50とMOST150のパケットデータ チャンネルのデータレートについては、セクション5.1と5.6を参照してください。

C 略語

8B10B	8B/10B Code (8B/10B符号化)
A/V	Audio/Video(オーディオ/ビデオ)
A2DP	Advanced Audio Distribution Profile (Bluetoothのプロファイルの1つ)
AAC	Advanced Audio Coding(先進的音響符号化)
ABY	All-Bypass
ACL	Asynchronous Connection-Less (Bluetoothの非同期リンクの1つ)
ADS	Asynchronous Data Service(非同期データサービス)
AG	Audio Gateway (Bluetoothオーディオ ゲートウェイ)
AH	Address Handler(アドレスハンドラ)
AM	Amplitude Modulation(振幅変調)
AMA	Active Member Address
AMS	Application Message Service(アプリケーション メッセージサービス)
ANT	Antenna Tuner(アンテナチューナ)
API	Application Programming Interface(アプリケーション プログラミング インターフェイス)
APWD	Average Pulse Width Distortion(平均パルス幅歪み)
ASYNC	Parallel Asynchronous Mode of the SP (NICのピン名)
AV/C	Audio-Video Compatibility Specifications
AVDTP	Audio/Video Distribution Transport Protocol (Bluetoothのプロファイルの1つ)
BAP	Bedien- und Anzeigeprotokoll(独) (Operation and Display Protocol)
BOSS	Bus-Owner/Supervisor/Selector
BPF	Bits-per-Frame
PACK	Preemptive Acknowledge
CACK	Complete Acknowledge
CAI	Cavity-as-Interface
CAL	CAN Application Layer (CANアプリケーション層)
CAN	Controller Area Network(コントローラ エリア ネットワーク)
CAPL	Communication Access Programming Language (Vector Informatik社によるC言語に似たプログラミング言語)
CDEF	CAN Data Exchange Format
CI	Configuration Interface(コンフィグレーション インターフェイス)
CM	ConnectionMasterまたはConnection Manager(接続マネージャ)
CMD	Command Interpreter(コマンド インタプリタ)

CMS	Control Message Service(制御メッセージ サービス)
CP	Control Port (NICの制御ポート)
CSMA	Carrier Sense Multiple Access(搬送波感知多重アクセス)
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance(搬送波感知多重アクセス/衝突回避)
CSMA/CD	Carrier Sense Multiple Access/Collision Detection(搬送波感知多重アクセス/衝突検出)
CSR	Control and Status Register(制御およびステータス レジスタ)
CU	Critical Unlock(クリティカル アンロック)
DAP	Diagnostic Adaptation Protocol
DDJ	Data-Dependent Jitter(データ依存ジッタ)
DHWG	Digital Home Working Group
DoC	Declaration of Compliance(適合宣言書)
DSD	Direct Stream Digital(ダイレクト ストリーム デジタル)
DSP	Digital Signal Processor(デジタルシグナル プロセッサ)
DSP	Digital Sound Processor(デジタルサウンド プロセッサ)
DTC	Diagnostic Trouble Code(診断トラブルコード)
DTD	Document Type Definition(文書型定義)
DTS	Digital Theatre System
ECL	Electrical Control Line
EHC	External Host Controller(外部ホスト コントローラ)
EHCI-State	External Host Controller Interface State Machine(外部ホスト コントローラ インターフェイス ステートマシン)
EMC	電磁適合性
EMI	電波障害
EOC	Electrical/Optical Converter (E/Oコンバータ)
EOP	End of Production(生産終了)
ePhy	electrical Physical Layer(電気物理層)
ET	EnhancedTestability (FBlockの1つ)
FBlock	Function Block(ファンクション ブロック)
FBlockID	Function Block Identifier(ファンクション ブロックID)
FCS	Frame Check Sequence(フレームチェック シーケンス)
FHSS	Frequency Hopping Spread Spectrum(周波数ホッピング方式、Bluetooth)
FIFO	First-In First-Out(先入先出バッファ)
FktID	Function Identfier(ファンクションID)
FM	Frequency Modulation(周波数変調)
FOR	Fiber Optic Receiver(光ファイバレシーバ)
FOT	Fiber Optic Transceiver(光ファイバ トランシーバ)
FOX	Fiber Optic Transmitter(光ファイバ トランスミッタ)
FPGA	Field-Programmable Gate Array
FS	Frame Synchronization(フレーム同期、サンプルレート)
FSY	Synchronization Pin (INICの同期ピン)

FTDMA	Flexible Time Division Multiple Access
FWHM	Full-Width at Half-Maximum(半値全幅)
GAVDP	Generic Audio/Video Distribution Profile (Bluetoothのプロファイルの1つ)
GMI	Golden MOST Implementations(ゴールデンMOSTインプリメンテーション)
GOEP	Generic Object Exchange Profile (Bluetoothのプロファイルの1つ)
GSM	Global System for Mobile Communications (Bluetoothのプロファイルの1つ)
GW	Gateway(ゲートウェイ)
HAVi	Home Audio-Video Interoperability
HCI	Host Controller Interface (Bluetoothのインターフェイスの1つ)
HF	Hands-Free Unit (Bluetoothのインターフェイスの1つ)
HMI	Human Machine Interface(ヒューマンマシン インターフェイス)
HU	Head Unit(ヘッドユニット)
IEEE	Institute of Electrical and Electronics Engineers
IFG	Inter Frame Gap(フレーム間隔時間)
INIC	Intelligent Network Interface Controller(インテリジェント ネットワーク インターフェイス コントローラ)
InstID	Instance ID(ファンクション ブロックのインスタンスID)
IP	Intellectual Property
ISM	Industrial Scientific Medical Band(免許不要で利用できる産業科学医療用周波数帯)
ITU	International Telecommunications Union(国際電気通信連合)
JPEG	Joint Photographic Experts Group
JTAG	Joint Test Action Group
L2CAP	Logical Link Control and Adaption Protocol (Bluetoothのプロトコルの1つ)
LAN	Local Area Network(ローカルエリア ネットワーク)
LED	Light Emitting Diode(発光ダイオード)
LEG	LEG OS8104 compatibility mode (NIC OS8104Aのピン名)
LFE	Low Frequency Effect
LLC	Logical Link Control(論理リンク制御)
LLD	Low Level Driver(ローレベル ドライバ)
LMP	Link Manager Protocol (Bluetoothのプロトコルの1つ)
MAC	Media Access Control(メディアアクセス制御)
MAMAC	MOST Asynchronous Medium Access Control
MBM	Message Buffer Management(メッセージ バッファ マネジメント)
MCA	MOST Compliance Administrator (MOSTコンプライアンス アドミニストレータ)

MCPL	MOST Compliance Product List (MOSTコンプライアンス製品リスト)
mCRA	Channel Resource Allocation Table
MCS	MOST Transceiver Control Service (MOSTトランシーバ制御サービス)
MCSB	MOST Compliance Supervisory Board (MOSTコンプライアンス監督委員会)
MCTG	MOST Compliance Technical Group (MOSTコンプライアンス技術グループ)
MCTH	MOST Compliance Test House (MOSTコンプライアンス テストハウス)
MDM	MOST Debug Message Module (MOSTデバッグ メッセージモジュール)
MDP	MOST Data Packets (MOSTデータパケット)
MEP	MOST Ethernet Packets (MOST Ethernetパケット)
MediaLB	Media Local Bus(メディア ローカルバス)
MHP	MOST High Protocol
MIMO	Multiple Input-Multiple Output
MIS	Message Interface Service(メッセージ インターフェイス サービス)
MLBCLK	MediaLB Clock (MediaLBクロック)
MLBDAT	MediaLB Data (MediaLBデータ)
MLBSIG	MediaLB Signal (MediaLB信号)
MNS	MOST NetServices Kernel (MOST NetServicesカーネル)
MPEG	Moving Picture Expert Group
MPR	Maximum Position Register(最大位置レジスタ)
MRT	MOST Routing Table (MOSTルーティング テーブル)
MSC	Message Sequence Chart
MSV	MOST Supervisor (MOSTスーパバイザ)
Mutex	Mutual Exclusion(相互排他)
NA	Numerical Aperture(開口数)
NAV	Navigation Device(ナビゲーション機器)
NB	NetBlock
NBEHC	NetBlock on EHC (NBMINと連携)
NBMIN	Minimum NetBlock (INICに実装される最小機能のNetBlock)
NCE	Network Change Event(ネットワーク変更イベント)
NDIS	Network Driver Interface Specification
NIC	Network Interface Controller(ネットワーク インターフェイス コントローラ)
NTFS	Notification Service(ノーティフィケーション サービス)
OEC	Optical/Electrical Converter (O/Eコンバータ)
OEM	Original Equipment Manufacturer
OpCodes	Operation Codes(オペコード)

oPhy	optical Physical Layer(光物理層)
OpType	Operation Type(オペレーション タイプ)
OS	Operating System(オペレーティング システム)
OSI	Open Systems Interconnection(開放型システム間相互接続)
OUI	Organizationally Unique Identifier(ベンダ識別子)
PAL	Protocol Adaption Layer
PAR_CP	Control Port Parallel (NICのピン名)
PAR_SRC	Source Port Parallel (NICのピン名)
PARC	Palo Alto Research Center (Xerox社の子会社)
PC	Physical Channel(物理チャンネル)
PCB	Printed Circuit Board(プリント基板)
PCI	Peripheral Component Interconnect
PCM	Pulse Code Modulation(パルス符号変調)
PCS	Polymer Clad Silica(ポリマクラッド シリカ)
PDC	Packet Data Channel(パケットデータ チャンネル)
PIM	Personal Information Management(個人情報管理)
PLL	Phase-locked Loop(位相ロックループ)
PM	Port Message Protocol(ポートメッセージ プロトコル)
PMA	Parked Member Address
PMMA	Polymethyl Methacrylate(ポリメタクリル酸メチル)
PMS	Port Message Service(ポートメッセージ サービス)
POF	Plastic Optical Fiber(プラスチック光ファイバ)
ppm	parts per million(百万分率)
PWD	Pulse Width Distortion(パルス幅歪み)
PWV	Pulse Width Variation(パルス幅の変動)
PXI	PCI eXtensions for Instrumentation (1997年にNational Instruments社によって初めて導入されたモジュール型計測プラットフォーム)
QA	Quality Assurance(品質保証)
QoS	Quality of Service(サービス品質)
RCLED	Resonant Cavity Light Emitting Diode(共振器型LED)
RBD	Ring Break Diagnostic(リングブレーク診断)
RFCOMM	Radio Frequency Communication (Bluetoothのプロトコルの1つ)
RMCK	Recovered Master Clock (NICまたはINICのピン名)
rRMS	Root Mean Square(二乗平均平方根)
RTCP	Real Time Control Protocol
RTP	Real Time Transport Protocol
RTR	Remote Transmission Request
SA	Super Audio
SCK	Clock Pin(クロックピン)
SCL	Clock of the serial CP Interface (NICのピン名)
SCO	Synchronous Connection Oriented (Bluetoothのリンクの1つ)

SCS	Synchronous Connection Service(同期コネクション サービス)
SDP	Service Discovery Protocol (Bluetoothのプロトコルの1つ)
SH	Socket Handle(ソケットハンドル)
SI	Step Index(ステップ インデックス)
SIG	Special Interest Group
SIM	Subscriber Identity Module(加入者識別モジュール)
SNAP	Sub Network Access Protocol
SOP	Start of Production(量産開始)
SP	Source Port (NICのソースポート)
SR	Streaming Data Receiver(ストリーミング データレシーバ)
SSO	Sudden Signal Off(突発的信号OFF)
STP	Shielded Twisted Pair(シールド付きツイストペア)
SX	Streaming Data Transmitter(ストリーミング データ トランスミッタ)
SyncML	Synchronization Markup Language
SW	Software(ソフトウェア)
TCU	Telephone Control Unit
TDM	Time Division Multiplexing(時間分割多重)
TDMA	Time Division Multiple Access(時間分割多元接続)
TP2.0	Transport Protocol 2.0 (Volkswagen社の独自プロトコル)
TSI	Transport Stream Interface(トランスポート ストリーム インターフェイス)
TV	Television Tuner(テレビチューナ)
UDS	Unified Diagnostic Services
UI	Unit Interval(データ信号の最も短いパルス)
UJ	Uncorrelated Jitter(非相関ジッタ)
UPnP	Universal Plug-and-Play(ユニバーサル プラグアンドプレイ)
USB	Universal Serial Bus(ユニバーサル シリアルバス)
UTP	Unshielded Twisted Pair(シールドなしツイストペア)
vMSV	Virtual MOST Supervisor(バーチャルMOSTスーパバイザ)
wADS	Asynchronous Data Service Wrapper(非同期データ サービスラッパ)
wAMS	Application Message Service Wrapper(アプリケーション メッセージ サービスラッパ)
wCMS	Control Message Service Wrapper(制御メッセージ サービスラッパ)
WLAN	Wireless Local Area Network(無線LAN)
wMCS	MOST Processor Control Service Wrapper (MOSTプロセッサ制御サービスラッパ)
wSCM	Socket Connection Manager Wrapper(ソケット接続マネージャラッパ)
www	World Wide Web(ワールドワイド ウェブ)

D 用語集

API
アプリケーション プログラミング インターフェイス。他のプログラムがソフトウェア要素(例: 通信インターフェイス)にアクセスできるようにするためのインターフェイス。

BMC (Biphase Mark Code)
マンチェスタ符号化によく似たライン符号化方式です。ビットの先頭で必ず遷移が発生します。「1」はビットの中央で遷移します。「0」はビットの最後まで遷移しません。

ConnectionMaster
MOSTシステムでストリーミング チャンネルを管理するファンクション ブロック。

DeviceID
DeviceIDはネットワーク内の物理デバイスまたはデバイスグループを表します。DeviceID (RxTxAdr)はノード位置アドレス(RxTxPos)、論理アドレス(RxTxLog)、グループアドレスのいずれかを表します。

FBlockシャドウ
FBlockをプロキシ経由で制御する実装コンセプトです。

Init Ready
Init Readyイベントは、MOSTシステムの状態がNetInterface通常動作に遷移した事をアプリケーションに通知します。

NetBlock
管理機能用に全てのデバイスに実装されるファンクション ブロック。

NetInterface
物理層、MOSTネットワーク インターフェイス コントローラ、ネットワークサービスで構成されるMOSTインターフェイス全体。

NetServices
「ネットワーク サービス」参照。

NetworkMaster
システムステートを制御し、セントラル レジストリを管理するファンクション ブロック。

PLL(位相ロックループ)
位相比較器を使ってオシレータを入力周波数の位相に同期させる制御回路。

POF(ポリマ光ファイバ、プラスチック光ファイバ)
ポリマ光ファイバは、データ転送用のプラスチック光ファイバの一種。一般に、POFのコアの材料にはポリメタクリル酸メチル(PMMA)を使います。通常、光ファイバは安定性向上のためにコーティングが施されます。

PowerMaster
PowerMasterはMOSTシステムの起動とシャットダウンを司り、電源管理を監視します。

RxTxAdr
MOSTノードの3種類のアドレス、すなわち論理ノードアドレス(RxTxLog)、ノード位置アドレス(RxTxPos)、グループアドレスの総称。TxAdrはトランスミッタのアドレスで、RxAdrはレシーバのアドレスです。

RxTxLog
論理アドレス。TxLogはトランスミッタの論理アドレスで、RxLogはレシーバの論理アドレスです。MOSTリング内で重複する事はありません。

RxTxPos
ノードのノード位置アドレス。リング内のノード位置で決まります。TxPosはトランスミッタのアドレス、RxPosはレシーバのアドレスです。

S/PDIF
Sony/Philips Digital Interfaceの略。異なるデバイス間でデジタル オーディオ信号を転送するためのバスおよびインターフェイスの仕様。Sony/Philips Digiconnectフォーマット、S/P-DIF、TOSLINK等と呼ばれる事もあります。

TimingMaster
フレームとブロックのためのシステムクロックはTimingMasterが生成します。TimingSlavesは、このシステムクロックに同期します。

$U_{Critical}$
通信は十分可能でも、アプリケーションの安全な動作を保証できなくなる低電圧レベル。この場合、ソースはゼロを送り、シンクは出力信号を保護する必要があります。Muteプロパティは変更されません。回復時、出力信号は即時復元できます。

U_{Low}
NetInterfaceが安定動作しなくなり、通信を維持できなくなるデバイス固有の低電圧レベル。U_{Low}に達すると、デバイスは変調信号をOFFに切り換え、DevicePowerOffモードに切り換わります。電源電圧が回復した場合でも、デバイスはDevicePowerOffモードにとどまります。

t_{Diag_Signal}
TimingSlaveノードがRx入力のアクティビティを検出しなくなってからTimingMasterモードに切り換わるまでの時間。

$t_{SSO\_ShutDown}$
シャットダウン フラグをセットしてから、出力信号をOFFにするまでの時間。

アンロック
TimingSlaveは、PLLを使ってRxの入力信号に同期できなくなるとアンロックを検出します。

アンロック イベント
MOSTネットワーク インターフェイス コントローラがPLLから安定クロックを生成できなくなると、ネットワーク サービスがアプリケーションに対してアンロック イベントを生成します。

ウェーブはんだ
電子機器製造におけるプロセス工程の1つ。噴流させた溶融はんだにPCBを浸漬してはんだ付けし、PCBと部品を確実に接続します。

エラッタ
仕様の誤りを訂正し、内容を明確化する補完的文書をエラッタといいます。MOST仕様では、コンプライアンス仕様にエラッタがあります。

置き換え法
テスト対象のファイバの特性を基準となるリファレンス ファイバの特性と比較して計測を行う方法。

押出成形
プラスチックに圧力をかけてノズルから押し出し、連続的に成形するプロセス。

オペレーション
MOST仕様では、プロパティとメソッドに対して実行できる標準のオペレーションを定義しています。

開口数
光ファイバに光を入射できる最大受光角。

キャビティ
ピグテールのくぼみ部分。

吸湿性材料
湿気を吸収する材料。吸湿性材料の取り扱いが不適切な場合、溶着プロセスで問題が発生する事があります。

クラッド
POFのコアをおおう部分。

クワドレット
1クワドレット = 4バイト。

ゲートウェイ
異なるデータ転送システムを接続するためのもの。ゲートウェイ接続では、各システムの完全な通信スタック(OSI参照モデルの全ての既存の階層)を実装する必要があります。

減衰
信号強度の低下を通常dB(デシベル)単位で表したもの。信号強度の低下を正の値、増大を負の値で示します。

コーティング
POFの外装被覆。車載アプリケーションでは二重コーティングを使います。

コーデック
コーダとデコーダで構成される混成語。データまたは信号をデジタル符号化および復号化するプロセスまたはプログラムを指します。

ゴールデン ファイバ
計測の基準となるリファレンス ファイバの事。

コントローラ
ファンクション ブロックを制御するもの。MOSTシステムでしばしば特定の機能(例: オーディオ機能)を司ります。

再結合
正または負の電荷を持ったキャリア (イオン、電子)が反対の電荷を持ったキャリアと結合し、電気的に中性な状態(原子、分子)になる事。

システムスキャン
NetworkMasterがNetworkSlaveのファンクション ブロックの設定データを読み出す動作。ネットワーク スキャンとも呼ばれます。

システムステート
通常動作中のシステムステートはOK、初期化中またはシステムでアドレス競合が発生した場合のシステムステートはNotOKです。

消灯状態
いずれかのノードが変調信号をOFFにした状態。この後、全てのノードが消灯状態に切り換わり、システムステートNotOKを経由して、最終的にNetInterface Offステートに遷移します。

ストリーミング データ
オーディオやビデオ等のデータ。MOSTでは、ストリーミング データ チャンネルで転送します。

スパイモード
解析ツールの動作モードの1つ。ネットワークからは透過的にバス通信全体をリッスンできます。

スレーブ
コントローラで制御されるファンクション ブロック、またはそのファンクション ブロックを実装したデバイス。

制御コマンド
MOSTシステムを制御するためのデータパケット。通常は制御チャンネルで転送されますが、パケットデータ チャンネルを使う事もあります。

セントラル レジストリ
システム内に存在するファンクション ブロックと、そのファンクション ブロックが実装されているデバイスの論理アドレスを登録したテーブル。セントラル レジストリはNetworkMasterに実装されます。

点灯状態
変調信号がONに切り換わった状態をいいます。

デセントラル レジストリ
NetworkSlaveにあるセントラル レジストリ(またはその一部)のコピー。MOSTシステム内の他のファンクション ブロックにアクセスするために使います。

デバイス
ネットワーク インターフェイス コントローラ経由でMOSTシステムに接続される物理デバイスをMOSTデバイスといいます。

デラミネーション
埋込用樹脂とICのリードフレームが剥離する事。熱応力の繰り返しによって発生する特徴的な損傷です。

トランシーバ
トランスミッタとレシーバから成る混成語。

導入
ファンクションをPCB上の電気的インフラストラクチャとして存在する実際のハードウェア構造に割り当てる事。

ネットワーク サービス
MOSTネットワーク インターフェイス コントローラを制御するためのドライバ層。MOSTシステムの全てのデバイスに実装が必要です。アプリケーションとシステムサービスはネットワーク サービスを介してMOSTシステムにアクセスします。NetServicesは、Microchip社が実装したネットワーク サービスの商標です。

ネットワーク スキャン
「システムスキャン」参照。

ネットワーク変更イベント(NCE)
リング内のアクティブなノードの数が変化した場合に発生するイベント。起動中にデバイスのブートアップに時間がかかり、後にバイパス無効になった場合、ノードの障害またはスリープへの移行によってバイパスが閉じた場合等に発生します。

バイパス
MOSTネットワーク インターフェイス コントローラの機能です。初期化フェイズ中、デバイスエラー発生時、デバイスを正しい手順でOFFにした場合にバイパスが

閉じます。この場合、入力(RX)で受け取った全てのデータを直接送信(TX)へ送ります。

バウンダリ ディスクリプタ
1フレーム内でのストリーミング データとパケットデータのレートを決定します。

パケットデータ
パケットデータ チャンネルで送信される大規模なデータパケット(例: MOSTバスを利用したTCP/IP接続のトンネリングの場合)。

ピグテール
光トランスミッタ、光レシーバ、2本のPOFファイバ、機械的アダプタで構成され、ファイバケーブルの一方の先端にPOFコネクタを取り付けたもの。

ファンクション
ファンクション ブロックに含まれるプロパティとメソッドの総称。

ファンクション ブロック(FBlock)
特定のアプリケーション(例: ラジオチューナ、電話)またはシステムサービスを制御するためのインターフェイス(すなわちメソッドとプロパティ)で構成されるオブジェクト。

フェルール
POFファイバの先端に取り付け、ファイバを固定する機械的インターフェイスとして機能する円筒形の部品。

フラッシュメモリ
消去/書き換え可能なメモリ。通常、車載分野ではマイクロコントローラのソフトウェアをフラッシュメモリに格納します。

フレーム
MOSTデータリンク層の基本的なデータ構造で、MOSTリングで周期的に送信されます。ストリーミング チャンネル、パケットデータ チャンネル、制御チャンネルに属するデータをフレームで伝送します。

フレネル損失
屈折率の異なる媒体間を光が通過する際に生じる損失。

プロパティ
ファンクション ブロックの特定の属性を表すファンクションをプロパティといいます。ファンクション ブロックの現在の状態を問い合わせる場合、または変更する場合にプロパティを使います。

冪等
関数またはメソッドを何回適用しても、1回適用した場合と同じ結果になる事。

ボンディング断線
チップをリードフレームに電気的に接続するワイヤの断線。

メソッド
ファンクション ブロックに含まれるファンクションのうち、何らかのアクションを開始し、一定時間の経過後に何らかの結果が得られるものをメソッドといいます。

モード分散
モード(光路)の伝搬定数の違いによって光ファイバ内で生じるパルス広がりの事。

リードフレーム
薄板の金属に接続ピンとICチップを支持固定するダイパッドをスタンピング加工したもの。

ロック
TimingSlavesがMOST信号に同期する事をロックといいます。

ロックイベント
MOSTネットワーク インターフェイス コントローラが最初に、またはアンロックイベントの後に信号に安定同期できる場合に生成されるイベント。

E 参考文献

[AppN Mig] Application Note MOST150 Migration, Rev 1.0; 05/2009; MOST Cooperation

[Bluetooth] Specification of the Bluetooth System-Profile, Version 1.1, 02/2001; Special Interested Group Bluetooth

[Burton 2004] Automotive Testing of Automotive Telematics Systems using TTCN-3, Simon Burton, Andre Baresel, Ina Schieferdecker, Proceedings of the 3rd Workshop on System Testing and Validation (SV04), December 2004.

[Cat 02] MOST Function Catalog Read Me First; Rev 1.0; 12/2002; MOST Cooperation

[Com_ePhyL] MOST ePHY Compliance Test Spec (with ePhy Compliance Test Patterns) Rev 1.0; 06/2006; MOST Cooperation

[ComPhyL 1.0] MOST Specification MOST Compliance Test of Physical Layer; Rev 1.0; 12/2003; MOST Cooperation

[ComPhyL 150] MOST150 oPhy Compliance Measurement Guideline; Rev. 1.1; 08/2010; MOST Cooperation

[ComPhyL_Err] ERRATA MOST Compliance Test of Physical Layer; Rev 1.0, 05/ 2006; MOST Cooperation

[ComPhyLP 1.0] MOST Compliance Verification Procedure Physical Layer; Rev. 1.0; 07/2004; MOST Cooperation

[ComPhyLP 150] MOST150 oPhy Compliance Verification Procedure – Physical Layer; Rev. 1.1; 07/2010; MOST Cooperation

[ComPhyLP_Err] ERRATA MOST Compliance Verification Procedure Physical Layer; Rev. 1.1; Rev. 1.0; 07/2005; MOST Cooperation

[ComProf] MOST Profile Compliance Test Specification; Rev 1.0; 10/2005; MOST Cooperation

[ComReq] MOST Specification - MOST Compliance Requirements, Rev 2.1; 04/2009; MOST Cooperation

[ContProt] MOST Content Protection Scheme DTCP Implementation; Rev. 3.0; 08/2009; MOST Cooperation

[Core] MOST Core Compliance Test Specification; Rev 1.1.01; 06/2005; MOST Cooperation

[Core_Err] ERRATA MOST Core Compliance Test Specification; Rev 1.3; 07/2009; MOST Cooperation

[Dau 01] Daum W. et al.:„ POF‐ optische Polymerfasern für die Datenkommunikation", Springer Verlag, 2001

[DPA] Diagnostic Protocols Adaptation Specification, Rev. 1.0.1; 08/2009; MOST Cooperation

[DS 8104] OS8104‐ MOST Network Transceiver‐ Final Product Data Sheet; Jan. 2003; www.smsc-ais.com

[DS FCM110] MOST Fiber Optic Transceiver, FCM110R/ FCM110D, Firecomms Ltd.

[DTD Cook] MOST DTD Cookbook; Rev 1.2; 12/2009; MOST Cooperation

[DynSpec] MOST Dynamic Specification; Rev 3.0; 07/2008; MOST Cooperation

[ECL] Electrical Control Line, Rev. 1.0.1; 08/2009; MOST Cooperation

[ETS 300 406] ETS 300 406 „ Methods for Testing and Specifications (MTS); Protocol and Profile Conformance Testing Specifications; Standardization methodology” http://portal.etsi.org/mbs

[ETSI 2003] ETSI ES 201873:The Testing and Test Control Notation TTCN-3, Part 1 (Core Language), Part 2 (Tabular Format), Part 3 (Graphical Format), Part 4 (Operational Semantics), Part 5 (TTCN-3 Runtime Interface), Part 6 (TTCN-3 Control Interfaces), Sophia-Antipolis, France, October 2003.

[FBADiskPl 2.4] MOST FunctionBlock AudioDiskPlayer; Rev 2.4; 09/2003; MOST Cooperation

[FBAmFmT 2.4] MOST FunctionBlock AmFmTuner, Rev 2.4.2; 09/2003; MOST Cooperation

[FBAudioA 2.4] MOST FunctionBlock AudioAmplifier; Rev 2.4.2; 09/2003; MOST Cooperation

[FBAuxIn 3.5] MOST FunctionBlock AuxIn, Rev 3.5; 10/2008; MOST Cooperation

[FBCM] MOST FunctionBlock ConnectionMaster; Rev 3.0.1; 09/2010; MOST Cooperation

[FBET] MOST FunctionBlock EnhancedTestability; Rev 3.0; 8/2010; MOST Cooperation

[FBGenPl 2.5] MOST FunctionBlock GeneralPlayer; Rev 2.5.1; 01/2008; MOST Cooperation

[FBVehicle 1.6] MOST FunctionBlock Vehicle, Rev 1.6 09/2002; MOST Cooperation

[FBX Cook] MOST FIBEX Cookbook; Rev 1.0; 03/2009; MOST Cooperation

[Found 04] MOST Foundation Training; Training Handout; Oasis SiliconSystems; 2004

[GADV] 4CS, GADV Gmbh, http://www.gadv.de/

[GenFB] MOST FunctionBlock GeneralFBlock; Rev 3.0.2; 09/2010; MOST Cooperation

[Gus 98] D. Gustedt, W. Wiesner:„ Fiber-Optic-Übertragungstechnik", Sende- und Empfangsgrundlagen, Franzis-Verlag, 1998

[HFP 1.5] Hands-Free Profile 1.5; HFP1.5_SPEC; 2005-11-25 www.bluetooth.org

[ISO 14229] Road vehicles -- Unified diagnostic services (UDS), ISO 14229.ISO, Geneve 2009

[ISO 14230] Road vehicles -- Diagnostic systems -- Keyword Protocol 2000, ISO 14230, Geneve 1999/2000

[ISO 9646] Information Technology, Open Systems Interconnection, Conformance Testing Methodology and Framework.International Standard IS-9646.ISO, Geneve 1991

[ITUZ120] Z.120:Message Sequence Charts, MSC-2000, Geneva, Switzerland, October 1999.

[Lehmann 2000] Test Case Design by Means of the CTE XL.E. Lehmann; Wegener, J Proceedings of the 8th European International Conference on Software Testing, Analysis & Review (EuroSTAR 2000), Kopenhagen, Denmark, December 2000.

[MAMAC] MAMAC Specification; Rev. 1.1; 12/2003; MOST Cooperation

[MediaLB] Media Local Bus Specification; Physical Layer and Link Layer; Version 3.0; SMSC; www.smsc-ais.com

[Mercury] Test Director, Mercury Interactive Limited, http://www.testerlyzer.de/

[MHP] MOST High Protocol Specification; Rev 2.3; 12/2008; MOST Cooperation

[MNS L1 1.x] MOSTNetServices; Basic Services (Layer I); Programmers' Library for Interfacing to MOST User Manual and Specification of Layer I; Preliminary Version 1.10; www.smsc-ais.com

[MNS L1 2.x] MOST NetServices Layer I; Wrapper for INIC; V2.1.x; User Manual/Specification; www.smsc-ais.com

[MNS L1 3.x] MOST NetServices Layer I; Wrapper for INIC; V3.x; User Manual/Specification; www.smsc-ais.com

[MNS L1 Exa] MOST NetServices; Layer 1 Example; Version 1.10-01; www.smsc-ais.com

[MNS L2] MOST NetServices Layer II; User Manual/Specification; Rev. 1.10.x / 2.1.x; www.smsc-ais.com

[MOST 2.4] MOST Specification; Rev 2.4; 05/2005; MOST Cooperation

[MOST 2.5] MOST Specification; Rev 2.5; 10/2006; MOST Cooperation

[MOST 2003] Empfehlungen für Optische MOST Interfaces in Automotive Anwendungen (Version 1.0), MOST Co, Audi AG, BMW AG, DaimlerChrysler AG, RMCTech GmbH, 2003.

[MOST 3.0] MOST Specification; Rev 3.0 E2; 07/2010; MOST Cooperation

[MOST AARSMD] Automotive Application Recommendation for optical MOST Components - Surface Mount Device (SMD) Rev. 1.0, 10/2010, MOST Cooperation

[MOST AARTHM] Automotive Application Recommendation for optical MOST Interfaces, Rev. 3.0-06, 06/2009, MOST Cooperation

[MOST AppTF] Application Note POF Transfer Function, Rev. 1.0, 03/2009, MOST Cooperation

[MOST BaPhy] MOST Physical Layer Basic Specification, Rev. 1.0, 12/2008, MOST Cooperation

[MOST ePhy] MOST Specification of Electrical Physical Layer, Rev. 1.0, 12/2004, MOST Cooperation

[MOST Org] MOST Cooperation Organizational Procedures Rev 3.5 04/2007; MOST Cooperation

[MOST Phy150] MOST150 oPHY Automotive Physical Layer Sub-Specification, Rev. 1.1, 05/2010, MOST Cooperation

[MOST Stream] Stream Transmission Specification Rev. 3.0.08/2009, MOST Cooperation

[MSC] MOST MSC Cookbook; Rev 1.2; 08/2005; MOST Cooperation

[NetBlock] MOST FunctionBlock NetBlock; Rev 3.0.1; 09/2010; MOST Cooperation

[Optitas] Tool4M-XL, Optitas GmbH, http://www.optitas.de/

[PBAP] Phone Book Access Profile; PBAP_SPEC; www.bluetooth.org

[PFLNet] MOST NetServices API – The Interface to the MOST Network; Product Flyer 04/03; SMSC; www.smsc-ais.com

[Por 2004] Vernetzte Systeme im Cayenne; Porsche Engineering Magazine; Ausgabe 01/2004; S. 12 - S.15

[Praesideo] Praesideo data sheet; www.bosch-sicherheitsprodukte.de (Nov. 2010)

[RFC 2616] Hypertext Transfer Protocol – HTTP/1.1; Juni 1999; R. Fielding, J. Gettys, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee

[Ruetz] Testerlyzer, Ruetz Technologies GmbH, http://www.testerlyzer.de/

[SAP] SIM Access Profile; SAP_SPEC; www.bluetooth.org

[Schr 01] A. Schramm, ”MOST Investigations at BMW, Migration or Evolution“, Annual All Members Meeting, Frankfurt, Germany, 11th March 2008

[Schr 02] A. Schramm, “MOST in der dritten Generation:MOST150 Systemaspekte und Migrationsalternativen“, ElektronikPraxis; 8th August 2008

[SMSC] Optolyzer G2, SMSC, Automotive Infotainment Systems GmbH, www.smsc-ais.com

[Stream] MOST Specification for Stream Transmission; Rev. 3.0; 07/2010; MOST Cooperation

[Tutorial] MOST Tutorial-CD; MOST Cooperation

[Vector] CANoe- Development and Test Tool for CAN, LIN, MOST, FlexRay, Ethernet and J1708, www.vector.com

[Vog 02] E. Voges, K. Petermann:„ Optische Kommunikationstechnik“, Springer Verlag, 2002

[Wei 99] Weinert A.:„ Plastic Optical Fibers“, Publicis MCD Verlag, Erlangen, 1999

[Z.120 04] ITU-T Message Sequence Chart (MSC); Geneva; (04/2004); http://www.itu.int

[Z.120 99] ITU-T Recommendation Z.120 (MSC2000) Message Sequence Chart (MSC); Geneva; (11/1999); http://www.itu.int

[Z.120AB 98] ITU-T Message Sequence Chart (MSC) Annex B:Formal sematics of Message Sequence Charts; Geneva; (04/1998); http://www.itu.int

[Z.120C1 01] ITU-T Z.LC-Text:Recommendation Z.120 Corrigendum 1; Geneva; (11/2001); http://www.itu.int

索引

N

O

P

Q

R

さ

た

な

は

ま

や

ら

わ

編集: Prof. Dr. Ing. Andreas Grzemba

MOST®

車載マルチメディア ネットワーク

MOST25からMOST150へ