MIDI 1.0 規格書

MIDI Standard & Recommended Practice（日本語版 98.1）

本書の編集について

本書は，英語版 MIDI 1.0 規格書「The Complete MIDI 1.0 Detailed Specification」Document Version 96.1 をもとに，各項目ごとに対応させて編集してあり，基本的には英語版の構成に従っているが，日本語版「MIDI1.0 規格 Document Ver. 4.1 日本語版」（1989 年発行）の表現を尊重しながら，なるべく解りやすくすることに努めた。

社団法人　音楽電子事業協会

Association of Musical Electronics Industry

〒 101-0061　東京都千代田区神田三崎町 2-16-9 イトービル 4F

電話：03-5226-8550

FAX：03-5226-8549

http://www.amei.or.jp/

Copyright ©1998 社団法人 音楽電子事業協会

Portions Copyright © 1885, 1989 Association of Musical Electronics Industry, 社団法人 音楽電子事業協会

目　次

5　GMシステム・レベル1

6　MIDI ショー・コントロール 1.1

7　MIDI マシン・コントロール 1.0

1. MIDI とミュージック・シンセシス入門

MIDI とミュージック・シンセシス入門

この文章は 1995 年に Crystal Semiconductor 社の Jim Heckroth により書かれたものを日本語化したものである。

Copyright ©1998 社団法人　音楽電子事業協会

MIDI とミュージック・シンセシス入門

Musical Instrument Digital Interface（MIDI）プロトコルは，1982/1983 にかけて創案されて以来，ミュージシャンや作曲家によって幅広く受け入れられ，活用されて来た。MIDI データは，楽曲の演奏の情報を表現するたいへん効率のいい方式であり，このため MIDI は，作曲家や演奏者のみならず，マルチメディア・プレゼンテーションやコンピューター・ゲームのような，サウンドを伴うコンピューター・アプリケーションにとっても，魅力的なプロトコルとなっている。しかしながら，シンセサイザーの機能が標準化されていないことが，アプリケーション開発者への普及を妨げ，また新しい MIDI ユーザーにとって習得困難なものとしていた。

幸いにも，General MIDI システム仕様の発表や，一般的な PC/MIDI インターフェースの普及，マイクロソフト Windows やその他の OS における MIDI のサポート，そして低価格なミュージック・シンセサイザーの発展のおかげで，MIDI プロトコルは今やますます多くのアプリケーションにおいて，広い範囲で使用されつつある。この文章は，MIDI プロトコルを使用したサウンドの生成に関して，規格や慣例，専門用語を概説するものである。

MIDI とデジタル化されたオーディオ

MIDI プロトコルは，元来ミュージシャンが複数のシンセサイザーを接続できるようにするために開発されたが，今ではゲームやマルチメディア・アプリケーションにおいて，デジタル化されたオーディオを代替，あるいは補完する伝送メディアとして，広い範囲で使用されつつある。ディスクや CD-ROM 上のサンプリングされた音を使用するのにくらべ，MIDI シンセサイザーによってサウンドを生成する方が，いくつかの利点がある。

第1の利点は,必要な記憶領域である。デジタル･サンプリングされた音を PCM フォーマットで記録した(.WAV ファイルのような)ファイルは，相当大きくなりがちである。これはとりわけ，高いサンプリング・レートを使用して，ステレオで収録された長い楽曲の場合に当てはまる。

一方 MIDI データ・ファイルは，サンプリングされた音声ファイルと比較した場合，非常に小さい。例えば，ステレオ・サンプリングされた高品位な音声を含むファイルは，1分のサウンドにつき約 10M バイトを必要とするが，典型的な MIDI シーケンスでは，1分のサウンドにつき約 10K バイト以下しか消費しない。これは MIDI ファイルが，サンプリングされた音声データではなく，シンセサイザーがサウンドを演奏するのに必要とされる指示のみを含んでいるからである。これらの指示は，MIDI メッセージの形式で，シンセサイザーに対し，どの音色を使用するか，各ノートをどの音程で，それぞれどれくらいの大きさで演奏するかを指示する。そして，実際のサウンドは，シンセサイザーによって生成されるのである。

コンピューターにとって，ファイル・サイズが小さいということは，サウンドを生成している周辺機器へ，このデータを準備して送り出すために必要なPCの情報量が，より小さくなることをも意味する。サウンド生成に MIDI を利用することの他の利点として，音楽を容易に編集できることや,再生スピードと音程(調性)を独立して変更できることが挙げられる。この最後の点は,ソングのキーとテンポがユーザーによって設定できるような，カラオケ機器などのアプリケーションにおいては，とりわけ重要である。

MIDI の基礎

Musical Instrument Digital Interface（MIDI）プロトコルは，楽曲の演奏の情報を電子データとして伝達するための，規格化された効率的な手段を提供する。MIDI 情報は “MIDI メッセージ” として送られるが，それはミュージック・シンセサイザーに，ひとつの楽曲をどのように演奏するかを示す指示と考えることができる。実際のサウンドは，MIDI データを受信するシンセサイザーが生成しなければならない。MIDI プロトコルの完全な説明は，「MIDI1.0 詳細」に収められている。

MIDI データ列は，各バイトが 10 ビット（1スタート・ビット，8データ・ビット，1ストップ・ビット）で送信される，31.25K ビット / 秒の単方向非同期のビット列である。MIDI 楽器は通常，IN，OUT，THRU と表示された3つの異なる MIDI コネクターを備えている。MIDI

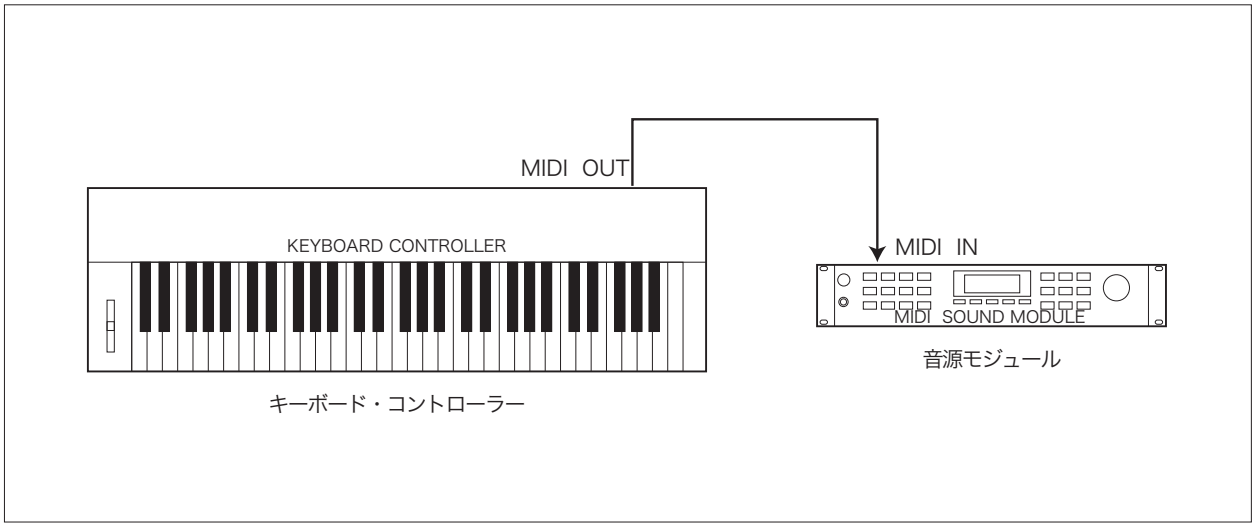
データ列は通常，鍵盤楽器のような MIDI コントローラーや，MIDI シーケンサーによって作り出される。

MIDI コントローラーは，楽器として演奏される機器で，演奏をリアルタイムに（演奏された通りに）MIDI データ列に変換する。MIDI シーケンサーは，MIDI データ･シーケンスを記録，編集，保存，結合，そして再生することができる機器である。MIDI コントローラーあるいはシーケンサーからの MIDI データ出力は，その機器の MIDI OUT から送信される。

この MIDI データ列は，通常，MIDI 音源や音源モジュールにより受け取られ，それらの受信機器は，MIDI IN から受信した MIDI メッセージに応じて，サウンドを演奏する。図1は，MIDI キーボード・コントローラーと MIDI 音源モジュールからなる単純な MIDI システムを示している。多くの MIDI 鍵盤楽器が，同一のユニットの中に，キーボード・コントローラーと MIDI 音源モジュールの機能の両方を含んでいることに注意されたい。これらのユニットでは，キーボードと音源モジュールは内部的に接続されていて，その楽器の“ローカル・コントロール”機能を ON または OFF に設定することにより，接続，または切断を切り替えられる場合もある。

物理的に1本の MIDI 信号線は,多くの MIDI チャンネル･メッセージに含まれる4ビットのチャンネル番号によって,16 の論理的なチャンネルに分割される。鍵盤楽器は通常，16 の MIDI チャンネルのうちの任意の1チャンネルで送信するように設定できる。MIDI 音源（音源モジュール）は，指定した（ひとつまたは複数の）MIDI チャンネルを受信するように設定できる。図1のシステムで音源モジュールがサウンドを演奏するためには，キーボード・コントローラーが送信しているチャンネルを，受信するように設定される必要があるだろう。

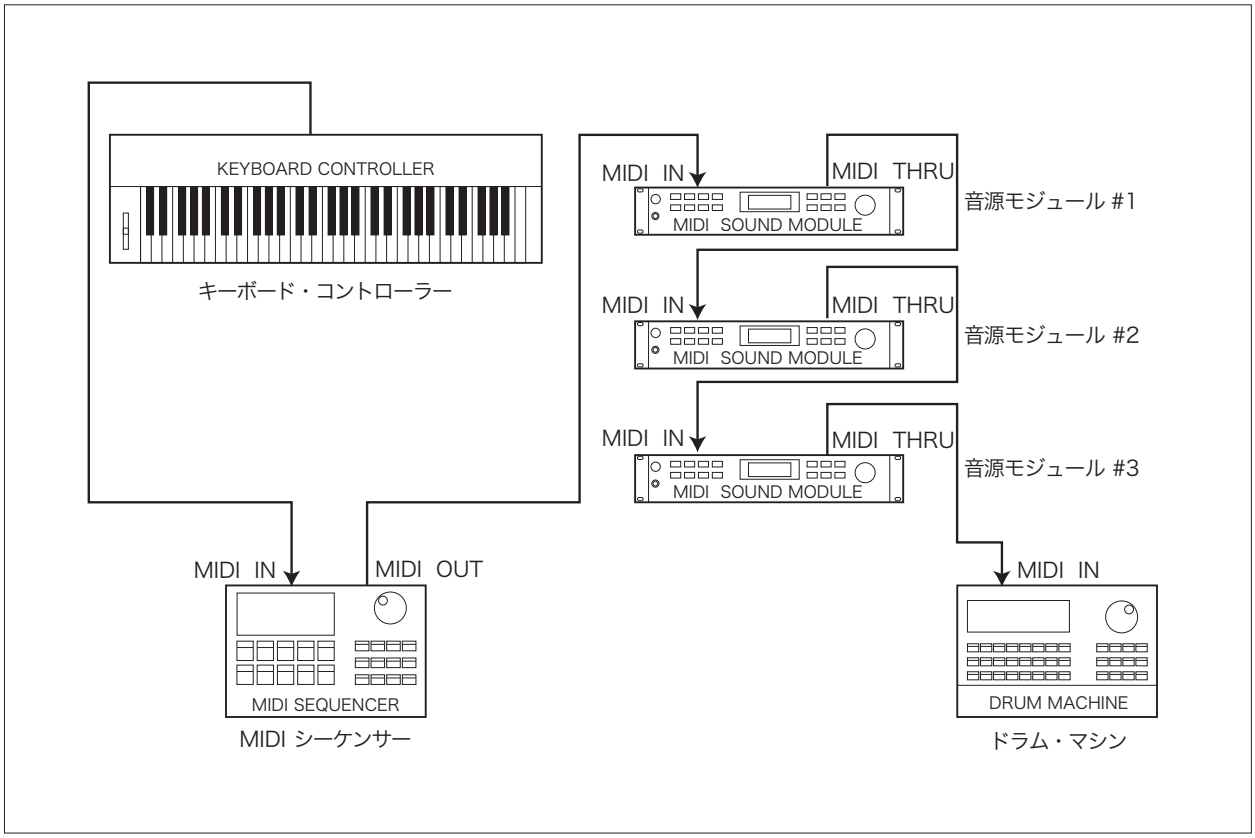
図1　シンプルな MIDI システム



MIDI 機器の MIDI IN で受信された情報は,その機器の MIDI THRU から再度送信される（繰り返される）。THRU 出力をチェーン中の後続の機器の IN に接続することによって，複数の MIDI 音源モジュールを数珠つなぎに連結することが可能である。

図2は，より複雑な MIDI システムを示している。この例では，MIDI キーボード・コントローラーは MIDI シーケンサーへの入力機器として使用され，シーケンサーの MIDI OUT ポートには，いくつかのサウンド・モジュールが接続されている。作曲者は，それぞれが異なる楽器のために書かれている，いくつかの異なるパートからなるひとつの楽曲を書くために，このようなシステムを利用することができる。作曲者は，キーボードでそれぞれのパートを1パートずつ演奏し，それら個々のパートをシーケンサーに記録することができる。そしてシーケンサーは，音源モジュールを通じて，それらのパートを合わせてプレイバックすることができる。この時，各パートは異なる MIDI チャンネルで演奏され,音源モジュールは,異なるチャンネルを受信するように設定されるであろう。例えば,音源モジュール1はチャンネル1で受信したパートをピアノ音で，モジュール2はチャンネル5で受信した情報をアコースティック・ベース音を使用して，そしてドラム・マシンは，MIDI チャンネル 10 で受信したパートを演奏するように設定することができる。

図2　拡張された MIDI システム



この例では，各パートを演奏するのに異なる音源モジュールが使用されている。しかし”マルチティンバー”の音源モジュールなら，いくつかの異なるパートを同時に演奏することができる。ひとつのマルチティンバー音源モジュールで，ピアノ・パートをチャンネル1，ベース・パートをチャンネル5，そして，ドラム・パートをチャンネル 10 で受信するように設定して，3つのパートを同時に演奏できるのである。

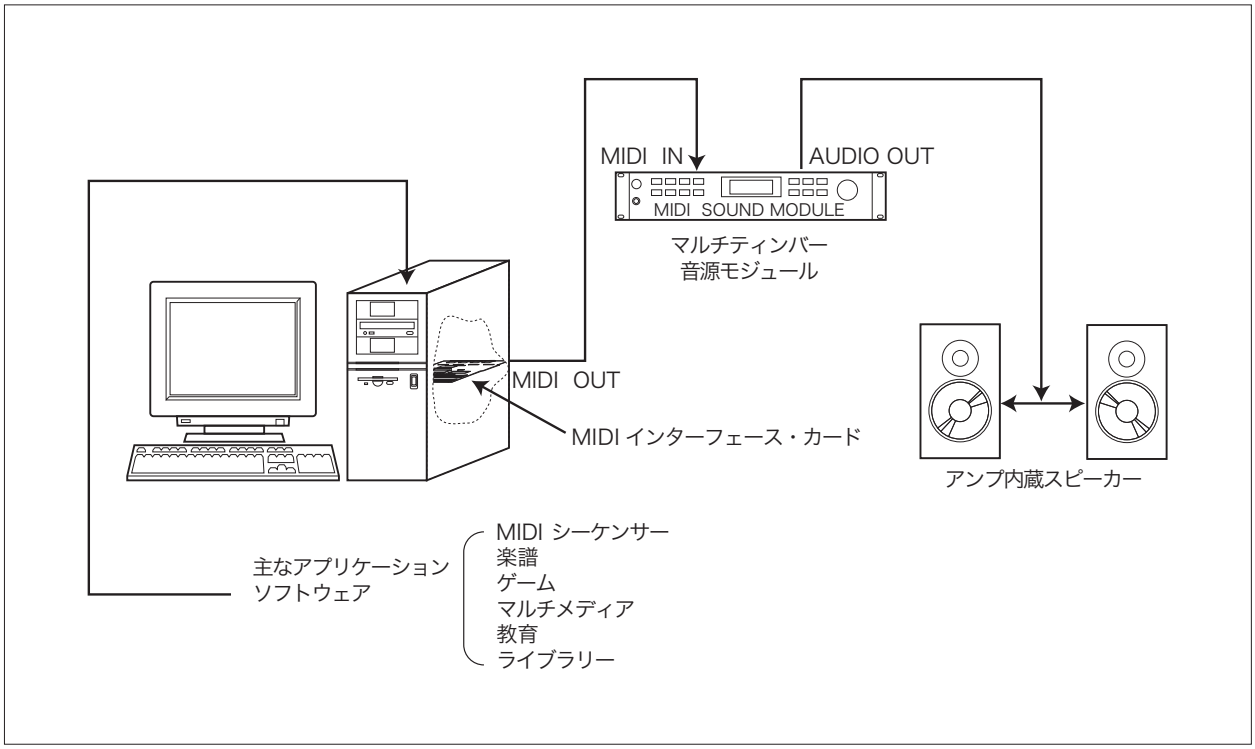
図3は，PC ベースの MIDI システムである。このシステムでは，PC は MIDI データを外部のマルチティンバー MIDI シンセサイザー・モジュールに送る，内蔵の MIDI インターフェース・カードを装着している。マルチメディア・プレゼンテーション・パッケージや教育的ソフトウェア，あるいはゲームなどのアプリケーション・ソフトウェアは，PC バスを通じて MIDI データを，パラレル形式で MIDI インターフェース・カードに送る。MIDI インターフェースは，この情報をシリアルの MIDI データに変換し，音源モジュールに送る。この音源モジュールは，マルチティンバーなので，ピアノ，ベース，そしてドラムスといった多くの異なる音楽パートを同時に演奏することができる。PC では，洗練された MIDI シーケンサー・ソフトウェア・パッケージも利用可能である。

このソフトウェアを PC で走らせ，MIDI キーボード・コントローラーを MIDI インターフェース・カードの MIDI IN ポートに接続することにより，前述の2つの例と同様な音楽作曲の作業を行うことができる。

PC ベースの MIDI システムは，数多くの異なる構成が可能である。例えば，MIDI インターフェースと MIDI 音源モジュールは，統合して PC に追加されるカードにすることができる。実際，マルチメディア PC（MPC）仕様は，すべての MPC システムがミュージック・シンセサイザーを持つことを要求しており，通常，オーディオ・アダプター・カード（サウンド・カード）には，シンセサイザーが MIDI インターフェース機能と共に含まれている。最近まで，大部分の PC サウンド・カードは，限られた能力と最低限に近い音質の FM シンセサイザーを含んでいた。

これらのシステムでは，外部にウェーブテーブル・シンセサイザー・モジュールを付け加えることで，より高音質なサウンド得られるだろう。最近では，高音質なウェーブテーブル・ミュージック・シンセサイザーを搭載した，あるいはドーターカードがオプションとして用意されている，より進んだサウンド・カードが注目されつつある。PC アプリケーションにおける MIDI プロトコル使用の増加と共に，この傾向は確実に続くであろう。

図3　PC ベースの MIDI システム



MIDI メッセージ

MIDI メッセージは，8ビットのステータス・バイトと，通常1または2の，それに続くデータ・バイトから成っている。MIDI メッセージには数多くの異なるタイプがある。MIDI メッセージは大別すると，チャンネル・メッセージとシステム・メッセージとに分類される。チャンネル・メッセージは，特定のチャンネルに適用されるメッセージで，これらのメッセージのステータス・バイトには，チャンネル番号が含まれている。システム・メッセージはチャンネルに依存しないメッセージで，ステータス・バイト中には，チャンネル情報は含まれない。

チャンネル・メッセージは，さらにチャンネル・ボイス・メッセージとチャンネル・モード・メッセージに分類することができる。チャンネル・ボイス・メッセージは，楽曲の演奏のデータを伝えるもので，典型的な MIDI データ・ストリーム中の大部分を占める。チャンネル・モード・メッセージは，受信側の楽器がチャンネル・ボイス・メッセージに応答する方法に影響する。

チャンネル・ボイス・メッセージ

チャンネル・ボイス・メッセージは，楽曲の演奏の情報を送るのに使用される。この分類に含まれるのは，ノート・オン，ノート・オフ，ポリフォニック・キー・プレッシャー，チャンネル・プレッシャー，ピッチ・ベンド・チェンジ，プログラム・チェンジ，そしてコントロール・チェンジ・メッセージである。

●ノート・オン／ノート・オフ／ベロシティ

MIDI システムにおいては，あるノートの開始と，その同じノートの離鍵とは，ふたつの独立したイベントとして考えられている。MIDI 鍵盤楽器や MIDI キーボード･コントローラーにおいて，キーが押された場合，そのキーボードは，MIDI OUT ポートからノート・オン・メッセージを送る。

キーボードは，16 の論理的な MIDI チャンネルの任意の1チャンネルで送信するように設定することができ，ノート・オン・メッセージのステータス・バイトで，選択されたチャンネル番号が示される。ノート・オン・ステータス・バイトの後には，ふたつのデータ・バイトが続

き，ノート番号（ノート・ナンバー：どのキーが押されたかを示す）とベロシティ（どのくらい強くそのキーが押されたか）を指定する。

ノート番号は，受信側のシンセサイザーにおいて演奏すべきノートを選ぶのに使用され，ベロシティは通常，ノートの音量をコントロールするのに使用される。キーが離鍵された場合，鍵盤楽器やMIDIキーボード・コントローラーは，ノート・オフ・メッセージを送る。ノート・オフ・メッセージも，ノート番号と離鍵時のベロシティのデータ・バイトを含んでいる。ノート・オフ・ベロシティ情報は，通常無視される。

●アフタータッチ

MIDI鍵盤楽器の中には，キーが押し下げられている間，キーに適用されるプレッシャーの量を感知する能力を持つものがある。このプレッシャー情報は，一般的に“アフタータッチ”と呼ばれるが，シンセサイザーが生成するサウンドの何らかの要素（例えばビブラート）をコントロールために使用することができる。キーボードが各キーに圧力センサーを持っている場合には，それによって得られる“ポリフォニック・アフタータッチ”情報をポリフォニック・キー・プレッシャー・メッセージとして送ることができる。このメッセージは，ノート番号とプレッシャー値というふたつのデータ・バイトを含んでいる。現時点の鍵盤楽器では，キーボード全体で単一のプレッシャー・レベルのみを感知する方が，より一般的である。この“チャンネル・アフタータッチ”情報は，チャンネル・プレッシャー・メッセージとして送られ，このメッセージは，プレッシャー値を指定するひとつのデータ・バイトのみを必要とする。

●ピッチ・ベンド・チェンジ

ピッチ・ベンド・チェンジ・メッセージは，通常，ピッチ・ベンド・ホイールの位置の変化に反応して送られる。ピッチ・ベンド情報は，該当するチャンネルにおいてプレイされているサウンドの音程を変化させるために使用される。ピッチ・ベンド・チェンジ・メッセージは，ピッチ・ベンド値を指定する2データ・バイトを含んでいる。2バイトが必要とされるのは，ピッチ・ベンド・ホイールの動きから生じる音程変化を，段階的ではなく，連続的と思わせるのに十分な細かい分解能を可能とするためである。

●プログラム・チェンジ

プログラム・チェンジ・メッセージは，該当するチャンネルでサウンドをプレイするのに使用されるべき楽器の種類を指定するのに使用される。このメッセージは，新しいプログラム番号を指定する1データ・バイトのみを必要とする。

●コントロール・チェンジ

コントロール・チェンジ・メッセージは，シンセサイザーにおいて，さまざまな機能をコントロールするために使用される。コントロール・チェンジ・メッセージは，他のMIDIチャンネル・メッセージと同様に，ステータス・バイトで示されたチャンネル番号のみに影響を及ぼす。

コントロール・チェンジのステータス・バイトの後には，“コントローラーの番号”を示す1番目のデータ・バイトと，“コントロール値”を指定する2番目のデータ・バイトが続く。コントロール番号は，そのメッセージによってシンセサイザーのどの機能がコントロールされるべきかを指定する。定義されているコントローラーの完全なリストは，「MIDI 1.0 詳解」に掲載されている。

バンク・セレクト

コントロール番号0（LSBを送る32と共に）は，バンク・セレクトとして定義されている。バンク・セレクト機能は，指定し得る異なる楽器音の数を拡張するために，一部のシンセサイザーにおいてMIDIプログラム・チェンジ・メッセージと共に使用されている（プログラム・チェンジ・メッセージのみでは128のプログラム番号のうちのひとつの選択が可能である）。

追加された音色は，コントロール番号0とコントロール番号32を使って新しいバンクの指定を送り，続けてプログラム・チェンジ・メッセージを送ることによって選択される。これにより，それぞれ128音を含む16,384バンクが指定可能となる。

MIDI規格は，バンク・セレクト・メッセージに対して，シンセサイザーのバンクをマッピングすべき方法を記述していないので，バンク・セレクト・メッセージにより，特定のシンセサイザーのバンクを選択する標準的な方法は存在しない。Roland（“GS”）やYAMAHA（“XG”）のように，自社製品の中で何らかの標準化を保証する，それぞれの慣例を採用しているメーカーもある。

RPN / NRPN

コントロール番号 6(データ・エントリー)は，コントロール番号 96(データ・インクリメント)，97(データ・デクリメント)，98(レジスタード・パラメーター番号 LSB)，99(レジスタード・パラメーター番号 MSB)，100(ノンレジスタード・パラメーター番号 LSB)，そして 101(ノンレジスタード・パラメーター番号 MSB)と組み合わせて，MIDI で利用可能なコントローラーの数を拡張する。最初にコントロール番号 98 と 99，あるいは 100 と 101 を使用してエディットされるべきパラメーター番号を選択し，それからコントロール番号 6，96，あるいは 97 を使用してそのパラメーターのデータ値を調節することにより，パラメーター・データが送信される。

RPN と NRPN は，音色パッチや他のデータをエディットするために，シンセサイザーにパラメーター・データを送るのに使用されるのが典型的である。レジスタード・パラメーターは，MIDI Manufacturers Association(MMA)と MIDI 規格協議会(JMSC：現在 AMEI)によって何らかの特定の機能を割り当てられたものである。例えば，シンセサイザーのピッチ・ベンド・センシティビティやマスター・チューニングをコントロールするよう割り当てられたレジスタード・パラメーター番号がある。ノンレジスタード・パラメーター番号は，特定の機能を割り当てられておらず，メーカーごとに異なる機能に使用してよい。ここでも，Roland や YAMAHA を含む各社は，何らかの標準化を保証するために，それぞれの慣例を採用している。

チャンネル・モード・メッセージ

チャンネル・モード・メッセージ(MIDI コントロール番号 120 ～ 127 に該当)は，シンセサイザーが MIDI データに反応する方法を変更する。コントロール番号 121 は，すべてのコントローラーをリセットするのに使用される。コントロール番号 122 は，ローカル・コントロール(鍵盤付きの MIDI シンセサイザーにおいて，キーボード・コントローラー機能とシンセサイザー機能を，ローカル・コントロールをオフにすることによって分離することができる)のオン / オフの選択に使用される。コントロール番号 124 から 127 は，オムニ・モードのオンとオフの選択，そしてモノ・モードとポリ・モードの選択に使用される。

オムニ・モードがオンの場合，シンセサイザーは，入ってくる全チャンネルの MIDI データに応答する。オムニ・モードがオフの場合，シンセサイザーは，ひとつのチャンネルの MIDI メッセージのみに応答する。ポリ・モードが選択されている場合，入ってくるノート・オン・メッセージは，ポリフォニックで演奏される。これは複数のノート・オン・メッセージが受信された場合，各ノートにそれぞれ別のボイスが割り当てられる(そのシンセサイザーで利用可能なボイス数に制限されるが)ことを意味している。その結果，複数のノートが同時に演奏される。モノ・モードが選択されている場合には，MIDI チャンネルごとにひとつのボイスが割り当てられる。これはそのチャンネルで同時にひとつのノートしか演奏されないことを意味する。MIDI シンセサイザーの多くは，オムニ・オン / ポリ・モードの動作が初期設定になっているであろう(現在はオムニ・オフ)。このモードでは，シンセサイザーは受信したすべての MIDI チャンネルのノート・メッセージを演奏し，それぞれのチャンネルで受信されたノートは，ポリフォニックで演奏される。オムニ・オフ / ポリ・モードの動作では，シンセサイザーはひとつのチャンネルのみ受信し，そのチャンネルで受信されたノートは，ポリフォニックで演奏される。このモードは，何台かのシンセサイザーが MIDI THRU を使用して数珠つなぎにされている時に役立つであろう。この場合，チューン中の各シンセサイザーは，ひとつのパート(ひとつのチャンネルの MIDI データ)だけを演奏し，他のパートは無視するように設定できる。

MIDI 楽器には“ベーシック・チャンネル”という MIDI チャンネルがひとつあることに注意しなければならない。ベーシック・チャンネルの割り当ては，ハードで固定されている場合も，選択可能な場合もある。モード・メッセージは，ベーシック・チャンネルでのみ受信される。

システム・メッセージ

システム・メッセージは，システム・コモン・メッセージ，システム・リアルタイム・メッセージとシステム・エクスクルーシブ・メッセージに分類される。システム・コモン・メッセージは，システム中のすべての受信機に対して送られる。システム・リアルタイム・メッセージは，MIDI クロックを使用した機器同士の同期を取るために使用される。システム・エクスクルーシブ・メッセージは，メーカーの ID コードを含み，

そのメーカーによって指定されたフォーマットにしたがい，任意のバイト数のデータを送信する時に使用する。

●システム・コモン・メッセージ

現在定義されているシステム･コモン･メッセージには，MTC クォーター･フレーム，ソング･セレクト，ソング･ポジション･ポインター，チューン・リクエスト，そしてエンド・オブ・エクスクルーシブ（EOX）がある。MTC クォーター・フレーム・メッセージは，MIDI 機材とオーディオまたはビデオ・デッキのような，他の機材との同期を取るのに使用する MIDI タイムコード情報の一部分である。

ソング・セレクト・メッセージは，複数の異なる曲を保存したり読み出したりするシーケンサーやドラム・マシンのような MIDI 機器で使用される。ソング・ポジション・ポインターは，曲の先頭以外のある場所から演奏をスタートするように，シーケンサーに指示する時に使用する。ソング・ポジション・ポインターの値は，曲の先頭からスタートさせたい場所までの時間を，MIDI クロック数で表した値である。このメッセージは，システム・リアルタイム・メッセージ（MIDI 同期）を認識する機器同士でのみ使用することができる。

チューン･リクエスト･メッセージは，一般にアナログ･シンセサイザー内部の発振器の再チューンを要求する時に使用される。このメッセージは，一般にデジタル・シンセサイザーには必要ない。

EOX メッセージは，可変長データを含むことができるシステム・エクスクルーシブ・メッセージの終了を知らせるために使用される。

●システム・リアルタイム・メッセージ

システム・リアルタイム・メッセージは，シーケンサーやドラム・マシンなど，MIDI クロックを使用するシステム内のすべての機器の同期を取るために使用される。鍵盤楽器やシンセサイザーは，通常システム・リアルタイム・メッセージの大部分を無視する。正確なタイミングを保証する手助けとして，システム・リアルタイム・メッセージは，他のメッセージよりも優先権が与えられており，これらの1バイトのメッセージは，データ列の任意の場所に入れることができる（システム・リアルタイム・メッセージは，MIDI メッセージのステータス・バイトとそれに続くデータ・バイトの間に現れてもよい）。

システム・リアルタイム・メッセージには，タイミング・クロック，スタート，コンティニュー，ストップ，アクティブ・センシング，そしてシステム・リセット・メッセージがある。タイミング・クロック・メッセージは，シーケンサーの演奏のテンポを設定するマスター・クロックである。タイミング・クロック・メッセージは，4分音符あたり 24 回送られる。スタート，コンティニュー，そしてストップの各メッセージは，シーケンサーの演奏を制御するために使用される。

アクティブ・センシング信号は，MIDI シーケンスの再生中に MIDI ケーブルが外れてしまった時に起こるかもしれない“音残り”を防ぐ手助けとして使用される。アクティブ・センシングを使わない場合，演奏中にケーブルが外れたなら，ノート・オン・メッセージにより発音したノートが，対応するノート・オフ・メッセージが送られてこないために，永遠に鳴り続けてしまうことが起こりうる。

システム・リセット・メッセージは，その名称が意味するように，このメッセージを受信するすべての機器をリセットし，初期化する時に使用される。このメッセージは，一般に送信側の機器によって自動的に送られることはなく，ユーザーが手動で実行しなければならない。

●システム・エクスクルーシブ・メッセージ

システム・エクスクルーシブ・メッセージは，パッチ・パラメーターやサンプル・データのようなデータを MIDI 機器間で送る時に使用される。MIDI 機器のメーカーは，システム･エクスクルーシブ･データのための独自のフォーマットを定義できる。メーカーには，JMSC（現在 AMEI）あるいは MMA によって独自の ID が与えられており，そのメーカー ID 番号がシステム・エクスクルーシブ・メッセージの一部分として現在使用される。メーカー ID の後に任意のサイズのデータが続き,送信データは EOX メッセージで終了する。メーカーは，システム・エクスクルーシブ・データ・フォーマットの詳細を公表することが要求され，他のメーカーはそのフォーマットを自由に利用する

ことが許されるが，そのフォーマットを改変したり，元のメーカーの仕様と矛盾する方法で利用してはいけないと規定されている。

若干のシステム・エクスクルーシブ ID 番号が，特別なプロトコルのために予約されている。これらの中には，MIDI ショー・コントロールや MIDI マシン・コントロール，そして MIDI 機器間でサンプル・データを送信するために MIDI 規格でシステム・エクスクルーシブ・データ・フォーマットが定義されている，サンプル・ダンプ・スタンダードがある。

ランニング・ステータス

MIDI データはシリアルで送信されるので，元来は同時に起こった，MIDI データ列の中で同時に送られるべき複数の音楽イベントが，実際，正確に同時には演奏されない可能性がある。MIDI データの1バイトは 10 ビットからなり，その転送レートは 31.25K ビット / 秒なので，3バイトからなるノート・オン / ノート・オフ・メッセージを送信するには約1ミリ秒かかり，これは一般に複数のイベントが同時に起きたと感じるのに十分なほど短かい時間である。実際，MIDI 鍵盤楽器を演奏している人にとって，10 個のキーを同時に押した時に，それらのノートを演奏するのにかかる時間差は 10m 秒を越えないはずであり，これは知覚されないであろう。

しかし，シーケンサーから送られる MIDI データは，たくさんのパートを含むことができる。拍によってはかなりたくさんの同時に起こるべき音楽イベントが存在する可能性があり，そのデータをシリアル化することによって引き起こされる遅延は，気が付くほどになるかもしれない。送信される MIDI データ列のデータの量を減らす手助けとして，“ランニング・ステータス”と呼ばれるテクニックが利用されることがある。

ランニング・ステータスは，連続したメッセージが同じタイプのメッセージであることが非常に多いという事実を考慮したものである。例えば鍵盤で和音を弾いた場合，10 個の連続したノート・オン・メッセージが生成され，その後に 10 個のノート・オフ・メッセージが続くだろう。ランニング・ステータスを使用した場合，ステータス・バイトは，そのメッセージが最後に送られたメッセージと同じチャンネル，同じタイプでない時だけ送られる。同じタイプの連続したメッセージのステータス・バイトは，省略してもよい（後続のメッセージは，データ・バイトのみが送信される）。

ランニング・ステータスの効果は，ノート・オフ・メッセージの代わりにベロシティ値が0のノート・オン・メッセージを送ることにより，高められる。この場合，ノート・オン・メッセージがかなりの長さで連続することがしばしばあるだろう。いくつかの MIDI コントローラーの変化や，楽器のピッチ・ベンド・ホイールの動きは，莫大な数の MIDI チャンネル・ボイス・メッセージを発生することがあり，ランニング・ステータスは，これらの場合にも大いに助けとなる。

MIDI シーケンサーとスタンダード MIDI ファイル

MIDI メッセージは，MIDI シンセサイザーによってリアルタイムで受信され，処理される。シンセサイザーが“ノート・オン”メッセージを受信すると，それにふさわしい音を発音する。それに相応する“ノート・オフ”メッセージを受信すると，シンセサイザーは，そのノートを消す。MIDI データの送信元が鍵盤楽器である場合には，MIDI メッセージと共にタイミング情報を送る必要はない。

しかし，MIDI データをデータ・ファイルとして保存したり，またはシーケンサーを使って編集する場合には，MIDI メッセージに対し，なんらかの形で“刻時”する必要がある。スタンダード MIDI ファイル規格は，タイミング情報が付いた MIDI データを扱うための標準化された方式を提供する。この標準化されたタイミング情報付き MIDI データのためのファイル・フォーマットは，シーケンサーや譜面ソフト，そしてマルチメディアを使用したプレゼンテーション・ソフトなどの異なるアプリケーションで，MIDI データ・ファイルを共有することを可能にしている。

スタンダード MIDI ファイル規格は，MIDI ファイルに3つのフォーマットを定義している。MIDI シーケンサーは，一般に複数の MIDI データ列である“トラック”を管理することができる。フォーマット0を使用したスタンダード MIDI ファイルは，すべての MIDI デー

タをひとつのトラックに収める。フォーマット1では，トラックの集合体として MIDI データを収める。フォーマット2では，複数の独立したパターンを収めることができる。フォーマット2は，MIDI シーケンサーでは通常，音楽用には使用されない。MIDI シーケンサーの大部分は，フォーマット0かフォーマット1のどちらかのスタンダード MIDI ファイルを読み込むことができる。フォーマット0のファイルの方がおそらく小さく，保管スペースを節約できる。そしてフォーマット1より少しだけ少ない情報量でデータを転送することができるだろう。しかし，フォーマット1のファイルは，より直接的に眺めたり編集したりできるので，一般にこちらの方が好まれる。

シンセサイザーの基礎

ポリフォニック（発音数）

音源装置の発音数は，複数のノートを同時に演奏することができる能力を示す。発音数は，一般にノートまたはボイスを単位として数えられる。初期の大部分のシンセサイザーは，モノフォニック，すなわち一度にひとつのノートしか演奏できなかった。モノフォニックのシンセサイザーの鍵盤を同時に5つ押さえても，ひとつのノートしか発音されないだろう。4音ポリフォニックのシンセサイザーの鍵盤を同時に5つ押さえると，通常は4つのノートが発音される。もしそのシンセサイザーがもっとたくさんのボイスを持っているなら（最近の多くのモジュールは 16，24 または 32 音ポリフォニックである），5つのノートすべてが発音される。

音色

シンセサイザーや音源装置が作り出す，それぞれの異なる音色のことをしばしば“パッチ”，“プログラム”，“アルゴリズム”または“ティンバー”と呼ぶことがある。プログラムが可能なシンセサイザーは普通，それぞれの音色に“プログラム番号”（パッチ番号）を割り当てている。例えば，ある音源モジュールは，パッチ番号1にアコースティック・ピアノの音色を，そしてパッチ番号 36 をフレットレス・ベースの音色に使っているかもしれない。すべての音色に対するすべてのパッチ番号の組み合わせは，しばしばパッチ・マップと呼ばれる。

指定チャンネルで受信している機器に対して，使用される楽器音色を変更するために，MIDI ではプログラム・チェンジ・メッセージが使用される。例えば，チャンネル4の機器をフレットレス・ベースの音色で演奏するように設定するために，シーケンサーはチャンネル4で，プログラム 36（これは GM のフレットレス・ベースのプログラム番号である）を指定したプログラム・チェンジを送ることができる。

マルチティンバー・モード

シンセサイザーや音源装置は，2つまたはそれ以上の異なる楽器の音色を同時に作り出す能力を持っている場合には，マルチティンバーであると言われる。もしあるシンセサイザーが同時に5つのノートを演奏することができ，そしてピアノの音色とアコースティック・ベースの音色を同時に作り出すことができれば，それはマルチティンバーである。十分な発音数とパート（マルチティンバー）数を持っていれば，1台のシンセサイザーでバンドやオーケストラの全体の音色を作り出すことも可能である。

マルチティンバー処理には，さまざまな MIDI メッセージを送るために，一般にシーケンサーを使用する必要があるだろう。例えばシーケンサーは，ピアノ・パートの MIDI メッセージをチャンネル1に，ベースをチャンネル2，サックスをチャンネル3，ドラムスをチャンネル10... と送ることができる。16 パートのマルチティンバー・シンセサイザーでは，MIDI の 16 論理チャンネルでそれぞれ異なるパートを受信することができるだろう。

普通，マルチティンバー・シンセサイザーの全ボイスは，使用される異なるパート（ティンバー）の間で動的に使用される。ある瞬間には5ボイスがピアノ・パートに，2ボイスがベースに，1ボイスがサックスに，そして6ボイスがドラムスに必要とされる，などである。シンセサイザーや音色によっては，実際にはひとつのノートでひとつ以上の“ボイス”を使用することがあり，音色によっては同時に発音する音数がシンセサイザーのポリフォニック数よりも少なくなってしまうことがあることに注意を要する。

General MIDI（GM）システム

普通，MIDI シーケンス・データの始めには，その曲で使用するそれぞれのパートに適切な楽器の音色を設定するために，それぞれのチャンネルにプログラム・チェンジ・メッセージを送る。プログラム・チェンジ・メッセージは，シンセサイザーに，その MIDI チャンネルでどのパッチ番号を使用すべきかを教える。もし，シンセサイザーがその MIDI シーケンス・データを作成した時に使用したものと同じパッチ・マップ（パッチ番号に対する音色の割り当て）を使って MIDI シーケンス・データを受信すれば，音色は意図した通りになるだろう。

GM以前には，パッチ番号とシンセサイザーの特定の音色との関係に標準はなかった。このため MIDI シーケンス・データは，異なるシンセサイザー上では，そのシンセサイザーが類似の音色を持っていたとしても，異なる音色で再生されることがあった。例えば，作成者がチャンネル1にエレクトリック・ピアノの音色を意図してパッチ番号5を選択したとしても，MIDI データを演奏するシンセサイザーのパッチ番号5にチューバが割り当てられていると，このシンセサイザーでは（たとえこのシンセサイザーが他のパッチ番号にすばらしいエレクトリック・ピアノの音を持っていたとしても）ピアノを意図した音がチューバで演奏されてしまうだろう。

General MIDI（GM）は，GM機器が持つべき一般的機能を定義している。GMにはGMサウンド・セット（パッチ・マップ），GMドラム・マップ（ノート番号への打楽器の音色の割り当て），そしてGMに必要な性能（ボイス数，認識する MIDI メッセージの種類など）の定義も含まれている。GM楽器用に作られた MIDI シーケンス・データは，他のいかなるGMシンセサイザーや，サウンド・モジュールでも正しく演奏されるはずである。

GMシステムでは，MIDI チャンネルの1から9と 11 から 16 は，音階のある楽器の音色のために使用し，チャンネル 10 は“キーに割り当てられた”打楽器の音色に使用する。これらの楽器の音色は，関連音色を組にしてグループ分けされている。例えば1から8はピアノ系の音色，9から 16 は音階のある打楽器系の音色，17 から 24 はオルガン系の音色，25 から 32 はギター系の音色，のようになっている。

チャンネル1から9と 11 から 16 の楽器の音色では，ノート・オン・メッセージのノート番号は，演奏される音色のピッチを選択するのに使用される。例えば，チャンネル3でビブラフォンの音色（プログラム番号 12）が選択されている場合，チャンネル3のノート番号 60 では，中央のCの音が演奏され（これはほとんどの楽器においてノート番号とピッチの対応の基準である），チャンネル3のノート番号 59 では，中央Cのすぐ下のBが演奏されるだろう。両方のノートともビブラフォンの音色で演奏される。

GMの打楽器の音色は，チャンネル 10 に設定される。これらの“キーに割り当てられた”音色では，ノート・オン・メッセージのノート番号は，別の意味で用いられる。チャンネル 10 のノート番号は，どのドラム音色を演奏するのかを選択するのに使用される。例えば，チャンネル 10 のノート番号 60 のノート・オン・メッセージは“ハイ・ボンゴ”のドラム音色で演奏される。チャンネル 10 のノート番号 59 は，“Ride Symbal2”の音色で演奏される。

GMシステムは，プログラム番号1から 128 を使用して音色を定義していることに注意を要する。これらの音色を選択するために使用する MIDI プログラム・チェンジは，8ビットからなる1バイトを使い，それに対応する 10 進数の0から 127 で希望するプログラム番号を指定する。したがって，GM音色番号 10 の“グロッケン”を選択するためには，プログラム・チェンジ・メッセージのデータ・バイトの値は，10 進数の9となる。

GMシステムは，それぞれのプログラム / パッチ番号に対応する楽器または音色を規定しているが，GMはそれらの音色をどのようにして作り出すのかは規定していない。したがって，プログラム番号1は，どのGM楽器でもアコースティック・グランド・ピアノが選択されるはずであろう。しかし，アコースティック・グランド・ピアノの音色は，異なる音源技術を使用したふたつのGMシンセサイザーでは，明らかに異なる音色になるだろう。

音源技術（FM 方式とウェーブテーブル方式）

シンセサイザーが音を作り出すために，多くの技術やアルゴリズムが使用されている。中でも，周波数変調（FM）方式とウェーブテーブル方式というふたつの技術は，幅広く使用されているものである。

一般的にFM方式は，周期的な信号（モジュレーター）を，別の信号（キャリア）の周波数を変調するために使用する。変調する信号が可聴範囲内ならば，その結果として，キャリア信号は著しい音色変化となるだろう。各FMボイスは，最小限2つの信号発生器を必要とする。これらの信号発生器は，通常“オペレータ”と呼ばれ，FM方式の音源の構成が異なれば，オペレータのパラメーターによるコントロールの度合いもさまざまである。

FM方式の音源では，ボイスごとに4ないし6のオペレータを使用し，各オペレータは，アタック・レイトやディケイ・レイト等の調節ができるエンベロープを持つ場合もある。初期のFM方式はアナログ回路であったが，現代のFM方式はデジタルで実現されている。

FM方式は，表現力に富んだ新しい音を作り出すのには，とても有効である。しかし，音源の目的が既存の楽器音を再現することであるならば，デジタル・サンプリングによる方式を用いることで，より正確に行うことが可能である。

デジタル・サンプリング音源は，音をデジタル方式による高品質なサンプル録音を行い，これらの音を要求に応じて再生する。デジタル・サンプルを元にした音源は，音のサンプルをストアしておくのに必要なメモリーの量を減らすため（あるいは，与えられた量のメモリーから，より多くの種類の音を得るために），ループ機能，ピッチ・シフト，補間，そしてデジタル・フィルターといったさまざまな技術を利用している。これらのサンプリング方式の音源は，しばしば“ウェーブテーブル”シンセサイザーと呼ばれる（これらの音源のサンプル・メモリーは，多数のサンプルされた音の部分から成り，必要な時に参照され，利用されるサウンド波形の“テーブル”と考えることができる）。

ウェーブテーブル方式

今日の本格的なシンセサイザーの多くは，何らかのサンプルされた音を利用する，ウェーブテーブル方式を使用している。さまざまな分野で求められている音源のトレンドは，ウェーブテーブル方式に向かっている。ここでは，この方式の音源に利用されているいくつかの技法について，以下に記しておく。

●ループ機能とエンベロープ

ウェーブテーブル方式では，サンプル・メモリーの容量を節約するための重要な技法のひとつとして，サンプルされた音の断片をループさせている。多くの楽器音は，2つの部分（アタック部分とサスティン部分）にモデル化される。アタック部分は，楽器音の最初の部分であり，そこでは楽器音の音量と音色の特徴は，とても急速に変化しているだろう。楽器音のサスティン部分は，アタック部分に続く部分で，そこにおける音の特徴の変化は，それほどダイナミックではない。

図4は，アタック部分とサスティン部分において考えられる波形を図示したものである。この例では，波形の特徴は，サスティン部分を通じて変わらず残っているが，その一方，音量はかなり一定の割合で減少している。これは誇張した例であり，ほとんどのアコースティック楽器音においては，音の持続と共に，波形の特徴と音量の両方が変化し続ける。サスティン部分は，それが識別されるならば，音の特徴が比較的一定である部分である。

図4　ウェーブフォームのアタック部分とサスティン部分

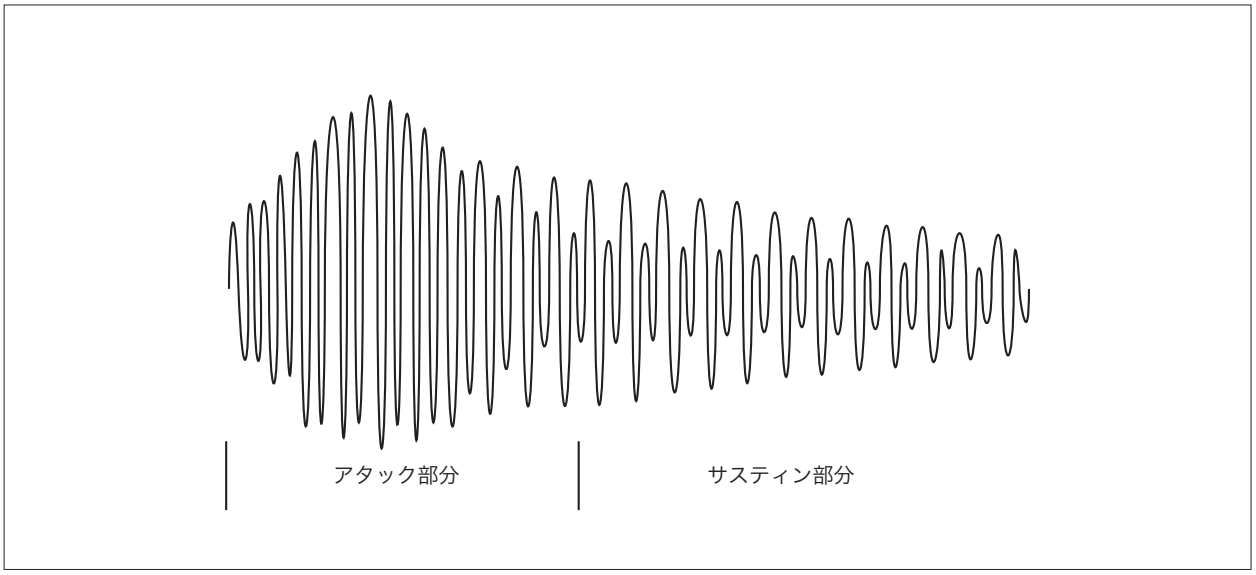
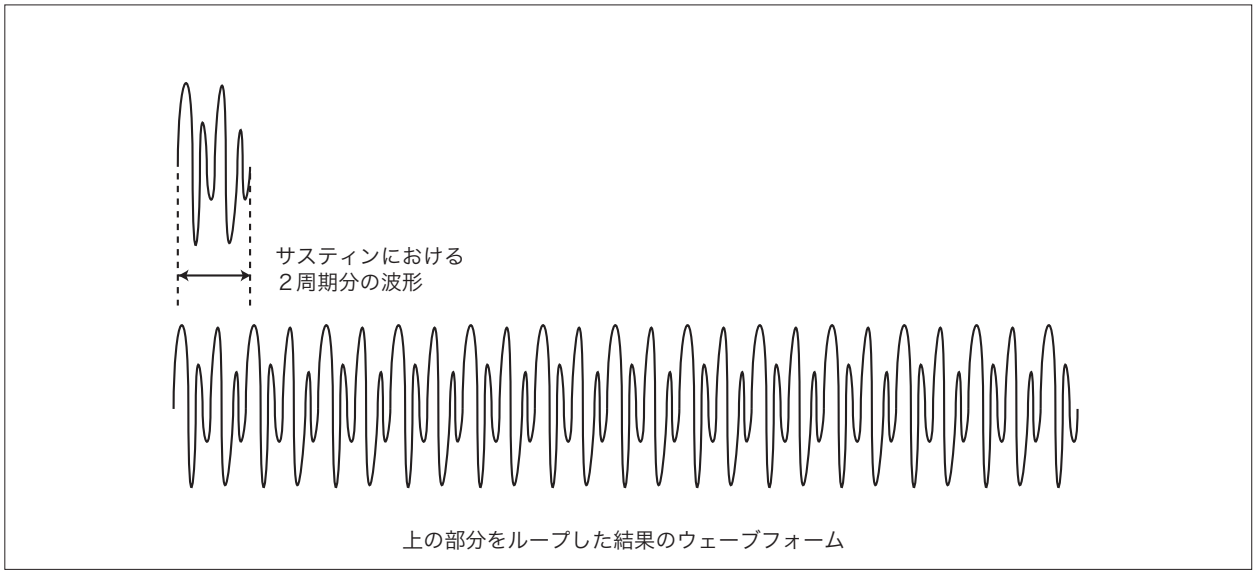


図5　サンプル部分のルーピング



ウェーブテーブル方式の音源では，サスティン部分の短い波形の部分のみを持ち，再生時にこの部分をループさせることによって，相当量のメモリーを節約することができる。図5は，図4の波形におけるサスティン部分の2周期の波形から，ループによって，一定周期の信号を作り出したものを示している。オリジナル波形の特徴と音量が，サスティン部分においてかなり一定している場合には，ループ動作によって生じた音は，オリジナルのサスティン部分の音にとても類似したものになるはずである。

多くのアコースティック弦楽器における波形の特徴は，サスティン部分ではかなり一定のままであるが，一方，音量は減衰している。これをシミュレートするには，サスティン部分に必要とされるシェイプ，すなわちエンベロープによって減衰の要素を加え，ループ再生を何度も繰り返すことで可能となる。音の音量エンベロープは,一般的にいくつかの直線区間から構成されるものとしてモデル化される。例は，一般的に使用されているもので，4つの各区間が直線のアタック - ディケイ - サスティン - リリース（ADSR）エンベロープのモデルである。図6は，典型的な ADSR エンベロープの形状を描いたもので，図7は図5からのループされた波形に，このエンベロープを当てはめた結果を示している。

図6　典型的な ADSR 音量エンベロープ

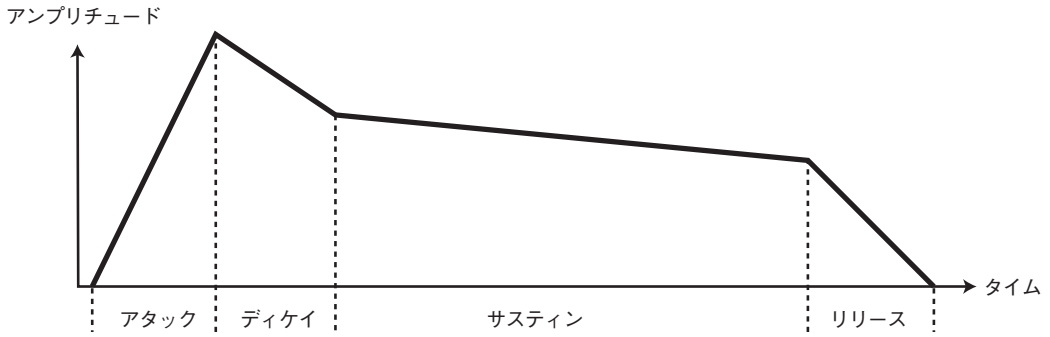
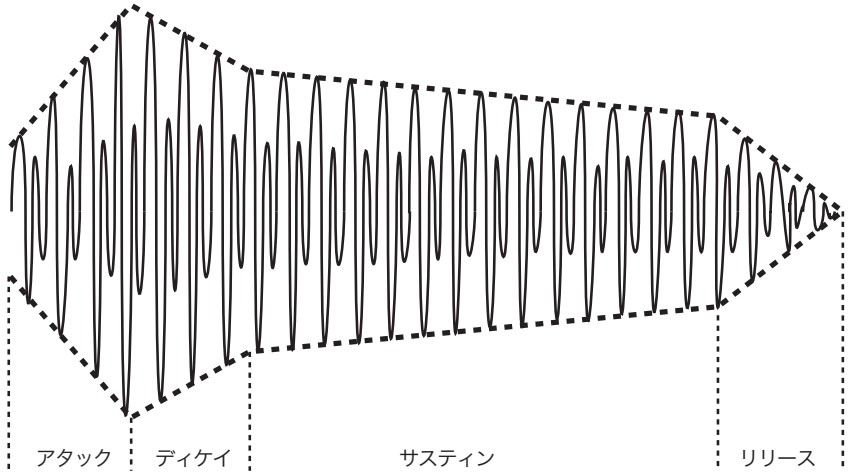


図7　ループ再生されたサンプルに適用された ADSR エンベロープ



典型的なウェーブテーブル方式は，ひとつの楽器音に対し，アタック部分とループ部分のためのサンプル・データを持っている。これらのサンプル・データは，イニシャル音とループ音などと呼ばれて区別されている。イニシャル音は，一度だけ通しで演奏され，続いてループ音がそのノートが終了するまで繰り返し再生される。エンベロープ・ジェネレーターの機能は，適切な楽器音のエンベロープを作り出すことで，このエンベロープは再生中のサンプル出力に適用される。

波形のイニシャル部分の再生（エンベロープのアタック部分が適用される）は，ノート・オン・メッセージが受信された時に開始される。イニシャル部分のサンプル波形の長さは固定されていて，そのエンベロープのアタック部分とディケイ部分の長さもまた，与えられた楽器音ごとに固定されているのが一般的である。

サスティン部分は，サスティン・エンベロープのスロープ（この例ではゆっくりと減衰する）が適用され，ノート・オフ・メッセージが認識されるまで，サンプルのループ再生を繰り返し続けるだろう。ノート・オフ・メッセージは，エンベロープのリリース部分をスタートさせる。

●ループ長

ループ長は,サンプル数で与えられる。演奏される音のループの長さは,もとのピッチの周期の整数倍である必要がある（そうでなければ，ループ再生が開始された時に望ましくない“ピッチ・シフト”が生ずるだろう）。実際，アコースティック楽器のサンプルのループ部分の長さは，元音のピッチに対して何周期分もあるかもしれない。その音が自然なビブラートやコーラス効果を持っている場合は，一般に，ループ長をビブラートやコーラスの周期の整数倍にすることが望まれる。

●ワン・ショット・サウンド

前項では,サンプル音をアタック部分とサスティン部分に分割し,サンプル･メモリーを最小化するために,サスティン部分にループ再生の技法を用いることについて述べた。しかし，特に持続時間の短いサウンドやダイナミックに変化するサウンドなど，音によってはループ再生がふさわしくない。短いドラム音などの短い音では，ループ再生しないで，通しで一度だけ再生されるシングル・サンプルとしてストアされる。これらの音は，“ワン・ショット”と呼ばれる。

●サンプル音の編集と処理

ウェーブテーブル方式の音源で使用する，サンプルされた音の準備に際して，多くのサンプル編集と処理段階がある。オリジナルのサンプル・データを編集し，イニシャル部分とループ部分を識別して取り出すことの必要性については，前述の通りである。

編集機能は，ループ部分のエンド・ポイントを適合させるためにも必要とされる。ループ部分の先頭と終わりにおいて，波形の振幅とスロープが一致しないと，ループ再生のたびに“グリッチ”が聴こえるであろう。信号 / 量子化ノイズ比を改善したり，あるいはサンプル・メモリーを節約する目的で，サウンドのダイナミック・レンジを“圧縮”するために，それ以上の処理が行われる場合もある。この話題は次に述べる。

シンセサイザーのサンプル・メモリーには，サンプル処理を施した，さまざまな楽器のサンプル音がマッピングされている。

●サンプル・データの圧縮

デジタルでサンプルされた信号の量子化ノイズは，サンプル点のサイズ（サンプルごとのビット数）と，デジタル化された信号のアンプリチュードによって決まる。多くのアコースティック楽器音は，その最大音量に素早く到達し，その音量ピークからゆっくりと減衰していく。耳は，ダイナミックに信号レベルを感知している。比較的小さなサンプル・サイズを使う楽器でも，量子化ノイズ・レベルは，信号が最大音量に近い場合には知覚できない。しかし，信号レベルが減衰するにしたがって，耳はより敏感になり，知覚できるノイズ・レベルは増大する。もちろん，もっと大きなサンプル・サイズを使用すれば，量子化ノイズを減らすことができるだろうが，サンプル数が多くなれば，相当な価格上昇となる。

データの圧縮は，サンプルされた音の量子化ノイズを軽減するために使用される。これにより，サンプル・メモリーに記録されるサンプル音のダイナミック・レンジを縮める。この信号のダイナミック・レンジを元に戻すため，サンプル・データの圧縮は，再生中に解かれる。これにより，小さなサンプル・サイズ（ダイナミック・レンジ）で使用するサンプル・メモリーでも，同じサンプル・サイズの他の機器より，大きなダイナミック・レンジが得られる。信号のダイナミック・レンジを圧縮するために，数多くのさまざまな圧縮方法が使用されている。

前述したループ機能にも，ある種の圧縮効果がある。ループ部分は，サンプル・メモリーで利用可能なダイナミック・レンジを最大限に利用した音量レベルで記録される。再生に使用されるプロセッサーとD/Aコンバーターが，サンプル・メモリーよりも広いダイナミック・レンジを持っている場合，再生中にディケイ・エンベロープを適用することで，前述のようなデコンプレッション効果が得られるだろう。

●ピッチ・シフト

サンプル・メモリーの必要量を最小化するために，ウェーブテーブル方式の音源は，ピッチ・シフト，つまりピッチ変換を，ひとつの楽器音のサンプルから多くの異なるピッチのノートを生成するために利用する。例えば，サンプル・メモリーがアコースティック・ピアノの中央Cのノートのサンプルを持つ場合，同じサンプル・データからピッチ・シフトを使用して，中央Cの上のC＃やDを生成するのである。

ピッチ・シフトは，記録されたサンプル・データを，記録時とは異なるレートで再生することで実現されている。例えば，ある音のサンプル・メモリーのアドレッシングにポインターが使用されているとして，そのポインターがメモリーを読み出すたびにひとつずつ増加する場合，このサンプル音は連続的に読み出され，ある特定の音程で演奏される。このポインターの増加が1ではなく2の場合は，サンプルはひとつおきに読み出され，その結果として音程は1オクターブ上にシフトされる（周波数が2倍になる）だろう。

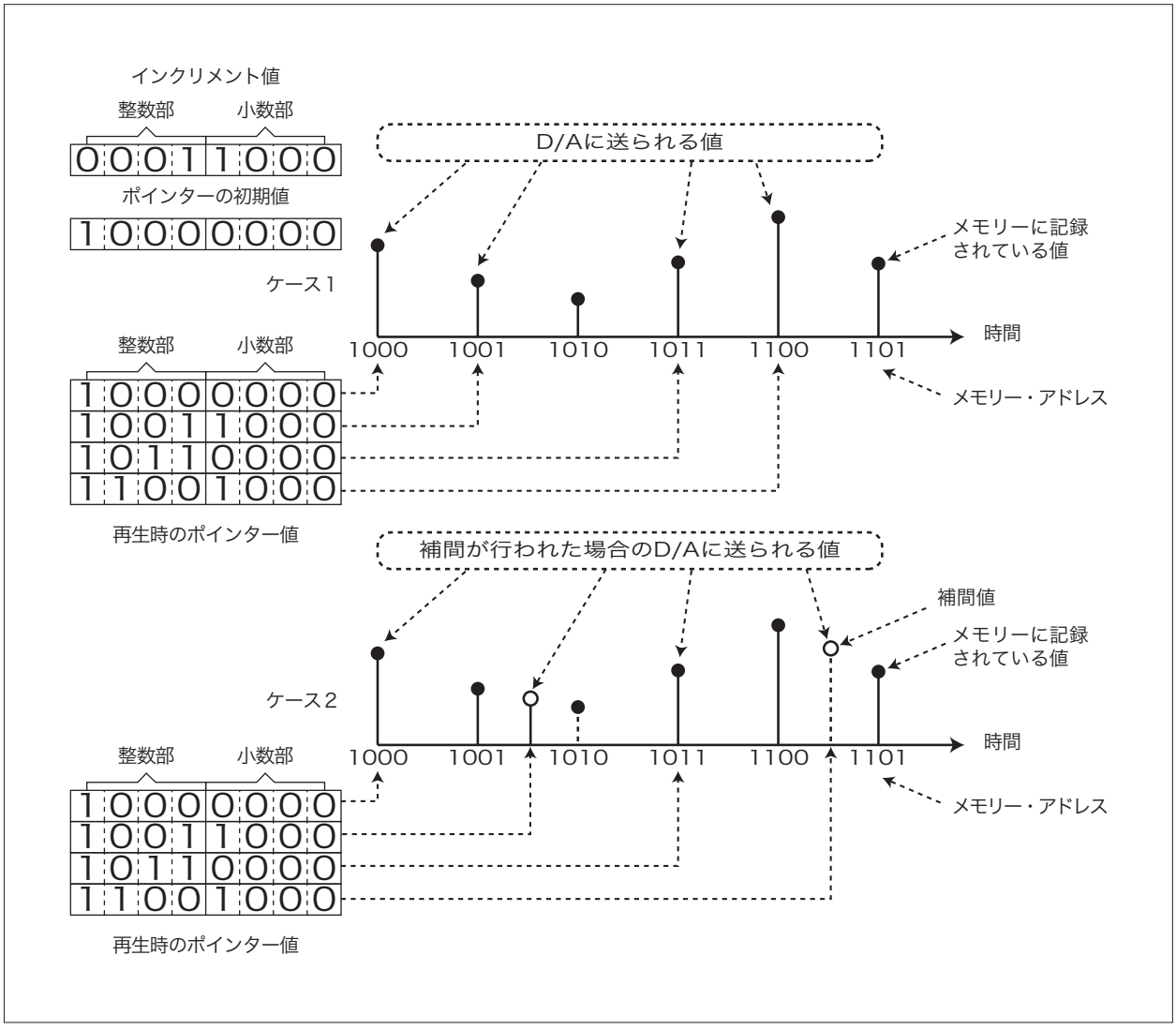
前例では，サンプル・メモリーのアドレス・ポインターのインクリメント値は，サンプルの整数倍であった。しかし，これでは限られたピッチ・シフトしかできない。多くの場合，アドレス・ポインターは整数部と小数部から成り，そのインクリメント値をサンプルの小数にすることができる。アドレス・ポインターは，しばしば“フェーズ・アキュームレーター（位相累算器）”として捉えられ，そのインクリメント値は，“フェーズ・インクリメント”と呼ばれる。フェーズ・アキュームレーターの整数部は，サンプル・メモリーのアドレッシングに使用され，小数部は周波数の精度を維持するために使用される。

例えば，フェーズ・インクリメント値が1/2ならば，そのピッチは1オクターブ下にシフトされる（周波数は半分になる）。それに比べ，1.05946（2の12分の1乗）というフェーズ・インクリメント値では，半音のピッチ・シフトが得られるだろう（CからC＃）。非整数のインクリメント値を利用する場合の再生時の周波数精度は，アドレス・ポインターとアドレス・インクリメント値の小数部を表すのに使用されるビット数によって決まる。

●補間

アドレス・ポインターの小数部が0でない場合，“望ましい値”は利用可能なサンプル・データの間を示す。図8は，アドレス・ポインターとインクリメント値が，それぞれ4ビットの整数部と4ビットの小数部を持つ場合の，単純化されたアドレッシングの仕組みを表している。この場合，インクリメント値は 1/ 2サンプルである。シンプルな音源では，ポインターの小数部を単純に無視して D/A コンバーターに送るサンプル値を決める。この方法で D/A コンバーターに送られるデータの値は，図8のケース1に示されている。

図8　サンプル・メモリーへのアドレッシングと補間



より良い方法としては，利用可能な最も近いサンプル・データを使用することであろう。さらに高度な音源では，再生に使用する値を得るために，利用可能なデータ・ポイントの間で，何らかの数学的な補間を行っている。補間が利用された場合，D/A に送られるであろう値は，ケース2に示されている。全般的な周波数の精度は，どちらのケースでも同じであるが，補間を使用しないケースでは，その出力波形が激しく歪んでしまう。

サンプル・データの値と値の間を補間するアルゴリズムには，多くの種類がある。最も単純なのは直線補間である。直線補間を用いる場合，補間された値は，小数アドレスを加重定数とした最も近いふたつのサンプルの単なる加重平均である。例えばポインターが(n+K)というアドレスを示し，そこではnがアドレスの整数部でKが小数部であるなら，補間された値は，s (n+K)=(1 − K) s (n)+(K) s (n+1) と計算される。ここでs (n) は，アドレスnのサンプル・データの値である。さらに，歪みを減らすために，より高度な補間技法を利用することも可能であるが，よりたくさんの計算時間を要する。

●オーバーサンプリング

ウェーブテーブル方式の楽器では，サンプル音のオーバーサンプリングも，歪みを改善するのに役立つ場合がある。例えば，4倍オーバーサンプリングが楽器音のサンプルに利用された場合，まったくピッチ・シフトのない再生において，4というアドレスのインクリメント値が使用されることになろう。再生時に選ばれるデータ・ポイントは，オーバーサンプリングを利用しない場合と比べて波形を表すデータ・ポイント数が増加する分，“望ましい値”により近くなるだろう。もちろんオーバーサンプリングは，サンプル・メモリーの必要量を増やすことになり，コストを高くする。

最良の方法は，直線補間とオーバーサンプリングを組み合わせて利用することであろう。より高度な補間を行うためには高い演算能力を必要とするが，直線補間はそれを必要としないため，リーズナブルに精度を上げることができる。さらに精度を必要とする音については，オーバーサンプリングを利用する。この方法によって，オーバーサンプリングのために必要とされる追加メモリーは，それを最も必要とするところでのみ利用すればよい。直線補間とオーバーサンプリングの選択的組み合わせの効果は，すばらしい結果を生み出すのである。

●スプリット

再生時にサンプル音のピッチを変更した場合，音色もいくらか変わってしまう。その変化は，大きなピッチ・シフトよりも，ピッチの小さな変更（数半音まで）の方が目立たない。自然なサウンドを得るには，ある楽器音のサンプルは，限られた範囲のノートを再生するためにのみ有効であろう。楽器の音域全体をカバーするには，多数の異なるサンプルを用意し，それぞれ限られたノート範囲で使用する。このような楽器は，“マルチサンプルされた”楽器と呼ばれる。この技法は，鍵盤楽器をいくつかの音域に分け，それぞれの音域に異なるサンプル音を割り当ててスプリットを作ることとして捉えることができる。それぞれの音域は，スプリット，あるいはキー・スプリットと呼ばれる。

ベロシティ・スプリットは，異なるサンプルを異なるノート・ベロシティで使用することを言う。ベロシティ・スプリットでは，あるノートが柔らかくプレイされた場合と，もっと大きいベロシティでプレイされた場合では，同じ楽器の同じノートの異なるサンプルが利用されることになろう。この技法は，メモリーの必要量が増えるため，基本的なサンプル音を作り出すのには通常使用されない。

しかし，キー・スプリットとベロシティ・スプリットは，共に演奏機能を高めるために利用できる。例えば，キー・スプリットを使えば，キーボードの低い方のオクターブでフレットレス・ベースを，高い方のオクターブでビブラフォンを演奏することが可能である。同様に，ベロシティ・スプリットを使えば，キーが強く打鍵された時だけアコースティック・ピアノにストリングスが重なって演奏されるようにもできるだろう。

●エイリアシング・ノイズ

前項では，ピッチ・シフトの結果として生じる音色変化について述べた。記録されたサンプル音のピッチをシフトするために使用されるリサンプリングも，また，楽器音にエイリアシング・ノイズを発生させる要因となる。エイリアシング・ノイズの発生により，サンプル音に効果的に適用できるピッチ・シフトの量も制限されてしまう。高音の倍音成分を多く含む音は，一般に，エイリアシング・ノイズの問題を生じることが多い。補間の後に使用されるローパス・フィルターは，エイリアシング・ノイズを除去する手助けとなりうる。オーバーサンプリングの使用も，またエイリアシング・ノイズの除去を手助けしてくれる。

●ビブラート及びトレモロ用の LFO

ビブラートとトレモロは，アコースティック楽器を演奏するミュージシャンによって，しばしば用いられる効果である。ビブラートは，基本的にノート･ピッチの低周波モジュレーションであり，一方トレモロは，音量のモジュレーションである。シンセサイザーにおけるこれらの効果は，ロー・フリケンシー・オシレーター（LFO）を用意し，作り出したサウンドの音程や音量を変調することでシミュレートしている。

自然なビブラートやトレモロ効果は，楽音が持続状態になるに従って，効果を増す傾向がある。シンセサイザーでは，エンベロープ・ジェネレーターに LFO を適用することによって作り出している。例えばフルート音は，ノートが発音した後，しばらくしてから始まるトレモロ効果を持っている。そのトレモロ効果は，最大まで徐々に増加し，その後はノートが発音を停止するまで，そのままの効果が続く。

●レイヤー

レイヤーは，ノートを発音する際，複数のサンプルを同時に演奏する方法である。この方法は，リッチなサウンドを得るために使用され，また限られたサンプル・セットから作り出すことのできる音色数を増加させるのにも有効であろう。レイヤーによるサウンドは，ノートの発音に1ボイス以上の同時発音を行う。したがって，これらのサウンドを使用する場合，同時発音可能なボイス数は事実上減少する。

●デジタル・フィルター

ローパス・フィルターは，ピッチ・シフトの際に発生するエイリアシング・ノイズを除去するために使用される場合があることは，前に述べた。楽器音を向上させるために，音色生成プロセスにおいて，デジタル・フィルターを使用する方法は数多くある。これらのデジタル・フィルター処理はポリフォニックで行われる。すなわち，各ボイスで生成された音に対して独立したフィルターが施され，そのフィルターは，ダイナミックに動くカットオフ周波数とQを持つべきである。

多くのアコースティック楽器で発せられる音色の特性は,その楽器が演奏される音量レベルによって大きく変化する。例えば，アコースティック・ピアノの音色は，力強く演奏された時にはたいへん明るいが，ソフトに演奏された場合はメロウである。ベロシティ・スプリットを使えば，異なるノート・ベロシティでは異なるサンプルを利用することができるため，これをシミュレートすることができる。

もうひとつのたいへんに効果的な方法は，ノート・ベロシティに連動してカットオフ周波数をコントロールできるデジタル・ローパス・フィルターを使用することである。このポリフォニック・デジタル・フィルターは，ノート・ベロシティによって，シンセサイズされたサウンドの出力周波数成分をダイナミックに変化させることで，効果的なアコースティック楽器音の再生を可能にする。

デジタル・フィルターのもうひとつの重要な用途は，キー・スプリットされたサンプル間の推移をなめらかにすることにある。スプリット境界の両側には，異なるサンプルを再生するふたつの隣り合ったノートが存在するだろう。通常，これらのサンプルのひとつは，必要とされるピッチのノートを作り出すためにシフト・アップされており，もうひとつは，そのピッチにシフト・ダウンされているだろう。このため，これらふたつの隣り合ったノートの音色は著しく異なり，スプリットされていることを目立たせてしまう。

これは，ノート番号によってフィルターの特性を変えることができるデジタル・フィルターを利用することで，緩和される場合もある。音色テーブルには，ノート番号におけるフィルター特性が，音色ごとに含まれるかもしれない。フィルターの特性は，キー・スプリットにおけるピッチ・シフトによって生じる音色変化を緩和するために選ばれるのである。

エンベロープ・ジェネレーターやLFOを用いて，デジタル・フィルターの特性をコントロールすることも，また一般的に行われている。その結果として，楽器音の音色の時間変化が得られる。エンベロープ・ジェネレーターによってフィルターのカットオフ周波数をコントロールすれば，開始時点ではたいへん明るく，ノートが減衰するにしたがって徐々に柔らかくなるような音色を作り出すことができるだろう。エンベロープ・ジェネレーターやLFOを使用し，フィルターのカットオフ周波数を高いQ設定でスイープさせることは，アナログ・シンセサイザーのサウンドをシミュレートする際にも有効である。

PCとMIDIの接続

パーソナル･コンピューターでMIDIを利用するためには,通常,PC/MIDIインターフェースが必要とされる(内蔵MIDIインターフェースを装備しているパーソナル・コンピューターもいくつかある)。PC用のMIDIインターフェースは数多い。IBM互換機用で最も使用されているMIDIインターフェースのタイプは，PCバスの拡張スロットに挿入するアドイン・カードであるが，シリアル・ポートMIDIインターフェース(PCのシリアル・ポートに接続する)やパラレル・ポートMIDIインターフェース(PCのプリンター・ポートに接続する)も使われている。他の多くのパーソナル・コンピューターでは，シリアル・ポート接続が使用されている。

PC用のMIDIインターフェースの基本的な機能は，PCデータ･バスからのパラレルなデータ･バイトを，シリアルのMIDIデータ･フォーマットに変換したり，その逆(UART機能)を行うことである。しかしながら“スマート”なMIDIインターフェースは，MIDIタイミング情報の生成，MIDIデータのバッファリング，MIDIメッセージのフィルタリング，外部機器との同期といった，数多くの高度な機能を持っている場合がある。

GMを使ったゲームや，他のアプリケーションの操作性を本質的にわかりやすいものとするため，マルチメディア・マーケットに向けたインターフェースの設計は，重要である。GMは，ゲームがどのシンセサイザーと接続できるかを規定していないので，正しい動作を保証するには，サウンド・カードの標準化も必要とされる。PCのオペレーティング・システム(OS)の中には，専用のデバイスが提供されているものもあるが，これはMIDIインターフェースの標準が必要とされるような，MS-DOSが動作している典型的なIBM-PCには当てはまらない。

IBM-PC用MIDIインターフェースのアドイン・カードの事実上の業界標準は，Rolandの MPU-401インターフェースである。MPU-401はスマートなMIDIインターフェースで，ダム・モード動作(しばしば“UARTモード”と呼ばれる)もサポートしている。市場には，MPU-401互換のMIDIインターフェースが多数存在するが，それらの中には動作としてUART(ダム)モードしかサポートしていないものもある。さらに，多くのIBM-PCアドイン・サウンド・カードは，MPU-401のUARTモード機能を持った内蔵MIDIインターフェースを搭載している。

PC互換問題

PCで動作するMIDIアプリケーション・ソフトにとって，考慮しなければならない互換性には，ふたつの段階がある。第1段階は，使用するMIDIインターフェースとアプリケーションの互換性である。第2段階としては，MIDIシンセサイザーとアプリケーションの互換性である。これは，他のPCシステムにも当てはまるが，ここではIBM-PCとその互換システムについてのみ話を進める。DOSとマイクロソフトWindowsオペレーティング・システム(OS)の下での互換性の考察は，次の項で述べる。

● MS-DOSアプリケーション

MIDIシンセサイザーを利用するMS-DOSアプリケーションとしては，MIDIシーケンス･ソフトウェア，楽譜作成ソフトウェア，さまざまなゲームなどがある。MIDIインターフェースの互換性の点において，これらのアプリケーションは実質的に，すべてMPU-401インターフェースをサポートしており，そのほとんどがUARTモードしか必要としない。これらのアプリケーションは，MPU-401の全機能を搭載したMPU-401互換インターフェース，あるいはMPU-401 UARTモード互換のサウンド・カードを

装備したすべてのPC互換機で正しく機能すべきである。シリアル・ポートやパラレル・ポートMIDIアダプターのような他のMIDIインターフェースについては，そのアプリケーションが，それらのMIDIインターフェースをサポートしている場合にのみ機能するだろう。

ある特定のアプリケーションが，多くのさまざまなシンセサイザーや音源モジュールをサポートしている場合もある。GM規格に先だって，シンセサイザーに幅広く受け入れられた標準的なパッチ・セットは存在しなかったため，アプリケーションは一般に，その時点で最もポピュラーないくつかのシンセサイザーをサポートする必要があった。アプリケーションが，そのPCに接続されている特定のシンセサイザーや音源モジュールをサポートしていない場合，そのアプリケーションによって作り出されるサウンドは，意図したサウンドではないだろう。現代のアプリケーションは，GMシンセサイザーをサポートすることができ，いかなるGM互換のサウンド・ソースに対しても，正しいサウンドを作り出すはずである。

●マルチメディアPC（MPC）システム

PCにおける高クオリティのオーディオ機能を持ったアプリケーション（ミュージック・シンセシスを含む）の数は，1991年のマルチメディア・エクステンションを伴うマイクロソフトWindows3.0（“Windows with Multimedia”）の発表後，爆発的に増えた。これらのエクステンションは，Windows 3.1オペレーティング・システム（OS）にも組み込まれている。マルチメディアPC（MPC）規格は，元来，1991年にマイクロソフトによって発表されたが，現在では，Software Publishers Associationの子会社であるMultimedia PC Marketing Councilによって維持されている。MPC規格は，マルチメディアを扱う能力のあるパーソナル・コンピューターが，Windows 3.1あるいはWindows with Multimediaをベースにしたマルチメディア・アプリケーションを動作させる場合の互換性を確保するための最小限の必要条件を述べている。

MPCシステムのオーディオ能力は，デジタル録音と再生（リニアPCMサンプリング），ミュージック・シンセシス，そしてオーディオ・ミキシングを含まなければならない。現在のMPC規格は，パフォーマンスの異なるふたつのレベルを定義している。MPCレベル1とMPCレベル2におけるデジタル・オーディオ録音と再生の必要条件は異なっているが，ミュージック・シンセシスの必要条件は，基本的に同一である。

MIDIにとって現在のMPC規格は2タイプのシンセサイザーを定義することによって，パフォーマンスとコストのバランスを取っている。“基本マルチティンバー・シンセサイザー”と“拡張マルチティンバー・シンセサイザー”である。基本と拡張のどちらのシンセサイザーも，GMパッチ・セットを使用することになっているが，どちらも実際には，GMで定められている同時発音数やティンバー数の必要条件を満たしてはいない。基本マルチティンバー・シンセサイザーは，3つの“メロディ・ティンバー”と2つの“パーカッション・ティンバー”を使って，6つの“メロディ・ノート”と2つの“パーカッション・ノート”を同時にプレイできなければならない。

拡張マルチティンバー・シンセサイザーの公式な必要条件は，基本的なマルチティンバー・シンセサイザーに指定された要件を上回る能力を持たなければならないということだけである。しかしながら，シンセサイザーの“目標”としては，16のメロディ・ノートと8つのパーカッション・ノート，9つのメロディ・ティンバーと8つのパーカッション・ティンバーを同時にプレイする能力が含まれている。

MPC規格は，MIDI楽曲のオーサリング基準も含んでいる。この基準は，利用可能な同時発音数とティンバーが異なってもよい。それぞれのMIDIファイルが同じソングについて，ひとつは基本シンセサイザー用，もうひとつ拡張シンセサイザー用の2つのアレンジを含むことを要求している。基本シンセサイザー用にアレンジされたMIDIデータは，MIDIチャンネル13～16で送られ（パーカッションはチャンネル16），拡張シンセサイザー用は，チャンネル1～10を利用する（パーカッションはチャンネル10）。

この方法は，両方のタイプのシンセサイザーでプレイするのに最適な MIDI ファイルを供給することを意図している。しかしこれは，GMシンセサイザーで演奏する際に起こる潜在的な問題を含んでいる。GMシンセサイザーは，16 の MIDI チャンネルすべてを受信し，両方のパフォーマンスをプレイするだろう。これはチャンネル 16 のパーカッション・トラックをメロディ楽器で演奏することを意味している。

マイクロソフトは，基本 / 拡張モデルの代わりに完全GMモデルを推奨することによって，この問題の解決を Windows の将来のバージョンに委ねている。しかしながら，マイクロソフトのデュアル・フォーマット用に作成された既存のデータは，次世代の Windows システムにとっても問題であり続けるだろうし，GM完全互換のシンセサイザーを含む，今日のいかなるシステムでも問題となる。

現在における唯一の解決方法は，以下に記述されているように，エクストラ・チャンネルの再生をブロックするために，Windows の MIDI マッパーを使用することである。残念なことに，これはGM互換スコアにおいて，必要とされるデータをもブロックしてしまうことになる。理想的な解決方法は，基本 / 拡張データを含むスタンダード MIDI ファイルを識別する方法を開発し，プレイするファイルのタイプを考慮する“ダイナミックな”MIDI マッピング方法を提供することである。これは Roland の GSや YAMAHA のXGフォーマットのような，GMハードウェアでは多少なりとも問題を生じる他の標準化されたフォーマットにも適用することができるであろう。

●マイクロソフト・ウインドウズの構成

Windows のアプリケーションは，MIDI インターフェースやシンセサイザーのようなハードウェア・デバイスへ，ドライバーを通してアクセスする。ドライバーはアプリケーションに，ハードウェアへアクセスする共通のインターフェースを提供し，これはハードウェアの互換性の問題を単純化する。シンセサイザーのドライバーは,コントロール･パネル内の Windows ドライバー･アプレットを使用してインストールされなければならない。

MIDI インターフェースやシンセサイザーが PC にインストールされ，必要なデバイス・ドライバーがロードされた場合，Windows MIDI マッパーがコントロール・パネル内に現れる。MIDI メッセージは，アプリケーションから MIDI マッパーに送られ，MIDI マッパーはそのメッセージを，適切なドライバーに送る。MIDI マッパーは，アプリケーションからドライバーへの経路の中で，MIDI メッセージのフィルタリングや変換を行うよう設定される場合もある。MIDI マッパーで行われる処理は，MIDI マッパー・セットアップ，パッチ・マップ，そしてキー・マップの中で定義される。

MIDI マッパーの設定によって，MIDI チャンネルをデバイス・ドライバーにアサインすることができる。例えば，MPU-401 インターフェースとGMシンセサイザーを所持していて，さらに Creative の Sound Blaster カードが搭載されている場合，チャンネル 13 ～ 16 を Ad Lib ドライバー（Sound Blaster 上の基本的な FM シンセサイザーをドライブする）に，そしてチャンネル 1 ～ 10 を MPU-401 ドライバーにアサインできる。この場合，MPC 互換の MIDI ファイルは，GMシンセサイザーと FM シンセサイザーの双方が同時にプレイすることになる。GMシンセサイザーは，MIDI チャンネル 1 ～ 10 で拡張されたアレンジでの演奏をし，FM シンセサイザーはチャンネル 13 ～ 16 で，基本的なアレンジでの演奏をするだろう。

MIDI マッパーの設定により，MIDI メッセージのチャンネル番号を変更することができる。GM楽器用に作成された MIDI ファイルを所持していて，それらを基本マルチティンバー・シンセサイザーでプレイしている場合，アプリケーションからチャンネル 10 で送られて来たパーカッション・データを，チャンネル 16 にしてデバイス・ドライバーに送りたい場合もあるだろう。

MIDI マッパーのパッチ・マップは，MPC あるいはGMファイルをGMパッチ番号を使用していないシンセサイザーでプレイしたい場合に，パッチ番号の変換に使用できる。パッチ・マップはまた，非GMシンセサイザー用にアレンジされた MIDI ファイルを，GMシンセサイザーでプレイする時にも使用できる。例えば，Windows で供給される MT-32 パッチ・マップは，GM互換の .MID ファイルを Roland の MT-32 音源モジュール，あるいは LAPC-1 サウンド・カードでプレイする場合に使用できる。

MIDI マッパーのキー・マップも同様の機能を果たすが，これは MIDI ノート・オンとノート・オフ・メッセージに含まれるノート番号を変換する。この機能は，GM互換のパーカッション・パートを非 GM シンセサイザーで再生するために変換する場合や，その逆の場合に有効である。Windows で供給される MT-32 キー・マップは，GMで使用されるキー対ドラム・サウンドの割り当てを，MT-32 と LAPC-1 で使用されるものへと変換する。

概要

MIDI プロトコルは，楽曲の演奏のデータを伝送する効率的なフォーマットを提供し，スタンダード MIDI ファイル規格は，異なるアプリケーションが時間情報を持った MIDI データを共有することを保証する。これだけでもミュージシャンにとっては十分なことである。MIDI データの送信効率とリアルタイムな編集能力もまた，MIDI をマルチメディア・アプリケーションやコンピューター・ゲーム，あるいはハイ・エンドのカラオケ装置における音源として，魅力あるものにしている。

GMシステムは，多くの同時発音数を持つマルチティンバー・シンセサイザーに，共通の音源機能と共通の音色マップを提供し，音楽シーケンスの作家やマルチメディア・アプリケーションのデベロッパーに，共通のプラットフォームを提供するものである。ウェーブテーブル･シンセシスによるリアルなサウンドと,より新しいインタラクティブなアプリケーションの登場により,MIDI で演奏されるシンセサイザーは，演奏機器とマルチメディア・アプリケーションにとって，重要な要素であり続けるであろう。

2. MIDI1.0 規格

Document Ver. 4.2 （日本語版）

MIDI1.0 規格　Docoment Ver.4.2(日本語版)
Copyright ©1989 社団法人 音楽電子事業協会
Portions Copyright © 1995, 1989 社団法人 音楽電子事業協会 , MIDI Manufactures Association

MIDI1.0 規格

Document Ver. 4.2

目　　次

概　　論

詳　　解

付　　録

序文

MIDIとはMusical Instrument Digital Interface (MIDI) の略で，異なる楽器間やシーケンサー，コンピューター，ライティング・コントロール，ミキサーなどを相互に結合し，情報交換を可能にするために制定されたハードウェア（送受信回路）とソフトウェア（データ・フォーマット）の規格である。MIDIは本来，ライヴ・パフォーマンスの情報伝達手段として考えられたものであるが，レコーディング・スタジオやオーディオ / ビデオ制作，作曲などのマルチメディア分野にも大きな影響を与えている。

この規格書は，MIDI規格協議会（JMSC）とMIDI Manufacturers Association（MMA）とによって共同作成されたものである。発行日以降の内容変更，追加項目等については，JMSCとMMAの合意の上で，Confirmation of Approvalが個別に発行される。

＜日本語版注＞

現在では，MIDI規格協議会（JMSC）の活動は，社団法人音楽電子事業協会（AMEI）によって執り行われており，MIDI規格の改訂はAMEIとMMAとの合意のもとで行われている。

ハードウェア

この規格では，インターフェースとして転送速度31.25 K bit/sec（± 1%）の非同期方式シリアル転送を用いる。転送はスタート・ビット，ビット0 ～ 7の8個のデータ・ビット，ストップ・ビットの順序に，計10ビットで行われ，1バイトを転送するためには320 μ s要する。ここで，スタート・ビットは論理“0”，ストップ・ビットは論理“1”とする。

回路は，5mAのカレント・ループ・タイプとし，論理“0”を電流が流れている状態とする。ひとつの送信回路はひとつの受信回路のみを駆動する。グランド・ループとそのために起きるデータ・エラーを避けるために，受信回路にオプトアイソレーターを使用して，送信回路と受信回路を電気的に分離する。なお，受信回路は5mA以下の電流で起動状態となり，立ち上がり及び立ち下がり時間は2 μ s未満でなければならない。

＜日本語版注＞

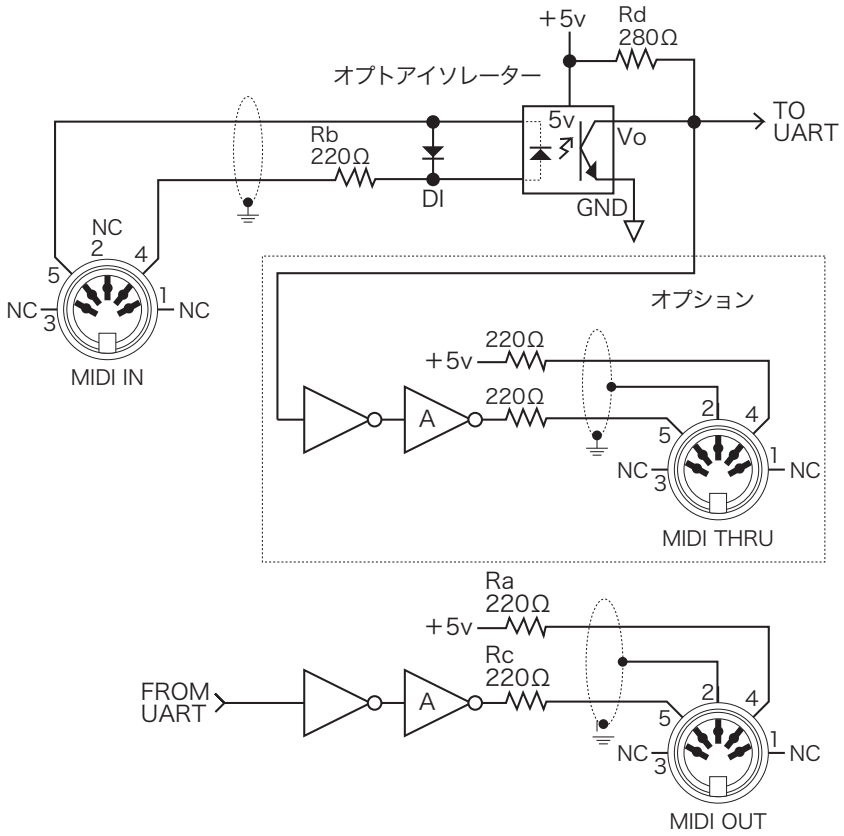
この規格に使用できる高速オプトアイソレーターには，次のような製品がある。

NJL-5127D, TLP513, PC-900V, PC-410(K), PC-910(K), HCPL-260L, HCPL-261A, HCPL-M600(Rd=1K Ω), QCPL-M605#500

コネクターは，5ピンのDIN（180°）を使用する。送受信とも機器パネル側にソケット（メス）を使用し，それぞれ「MIDI OUT」「MIDI IN」と表記しなければならない。ピン番号1と3は使用してはならず，送受信ともに解放（NC）にしなければならない。ピン番号2は，送信側ソケットにおいてのみ接地する。

MIDI INジャックのシェルの接続する端子は，グランド・ループを避けるため，回路やシャーシ・グランドに接続しない方がよい。

MIDI THRU情報がMIDI IN信号から得られる場合には，方形波の上昇点と下降点との間での信号の下落によって不正確なものとなり得る。この信号の下落は，オプトアイソレーターの応答時間に起因するものである。このようなタイミング・エラーは，MIDI THRUとMIDI INジャックとの間に接続される機器の数に比例して増加する傾向にある。これは回路の品質の欠如であって，いかなる高性能の素子を使っても，MIDI THRUで直列に接続できる数には制限があることを示すものである。



MIDI 標準ハードウェア（回路図）

＜備考＞

1. ゲート“A”は，集積回路またはトランジスターである。
2. 抵抗は，± 5%。また，Rd は使用するオプトアイソレーターに依存する。
3. DI は，ダイオード。

MIDI ケーブルは，最長で 15 mのシールドされたツイストペアとし，5 ピンの DIN プラグ（オス）を両端に接続する（SWITCH CRAFT 05GM5M などが，この規格に使用できる）。シールドは，両端でピン番号2に接続しなければならない。

必要に応じて“MIDI THRU”と表記された出力端子を備えてもよい。これは，MIDI IN の信号をそのままの形で出力するものとする。また，MIDI THRU によって3台を越える機器を対象とする長いチェーン接続を行うためには，より高速のオプトアイソレーターを使用して，立上り，立下りの時間差によるエラーを防止する必要がある。

データ・フォーマット

MIDI におけるデータ交換の実際は，複数バイトの“メッセージ”によって行われる。そして，リアルタイムまたはエクスクルーシブ・メッセージを除いて，ひとつのメッセージは，ひとつのステータス・バイトとそのステータスによって導かれる，ひとつまたは2つのデータ・バイトとで構成される。

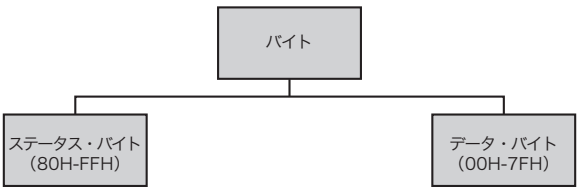
MIDI 機器は，一般に送信器と受信器とを有するが，送信器だけまたは受信器だけを有するものであってもよい。送信器は，MIDI のデータ･フォーマットに従ってメッセージを作成し，それを UART（Universal Asynchronous Receiver ／ Transmitter）及びライン・ドライバーを通して送り出す。

受信器は，オプトアイソレーター（フォトカプラー），UART 及びその周辺回路で構成され，送られて来たメッセージを MIDI のフォーマットに則して解釈，実行する。

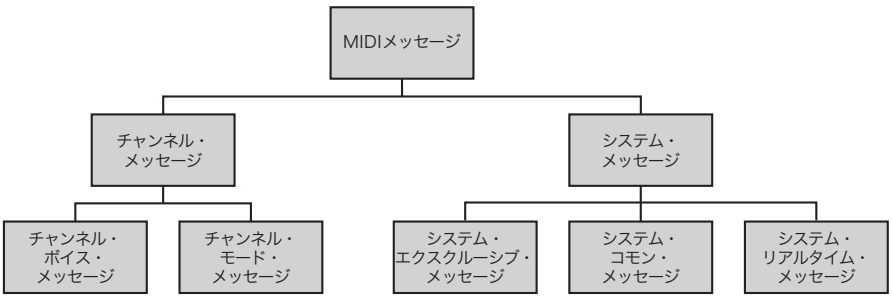
MIDI では，演奏（パフォーマンス）情報を送るために 16 のチャンネルが用意されており，他にもさまざまな情報を送れるようにして，拡張性や柔軟性を持たせている。MIDI メッセージには，チャンネル・ボイス（Channel Voice），チャンネル・モード（Channel Mode），システム・コモン（System Common），システム・リアルタイム（System Real Time）及びシステム・エクスクルーシブ（System Exclusive）の，主要5メッセージがある。

MIDI では，事象は“メッセージ”として伝送され，メッセージは1バイト以上で構成される。MIDI で送られるデータを分類して図示すると次のようになる。

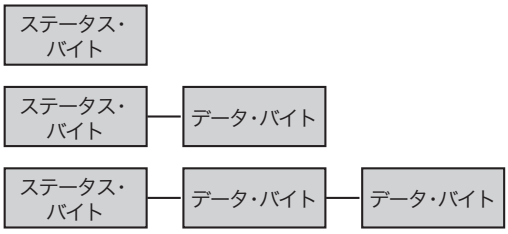
MIDIバイトの種類



MIDIメッセージの種類



MIDIメッセージ（シングル）の構成



エクスクルーシブ・メッセージの構成

メッセージの種類

MIDI のメッセージは，チャンネル・メッセージとシステム・メッセージとに大別される。

チャンネル・メッセージ

チャンネル・メッセージではステータス・バイトの下位4ビットを使用して，1から 16 までのチャンネルを指定し，その上位 4 ビットでメッセージの種類を表す。このチャンネル指定によって，メッセージはシステム全体の中で，そのチャンネル番号がステータスのチャンネル番号と合致する受信器にて受け入れられることとなる。

機器の設定によっては，一度に複数チャンネルのメッセージを受け入れて動作する場合があるが，それらのチャンネルをまとめて “ボイス・チャンネル”と呼び，その中で “どの様なモードになるか”といった重要な指令を受け取ることになっているチャンネルを “ベーシック・チャンネル”と呼ぶ。

チャンネル・メッセージは，さらにボイス・メッセージとモード・メッセージとに細分される。

ボイス： ボイス・メッセージは，受信側の機器のボイス（発音）をコントロールするもので，発音させたいボイス（またはモジュール）が所属するチャンネル（ボイス・チャンネル）で送られる。

モード： モード・メッセージは，受信側の機器がボイス・メッセージをどのように処理するかを決めるもので，機器のベーシック・チャンネルで送られる。

システム・メッセージ

システム・メッセージは，そのシステム全体に接続された機器に共通に必要な情報であり，コモン・メッセージ，リアルタイム及びエクスクルーシブの3種類がある。

コモン　コモン・メッセージは，そのシステムに接続されたユニット全部に対し有効である。

リアルタイム　リアルタイム・メッセージは，コモン・メッセージと同様に，すべての機器に対し有効である。ステータス・バイトのみで，データ・バイトを有さず，かつ他のメッセージのバイト間でも送ることができる。機器によっては，リアルタイム・メッセージを無視してもよいが，その場合は，リアルタイム・メッセージを受ける直前の状態（例えば，あるメッセージの1バイト目のデータの直後など）にもどって，以後の処理を続けるものとする。

エクスクルーシブ　エクスクルーシブ・メッセージは，ステータス・バイトに続き，何バイトのデータをも送ることができる。送信側は，エクスクルーシブの終わりには速やかに EOX（End of eXclusive）を送らなければならない。受信側は，EOX または他のメッセージ（リアルタイム・メッセージを除く）のステータス・バイトによって終了する。

エクスクルーシブ・メッセージは，ステータス・バイトの次に各製造者の ID コードを有し，受信側がその ID コードを認識することにより，継続するデータを無視するか実行するかを決定する。ID コードを有する製造者は，その ID によって決定される継続のデータのフォーマットを公表しなければならない。また，他の製造者は，自由にその ID を使用してもよいが，フォーマットの変更や追加はその ID を所有している製造者のみが行うことができる。

データの種類

MIDI データは，ステータス・バイトとデータ・バイトとに分けられる。

ステータス・バイト

ステータス・バイトは，MSB が1である8ビット・バイナリィであり，メッセージの種類を示す。ステータス・バイトによって，継続するデータ・バイトの数及びそれらの意味が定義される。システム・リアルタイム・メッセージのステータス・バイトを除いて，新しいステータス・バイトは，たとえその直前のメッセージが完結していなくても，認識されなければならない。次のデータ・バイトは，新しいステータス・バイトに所属することになる。また，完結されなかった最終のメッセージは無視されることになる。

＜日本語版注＞

ここでの MSB は，1 バイトの中での 8 番目の最上位ビットの意味である。

ランニング・ステータス

チャンネル・ボイス・メッセージ及びチャンネル・モード・メッセージにおいては，次のメッセージのステータス・バイトが，前のメッセージのステータス・バイトとチャンネル番号を含めて同一であれば，省略することができる。したがって受信側は，あるステータス・バイトを受信すると，それを記憶しておいて，次の新しいステータス・バイトが来るまで，記憶されたステータス・バイトに所属するメッセージとして，データ・バイトを処理しなければならない。つまり，データ・バイトの数は常に正しい必要がある。例えば，2 バイトのデータ・バイトを持つステータス・バイトで，勝手に3バイトのデータ・バイトを送り，3バイト目に独自の意味を持たせたりしてはならない。この3バイト目は，受信側では（同じステータスのもとでの）次のメッセージのデータ・バイトの 1 バイト目と判断される。

このように，送信側では省略され，受信側で記憶されているステータス・バイトを “ランニング・ステータス”と呼ぶ。

このルールによって，同一チャンネルの連続するノート・オン・メッセージのように，同一のステータスで送られるメッセージは速く送信できるので便利である。なお，この場合ノート・オフは，ノート・オン・メッセージのベロシティを “0”にすることで代用できることを利用して，ランニング・ステータスを変更させないようにすれば，いっそう効果的である。

システム・リアルタイム・メッセージのステータスは、ランニング・ステータスの効力を消滅させない。特に受信側の処理は，以上のルールを十分考慮に入れてなされなければならない。

使用しないステータス

受信側が備えていない機能を有するステータス・バイトは，無視するとともに，その後に続くデータ・バイトも，同様に無視するものとする。

未定義のステータス

未定義のステータス・バイトは，絶対に送出してはならない。特に，電源オン／オフ時にパルス，ノイズが不用意に送り出されることによって，未定義のステータス・バイトまたはルール違反のデータ・バイトになったり，フレーミング・エラーを引き起こすため，十分に注意する必要がある。受信側は，未定義のステータスと，その後のデータ・バイトとを共に無視しなければならない。

データ・バイト

ステータス・バイトの後には，システム・リアルタイム・メッセージを除いて，ひとつまたは2つのデータ・バイトが続くことによって，メッセージの内容を伝える。このデータ・バイトは，8ビット・バイナリィの最上位ビットが0であることによって，ステータス・バイトと区別される。

各々のステータス・バイトの後に続くべきデータ・バイトの数と，そのデータ・バイトがとる値の範囲は，メッセージごとに定められている。送信側は，各々のステータス・バイトに対して，常に正しい数のデータ・バイトを送らなければならない。受信側は，ステータス・バイトに続くひと組のデータ・バイトを受け終わるまで，そのメッセージの実行を待つものである。受信側は，ランニング・ステータスの場合を除いて，適当なステータス・バイトの先行がないデータ・バイトを無視しなければならない。

チャンネル・モード

シンセサイザー，その他の機器には複数の音源エレメント（モジュール）があるが，これを“ボイス”と呼ぶ。“ボイス・アサイメント”とは，自己のキーボードや MIDI IN より送られてくるノート・オン／ノート・オフの情報によって，ボイスをコントロールするためのアルゴリズム処理方法である。

＜備考＞ 本文中で使われる“機器”の意味

本文中でひとつの“機器”と言った場合，それは実際の動作の単位として使われるので注意が必要である。例えば，外見上は1台の機器がスプリットなどによって実質的に複数の“機器”として動作することがある。

MIDI の 16 チャンネルと機器のボイス・アサイメントとの関係は，4つのモードによって定義づけることができる。それらのモードは，オムニ・オン／オムニ・オフと，ポリ／モノとの組合せで作られる。オムニ・オンとオムニ・オフ，ポリとモノとは，それぞれ排他的である。例えば，ポリ・オンは，ポリ・モードがオンでかつモノ・モードがオフとなることを意味する。受信側が，オムニ・オンの状態では，すべてのチャンネル番号のボイス・メッセージを無条件に受け入れ，実行する。オムニ・オフでは，特定の，ひとつまたは複数のボイス・チャンネルのボイス・メッセージだけを受け入れる。モノ・モードの状態では，ひとつのチャンネルのボイス・メッセージは，ひとつのボイスにのみ割り当てられる（チャンネルごとに各々モノフォニックである）。ポリ・モードでは，受信側の有するアサイン方法によって，複数のボイス・メッセージが各々複数のボイスに割り当てられる。

ベーシック・チャンネル “N” (N=1 ～ 16)の受信側機器は，2組のモード・メッセージによって，4つの状態(Mode 1 ～ 4)のいずれかになり得る。

Mode	OMNI	POLY/MONO	
1	ON	POLY	すべてのチャンネルのボイス・メッセージを受け入れて，自己の有する方法により，ポリフォニックに割り当てる。
2	ON	MONO	すべてのチャンネルのボイス･メッセージを受け入れて，ひとつのボイスだけをコントロールする。
3	OFF	POLY	チャンネルNのボイス・メッセージを受け入れて，それぞれのボイスに，ポリフォニックに割り当てる。
4	OFF	MONO	チャンネルNから N+M-1 までのボイス･メッセージを受け入れて，ボイス1からMまでに，それぞれ固定的に割り当てる(“M” はモード・メッセージの3バイト目の値)。

送信側機器における4つのモードは次の通りとなる(ベーシック・チャネルは “N”とする)。チャンネル番号を変更する機能を有しないのものは，通常，ベーシック・チャンネルが “1” (N=1)でデータを送る。

Mode	OMNI	POLY/MONO	
1	ON	POLY	すべてのボイス・メッセージはチャンネルNで送られる。
2	ON	MONO	ひとつのボイスを対象として，ボイス・メッセージはチャンネルNで送られる。
3	OFF	POLY	すべてのボイスに対するボイス・メッセージは，チャンネルNで送られる。
4	OFF	MONO	ボイス 1 から M までに対するボイス・メッセージは，それぞれ固定的に，チャンネルNから N+M-1 で送られる。

受信側機器または送信側機器は，一度に複数のモードを取ることはできない。受信側が指定されたモードの機能を持っていない場合には，そのモード・メッセージを無視するか，代用できるモードに切り換わる(通常は，オムニ・オン，ポリ)。

モード・メッセージは，受信側のモードに関係なく，ベーシック・チャンネルで送られて来たものだけが受け入れられ，認識される。ボイス・メッセージは，上記のルールにしたがって，受信側が用いたモードによって定められたチャンネルのものが認識される。

1台の機器であっても複数の “機器” として動作するものの場合，それぞれの “機器” は，異なるベーシック・チャンネルとモードをとることができる。このモードの変更は，機器のパネル操作か，またはそれぞれのベーシック･チャンネルで送られるモード･メッセージによって行われる。MIDI で規定しているモードではないが，このような動作の状態は “マルチ・モード” と呼ばれている。

一台の機器の受信と送信の部分が，異なるチャンネル，異なるモードで動作してもよい。例えば，モノ・モードで受信し，ポリ・モードで送信する機器があってもよいし，MIDI の受信をチャンネル1，送信をチャンネル3で行ってもよい。

電源オン時の状態

電源オン時にはオムニ・オン，ポリで，チャンネル1の状態になるのがよい。しかしモードやチャンネル，その他の MIDI に関するパラメーターが電源オフ時に保持されるという方法も考えられる。つまり電源オン時の状態をどうするかは，各機器の仕様によって決められることである。ここで言う「電源オン時」とは，機器の電源スイッチをオンした後のデフォルト状態を指す。すなわち，パネル操作や MIDI IN からの MIDI メッセージをまだ受けていない状態を言い，たとえスイッチをオンしてから数時間経過していても，上記の状態を保っていれば「電源オン時」である。

チャンネル・ボイス・メッセージ

ノート・オフ	**8nH**
ノート・オン	**9nH**
ポリフォニック・キー・プレッシャー（アフタータッチ）	**AnH**
コントロール・チェンジ	**BnH（第2バイト 0-119）**
プログラム・チェンジ	**CnH**
チャンネル・プレッシャー（アフタータッチ）	**DnH**
ピッチ・ベンド・チェンジ	**EnH**

MIDI 機器間で伝送される情報は，チャンネル・ボイス・メッセージが大部分を占める。それはノート・オン，ノート・オフ，プログラム・チェンジ，ピッチ・ベンド・チェンジ，アフタータッチ，コントロール・チェンジで構成される。これらの用語については後述する。

ひとつのノート・オン・メッセージは，3バイトで構成される。これを伝送するには 960 μ s を要するため，数音を同時に演奏する時には，それらのノート・オンを伝送するために数 ms かかることになる。同時に多数のイベントが起こった場合，MIDI を使って，聴感上のわずかな遅れもなく，これに対応させるのは困難である。この問題は，ランニング・ステータスを用いることにより，いくぶん緩和される。

チャンネル・ボイス・メッセージのタイプ

チャンネル・ボイス・メッセージのメッセージ・タイプ（型）は，次に大別される。

ノート・オン　鍵を押すか，他のトリガー機器から送出される。

ノート・オフ　鍵を離すことによって，送出される。送信側は，同一チャンネル，同一ノートに対して，複数のノート・オンを送った場合には，同じ数のノート・オフを送らなければならない。

コントロール・チェンジ　主に鍵以外の操作子（ペダル，ホイール，レバー，スイッチなど）を動かした時に送出される。

プログラム・チェンジ　“プログラム（例えば，サウンド，ボイス，トーン，プリセット，パッチ）”を切り替える時，新しく選択されたプログラム番号が送出される。

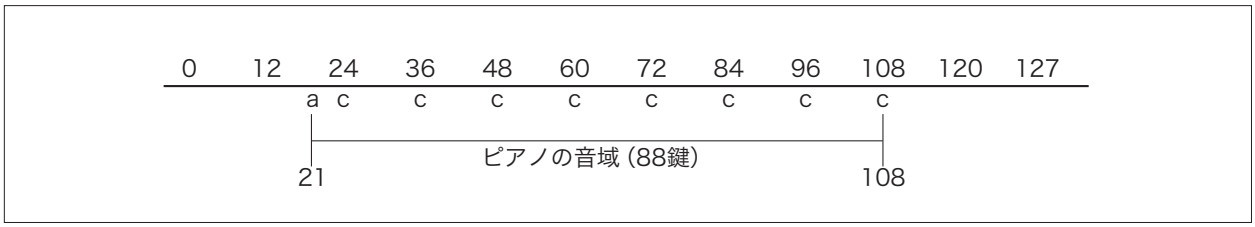
アフタータッチ　キーオン後の鍵を押す力により送出され,演奏されている音を変化させるために用いられる。アフタータッチは，ポリフォニック・キー・プレッシャー，またはチャンネル・プレッシャーで送られる。

ピッチ・ベンド・チェンジ　ピッチを変えるために用いられる。最大 14 ビット（2バイト）の分解能を持つ。

チャンネル・ボイス・メッセージは，鍵盤を有する機器に使われるだけでなく，さまざまな音楽的目的のために送出される。例えば，通常のキーボード・シンセサイザーからのノート・オン・メッセージが，パーカッション・シンセサイザーをトリガーしたり，照明の制御のために使われたりすることもあり得る。しかし，これらの使用法は，本来の使用方法による接続体系に組み入れられた時に，本来の目的を阻害しないように十分に考慮する必要がある。

ノート番号(鍵番号)

各鍵は数値に割り当てられ，ノート・オン／ノート・オフ・メッセージを送出する。中央Cが 60 と定められている。これは，88 鍵ピアノの中央Cであり，物理的にキーボードの中央に位置される必要はない。

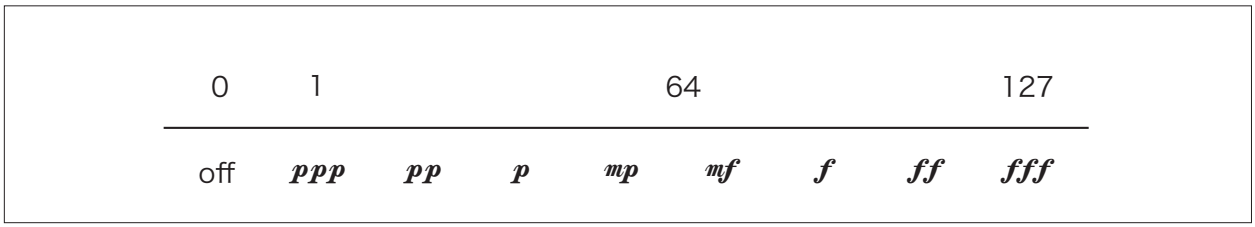


ベロシティ

ベロシティ・バイトの解釈は受信側に任される。オン・ベロシティは，音に強弱の差を付けるのに利用されるのが普通である。ベロシティが音量(出力レベル)に適用された場合，値が大きいほど大きな音を生成する。ベロシティと音量の関係は指数関数的になることが望ましいが，音楽表現上，音の強さは必ずしも音量だけに対応するとは限らない。受信側で MIDI ベロシティと内部のベロシティ応答との変換表(マッピング・テーブル)を複数持ち，それを切り替えて使用することとしてもよい。

オフ・ベロシティは離鍵後のエンベロープの減衰時間を変更することにも使用できる。

64(40H)の値は，ノート・オンで言えばメゾフォルテ付近に相当し，ベロシティを検出しない送信側機器は，原則としてこの値を送る。ただし，特殊な効果を得る目的で 64 以外の値を送ってもよい。



ノート・オフ

MIDI には，ふたつのノート・オフの送信方法がある。ノート・オン・メッセージと同一チャンネル，同一ノートのノート・オフ・メッセージを送る方法と，ベロシティ値 “0” で同一チャンネル，同一ノートのノート・オン・メッセージを送る方法とである。後者はランニング・ステータスを用いることによって，ステータス・バイトを省略することができるため，最も一般的に用いられている方法である。したがって受信側は，どのノート・オフの方法によっても認識し，同様に取り扱わなければならない。

ノート・オン(9nH)，ノート・オフ(8nH)の使い方は次の通りである。

1. ベロシティを検出しないキーボードでは，キー・オン時には，9nH, kk, 40H(64)を送り，キー・オフ時には，9nH, kk, 00H または，8nH, kk, 40H(64)を送る。

2. キー・オン・ベロシティだけを検出するキーボードでは，キー・オン時には，9nH, kk, vv を送り，キー・オフ時には，9nH, kk, 00H または，8nH, kk, 40H(64)を送る。

3. キー・オン・ベロシティ，キー・オフ・ベロシティのどちらも検出するキーボードでは，キー・オン時には，9nH，kk，vv を送り，キー・オフ時には，8nH，kk，uu を送る。

＜日本語版注＞

kk は，ノート番号，vv は，オン・ベロシティ，uu は，オフ・ベロシティ。

オフ・ベロシティに対応した音源が，ベロシティ値 “0” のノート・オン・メッセージを受け取った時は，通常ベロシティ値 “64” でノート・オフ・メッセージを受け取ったものと見なして処理をする。

コントロール・チェンジ

コントロール・チェンジ・メッセージは，音色の中に含まれるパラメーター（VCF のカット・オフやエンベロープのディケイなど）を割り当てて送るためのものではなく，定まった音色を鍵盤以外の操作子によってモディファイするような時に使うものである。

コントロール・チェンジ・メッセージには，バンク・セレクト・メッセージ，レジスタード・パラメーター番号（RPN）及びノンレジスタード・パラメーター番号（NRPN）メッセージのような，特別なメッセージもある。

コントロール番号

すべてのコントロール番号の定義は,MMA と AMEI との合意の上で決定される。Table Ⅲでリストされているコントロール番号は，標準的な電子楽器に関連するものである。しかしながら，ライティング・コントロールのように，MIDI を使う多くの非楽器分野においては，それらの裁量で決められたコントロール番号を使用するであろう。この時，コントロール番号の数には制限があるため，楽器及び非楽器分野における，現在と今後で採用するすべての可能な効果に番号を割り当てることは不可能であろう。したがって，コントロール番号は，一般的に楽器に関連する目的のためにだけ割り当てられる。

処理系作成者は非標準的なコントローラー割当てをしている装置があれば,利用者にそのことを知らせるべきである。コントローラーが非楽器分野で使用されていたとしても,Table Ⅱで詳述されるフォーマットの範囲内で使用しなければいけない。処理系作成者は，MMA または AMEI を通じて新たな番号の割り当てを要求できる。コントローラーの割当てテーブルを，すべての製品の取扱い説明書に載せることが望ましい。

デバイス固有パラメーター制御をしたい場合には，多くのパラメーターに対応するために，ノンレジスタード・パラメータ番号とデータ・エントリー・コントローラー（データ・エントリー，データ・インクリメント及びデータ・デクリメント・メッセージ）とを使用すべきである。このことによって，同じコントロール番号に応答する装置の予期しない干渉による誤動作が緩和される。

コントロール番号は現在，0 ～ 119（コントロール番号 120 ～ 127 は，チャンネル・モード・メッセージとして採用され，コントロール・チェンジではない）の 120 個があり，次のようにコントロール番号 32 ～ 63 は，コントロール番号 0 ～ 31 に対応する LSB として定められている。コントロール番号の分類は次の通りである。

0	**バンク・セレクト（MSB）**
1 ～ 31	**2 バイト・データの操作子の MSB**
32	**バンク・セレクト（LSB）**
33 ～ 63	**1-31 操作子の LSB**
64 ～ 95	**1 バイト・データの操作子**
96 ～ 101	**インクリメント／デクリメントとパラメーター番号**
102 ～ 119	**未定義の 1 バイト操作子**

コントロール番号は大きく2つの意味に分けられる。第一は,受信側の “効果” に割り当てられ,それは送信側のどんな操作子によってコントロールされてもよい。第二は，送信側の “操作子” の種類に割り当てられ，それは受信側でどんな効果のコントロールに用い

てもよい。例えば，コントロール番号が1番のモジュレーション・ホイールは，モジュレーション・デプスという“効果”として定義され，2番のブレス･コントローラーは，ブレス･タイプの操作子であると定義されている。さらに単一のコントロール番号は，いくつかのパラメーターを変更するように用いても良く，コントロール番号は各種のカテゴリーによって分類される。

コントロール番号の 0 及び 32 は，バンク・セレクトに割り当てられて，プログラム・チェンジ・メッセージの拡張として用いられる。

コントロール番号 1 ～ 31 は，ペダル，レバーまたはホイール操作などによって主に連続可変的なコントロールをするものに割り当てられる。コントロール番号 33 ～ 63 は，コントロール番号 1 ～ 31 のそれぞれに対応した下位 7 ビットとして，より高精度が要求される時に使用される。例えば，コントロール番号 7 は，その次のデータ・バイトで 128 段階の値を表現できるが，さらにコントロール番号 39 によって下位を 128 段階まで高精度に表せる。すなわち，コントロール番号7と 39 とを使って，最大 14 ビット（16384 段階）の精度を得ることができる。上位の数ビットだけを有効にして利用することはさしつかえない。

もしも，128 段階の精度で充分であれば LSB を省略することができる。MSB と LSB の両方を使う場合でも，LSB だけが変化した時には，送信側は，それだけを送って MSB の送信を省略してもよい。もし大きな変化があれば，再び MSB を送る。MSB を受信した時は，受信側は LSB を“0”に設定しなければならない。

コントロール番号 64 以上で扱う値は,単一バイトであり LSB をもたない。それらのうちコントロール番号 64 ～ 69 はスイッチ機能(ダンパー・ペダルなど)として定義され，コントロール番号 91 ～ 95 はエフェクト用のデプス制御に定義されている。

コントロール番号 64 ～ 69 は，本来，スイッチ動作（サステイン，ソフト・ペダルなど）を想定して割り当てられているが，連続的な値を送ることもできる。例えば，コントロール番号 64（ダンパー・ペダル）で連続的な値を送り，受信側でそれに応じて減衰時間を変化させれば，ハーフ・ダンプの効果を得ることができる。送信側がスイッチ動作をする場合は，データは“0”（オフ）と“127”（オン）とを送る。一方，受信側がスイッチ動作をする場合は，コントロール番号“0 ～ 63”をオフ，コントロール番号“64 ～ 127”をオン，として認識しなければならない。

コントロール番号 96 ～ 119 は，使用方法または使用目的が特殊なものに割り当てられている。例えば，その内のコントロール番号 98（NPRN-LSB），99（NRPN-MSB），100（RPN-LSB），101（RPN-MSB）は，コントロール番号6（データ・エントリー MSB），38（データ・エントリー LSB）及びコントロール番号 96（データ・インクリメント），97（データ・デクリメント）と組み合わせて使用する。

グローバル・コントローラー

受信側機器がモード 4（オムニ・オフ，モノ）になっていて，複数の MIDI チャンネルに応答できる時は，個々のチャンネルの代わりに，あるチャンネルにのみコントロール・チェンジ・メッセージを送ることで，すべてのボイスをコントロールすることができる。このチャンネルをグローバル・チャンネルと呼び，“ベーシック・チャンネル － 1”に設定される。例えば，受信側がモード4の状態でチャンネル6～ 12 に応答している時，すなわちベーシック･チャンネルが6の時，チャンネル5で受信されたコントロール･チェンジメッセージは，同じメッセージをチャンネル6～ 12 でそれぞれ受け取ったのと，同じ結果をもたらす。ベーシック・チャンネルが1の時は，グローバル・チャンネルは 16 になる。ただし，受信側は必ずしもこの機能を持つとは限らない。

汎用操作子

コントロール番号 16 ～ 19 及び 80 ～ 83 は，汎用操作子として定義されている。それらは特定の機能を持つものではなく，どのように用いてもよい。コントロール番号 16 ～ 19 は2バイト操作子であって,コントロール番号 48 ～ 51 がその LSB となる。なお,コントロール番号 80 ～ 83 は1バイト操作子である。

コントローラーの効果

送信側は，操作子の動作する可変範囲全体を，MSB の 0(00H)から 127(7FH)までに対応させて送るのがよい。その場合，効果が最小になるべき位置を“0”，最大の効果を期待する位置を“127”で表す。通常時の位置が可変範囲のほぼ中央にあって，その両側に移動することで何らかの効果を生じるような場合は，中央位置の値として“64”(40H)を送る。

コントローラーの値は，ほとんどの場合で“0”はその効果がないことを，“127”は効果が最大であることを意味するが，バランス，パン，エクスプレッションは例外で，次のように定義される。

バランス　コントロール番号8(08H)として採用されており，0 は左またはロワーの音源が最大音量に，64(40H)は等バランス，127(7FH)は右またはアッパーの音源が最大音量となる。この操作子は，2つの異なる音源の間の音量バランスを設定する。

パン　コントロール番号 10(0AH)として採用されており，0 は左端，64(40H)は中央，127(7FH)は右端に定位する。この操作子は，単一の音源をステレオ音場に配置するために用いる。

エクスプレッション　コントロール番号 11(0BH)として採用されている。エクスプレッションは，プログラムされたボリュームまたはメイン・ボリュームに対して，音量などのアクセントを付けるために用いる。

バンク・セレクト

バンク・セレクトはコントロール・チェンジに割り当てられているが，特別なコントローラーである。バンク・セレクト・メッセージは，プログラム・チェンジ・メッセージの拡張機能であり，複数のバンクを切り替えるために使用する。バンク・セレクト・メッセージによって，128 を越えるプログラムを選択することができる。また，内部メモリーと外部メモリーとの間を切り替えることもできる。

コントロール番号 00H(MSB)と 20H(LSB)とがバンク・セレクト・メッセージに定義される。送信側は，バンク・セレクトの MSB と LSB とを一対として，MSB，LSB の順で送出しなければならない。ならびにプログラム・チェンジは，バンク・セレクトの対の後に送らなければならない。

バンク・セレクトの MSB，LSB 及びプログラム・チェンジによって特定のプログラムが選ばれる。あるバンクに変わった後に，異なるプログラム・チェンジだけを送ると，そのバンクで他のプログラムが選ばれる。受信側は，バンク・セレクトを受信しただけでプログラムを変えてはならない。バンク・セレクトを受信した時にはそのバンクを記憶しておき，プログラム・チェンジ・メッセージの受信契機をもって，プログラムを変化しなければならない。

14 ビットのバンク・セレクト値は次のようになる。

MSB	LSB	バンク
00H	00H	バンク 1
00H	7FH	バンク 128
01H	00H	バンク 129
7FH	7FH	バンク 16384

プログラムと同様に，バンクも1から数え始める。

レガート・フット・スイッチ

Bn 44 vv Legato Footswitch

vv = 00 ～ 3F Normal

vv = 40 ～ 7F Legato

このコントローラーは，受信機器のモノフォニックのレガートをオンまたはオフするために使用される。レガート・オンされた時，受信機器はモノフォニックなモードに入る。この状態で，現在発音中のノートをオフするためのノート・オフ・メッセージの前に新しいノート・オン・メッセージが受信されると，エンベロープを再アタックせずにピッチが変化させられる。また，その音のアタック部分を発音することなしに，ピッチが変化させられることもある。レガート・オフされた時は，ボイス・アサイメント・モード（ポリフォニックまたはモノフォニック）は，レガート・オン・コマンドを受信する前の状態にもどる。

< 備考 >

このメッセージは，モード 4 によるレガート効果とは異なるものである。また，送信済みのすべてのノート・オンのためのノート・オフの送信の代用にもならない。演奏コントローラーとして明確に意図されているものである。

エフェクト・コントローラー

コントロール番号 91 ～ 95 は，エフェクト1デプスからエフェクト5デプスとして定義され，任意のエフェクトの制御に使用可能である。エフェクト1～5のデプスは順にそれぞれ，外部エフェクト・デプス，トレモロ・デプス，コーラス・デプス，セレステ（デチューン）デプス，フェイザー・デプスが推奨された省略時解釈（デフォルト）である。

サウンド・コントローラー

コントロール番号 70 ～ 79 は“サウンド・コントローラー”として定義されている。処理系作成者及び使用者は，それらの 10 種類のコントローラーに対して，使いたい機能を割り当ててよい。しかしながら，標準化を押し進め，利用者が簡単にセットアップできるようにするために，AMEI と MMA とは，それらのコントローラーに対するデフォルト設定を決めている。処理系作成者は，それらのコントローラーに対し，独自に他の機能を割り当ててもよいが，AMEI と MMA が次に示すデフォルト設定を割り当てたことを理解した上で，独自の機能を定義すべきである。

次の五つのサウンド・コントローラーについてのデフォルト設定が，現在 AMEI と MMA とで定義されている。

コントロール番号	名称	機能
46H(70)	サウンド・コントローラー1	サウンド・バリエーション
47H(71)	サウンド・コントローラー2	ティンバー／ハーモニック・インテンシティ
48H(72)	サウンド・コントローラー3	リリース・タイム
49H(73)	サウンド・コントローラー4	アタック・タイム
4AH(74)	サウンド・コントローラー5	ブライトネス

サウンド・バリエーション・コントローラー（SVC）

Bn 46 vv　　Sound Variation

このコントローラーは，現在使用中の音色のバリエーションを演奏中に選ぶために使用する。プログラム・チェンジとはいくつかの点で異なっていることに注意が必要である。

1. 音色バリエーションはプログラムの一部分である。

2. バリエーションは，例えばサックスとオーバーブロー・サックス，弓で弾いたストリングスとピチカート，普通のギターとミュート・ギターなどのように，通常は，基本音色と関連している。

3. 使用されるべきバリエーションは,ノート･オンのタイミングで決定される。例えば SVC 値が “0”に設定され，ノートが発音し，その後に SVC 値が “36”（24H）に設定されても，発音中のノートは変化しない。SVC 値は,新たなノートによって初めて有効となる。古いノートが減衰する時は,新しい SVC 値に従うのではなく，そのノートがオンした時の SVC 値に従って発音を終える。

SVC は，マルチ・レベル切り替えとして用いられる。バリエーションの楽器のレベルは，00H ～ 7FH の範囲いっぱいにマッピングされるべきである。

ティンバー・コントローラー

Bn 47 vv　　Timber/Harmonic Intensity

Bn 4A vv　　Brightness

ティンバー／ハーモニック・インテンシティは，音色の倍音量の変更に使用する。一般的にはレゾナンスとして使用されることが多い。

ティンバー／ハーモニック・インテンシティは，相対値よりは絶対値として扱われるべきである。

ブライトネス･コントローラーは音色の明るさを変えることを目的とし，ブライトネスは通常，ローパス･フィルターのカットオフ･フリケンシーで対応するのが望ましいが，イコライザーまたはハーモニック・エンハンサー／ハーモニック・エキサイターで対応してもよい。

ブライトネスは,相対的なコントローラーとして扱われるべきであり,データの値が 40H の時は変化せず,値が 40H より小さくなるに従って明るさが失われていき，40H より大きくなるに従ってより明るさを増していく。

エンベロープ・タイム・コントローラー

Bn 48 vv　　リリース・タイム

Bn 49 vv　　アタック・タイム

vv=00 ～ 3F　　より短め時間（00= もっとも短く）

vv=40　　変化なし

v=41 ～ 7F　　より長め時間（7F= もっとも長く）

これらのコントローラーは，サウンド（音）のアタック・タイムとリリース・タイムとを，その設定値に対して相対的に変化させるものである。どのエンベロープを変化させるかは任意である。これらのコントローラーは，リリースまたはアタック動作に入るすべてのエンベロープを変化させるべきであるが，すでに動作開始しているエンベロープに対して変化させるかどうかは，任意である。

ポルタメント・コントローラー

Bn 54 kk

n = チャンネル

kk= ソース・ノート番号

ポルタメント・コントローラー（以下 PTC）は，ポリフォニックのポルタメントをコントロールするために，どのノート番号（ソース・ノート番号）からポルタメントを開始するかを指定するものであり，それに続くノート・オンのノート番号（デスティネーション・ノート番号）に関連づけられる。通常，ノンリアルタイムで作成されたシーケンス・データの中で使用され，和音の特定のノートだけをレガートで音程を変えたりすることができる。

PTC 直後に受信したノート・オンは，ソース・ノート番号のピッチから連続的にピッチが変化して発音する。この時，ソース・ノート番号と一致するノート番号で発音中のボイスがある場合,このボイスは PTC 直後のノート・オンによって,再アタックなしで新たな音程にピッチが変化し発音が継続される。なお，ソース・ノート番号と一致するノート番号で発音中のボイスがある時，新しいボイスはアサインされるべきではない。ピッチ変化の速さは，ポルタメント・タイムの設定値で決まる（ポルタメント・オン／ポルタメント・オフは無視される）。

ポリ・モードの場合，PTC の受信は，サステイン中あるいはリリース中のボイスのピッチに影響を与えない。モノ・モードまたはレガート・フット・スイッチがオンの場合，発音中の音のピッチは，新たなノート・オンとともに PTC で指定されたソース・ノート番号のピッチにジャンプし，その後，新たなノート・オンのノート番号のピッチに向かって，ポルタメント・タイムで設定されているレートで変化する。

すべてのモードにおいて，オンしているノートは，それと対になる同一チャンネル，同一ノート番号（PTC で指定されたソース・ノート番号ではない）のノート・オフによってオフされる。PTC 及びそれに続くノート・オンによってピッチ変化させた場合においても，ピッチ変化する元のノートに対応するノート・オフを送信しなければならない。ピッチ変化したボイスは，PTC に続くノート・オンと対になるノート・オフによってオフされる。

PTC は，同一チャンネルの次のノート・オンにのみ作用する（言い替えれば，ノート・オン受信後はリセットされる）。また，ソース・ノート番号と一致しないノートのピッチには影響を及ぼさない。

例 1:

MIDI メッセージ	説明	結果
90 3C 40	Note On#60	#60 オン（中央 C）
B0 54 3C	PTC from#60	変化なし
90 40 40	Note On#64	ピッチが #60 から #64 へ滑らかに変化
80 3C 40	Note Off#60	変化なし
80 40 40	Note Off#64	#64 オフ

例 2:

MIDI メッセージ	説明	結果
B0 54 3C	PTC from#60	変化なし
90 40 40	Note On#64	#60 でオンし，ピッチが #64 へ滑らかに変化
80 40 40	Note Off#64	#64 オフ

レジスタード／ノンレジスタード・パラメーター番号

レジスタード及びノンレジスタード・パラメーター番号は，音色や演奏表現などに関するパラメーターを示すために用いられる。後述するように，レジスタード・パラメーター番号（RPN）は，MMA 及び AMEI によって合意されたものであり，一方ノンレジスタード・パラメーター番号（NRPN）は，個々の処理系作成者が必要に応じて自由に割り当てて使用できるものである。

基本的なパラメーター番号の使い方のルールを以下に示す。

1. ノンレジスタード・パラメーター番号（NRPN）は，メーカーが自由に設定して使用できる。そのリストは，取り扱い説明書等に明示。

2. ノンレジスタード・パラメーター番号（NRPN）は，電源オン時にディスエーブルにすることにより，異なる機器が接続された場合の混乱を避けることができる。

3. 受信側は，ノンレジスタード（及びレジスタード）パラメーター番号の受信がイネーブルされた後，パラメーター番号の LSB と MSB の両方を受信するまでは，パラメーターの変更をすべきでない。

4. パラメーター番号の LSB 及び MSB だけが送られて来た場合，初期状態ではもう一方のパラメーター番号が送られて来るのを待つが，一度正しく認識された後は，どちらか一方だけでもパラメーター番号は変更される。

5. レジスタード・パラメーター番号（RPN）は，AMEI 及び MMA によって合意されたものである。RPN は，電源オン時にイネーブルになっていてもよい。

6. パラメーター番号が認識されると，データ・エントリー，データ・インクリメント，データ・デクリメントを受信することで，パラメーターの値が変更される。

＜日本語版注＞

基本的なパラメーター値の設定手順は，次の通りである。

第 1 にモディファイすべきパラメーターに応じた NRPN または RPN を，コントロール番号 98，99（62H，63H）または 100，101（64H，65H）を使って送る。次に 6，38（06H，26H）（データ・エントリー）を送ってそのパラメーターの値をセットするか，または 96，97（60H，61H）（データ・インクリメント，データ・デクリメント）で値を増減する。

RPN 及び NPRN の詳細は，次に示す通り。なお，レジスタード及びノンレジスタード・パラメーター番号を，総してパラメーター番号と称する。

受信側は，RPN（または NRPN）の受信がイネーブルにされた後，パラメーター番号の LSB と MSB の両方を受信するまでは，送られてくるデータ値を認識すべきではない。また，NRPN と RPN の 2 種の番号は，それぞれ別々に管理しておかなければならない。いちど正しくパラメーター番号を認識した後は，LSB または MSB だけを受け取った場合でも，パラメーター番号を変えることができなければならない。

送信側は，パラメーター番号は MSB と LSB の両方を送るようにする。すでにこれらが送られている場合は，その後はどちらか必要な方だけを送ってもよいが，受信側がいつイネーブルになったかを知ることができないので，新しいパラメータが選ばれるたびになるべく LSB と MSB の両方を送ることが望ましい。

送信例（NRPN）　BnH 62H PL（BnH）63H PM　または　BnH 62H PL　または　BnH 63H PM
（PL，PM は NRPN の LSB 及び MSB）

送信例（RPN）　BnH 64H QL（BnH）65H QM　または　BnH 64H QL　または　BnH 65H QM
（QL，QM は RPN の LSB 及び MSB）

受信側が新しいパラメーター番号を受け取った場合でも，データ・エントリー，データ・インクリメントまたはデータ・デクリメントを受信するまでは，そのパラメーターは以前の値をそのまま維持する。

あるパラメーター番号に対する値を送る時は，データ・エントリーを使う。MSB と LSB のふた通りがあるが，これを片方のみ送るか，両方送るかといった送受信の規則に関しては，コントロール・チェンジメッセージのコントロール番号1～ 31 の場合と同じものが適用される。

送信例　BnH 06H VM 26H VL　または　BnH 06H VM　または　BnH 26H VL
または　BnH 06H VM 06H VM' 06H VM" ……
（VM，VL はデータ値の MSB 及び LSB）

あるパラメーター番号によってすでに設定されている値を増減する時は，データ・インクリメント，データ・デクリメントを送る。このメッセージによるデータの増減の単位については，本来パラメーター番号ごとに定められるべきであるが，そうでない場合は受信側の自由とする。

送信例　BnH 60H xx（インクリメント）
BnH 61H xx（デクリメント）
（xx はダミー。送信側では 0 ～ 7FH のどんな値を用いてもよいが，受信側では無される。）

■パラメーター番号の受け方の例

受信メッセージ	動　作	
Bn 64H QL	パラメーター番号を記憶する。	(xxQL) RPN の LSB
Bn 63H PM	パラメーター番号を記憶する。	(PMxx) NRPN の MSB
Bn 62H PL	パラメーター番号を記憶する。	(PMPL) NRPN の LSB
Bn 65H QM	パラメーター番号を記憶する。	(QMQL) であって QMPL ではない。 (xx はダミー)

■ PM (QM) または，PL (QL) が不定時の VM，VL の受信処理の例

受信メッセージ	動　作
Bn 63H PM	パラメーター番号を記憶する。(PMxx)
Bn 06H VM	電源 ON 直後などで xx がまだセットされていない時は,VM は無視するのが望ましい。
Bn 62H PL	パラメーター番号を記憶する。(PMPL)
Bn 06H VM	VM をパラメーター番号 (PMPL) の上位 7bit にセットする。
Bn 26H VL	VL をパラメーター番号 (PMPL) の下位 7bit にセットする。
Bn 62H PL'	パラメーター番号を記憶する。(PMPL')
Bn 06H VM'	VM' をパラメーター番号 (PMPL') の上位 7bit にセットする。
Bn 65H QM	パラメーター番号を記憶する。(QMxx)
Bn 64H QL	パラメーター番号を記憶する。(QMQL)
Bn 06H VM	VM をパラメーター番号 (QMQL) の上位 7bit にセットする。
Bn 62H PL"	パラメーター番号を記憶する。(PMPL") であって QMPL" ではない。
Bn 06H VM	VM をパラメーター番号 (PMPL") の上位 7bit にセットする。
Bn 06H VM'	VM' をパラメーター番号 (PMPL") の上位 7bit にセットする。

RPN のパラメーター番号の定義は，AMEI と MMA との合意の上で決定される。処理系作成者は AMEI または MMA を通じて新たな番号定義の提案をすることができる。RPN ですでに定義済みの番号は，Table Ⅲ a に示されている。NRPN に対してはどんなパラメーターを割り当ててもよいが，そのリストをその製品の取扱い説明書に公表することとする。

RPN の内で，すでに定義済みのものに関して，次に示す。

ピッチ・ベンド・センシティビティ

ピッチ・ベンド・センシティビティは，RPN 00(MSB)00(LSB)で定義される。データ・エントリーの MSB は半音(100 セント)単位で感度を表し，LSB は 100 ／ 128 セント単位で感度を表す。例えば，MSB=01，LSB=00 は，± 1 半音(計 2 半音)を表す。

マスター・チューニング

RPN 00(MSB)01(LSB)及び 00 02 は，マスター・チューニング・コントロールに使用され，次のように定義される。

RPN 00 01： ファイン・チューン

分解能： 100 ／ 8192 セント

範囲 ： (100 ／ 8192)*(-8192)～(100 ／ 8192)*(+8191)

コントロール値		変位[セント]
MSB	LSB	
00H	00H	(100 ／ 8192)*(-8192)
40H	00H	(100 ／ 8192)* 0
7FH	7FH	(100 ／ 8192)*(+8191)

RPN 00 02： コース・チューン

分解能： 100 セント

範囲 ： 100*(-64)～ 100*(+63)

コントロール値		変位[セント]
MSB	LSB	
00H	xx	100*(-64)
40H	xx	100*0
7FH	xx	100*(+63)

プログラム・チェンジ

このメッセージは MIDI 機器で，主に音色を変える時に用いられる。このメッセージは選択された音色のパラメーターをいっさい含んでいない。プログラムを構成する各種パラメーターは，MIDI 機器それぞれの間で非常に異なるため，音色を単にその内部番号だけで選択するのが適切である。

プログラム・チェンジ・メッセージは，通常，送信側のパネル操作によって自己の音色を切り換えるとともに，対応する番号を送るようになっている場合が多い。自己では発音しないような機器でも，パネル上のスイッチを使って相手にプログラム・チェンジを送ることができる。

送信側と受信側にまったく同じ音色が存在することは，あまり多くないので，与えられた音色番号に対して音色を割り当てる時には，なんらかの注意が必要である。すなわち，プログラム・チェンジ番号と自己の音色との対応を，送信側または受信側で任意に変更できる機能が望ましい。音色番号の呼び方は機器によりまちまちであるが，MIDI 上では，必ず 00H から始まって 7FH まで順に増える番号を使う。

送信側で音色を変えた時，同時に受信側でも音色変化を生ずるのが必ずしも望ましくないことがある。したがって，プログラム・チェンジの送信または受信をディスエーブルにする，何らかの手段があることが望ましい。

プログラム・チェンジ・メッセージは，音色の切り換え以外の目的にも使用できる。機器によっては，例えばドラム・マシンのようなものでは，プログラム・チェンジ・メッセージが別のリズム・パターンに切り換えるために用いられることがある。MIDI で制御されるエフェクターでは，プログラム・チェンジ・メッセージはプリセットされた各種のエフェクトを選択することに用いられる。

＜備考＞
バンク・セレクトの説明を参照のこと。

ピッチ・ベンド・チェンジ

この機能は，ピッチの制御を目的とするものであり，メッセージは常に 14 ビット（2バイト）で送られる。コントロール・チェンジでは，LSB または MSB のいずれかのみを送信することができるが，ピッチ・ベンド・メッセージは，常に2つのデータ・バイトを伴って送信される。これは，ピッチの変化に特に敏感な人間の聴力を考慮に入れている。

ピッチ・ベンド・メッセージは，ステータス・バイト，LSB 及び MSB の3バイトから構成される。ピッチが最低になる時のデータ・バイトの値は，00，00（LSB=00H，MSB=00H）である。ピッチの中央（非効果）位置は 00，64（LSB=00H，MSB=40H）のデータ・バイト値で得られる。ピッチが最高になる時のデータ・バイトの値は 127，127（LSB=7FH，MSB=7FH）である。

ピッチ・ベンドの感度は，受信側で設定される。この感度はまた，レジスタード・パラメーター番号（LSB=00H，MSB=00H）を介して送信側より設定することもできる。

アフタータッチ

アフタータッチには，2種類のメッセージがある。チャンネルごとのものと，演奏された各ノートにかかるものである。これらは，そのステータス・バイトによって区別される。いずれの場合も，アフタータッチ・バリューは，キーを水平方向（前後または左右）に動かすことによって，あるいはキーを押し下げることによって決定される。

ウインド・コントローラーのような場合には，アタックの後に吹き込みの強さを増加することで，アフタータッチを送ることができる。アフタータッチによって生ずるモディファイの種類は，受信側によって決定され，音量，音質，またはビブラートなどに割り当てることができる。

“チャンネル・プレッシャー（DnH，0vvvvvvv）”メッセージを受けた場合，アフタータッチは，そのチャンネルで演奏するすべての音にかける。

“ポリフォニック・キー・プレッシャー（AnH，0kkkkkkk，0vvvvvvv）”メッセージを受けた場合，該当するチャンネルの各ノートごとに独立にアフタータッチをかける。

ここで 0kkkkkkk はノート番号，0vvvvvvv はプレッシャー値である。

チャンネル・モード・メッセージ

オール・サウンド・オフ	120
リセット・オール・コントローラー	121
ローカル・コントロール	122
オール・ノート・オフ	123
オムニ・オフ	124
オムニ・オン	125
モノ・モード・オン	126
ポリ・モード・オン	127

チャンネル・モード・メッセージは，コントロール・チェンジと同じステータス・バイトで送られる。そのメッセージの第2バイトはモード・メッセージを意味する 120(78H)と 127(7FH)との間にある。チャンネル・モード・メッセージは，ボイス・メッセージがどのように受信されるかを決定する。これによって，受信側はノートをモノフォニックで演奏するか，あるいはポリフォニックで演奏するか，そして受信側がひとつの特定のチャンネル上で送られるデータのみを受信するか，あるいはすべてのチャンネルを受信するかを決定する。

オール・ノート・オフ・メッセージとしてのチャンネル・モード・メッセージ

チャンネル・モード・メッセージ(124 ～ 127)はオール・ノート・オフ・メッセージとしても機能する。オール・ノート・オフ・メッセージ(123 ～ 127)は，ベーシック・チャンネルに基づくすべてのボイスをオフする。いかなる場合においても，ノート・オンされた音をオフするためのノート・オフ・メッセージの代わりにオール・ノート・オフ・メッセージ(123 ～ 127)が用いられてはならない。ノート・オフ・メッセージは，ノート・オン・メッセージと対応しなければならないため，オール・ノート・オフ・メッセージ(123 ～ 127)は，受信側で無視してもよい。

機器のベーシック・チャンネル

チャンネル・モード・メッセージは，現在のモードに関係なく，受信側のベーシック・チャンネル上で送信された時のみに認識される。ベーシック・チャンネルは，送信側または受信側で，操作ボタンまたはシステム・エクスクルーシブ・メッセージのいずれかにより設定される。チャンネル・モード・メッセージは機器のベーシック・チャンネル上でのみ，送信及び受信される。

受信側のモード(オムニ・オン／オムニ・オフ及びポリ／モノ)

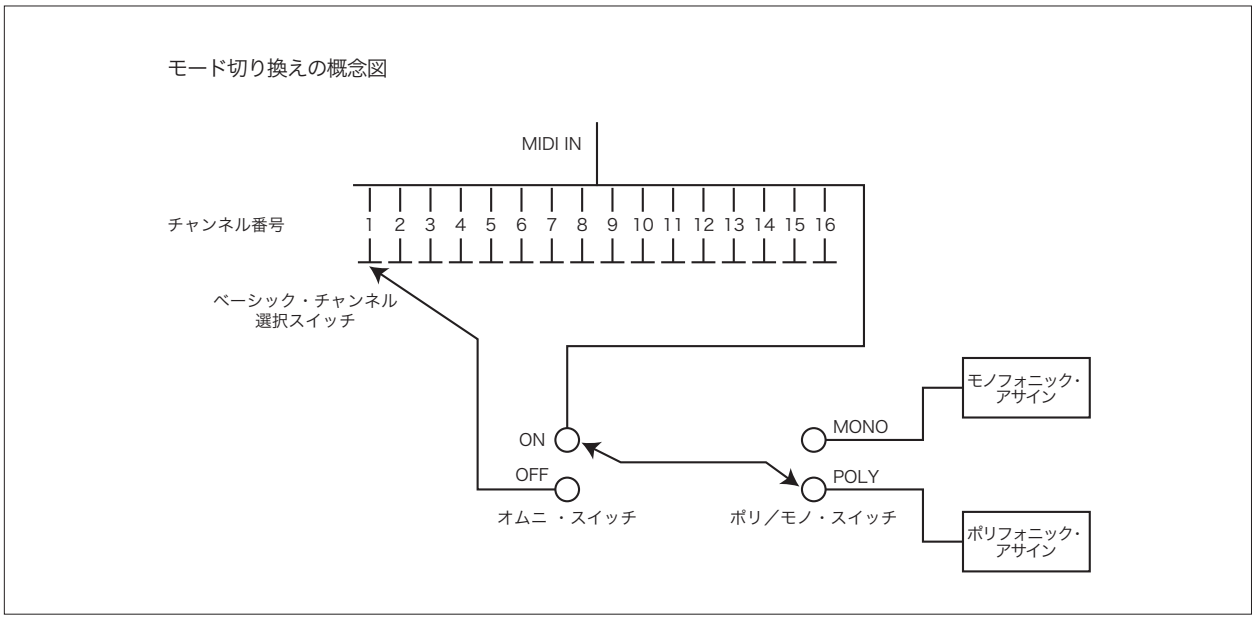
受信側がどのようにしてボイス・メッセージを認識するかを，オムニ・オン／オムニ・オフ及びポリ／モノの組合せによる4つのモードのひとつに設定することができる。

モード	オムニ・オン／オフ	ポリ／モノ
Mode 1	Omni on	Poly
Mode 2	Omni on	Mono
Mode 3	Omni off	Poly
Mode 4	Omni off	Mono

ポリとモノとが，複数のノートが同時に受信された時に受信側のボイスがどのようにして割り当てられるかを決定する。モノ・モードにおいては，受信側における各ボイスが特定の MIDI チャンネル上のノート・メッセージにモノフォニックに応答する。これは単一の機器内に複数個の単音シンセサイザーを有するようなものである。ポリ･モードにおいては，受信側のボイスがノート･メッセージにポリフォニックに応答する。

これら4つのモードは受信側の操作ボタンで選択できるようにしてよい。しかし，受信側が送信側からのデータを認識しない，あるいは正しく応答しないようなモードに設定されることもあるので注意が必要である。受信側が送信側のモードを知る方法がないため，受信側が異なるモードに手動で設定されている場合には，送信側が期待したように受信側がメッセージを読み取る保証はない。

オムニ・オンにすると，受信側はすべての MIDI チャンネルのボイス・メッセージに応答するため，機器のベーシック・チャンネルには関係なく動作させることができる。オムニ・オフでは，受信側は設定されているベーシック・チャンネルのボイス・メッセージにだけ応答する。



受信側がポリ・モードにあり，受信側のチャンネルで複数のノートが受信された場合，これらのノートは受信側のボイス数の範囲内で同時に演奏される。オムニ・オンの場合はすべての MIDI チャンネルで受信可能であり，オムニ・オフの場合は受信側のベーシック・チャンネルのみ受信可能である。

受信側がモノ・モードにある場合，オムニ・モードがオンであるかオフであるかにより，ノートの割り当て方が異なる。

モノ・モード

モノ・モードは，特にギター・コントローラーから MIDI データを受信する場合に有用であるが，キーボードその他のコントローラーでも，同様に使用できる。これは個々の音の独立したピッチ・ベンド，2音間での異なるスピードのポルタメントなどの目的に有用である。

受信側が，次に来るノート・メッセージに対してレガート風に応答してもいいという点も，モノ・モードを使用するひとつの理由として挙げられる。ノート･オンが受信され，そのノートのオフが受信される前に次のノート･オンが受信された場合には，受信側はエンベロープをリスタートせず，ピッチを後者のノートに変更すべきである。送信側が受信側にレガート風の応答を望むためには，ノート・メッセージのタイミングは次のようにする。

<Note On #1> <Note On #2> <Note Off #1>, etc.

ノート・オフはすべてのノートに対して送られなければならないという，MIDI の規定はそのまま適用される。レガート・フット・スイッチの項も参照のこと。

オムニ・オフ／モノ

モノ・メッセージの第3バイトは，単音のボイス・メッセージが送られるチャンネル数を指定する。文字Mが受信可能な1から 16 の範囲にわたる MIDI チャンネルの数を表し，文字Nがベーシック・チャンネル番号（1 ～ 16）を表すとすれば，使用できるチャンネルは 16 を最大として，現在のベーシック・チャンネル（N）から N+M − 1 までとなる。“M=0”は特殊な場合で，受信側をベーシック・チャンネルNから 16 まで，チャンネルあたりひとつずつ，そのすべてのボイスに割り当てるように動作する。

送信側

送信側がオムニ・オフ，モノ・モードに設定された場合，ボイス・メッセージはチャンネル N から N+M − 1 上で送られる。これは個々のボイスが別々のチャンネル上で送られることを意味する。ベーシック・チャンネル番号 N が 1 より大きな値に設定されている 16 ボイスの機器から送信する場合，N+M − 1 は 16 より大きくなり，実在しない 16 より上のチャンネルに対して割り当てられたノートは送信されない。16 ボイス全部の送信を可能とする場合には，ベーシック・チャンネル N は 1 に設定されるべきである。

例えば，ベーシック･チャンネルが3に設定された4ボイスの場合は，チャンネル3から6でノート･メッセージを送信することになる。

受信側

オムニ・オフ，モノ・モードに設定された受信側においては，チャンネルNから N+M − 1 で受信されたボイス・メッセージは，モノフォニックにその内部のボイス1からMに割り当てられる。もし N=1 で M=16（最大）であれば，メッセージはチャンネル1から 16 で受信される。ひとつのチャンネルに複数個のノート･オン･メッセージが送られた場合，受信側の動作は特定できない。このモードでは，ひとつのノート（またはボイス）のみが，ひとつの MIDI チャンネルに割り当て可能である。M=0 は特殊な場合で，ベーシック・チャンネルNから 16 まで，チャンネル当たりひとつずつのボイスを，可能な限り割り当てることができる。

オムニ・オン／モノ

送信側がオムニ・オン，モノ・モードに設定された場合，ひとつのボイスを対象としてボイス・メッセージはチャンネルNで送られる。受信側がオムニ・オン，モノ・モードに設定された場合，任意のチャンネルで受信されたボイス・メッセージは単一のボイスに割り当てられる。受信されている MIDI チャンネルの数，チャンネル当たり複音であるかどうかに関係なく，受信側は一度にひとつのノートしか発音しない。

送信側

送信側は受信側をモノ・モードに設定するためにモノ・モード・オン・メッセージを送信できる。しかしながら，受信側がモノ･モードになれないことがあるので，送信側は複数個のノート･メッセージを同時に送り続けてもよい。また，たとえ送信側と受信側の双方がモノフォニックに演奏している場合でも，複数のノート・オン・メッセージを同時に送ることができる。

受信側　　ノート・オン・メッセージがオムニ・オン，モノ・モードに対して送られている場合，受信側はチャンネル番号に関係なく，そのノートを発音する。モノ・モード・オン・メッセージを受信し，オムニ・オンである場合，Mは無視されて受信側は依然として単音で動作する。

受信側で取り得ないモード

送信側が，受信側で取り得ないモードを指定する場合がある。例えば，送信側がオムニ・オフ，モノで M=2 を指定しても，受信側はオムニ・オンのモノまたはポリにしかなり得ない場合である。このような状況で，受信側は(1)または(2)のうち，いずれか一方の動作を行なってよい。

(1) メッセージを無視する。

(2) オムニ・オンのポリとなり，どちらのチャンネルによる発音も行われる。

なお，(2)の動作の方が，受信側が両方のチャンネルの音を発音できるので望ましい。

＜備考＞

受信側はモード 4 を取り得ない場合には，メッセージを無視するかまたは次に示す動作を行ってよい。

受信側のモード選択

	MONO		POLY
	M＝1	M≠1	
Omni Off	use Mode 2	use Mode 1	Mode 3
Omni On	Mode 2		Mode 1

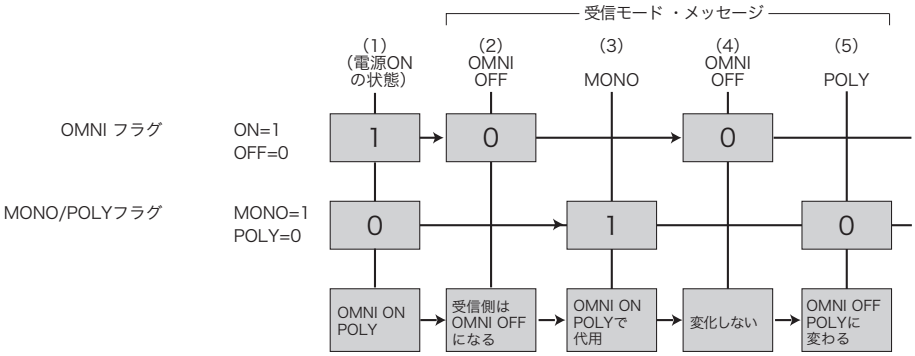
送信側では，個々のモード・メッセージに対して受信側が正しく反応したかどうかを知ることができないが，受信側が上記の表のように対応していれば，予期せぬ結果を最小限にできる。しかし，その場合でも，送信側で特定のボイスに対してピッチ・ベンドまたはモジュレーションをかけたりすると，受信側ではすべてのボイスが反応して，送信側で期待した結果が得られないのは明白である。したがって，受信側が上記のような代用のモードになっている時は，可能ならばノート・オン／ノート・オフ・メッセージ以外は無視するようにするのがよい。

送信側は，オムニ・オンまたはオムニ・オフ，ならびにポリまたはモノのメッセージを，任意の順番で送信してよい。したがって，受信側でこれらのモード・メッセージを正しく認識するためには，オムニ・オン／オムニ・オフ用と，ポリ／モノ用のふたつのフラグが必要で，これらの組合せによって生じる4つの場合に応じて，適切なモードを取る必要がある。

例えば，オムニ・オフ，モノのモード・メッセージを受けた受信側が，代用のオムニ・オン，ポリのモードになっている状態で，さらにオムニ・オフを受信しても，それを認識してオムニ・オフ，ポリに変わるのは正しくない。なぜなら，その時点では送信側からはまだオムニ・オフ，モノを指定していることにかわりがないからである。その後ポリ・モード・オンを受信して初めて代用のオムニ・オンが解除され，オムニ・オフ，ポリに変わることになる。

送信側は，モードを変える時に，オムニ・オン／オムニ・オフまたはポリ／モノを重複して送ってよい。例えば，受信側にモード3を指定するためにオムニ・オフ，ポリを送っておいて，後にモード1を指定するためにオムニ・オン，ポリを送った場合，ポリ・モード・オン・メッセージは前回に続いて2度送られることになる。受信側では再度送られたポリ・モード・オン・メッセージで問題が起こってはならない。

■代用されるモードの例



受信モード ・メッセージ

	Power Up	OMNI-OFF	MONO M>=2	OMNI-OFF	POLY
OMNI MONO/POLY	1 0	0 0	0 1	0 1	0 0
受信可能 モード	1	3	N/A	N/A	3
受信代用 モード			1 (OMNI-ON, POLY)	1 (NoChange)	

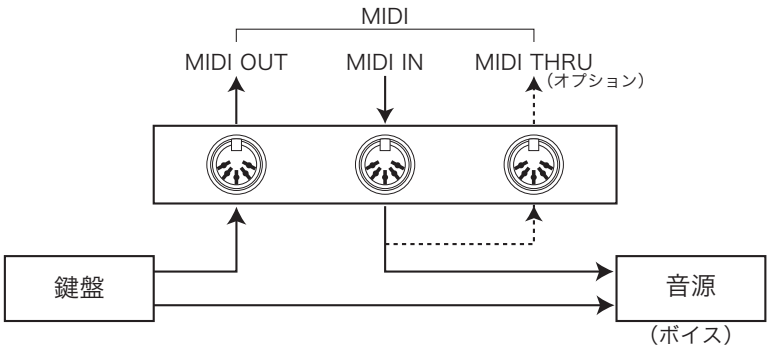
OMNI ON=1, OMNI OFF=0
MONO=1, POLY=0

N/A=受信側に該当するモードが存在しない

オール・ノート・オフ

オール･ノート･オフ(123)は，MIDI によって発音したすべてのボイスを停止する，有効なモード･メッセージである。このメッセージは，さまざまな応用に便利であるが，受信側はこれを認識しなくてもよい。オール・ノート・オフの認識は受信側にとって必須でないので，すべての発音はオール・ノート・オフを送る前に，個々のノート・オフ・メッセージにより停止させなければならない。

MIDI キーボードにおいて，本体のキーボードによる発音と MIDI IN による発音とは区別されるべきである。



機器の構成が前記の図のような時，この機器は，MIDI IN と本体のキーボードによる命令を区別できないことがあり得る。もしオール・ノート・オフを MIDI によって受信すると，本体のキーボードによって演奏している音までも含んで，すべての発音を停止してしまう。これは，オール・ノート・オフ・メッセージの正しい実行ではない。オール・ノート・オフは，MIDI によって発音された音だけを停止しなければならない。もし機器が本体のキーボードと MIDI メッセージを区別できないのなら，オール・ノート・オフは無視されなければならない。

オムニ・オン（モード 1 または 2）の間は，受信側はオール・ノート・オフを無視しなければならない。受信側がオムニ・オフ，ポリ（モード 3）に設定されている時，オール・ノート・オフはベーシック・チャンネル（送られたチャンネル）のノート・オン・メッセージだけをキャンセルする。オムニ・オフ，モノ（モード 4）の場合は，ベーシック・チャンネル（N）から N+M − 1 までのチャンネルのノート・オン・メッセージに対してベーシック・チャンネルのオール・ノート・オフが有効である。

＜備考＞
オール・ノート・オフについては，補足説明と運用上の注意も参照のこと。

オール・サウンド・オフ

オール・サウンド・オフ（120）は，特定の MIDI チャンネルで受信した機器の演奏しているすべての音を消音するためのモード・メッセージである。

受信に際してはオール・ノート・オフと同じ決まりに従う。例えば，オムニ・オン状態では無視すべきである。

リセット・オール・コントローラー

リセット・オール・コントローラー（121）を受信した時は，すべてのコントローラー（コンティニュアス・コントローラー，スイッチ・コントローラー，ピッチ・ベンド，アフタータッチ）の状態を理想的な初期値（モジュレーション・ホイールは0，ピッチ・ベンドはセンターなど）にリセットしなければならない。受信に際しては，オール・ノート・オフと同じ決まりに従う。例えば，オムニ・オン状態では無視すべきである。

このコマンドを解釈しない機器が接続される事を考慮するならば，すべてのコントローラーの初期値となるべき値を最初に送り，その後，このメッセージを送信するべきである。このメッセージを認識する機器は，規定された状態になり，一方，メッセージを認識しない機器は，初期状態のままに置かれる。

ローカル・コントロール

ローカルコントロール(122)は，内部での情報の伝達(例えば，キーボードから音源へ)をオン／オフするために使われる。3バイト目が0であるメッセージ(ローカル・オフ)を受信すると，内部結線はオフとなり，キーボードのデータは MIDI OUT へ出るだけとなり，音源は MIDI IN より受けるデータでのみ動作するようになる。3 バイト目が 127(7FH)のメッセージ(ローカル・オン)を受けると，通常の動作に戻る。

スレーブとして接続されている鍵盤楽器の場合，その楽器のキーボードを内部音源から切り離せば，MIDI による演奏がキーボードによって妨げられることはない。これはまた，自身のキーボードの演奏情報を見ない分，MIDI の処理速度を向上させるかもしれない。楽器は，電源 ON 時にはローカル・オンであるべきである。また楽器は，ローカル・オフの間も，自身のキーボードからの MIDI 情報は送り続けるべきである。

システム・コモン・メッセージ

MIDI タイムコード・クォーター・フレーム	F1H
ソング・ポジション・ポインター	F2H
ソング・セレクト	F3H
チューン・リクエスト	F6H
EOX（エンド・オブ・エクスクルーシブ）	F7H

MTCクォーター・フレーム

このメッセージは MIDI タイムコード（MTC）のひとつであり，8つのメッセージ・シーケンスの中でフレーム，秒，分，時間カウントの情報を扱う。

＜備考＞

MIDI タイムコードのクォーター・フレーム（システム・コモン），フルフレームまたはユーザー・ビット（システム・エクスクルーシブ）

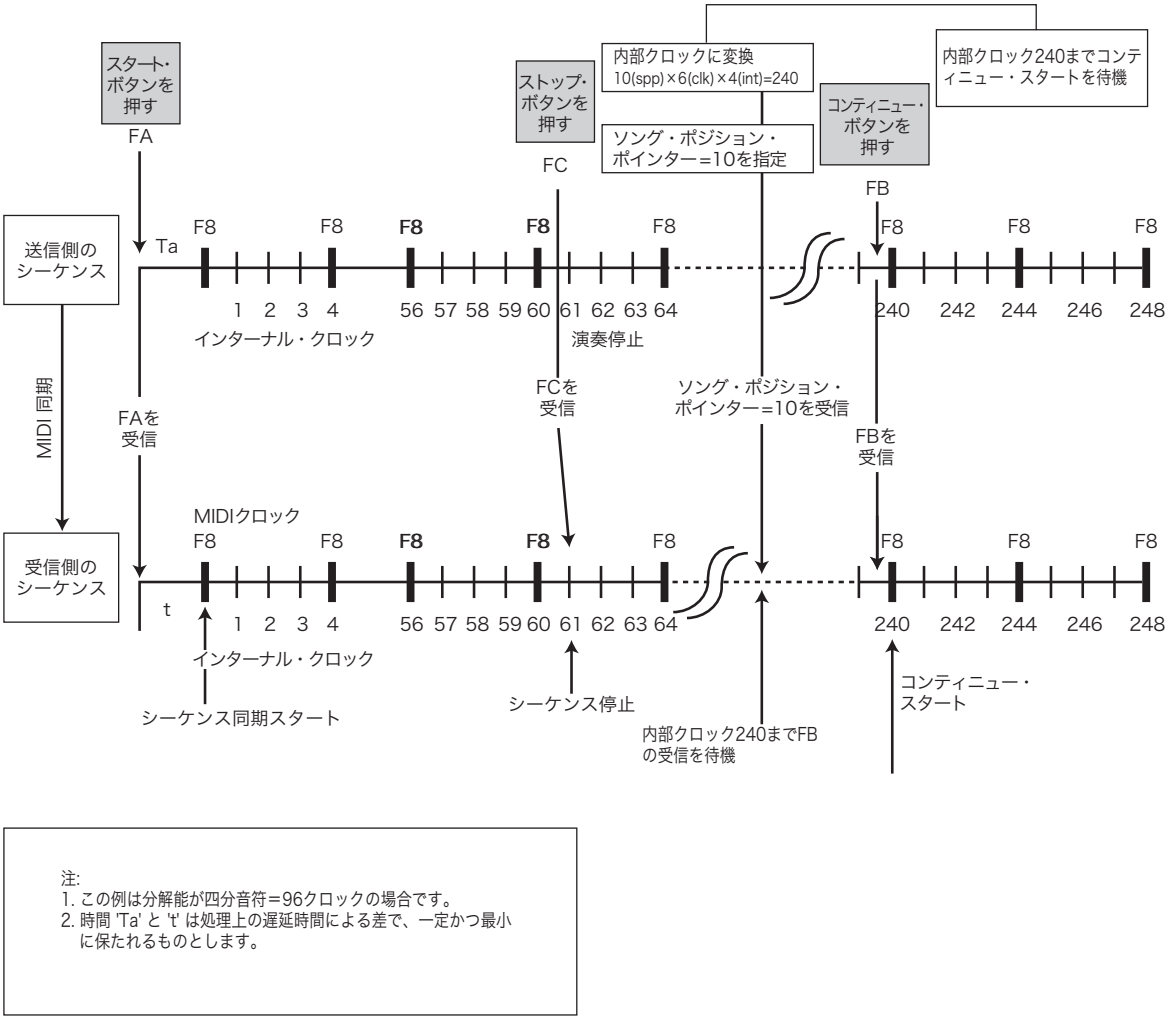
詳しくは，“MIDI タイムコード”を参照。

ソング・ポジション・ポインター

シーケンサーのソング・ポジション（SP）とは，曲の最初からの経過時間を1ビートを6MIDI クロックとする，MIDI ビートの数で表したもので，曲の頭以外の位置から演奏を始めるのに使用される。通常，シーケンサーのスタートが指示された時に，0にセットされる。そして，ストップが指示されるまで，6MIDI クロックごとに増加する。コンティニューが指示された時は，現在のソング・ポジションから増加する。この現在のソング・ポジションは，ソング・ポジション・ポインター・メッセージによって送受信することが可能で，受信側のソング・ポジションは，このメッセージの受信によって変更される。

ソング・ポジションは，常に MIDI クロック（F8H）6回の倍数なので，ソング・ポジションの最小単位は6MIDI クロック，つまり 16 分音符になる。これは，シーケンサー内部のタイムベースに変換される。

例を示すと，ソング・ポジション・ポインターが 10 の時，6MIDI クロックで乗算して（10 × 6=60），さらにシーケンサーのタイムベースを乗算する。タイムベースが1拍あたり 96 クロックだとすると，MIDI クロック間のタイムベースは4クロックなので，結果は 240（60 × 4=240）となり，演奏開始のために，シーケンサー内部のポインターを 240 に合わせる。



注:
1. この例は分解能が四分音符=96クロックの場合です。
2. 時間 'Ta' と 't' は処理上の遅延時間による差で、一定かつ最小に保たれるものとします。

スタート(FAH)は，ソング・ポジションの値0とコンティニュー(FBH)とが合成されたコマンドであると考えられる。しかし，このスタート(FAH)は，コンティニュー(FBH)より多く使われると思われるので，受信側では，シーケンスが曲の途中でストップ状態の時でも，次にスタートかコンティニューかのいずれにも対応できるようにしておく必要がある。

MIDI シンク・モードをもたない機器では，ソング・ポジション・ポインター・メッセージは無視されるべきである。

＜備考＞

MIDI シンクモードに関しては，システム・リアルタイム・メッセージの項を参照のこと。

ソング・ポジション・ポインターの推奨される使用法

以前は，MIDI 機器がソング・ポジション・ポインター(SSP)に続いてコンティニュー・メッセージを送信した後，MIDI クロックの送信を始める前に，5秒待つことが推奨されていた。しかし現在，ソング・ポジション・ポインターを受信した機器は，曲中の新しいポジションへロケートを行っている最中であったとしても，コンティニュー・メッセージと，それに続く MIDI クロックを正しく受信することが推奨される。新しいポジションにロケートしたらすぐに機器は，SSP を送信した機器との同期を行わなければならない。

例えば，送信側から4(24MIDI クロック)という SSP メッセージを受け取り，ロケートを行っている間に，コンティニューとさらに3つの MIDI クロックを受信した場合，受信側の機器は，曲頭から 27 クロック目から開始すべきである。

ソング・セレクト

ソング・セレクトは，複数のソングやシーケンスを扱うことのできるシーケンサーなどにおいて，スタート・メッセージが指示された時に演奏するソングやシーケンスを設定するものである。

ソング・ポジションとソング・セレクト

ソング・ポジション・ポインター・メッセージまたはソング・セレクト・メッセージを受信して認識すると，その処理には相当時間を要すると考えられる。演奏メモリー・アドレスへのロケーティング処理の最中にコンティニューが送られ，MIDI タイミング・クロックが送られて来たら，その数を認識してソング・ポジションに加える必要がある。

例えば，4MIDI ビート（24MIDI クロック）のソング・ポジション・ポインター・メッセージを受信し，ロケーティング処理中にコンティニュー及び3つの MIDI タイミング・クロックが送られて来たとすると，シーケンサーは 27MIDI クロック目に相当するポイントから演奏すべきである。ロケーティング処理中に送られて来た MIDI クロックを取り損なうと，同期演奏はできない。

チューン・リクエスト

チューン・リクエストは，シンセサイザーなどのアナログ・タイプの音源の周波数を，自動的にチューニングする時に使用する。どのような方法でどの高さにチューニングするかは，受信側に任されている。

EOX

EOX（エンド・オブ・エクスクルーシブ）は，システム・エクスクルーシブの送信の終わりを示すために使用する。F0H で始まるエクスクルーシブ・メッセージでは，ステータス・バイトに続き，何バイトのデータをも送ることができる。送信側は，エクスクルーシブ・メッセージの終わりには，速やかに EOX（F7H）を送らなければならない。受信側は，安全のために，EOX または他のメッセージ（リアルタイム・メッセージを除く）のステータス・バイトによって終了する。

エクスクルーシブの送信中であっても，そのデータ・バイト中にリアルタイム・メッセージを挿入できる。

システム・リアルタイム・メッセージ

タイミング・クロック	F8H
スタート	FAH
コンティニュー	FBH
ストップ	FCH
アクティブ・センシング	FEH
システム・リセット	FFH

システム・リアルタイム・メッセージは，システム共通の情報であり，チャンネル番号は有しない。システム・リアルタイム・メッセージのうち F8H ～ FCH は，MIDI クロックをベースとした MIDI 機器の同期を行うために使われる。

システム・リアルタイム・メッセージは，いつでも他のステータスやデータ・バイトに対して優先して送られ，MIDI データ列の中やステータス・バイトとデータ・バイトとの間にも挿入することができる。また，システム・リアルタイム・メッセージは，他のメッセージよりも優先処理しなければならない。

タイミング・クロック　MIDI 接続されたシステムは，4分音符あたり 24 の割合で送られるこのメッセージによって同期をとることができる。送信側は演奏時でなくても，タイミング・クロック（F8H）を送信し続けることはさしつかえないが，その場合，自己の持つテンポ情報に従って，正しいタイミング・クロックを出力すべきである。MIDI シンク・モード（MIDI IN のタイミング・クロックに同期するモード）に設定された受信側は，スタート（FAH）またはコンティニュー（FBH）が送られて来るのを待った状態で外部のクロックに同期させることができる。

小節の区切りなどを表すメッセージは，別に設けられていないので，クロックの認識をひとつでも誤ると，最後までそれを回復する手段はない。したがって，これを確実に認識できるように，特に留意しなければならない。

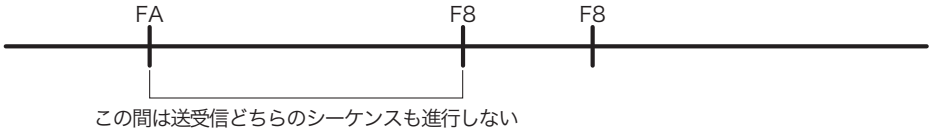
スタート　スタート（FAH）は，マスター（シーケンサーなど）の PLAY ボタンが押された時に送信される。このメッセージを，スレーブとして接続されている受信側が認識し，次に送られて来るタイミング・クロック（F8H）を受信すると，ソングやシーケンスの先頭から演奏を開始する。

コンティニュー　コンティニュー（FBH）は，CONTINUE ボタンが押された時に送信される。次に送られて来るタイミング・クロック（F8H）を受信すると，シーケンスの現在の位置より演奏が再開される。

ストップ　ストップ（FCH）は，STOP ボタンが押された時に送信され，これを受けた受信側はシーケンスの演奏を停止する。

スタート・メッセージ及びコンティニュー・メッセージ

受信側が，MIDI シンク・モードになっている時，次の図のように，スタート（FAH）またはコンティニュー（FBH）が送られて来ても，次にタイミング・クロック（F8H）が送られて来なければ，受信側はシーケンスを進行させてはならない。また，FAH（または FBH）と F8H は，受信側の処理上の理由を考慮して，少なくとも 1ms 以上の間隔をあけたほうが無難であろう。しかし，受信側はスタートやストップの後，最初に送られて来る F8H の受信で，すぐにもスタートできるようにすべきである。



受信側が，内部クロックで動作するモードになっている時，受信したFAHまたはFBHによって，自己のクロックでシーケンスを開始することはさしつかえない。

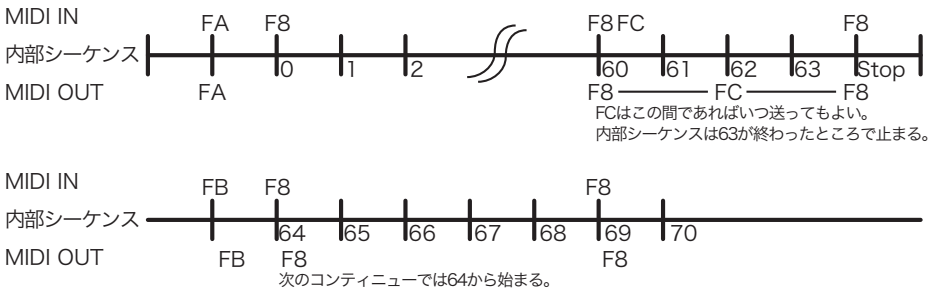
ストップ・メッセージ

マスターのシーケンサーを停止させた時，ただちにストップ(FCH)を送信しなければならない。すると，スレーブに接続された機器も停止する。この時，シーケンサーは演奏停止時の位置を記憶しておく必要がある。そうすることで，コンティニューが送られて来た時に，同じ位置からシーケンスを再開することができる。

受信側は，ストップ(FCH)を受けるとシーケンスの進行を停止し，その後のクロックは無視しなければならない。ただし受信側の都合で，ストップ受信後もしばらくシーケンスを進行させる場合は，ストップ時の位置を記憶しておき，次にコンティニューされた時に，そこから演奏をスタートさせて同期させる。

シーケンス進行によってノート・オンされ，ストップが押される前に，まだノート・オフされていない音に対し，送信側はノート・オフ・メッセージを送って，すべての発音を停止させなければならない。オール・ノート・オフ・メッセージを送ることもできるが，これに対応していない機種もあるので，残ったノートへのノート・オフ・メッセージの代わりに，オール・ノート・オフ・メッセージを送ってはいけない。同様に，コントローラー(ピッチ・ホイール，サステイン・ペダル，など)についても，初期ポジションにもどすことが望ましい。

次の図は，リアルタイムのシーケンサーのように，MIDIクロックを補間して内部クロックを持っている場合の同期の仕方を示す。4分音符=96クロック，すなわちMIDIクロック=4内部クロックの例。



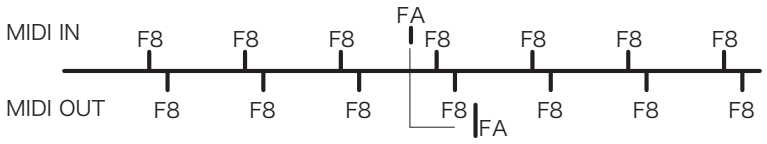
＜日本語版注＞

シーケンスが停止した時，シーケンスが正しくMIDIビート(MIDIクロックで計算して6の倍数)上にある必要はない。この状態でソング・ポジション・ポインターを使用すれば，正しいMIDIビートの位置を受信側に知らせることができる(この時，送信側，受信側共にMIDIクロックの6の倍数に訂正されなければならない)。

クロックとコマンドの関係

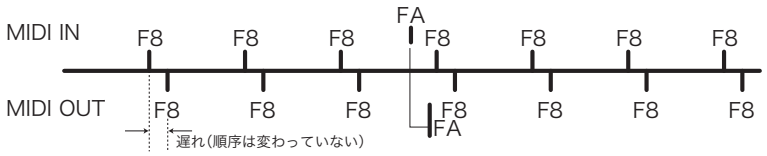
シーケンサーの演奏中には，タイミング情報とボイス情報が MIDI 出力されるが，タイミング情報であるリアルタイム・メッセージは，他のメッセージに優先して送信される。そのために，メッセージの送信順がかわっても差し支えない。しかし，受信したリアルタイム・メッセージ（F8H，FAH，FBH，FCH））を MIDI OUT からエコー出力する場合，クロックとコマンドとの順序が入れかわってはならない。

悪い例：



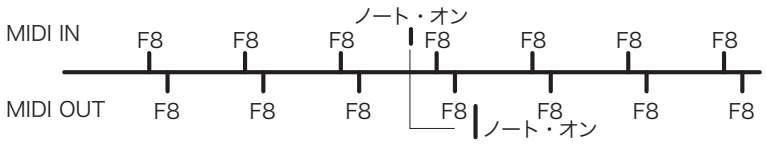
次の図は，MIDI IN と MIDI OUT の関係において，その処理の関係上許される例である。（MIDI OUT は MIDI IN より一定時間遅れているが，順序は変わっていない。）

良い例：



下記は，F8H を優先して送信したために，ノート・オフに遅延が生じた例。

順序が変わっても問題ない例：



クロック情報は，できるだけ早く送り出したいので，上の図のようにボイス・メッセージと送りの順序がかわっても差し支えない。さらには，ボイス・メッセージやコモン・メッセージなどの中に割り込んで送信してもよい。クロック情報と他のメッセージが同タイミングの場合は，正確なタイミングを維持するために送信する順序をかえることができる。

シーケンサーは，ストップ状態の時にタイミング・クロック（F8H）を送ることは差し支えない。その利点としては，受信側で次のスタート以後のテンポを予測できることが挙げられる。

コマンドの重複

すでに停止している状態でストップ・メッセージを受信したり，すでに演奏状態にある時に，スタートやコンティニュー・メッセージを受信した場合は，これらのコマンドを無視すべきである。

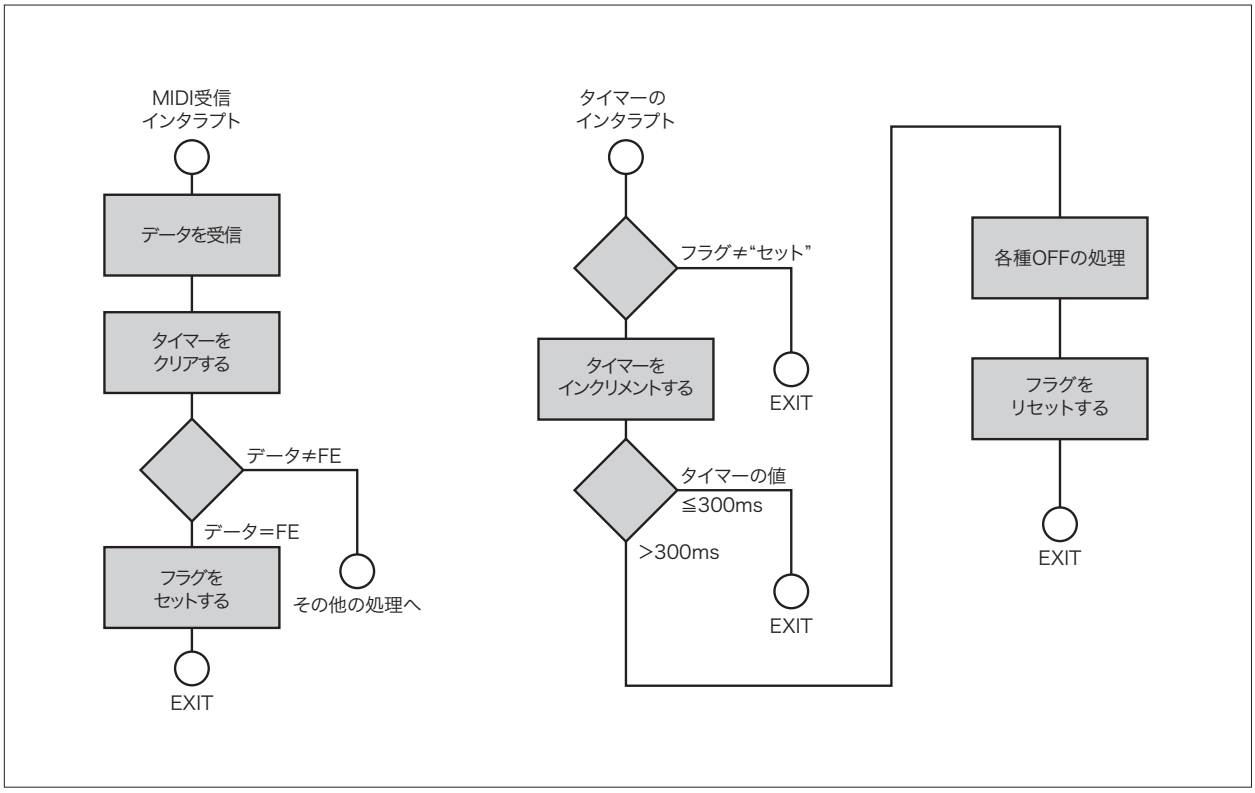
自己のパネル操作と，MIDI 入力されるメッセージの両方があった場合は，両者が作用してもよい。また，操作モードによってパネル操作か MIDI メッセージかを選択してもよい。例えば，MIDI クロック同期の場合はパネル操作を無視し，自己のクロックで演奏する場合はシステム・リアルタイム・メッセージ（F8H ～ FCH）を無視するようにしてもよい。

アクティブ・センシング

アクティブ・センシングは，送信側や受信側で必要に応じて使用できる。これは，他のデータが存在しない時に，最大 300ms ごとに送られる。受信側はアクティブ・センシングを 1 度も受けない時は，通常動作をするが，1 度これを受けて認識した場合は，300ms 以内になんらかのメッセージが送られて来ることを期待するようになる。もしも送られて来ないと，MIDI ケーブルがなんらかの原因で断線したものと判断し，すべてのボイスの発音を停止して通常動作にもどる。

実際には 10％程度の余裕をとって，例えば送信側は 270ms 以下で送信し，受信側は 330ms 以上の時間間隔で判定することが望ましい。

次のチャート図は，アクティブ・センシングに対する処理法である。



システム・リセット

システム・リセットは，システムを初期状態にしたり，電源を ON した時の状態にする。このメッセージは控えめに使用しなければならず，マニュアル操作でのみ使用すべである。電源オン時などに自動的に送ったりしてはならない。

もし，システム・リセットを認識するとしたら，次のような動作が望ましい。

1) ローカル・オンにする
2) ボイスをオフにする
3) すべてのコントローラーをリセットする
4) ソング・ポジションを0にする
5) 演奏をストップする
6) ランニング・ステータスをクリアする
7) その他，機種特有の MIDI の初期設定をする

システム・エクスクルーシブ・メッセージ

システム・エクスクルーシブ　　F0H

システム・メッセージは，チャンネル番号を有さず，MIDI で接続されたシステム全体で共通に使用される。しかし，システム・エクスクルーシブ・メッセージは，ID 番号によって特有のフォーマットがあるので例外である。

システム・エクスクルーシブ・メッセージは，シーケンス・データや音色のパラメーターなど，主に実時間のインフォメーションではないものを送信する時に利用される。これらについては送受信側の構成によって，異なるフォーマットが要求されている。したがって，ID 番号の登録 / 管理機関である AMEI ／ MMA から与えられた ID 番号の下に，独自のフォーマットが設計されている。

MIDI の趣旨は，いろいろな楽器や周辺機器を共通に結合することであるので，ボイス・メッセージなどで送ることができる実時間の演奏情報の伝達に，システム・エクスクルーシブ・メッセージを使用しない方がよい。

ID 番号が必要な場合には，AMEI ／ MMA に申請して，その運用に従う。なお，ID 番号を授与された場合には，1年以内にその ID 番号を使用したシステム・エクスクルーシブ・メッセージのフォーマットを公表しなければならないことになっている。ここに言う公表とは，そのフォーマットを AMEI に提出することはもちろん，MIDI の基本的な趣旨である公開の原則を十分考慮して，その製品の取扱い説明書や，各社が発行する技術資料などにも記載することなど，すべて行なうことを意味する。ユーザーや報道機関から請求があった場合には，正しい資料を提供することが望ましい。

また，前述の公開の原則に加えて統一の原則をも考慮し，すでに公表された他の ID 番号のフォーマットで目的を達成するものがあれば，なるべくそれを利用することが望ましい。公表されたエクスクルーシブのフォーマットは，一般の他のフォーマット（チャンネル・メッセージやコモン・メッセージ）とまったく同じ扱いとなり，そのフォーマットに忠実である限り，誰でも使用することができるが，その ID 番号の所有者に連絡することが望ましい。連絡を受けた ID 番号所有者は,正当な理由なしに,その使用を断わることはできない。

公表されたエクスクルーシブのフォーマットは，よほどの不都合が生じない限り，変更してはならない。また，新しいフォーマットを追加した場合は，すみやかに発表しなければならない。ID 番号所有者以外の者が，その ID 番号上のフォーマットを変更したり，新規に追加したりしてはならない。

ID 番号の区分

	American	European	Japanese	Other	Special
1バイト ID	01-1F	20-3F	40-5F	60-7C	7D-7F
3バイト ID	00 00 01	00 20 00	00 40 00	00 60 00	
	00 1F 7F	00 3F 7F	00 5F 7F	00 7F 7F	

ID 番号の 00 及び 00 00 00 は使用しない。特定の使用目的のため確保されているシステム・エクスクルーシブ ID 番号が3つある。7DH は非営利用（例，学校，研究等）に予約されており，一般に発売する製品には使用してはならない。そして，7EH 及び 7FH は，ユニバーサル・システム・エクスクルーシブのための ID 番号である。

ユニバーサル・システム・エクスクルーシブ

システム・エクスクルーシブの ID 番号の 7EH（ノンリアルタイム）及び 7FH（リアルタイム）は，MIDI 仕様の拡張用に予約されている。リアルタイム及びノンリアルタイム・メッセージの標準フォーマットは，次の通りである。

F0H <ID 番号 > < デバイス ID> < サブ ID 番号 #1> < サブ ID 番号 #2> ... F7H

リアルタイム及びノンリアルタイム・ユニバーサル・エクスクルーシブの ID 番号は，Table Ⅷ a に記載されている。

デバイス ID 番号

システム・エクスクルーシブ・メッセージは，MIDI チャンネルを持たないため，デバイス ID（以前は“チャンネル”と呼ばれていた）によって，システム中のどの機器へのメッセージかを指定する。デバイス ID の 7F は，all call（全員呼び出し）と呼ばれ，すべての機器に対するメッセージであることを示す。

多くの場合，デバイス ID は MIDI チャンネルのように機器内の仮想的なデバイスを指定するためではなく，MIDI システム内の物理的に独立した機器を特定するために使用されるべきである。

しかしながら例外もある。例えば，MMC に対応したデュアル・トランスポートのテープデッキにおけるデバイス ID の場合は，デバイス ID をひとつのアドレスと捉え,ひとつの機器が複数のアドレスを有することを認める方がよいという考え方もあり得る（これは機器が，単体のユニットからコンピューター内部のカードに移るにつれ，いっそう妥当な考え方となろう）。

デバイス ID は，デフォルトとしてはある特定の機器や I/O ポートを指定するものであるが，高度な機器（マルチ・トランスポートのテープデッキ，複数のカード・スロットを持ったコンピューター，ネットワーク接続された機器など）では，複数のデバイス ID を持つことが考えられる。そのような場合,マニュアルに明記すべきである。またデバイスによっては,それぞれのデバイス ID についてさらに MIDI チャンネル番号を使って，最大 16 の仮想的な機器へのアクセスが行われることもある。

サンプル・ダンプ

サンプラーのデータ・ダンプ用の規格であり，これはオープン・ループ，クローズド・ループのいずれでも動作する。クローズド・ループ・システムでは，スピード及びエラー復帰の改善のため，ハンドシェイクが実施される。クローズド・ループには，入力データの処理時間が遅い機器を含むことが可能である。オープン・ループ・システムは，ハンドシェイクを省略した簡易型のシステムを希望する場合に使用できる。

ハンドシェイクのためのメッセージとして，5つのジェネリック・メッセージ（ACK，NAK，Cancel，Wait，EOF）があり，これはファイル・ダンプなどでも使用される。サンプル・ダンプでは，加えてダンプ・リクエスト，ダンプ・ヘッダー，データ・パケット，そしてサンプル・ダンプ・エクステンションが使用される。

ジェネリック・ハンドシェイク・メッセージ

ACK

F0 7E <device ID> 7F pp F7
pp　　パケット番号

一番目のハンドシェイク・フラグで,「最後のデータ・パケットを正しく受信した, 次のデータを送信せよ」を意味する。パケット番号は, 正しく認識されたパケット番号を表す。

NAK

F0 7E <device ID> 7E pp F7
pp　　パケット番号

2番目のハンドシェイク・フラグで,「最後のデータ・パケットを正しく受信できなかった, 再送せよ」を意味する。パケット番号は, 受け入れられなかったパケット番号を表す。

Cancel

F0 7E <device ID> 7D pp F7
pp　　パケット番号

3番目のハンドシェイク・フラグで,「ダンプを中止した」を意味する。パケット番号は, 中止した時のパケット番号を表す。

Wait

F0 7E <device ID> 7C pp F7
pp　　パケット番号

4番目のハンドシェイク・フラグで,「指示あるまでこれ以上パケットを送信するな」を意味する。これは次のようなシステムにとっては非常に重要である。すなわち, 受信側がコンピューター等の場合, 次のダンプを受信する前に済ませておかなければならない動作, 例えば, ディスク・アクセスなどがあるからである。ダンプを再開させるには ACK を, 中止させるには Cancel を使用する。

EOF

F0 7E <device ID> 7B pp F7
pp　　パケット番号（無視される）

これはファイル・ダンプのために新しく追加されたハンドシェイク・フラグで, ファイル・ダンプの完了を表す。

ダンプ・ヘッダー

F0 7E <device ID> 01 ss ss ee ff ff ff gg gg gg hh hh hh ii ii ii jj F7

ss ss	**サンプル番号（LSB から）**
ee	**サンプル・フォーマット（8〜 28 ビット）**
ff ff ff	**サンプル周波数（ナノ秒単位で LSB から）**
gg gg gg	**サンプル長（word 単位で LSB から）**
hh hh hh	**ループ・スタート・アドレス（word 単位で LSB から）**
ii ii ii	**ループ・エンド・アドレス（word 単位で LSB から）**
jj	**ループ・タイプ（00= フォワード, 01= フォワード／バック, 7F= ループ・オフ）**

ダンプ・リクエスト

F0 7E <device ID> 03 ss ss F7

ss ss	**サンプル番号（LSB から）**

本メッセージを受信したサンプラーは，要求されているサンプル番号が有効範囲のものかどうかをチェックしなければならない。正当な場合は，要求されたサンプルがカレント・サウンド番号となり，下述の手順で要求先のマスターへダンプされる。受信側は，有効範囲外のメッセージは無視しなければならない。

データ・パケット

F0 7E <device ID> 02 kk <120 bytes> ll F7

kk	**パケット・カウント**
ll	**チェックサム（7E から <120 bytes> までを XOR した値）**

ひとつのデータ・パケットの全体の長さは 127 バイト。これは，メッセージ全部を受信してから，それらの処理を行おうとする機器の MIDI 入力バッファがオーバーフローするのを防止するためのもの。現在の MIDI 機器では，最低 128 バイト，つまり 1/2 ページのメモリー容量のバッファを備えていると思われる。

サンプル・ダンプ・エクステンション

サンプル・ダンプ・スタンダードの拡張は，ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・メッセージのサブ ID 番号 #1（05）で行われる。

マルチプル・ループ・ポイント・メッセージ

サンプル・ダンプ規格の拡張メッセージとして追加された。これにより，サンプルあたり最大 16,383 のループ・ポイントのペアを定義することができる。これを使用すれば，サンプルそのものを送信することなしに，ループ・ポイントを変更することが可能である。

サンプル・ダンプ規格の拡張は，すべてノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・メッセージのサブ ID 番号 #1（05）に続ける。

これらのメッセージのフォーマットは，次の通り。

ループ・ポイント・トランスミッション（17 バイト）

F0 7E <device ID> 05 01 ss ss bb bb cc dd dd dd ee ee ee F7

F0 7E <device ID>	ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー
05	サンプル・ダンプ・エクステンション（サブ ID 番号 #1）
01	マルチプル・ループ・メッセージ（サブ ID 番号 #2）
ss ss	サンプル番号（LSB から）
bb bb	ループ番号（LSB から, 7F 7F=delete all Loop）
cc	ループ・タイプ 00 ＝ Forwards Only 01 ＝ BackWards/Forwards 7F ＝ Off
dd dd dd	ループ・スタート・アドレス（LSB から）
ee ee ee	ループ・エンド・アドレス（LSB から）
F7	EOX

ループ・ポイント・リクエスト（10 バイト）

F0 7E <device ID> 05 02 ss ss bb bb F7

F0 7E <device ID>	ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー
05	サンプル・ダンプ・エクステンション（サブ ID 番号 #1）
02	ループ・ポイント・リクエスト（サブ ID 番号 #2）
ss ss	サンプル番号（LSB から）
bb bb	ループ番号（LSB から, 7F 7F=request all Loop）
F7	EOX

メッセージがひとつ送られると，ループ要求またはループ送信ごとにひとつのループが影響を受ける。ただし，全ループ要求及び全ループ削除は別である。ループ･メッセージが現存するループと同じ番号で送信された場合は，更新される。ループ番号 00 00 は，サンプル・ダンプ規格で定義されているサステイン・ループと同じである。

サンプル・ダンプの使い方

パネルまたは MIDI により，ダンプが要求されると，ダンプ・ヘッダーが送信される。マスターは，ダンプ・ヘッダーを送信後，最低2秒間は送信を休止し，受信側にダンプを受け入れられるかどうか（メモリーに余裕があるかどうか，等）を判断する時間の余裕を与えなければならない。マスターは,Cancel を受信した場合,ただちにダンプを中止しなければならない。ACK を受信した場合は,データ・パケットの送信を開始する。Wait を受信した場合は，次のメッセージを受信するまで中断する。マスターは，ダンプ・ヘッダー送信後の休止中に何のメッセージも受信しない場合は，オープン・ループであると判断し，パケットの送信を開始する。

データ･パケットは，そのパケット自身のヘッダー，パケット番号，120 バイト･データ，チェックサム及びエンド･オブ･エクスクルーシブ（EOX）からなる。パケット番号は，00 から始まり，新しいパケットごとに増加し，7FH に達すると次は，00 にリセットする。受信側はこれにより，パケットが新しいものか，または再送されているものかを判断する。サンプル・フォーマットにしたがって，パケット番号に続いて 30，40，または 60 ワードを構成する 120 バイトのデータが送信される。

各データ・バイトは7ビット構成で，サンプル・フォーマットが8～ 14 ビットの場合には，2バイトで1ワードとなる。15 ～ 21 ビットのサンプル・フォーマットでは，3バイト / ワード（40 ワード / パケット）を必要とし，また 22 ～ 28 ビットでは4バイト / ワード（30 ワード / パケット）が必要となる。7ビットのインフォメーションは左詰めとし，未使用のビットにはゼロを挿入する。サンプル・ワードは，オフセット・バイナリィで，例えば 12 ビットの場合，サンプル・ワード FFFH は 01111111B 01111100B と送信される。FFFH は，正の最大値（000H は負の最大値）を表している。チェックサムは，“7E <device ID>02< パケット・カウント > <120 バイト >”の排他的論理和（XOR）である。

データ・ダンプ受信中，サンプラーはチェックサムをとる。チェックサムが合致した時には，ACK を送信して次のパケットを待つ。合致しない時には，NAK を送信して次のパケットを待つ。もし次のパケット番号が前のものと同じでなければ，送信側に NAK によるパケット再送の機能がないものとして，エラーを無視し，受信を続けるようにする。

データ･ダンプ送信中，サンプラーはパケットを送信した後，自己の MIDI IN ポートを監視し，ACK を受信した場合は，次のパケットを送信する。もし NAK を受信した際のパケット番号が，前回送信のパケット番号と同じ場合は，そのパケットを再送する。パケット番号が一致せず，かつサンプラーにはパケットを順不同で送信する手段がない場合には，NAK を無視しなければならない。サンプラーは Wait を受信した場合，自己の MIDI IN ポートを次のメッセージが来るまで監視し，メッセージが来た場合，そのメッセージを通常の ACK，NAK，Cancel または不正メッセージ（通常はダンプを中止する）として処理しなければならない。20ms 以内にいかなるメッセージも受信しなかった時は，オープン・ループと見なして，次のパケットを送信することができる。

ハンドシェイク･フラグ（ACK, NAK, Cancel, Wait）は，パケット番号を含んでいる。これは，将来出現するであろう高度な頭脳を持った機器をも対象にしている。つまり，同期が失われた後でも，特定のパケットが再送信可能な機器である。

本プロセスは，送信すべきバイトが 121 未満になるまで継続する。最後のデータ・パケットは，実際に残っている有効バイトの数に関係なく 120 データ・バイトで構成される。使用しないバイトはゼロで埋める。受信側は，最後のパケットにもハンドシェイクをしなければならない。受信側は，自己のメモリーがいっぱいになった時には，マスターに対し Cancel を送信しなければならない。

デバイス・インクワイアリー

受信側機器に固有の情報を照会する時に用いる。

メッセージのフォーマットは，以下の通り。

アイデンティティ・リクエスト（6バイト）

F0 7E <device ID> 06 01 F7

F0 7E <device ID>	**ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー**
06	**ジェネラル・インフォメーション（サブ ID 番号 #1）**
01	**アイデンティティ・リクエスト（サブ ID 番号 #2）**
F7	**EOX**

アイデンティティ・リプライ（15 または 17 バイト）

F0 7E <device ID> 06 02 mm ff ff dd dd ss ss ss ss F7

F0 7E <device ID>	**ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー**
06	**ジェネラル・インフォメーション（サブ ID 番号 #1）**
02	**アイデンティティ・リプライ（サブ ID 番号 #2）**
mm	**メーカー ID**
ff ff	**ファミリー・コード（14 ビット，LSB から）**
dd dd	**ファミリー・ナンバー（14 ビット，LSB から）**
ss ss ss ss	**バージョン情報など（フォーマットはデバイスにより異なる）**
F7	**EOX**

＜備考＞

メーカー ID が 00H で始まる時は，拡張された ID 番号であるため，mm は3バイトになる。

ファイル・ダンプ

ファイル・ダンプは，あるコンピューターから別のコンピューターに，MIDI を利用してファイルを送信するプロトコルである。このプロトコルには，ふたつの主要な目的がある。コンピューターと専用シーケンサー間の MIDI ファイル（特にテンポ･マップ）の送信。そして，異なる種類のコンピューター間の，MIDI ファイルを含む，あらゆる種類のファイルの送信である。ファイルと共にファイル・ネームが送られるので，必要最小限の操作で，いくつかのファイルを次々に送ることができる。

すべてのファイル・ダンプ・メッセージは，ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・メッセージ（サブ ID 番号 #1 = 07）であり，以下のヘッダーで始まる。

F0 7E <device ID> 07 <sub-ID#2> ss ...

<device ID>	**メッセージの宛先のデバイス ID（7F を使用できる）**
07	**ファイル・ダンプ（サブ ID 番号 #1）**
<sub-ID#2>	**ファイル・ダンプ・メッセージの種類**
01	**ヘッダー**
02	**データ・パケット**
03	**リクエスト**
ss	**メッセージの送信元のデバイス ID（7F“all-call”は使用できない）**

送信元のデバイス ID を含んでおり，受信側の機器から送信者への応答や，パケットの送信に，この送信元のデバイス ID が使用されることがある。すなわち,送信時の機器Aと機器B間のハンドシェイクにおいて,AからBへのメッセージにはBを宛先として指定し，BからAへ返されるメッセージには，Aを宛先として指定する。これを行うために，Bへ送る最初のメッセージに，送信元である A の ID が指定されていることが必要である。そうすればBは，応答のメッセージを送り返す相手のデバイス ID がわかる。

リクエスト

F0 7E <device ID> 07 03 ss <type> <NAME> F7

<device ID>	**リクエストの宛先のデバイス ID（ファイルの送信側となる）**
ss	**リクエストした側のデバイス ID（ファイルの受信側となる）**
<type>	**4つの7bit の ASCII バイト（ファイルの種類）**
<NAME>	**ファイル・ネーム（7bit の ASCII バイト，メッセージは F7 で終結）**

<type> は，一般的な意味で，どんな種類のファイルがリクエストされているかを記述する。

以下にタイプの例を示す。これ以外のタイプは，受信側がそれを認識するとわかっている場合にのみ使用すべきである。リクエストを受信した機器が,リクエストされたファイルのタイプをサポートしていない場合は,後で述べる Cancel メッセージを送信すべきである。

type	推奨される DOS 拡張子	内容
MIDI	MID	MIDI ファイル
MIEX	MEX	MIDIEX ファイル
ESEQ	ESQ	ESEQ ファイル
TEXT	TXT	7 ビットの ASCII テキスト・ファイル
BIN<space>	BIN	（あらゆる MS-DOS ファイルのような）バイナリィ・ファイル
MAC<space>	MAC	Macintosh ファイル（Mac バイナリィ・ヘッダー付き）

<type> が MAC の場合は，Mac ファイルがリクエストされたことを意味する。Mac ファイルは，ふたつの“フォーク”と重要なファインダー情報を含んだ，Mac バイナリィ･イメージとして送られるが，ファイル･ダンプで MIDI ファイルを送信する場合，たとえ Mac であっても，Mac バイナリィを意識する必要はない。MIDI ファイルは，どんな場合でも MIDI ファイルとして認識されるように MIDI タイプを使わなければならない。

ファイル・ネームの長さは任意であり，また省略してもかまわない。省略した場合は，“何であれ，現在ロードされているもの”を意味する。ファイル･ネームは，印刷可能な ASCII 文字（20H ～ 7EH）のみが許される。コロンとバックスラッシュ（日本語版注:日本語では ¥）がパスとして解釈される場合もある。パスの使用を望まない場合，ファイル・ネームには避けるべきである。リクエスト・メッセージを受信した機器がファイル・システムを備えていない場合，null ファイル・ネームを使用して，現在ロードされているものを送るべきである。

リクエスト･メッセージを受信した機器がコンピューターであれば，ファイル･ネームを認識した場合，あるいはファイル･ネームはないが，“現在ロードされている”ファイルがある場合には，送信を始めるべきである。どちらにも該当しない場合，ユーザーにダンプ・メッセージ内のファイル・ネームを表示して，妥当なファイル・ネームの入力を促したり，リクエストした側に Cancel メッセージを送り返してもかまわない。ファイル・ネームの入力を促す場合，送信が開始されるまで，あるいは Cancel が送られるまで時間がかかるので，リクエストした側に Wait メッセージを送っておくべきである。

ヘッダー

```
F0 7E <device ID> 07 01 ss <type> <length> <NAME> F7
```

`<device ID>`	**リクエストした側（受信側）のデバイス ID**
`ss`	**送信側のデバイス ID**
`<type>`	**4つの7ビットの ASCII バイト（ファイルの種類）**
`<length>`	**4つの7ビットのバイト（実際の（エンコード前の）ファイルの長さ，LSB から）**
`<NAME>`	**ファイル・ネーム（7ビットの ASCII バイト，メッセージは 7F で終結）**

<type> と <NAME> は，ダンプ・リクエスト・メッセージで記述された通りである。

（送信中にコンバートされるため）ファイル長がわからない場合，<length> をゼロとしてもかまわない。

送信側が小規模な専用シーケンサーなどの場合には，ファイル･ネームを送信する必要はない。送信側がコンピューターで，ファイル･ネームがある場合はヘッダーで送られ，前述のように，ファイル・ネームの長さは任意であるが，印刷可能な ASCII 文字のみを使用する。パスを記述するキャラクターを含んでもよいが，最大限の互換性を確保するためには，パス情報を送るべきではない。DOS ライクなマシンのネームでは，ピリオドによって区切られたファイル拡張子を，ピリオドの前にスペースを入れずに続ける。

受信側がコンピューターで，起動しているプログラムがファイルの受信をサポートしている場合，必要があればそのファイル･ネームを，それ自身のファイル・システムに適したものにモディファイすべきである。例えば DOS マシンの場合，受信したファイル・ネームがピリオドを含んでいたら，ピリオドの後をファイルの拡張子として解釈すべきである。ピリオドがなければ，前述した拡張子を使用すべきである。プログラムがユーザーと対話しながら走行している場合，受信したファイル・ネームをユーザーに提示して，ファイル・ネームの入力を促すべきである。ユーザーが必要としたり，ファイル・ネームが送られてこなかったり，あるいはその名前のファイルがすでに存在している場合，そのファイル・ネームをモディファイすることができる。ユーザーに入力を促す場合には，送信が続行されるまで，あるいは Cancel が送られるまで時間がかかるので，送信側に Wait メッセージを送っておくべきである。

受信側がファイルを持たない小規模な専用シーケンサーであったり，コンピューター・メモリー上のデータを入れ替えるだけの場合，サポートされているタイプのファイルであれば，ファイル・ネームを無視し，現在のメモリー内容を受信したファイルの内容に入れ替えるべきである。

データ・パケット

```
F0 7E <device ID> 07 02 <packet #> <byte count> <data> <chksm> F7
```

<device ID>	**受信側のデバイス ID**
<packet #>	**パケット・カウント（1バイト）**
<byte count>	**パケット・サイズ（1バイト。エンコードされたデータ・バイトの数－1）**
<data>	**以下に記述されるようにエンコードされたデータ**
<chksm>	**チェックサム（1バイト。サンプル・ダンプと同様に，F0 の次からチェックサム・バイト前までのすべてのバイトの XOR）**

データ・パケットのトータル・サイズは，サンプル・ダンプよりも若干大きく，最大 137 バイト。パケット・カウントは 00 からスタートし，新しいパケットごとにインクリメント，7FH に到達した後は 00 にリセットする。これは，受信側が受けそこねたパケットを見つけるために使用される。バイト・カウントは，エンコードされたデータ・バイトの数から1を引いたものである。例えば，ファイルにストアされた 64 バイトは，74 バイトにエンコードされ（以下に記述される通り），そのバイト・カウントは 73 である。（1を引くことで 128 のデータ・バイトを送ることができる。ゼロ・バイトの送信はない。）

送信時間を倍増させる4ビット分割の代わりに，データは“7ビット化”され，送信時間は，8ビットとして送る（MIDI では不可能）場合より 12％ほど多くなる。7バイトずつグループ化されたデータは，8バイトとして送信される。最初にバイト7つそれぞれのサイン・ビットが1バイト中にマージされ，それに続いて各バイトの下位の7ビットが送られる（例外なく，8バイトごとに補助バイトが現れ，受信する機器にとっては，若干デコードしやすい）。

AAAAaaaa BBBBbbbb CCCCcccc DDDDdddd EEEEeeee FFFFffff GGGGgggg

上記の7バイトは，次のように送られる。

0ABCDEFG
0AAAaaaa 0BBBbbbb 0CCCcccc 0DDDdddd 0EEEeeee 0FFFffff 0GGGgggg

データは7バイトずつにグループ化され，各グループは8バイトへとエンコードされる。データのサイズが7の倍数でない場合，最後にいくつかのバイトが残される。この7バイトに満たないグループも同様に，送信されるバイトの最上位ビットの右からサイン・ビットと共に送信される。以下に例を示す。

AAAAaaaa BBBBbbbb CCCCcccc

上記の3バイトは，次のように送信される。

0ABC0000 0AAAaaaa 0BBBbbbb 0CCCcccc

最大パケット･サイズは 128 の送信されるバイトなので，これは 16 グループの7バイト，すなわち 112 のストアされたバイトに相当する。

ハンドシェイク・フラグ

ハンドシェイク・メッセージとしては，元来サンプル・ダンプ・スタンダードのために作成された汎用的なセットに加え，新たに EOF メッセージが使用される（ノンリアルタイムのサブ ID 番号＃ 1=7B ～ 7F）。その中の4つのメッセージは，サンプル・ダンプ・セクションで説明されているので，ここでは新たに追加されたメッセージのみを示す。

```
F0 7E <device ID> <sub-ID #1> pp F7
<device ID>          パケット送信側のデバイス ID（メッセージの宛先）
<sub-ID #1>          ハンドシェイク・メッセージ
        7B             End of File（エンド・オブ・ファイル）
        7C             Wait（ウエイト）
        7D             Cancel（キャンセル）
        7E             NAK
        7F             ACK
pp                   パケット番号
```

NAK

```
F0 7E <device ID> 7E pp F7
      <device ID>    パケット送信側のデバイス ID（ACK の受信側）
      pp             パケット番号
```

これは，受信したメッセージの長さが誤っていたり，チェックサムが正しくない時に送られる。パケット送信側が NAK を受信したら，パケットを再送すべきである。NAK を送信した後，パケット受信側は，同じパケットが再び送られるのを待つ。同じパケットの受信を3回連続してエラーした場合,NAK に代わって Cancel が送られるべきである。パケット（あるいは NAK）が見落とされたりしてパケット番号が誤っていた場合，NAK の代わりに Cancel メッセージが送られるべきである。

ACK

```
F0 7E <device ID> 7F pp F7
      <device ID>    パケット送信側のデバイス ID（ACK の受信側）
      pp             パケット番号
```

パケット番号は，正しく認識されたパケットを表している。ヘッダーの受信に対して応答する ACK では，パケット番号は未定義である。

Wait

```
F0 7E <device ID> 7C pp F7
      <device ID>    Wait 受信側のデバイス ID
      pp             パケット番号（無視される）
```

このハンドシェイク・フラッグは，ファイル・ヘッダーやデータ・パケット，あるいはファイル・ダンプ・リクエストを受信した後に使用される。ヘッダーに対する応答の場合，“ACK（あるいは Cancel）を受信するまで，データ・パケットを送るべからず”を意味する。データ・パケットに対する応答の場合，“ACK または NAK（あるいは Cancel）を受信するまで，これ以上データ・パケットを送信するべからず”を意味する。ファイル・ダンプ・リクエストへの応答として使用された場合には，“ファイル・ヘッダー（あるいは Cancel）が続く，待ちなさい”を意味する。

このメッセージは，データの受信側が残りのダンプ・データを処理する前に，ディスク・アクセスやユーザーに入力を促すなど，他のオペレーションを行なう必要があるシステムにとって重要である。時間の必要な装置は，ファイル・ヘッダーやデータ・パケット，あるいはファイル・ダンプ・リクエストを受信するたびに Wait の送信を必要とするだろう。こうすることで，受信されたデータを処理し，適切に応答するのに必要な時間を確保することができる。

Cancel

F0 7E <device ID> 7D pp F7

<device ID>	**Cancel 受信側のデバイス ID**
pp	**パケット番号（無視される）**

このハンドシェイク・フラッグは，いつでも使用でき，“ダンプの中止”を意味する。パケット番号は，中止が起こるパケットを表すが，受信側では無視される。これは，データ・パケットや，ハンドシェイク・メッセージ中に正しくないパケット番号があった時のような，何らかのエラーが見つかった場合，あるいはユーザーがダンプを中止した場合に，送信側あるいは受信側のどちらから送られてもかまわない。送信側が送信を中止する場合，Cancel メッセージのヘッダー中に受信側のデバイス ID を指定しなければならない。また，受信側が送信を中止してもらいたい場合は，Cancel メッセージの中に送信側のデバイス ID を使用しなければならない。

END OF FILE(EOF)

F0 7E <device ID> 7B pp F7

<device ID>	**受信側のデバイス ID**
pp	**パケット番号（無視される）**

これは MIDI における5番目の汎用的なハンドシェイク・フラッグで，サブ ID 番号 #1(7B) である。（ファイル・ダンプのような）非常に長いメッセージの最後のパケットを送った後，送信側は受信側に対し，ファイル全体の送信が完了したことを知らせるために EOF メッセージを送らなければならない。これは，ファイル・ダンプ・ヘッダー中のレングスが0（ファイルの長さがわからないことを意味する）の場合，絶対に必要である。なぜなら，これが受け手にとって，送信が完了したことを知る唯一の方法だからである。このメッセージは，たとえ最初に正しいレングスがわかっていても，送られなければならない。EOF に対する応答は必要ない。

ファイル・ダンプ送信シナリオ

ファイル・ダンプ・リクエストの使用は，オプションである。機器がリクエスト・メッセージを使ってファイル（あるいはメモリーの内容）を要求する場合もあるだろうし，あるいはリクエスト・メッセージを受信しなくても，ユーザーがファイル・ダンプを開始することもあるだろう。ファイルの送信側がリクエスト・メッセージの EOX（F7H）を受信したら，200ms 以内にファイル・ダンプ・ヘッダー，Wait，あるいは Cancel で応答しなければならない。Wait で応答した場合，準備が整ったらいつでもファイル・ダンプ・ヘッダー，あるいは Cancel メッセージを送ってかまわない。

ファイルの送信側は，ファイル・ダンプ・ヘッダー・メッセージを送る。受信側は，ヘッダーの EOX（F7H）を受信した後，200ms 以内に ACK,Wait,あるいは Cancel で応答しなければならない。Wait で応答した場合は,準備が整ったらいつでも ACK か Cancel メッセージを送ってかまわない。200ms 以内に受信側からの応答が受信できなかった場合，ファイル送信側はオープン・ループ送信であると考えて，ACK が受信されたかのように次に進む。

次に，ファイル送信側はデータ・パケットを送る。受信側は，データ・パケットを受信しながら，チェックサムの計算を行う。チェックサムが合致し，ただちにそのデータを処理できる場合，受信側は ACK を送って次のパケットが送られてくるのを待つ。受信側がデータをストアするのに 50ms より多くの時間を必要とする場合，Wait メッセージを送る（データをストアした後，送信を再開してもらうには

ACK を送る)。チェックサムが合致しなかったり，データの長さが誤っていたら，受信側は NAK を送り，同じパケットが再び送られて来るのを待つ。パケット番号が予期していたものでない場合，ファイル受信側は Cancel メッセージを送り，新しいヘッダーが送られるまで，それ以降のデータ・パケットをすべて無視する(オープン・ループの場合は，送り手は決して Cancel メッセージを受信しない)。最後のパケットの途中であっても，受信側のメモリーがフルになってしまった場合は，送信側に対し，Cancel を送るべきである。

機器がファイル・ダンプ送信を行っている場合，パケットを送ったら，それ自身の MIDI IN ポートを監視すべきである。ACK が受信された場合は，ただちに次のパケットを送り，NAK が受信され，そのパケット番号がひとつ前に送ったものと一致したら，そのパケットを再送する。ACK あるいは NAK のパケット番号が，今送ったばかりのパケットのパケット番号に一致しなかった場合，送信側は Cancel メッセージを送って，送信を中止すべきである。Wait が受信された場合は，別のメッセージが来るまで，その MIDI IN ポートを監視すべきである。ACK あるいは NAK を受信した場合，それを通常通りに処理し，続行すべきである。Cancel あるいは不法なメッセージを受信した場合，送信側はファイル・ダンプを中止すべきである。データ・パケット送信の後 50ms，あるいはヘッダー送信の後 200ms，何の応答も受信されなければ，オープン・ループであると考えて，次のパケットを送る。

最後のパケットに対する ACK が受信側から送られて来たら，ファイル送信側は EOF を送信する。このメッセージに対する受信側の ACK は必要とされない。そしてファイル・ダンプは完了する。

最大 128 データ・バイトまでなら，各パケットのバイト数は任意である。多くの機器は，等しいサイズのパケットをいくつか送信し，そして残ったものを最終パケットとして送信するだろうが，受信側は任意のパケット・サイズに対応できる必要がある。

MIDI チューニング

これは，“マイクロチューニング”（12 音平均律の他にユーザーが定義するスケール）を異なるメーカーの楽器間で互換性を持たせたり，チューニングをリアルタイムで切り換えることを可能にするために，MIDI 規格に追加された。

メッセージは，以下のものを含んでいる。

バルク・チューニング・ダンプ・リクエスト（ノンリアルタイム）

バルク・チューニング・ダンプ（ノンリアルタイム）

シングル・ノート・チューニング・チェンジ（リアルタイム）

最初の2つのメッセージは，ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・メッセージとして，そして最後のメッセージはリアルタイムのメッセージとして定義されたが,これらは同一のサブ ID 番号を使っている。個々の MIDI ノート番号に対するパフォーマンス・コントロールとして，新しい周波数にリアルタイムで再チューニングするシングル・ノート・チューニングは，この提案の一部である。

この規格は，メーカーがマイクロチューニングをインプリメントする方法を示すのではなく，異なる楽器間でチューニング・データを相互利用するための，一般的な手法を提供するためのものである。

この規格を使用するためには，楽器にいくつかの機能が備わっていることが前提となる。128 の MIDI ノート番号の定義域（あるいは少なくともその楽器が演奏可能な範囲の MIDI ノート番号）のどれでも，提案されている周波数レンジの範囲内で，いかなる周波数にもチューニング可能であることが必要とされる。また，楽器の周波数レンジにわたって，リニア（Hz 表記）よりむしろ，エクスポネンシャル（セント表記）チューニング分解能が強く提唱される。

この規格は，プリセットの選択及び各ノートへの設定による，リアルタイムでのチューニングの変更を可能とする。発音中のノートに，そのどちらかのリアルタイム・チューニング・メッセージの効果が反映される時，そのノートの発音を持続したまま，即座に新しい周波数に再チューニングすべきである。この時，音の途切れや強制的な発音停止，再トリガーやその他の耳に聴こえる不自然さがないように動作すべきである（「追記事項」を参照のこと）。

この規格は，MIDI プログラム・チェンジと同様，最大 128 のチューニング・メモリー・プログラムの切り替えを可能にする。楽器がサポートするチューニング･プログラム数は，これより少なくてもかまわない。存在するすべてのプログラムについて，定義された通り，メッセージに応答すればよい（「チューニング・プログラムの変更」を参照のこと）。

この規格は，いくつかの既存の楽器に直接的に当てはまるようにできているが，将来にわたり有益に活用できるよう，論理的に組み立てられている。これは，MIDI 楽器におけるマイクロチューニングを，より容易にインプリメントできるように，そしてより一般的にすることを望むものである。

周波数データ・フォーマット

この規格の周波数分解能は，音楽及び実験における多くの要求を十分満足させるものであり，1セントの 100 分の1よりもさらに細かい分解能を持つ。ただし，楽器がこの規格をサポートする際，ハードウェアが必ずしもこの分解能を持つ必要はない。この規格は，単にこの限度まで，いかなる分解能においても，チューニング・データの送信を可能にするということである。

周波数データは，システム・エクスクルーシブ・メッセージで送られる。システム・エクスクルーシブのデータ・バイトは，上位ビットを下位からセットし，データ・バイトは7ビットなので，提案されている分解能で周波数を指定するのに3バイト（21 ビット）周波数データ・ワードを使用する。この分解能をサポートしない楽器でも，必要のない下位ビットを放棄して使用してもかまわないが，大きな分解能を持

つ他の楽器に送信できるように，フル分解能を保持していることが望ましい。

周波数データは,半音を分割した単位で定義されるべきである。周波数レンジは MIDI ノート 0,C=8.1758Hz から始まり,MIDI ノート 127，G=12543.854Hz におよぶ。周波数データ・ワードの最初のバイトは，その周波数より低い平均律の半音で，最も近いものを指定する。次の2バイト(14 ビット)は，その半音より何セント高いかを指定する。その分解能は 100 セント / 2^{14}=.0061 セントである。

3バイトの値のひとつ(7FH 7FH 7FH)は，周波数データではなく，“変更なし”を示すために予約されている。楽器がこの3バイトを周波数データとして受信した場合，その MIDI ノート番号に設定されている周波数データに変更を加えてはならない。これは，128 の MIDI ノート番号の全レンジを使用しない楽器が，全レンジを使用する楽器に対して間違ったチューニング･データを送るのを防ぐ。3バイトによる周波数の表現は，以下のように解釈されるだろう。

0xxxxxxx 0abcdefg 0hijklmn

xxxxxxx　　＝半音

abcdefghijklmn　＝半音以下の部分（.0061 セント単位）

周波数データの例：

00 00 00	＝	8.17580Hz	（C - MIDI ノート番号 0 の通常のチューニング）
00 00 01	＝	8.17583Hz	
01 00 00	＝	8.6620Hz	
0C 00 00	＝	16.3516Hz	
3C 00 00	＝	261.6256Hz	（中央 C）
3D 00 00	＝	277.1826Hz	（C# - MIDI ノート番号 61 の通常のチューニング）
44 7F 7F	＝	439.9984Hz	
45 00 00	＝	440.0000Hz	（A-440）
45 00 01	＝	440.0016Hz	
78 00 00	＝	8372.0181Hz	（C - MIDI ノート番号 120 の通常のチューニング）
78 00 01	＝	8372.0476Hz	
7F 00 00	＝	12543.8540Hz	（G - MIDI ノート番号 127 の通常のチューニング）
7F 00 01	＝	12543.8982Hz	
7F 7F 7E	＝	13289.6566Hz	（レンジのトップ）
7F 7F 7F	＝	変更なし	（予約済み）

バルク・チューニング・ダンプ・リクエスト

バルク・チューニング・ダンプ・リクエストは以下の通りである。

F0 7E <device ID> 08 00 tt F7

F0 7E	ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー
<device ID>	ターゲットとなる機器の ID
08	サブ ID 番号 #1 ＝ MIDI チューニング・スタンダード
00	サブ ID 番号 #2 ＝バルク・ダンプ・リクエスト
tt	チューニング・プログラム番号（0～ 127）
F7	EOX

受信する楽器は，指定されたチューニング・プログラム番号に対し，以下で記述されるバルク・チューニング・ダンプ・メッセージにより応答する。

バルク・チューニング・ダンプ

バルク･チューニング･ダンプは，128 すべての MIDI ノート番号について，ノート 0（最初に送られる）からノート 127（最後に送られる）までの順で，前述した3バイト・フォーマットの周波数データを含む。このメッセージは，チューニング・ダンプ・リクエストを受信した楽器によって送られる。

F0 7E <device ID> 08 01 tt <tuning name> [xx yy zz] ... chksum F7

F0 7E	**ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー**
<device ID>	**応答する機器の ID**
08	**サブ ID 番号 #1 ＝ MIDI チューニング・スタンダード**
01	**サブ ID 番号 #2 ＝バルク・ダンプ**
tt	**チューニング・プログラム番号**
<tuning name>	**16 の ASCII 文字**
[xx yy zz]	**1ノートのための周波数データ（128 回繰り返される）**
chksum	**チェックサム（7E <device ID> 08 01 tt <388bytes> を XOR した値）**
F7	**EOX**

ある楽器が，128 の MIDI ノート番号の全レンジを使用していない場合，演奏できないノート部分のデータを受信時に無視してもかまわないが，広いキー･レンジを持つ他の楽器への送信が可能なように，128 キー全部のチューニングを保持しておくことが望まれる。演奏できないノートの周波数データには，必要なら上記の“変更なし”周波数データ･ワード（7FH 7FH 7FH）を使用することができる。

その楽器のレンジ範囲内のキーの場合，そのキーのノート･オン･メッセージによって演奏されるピッチのデータを送るべきだが，キー･レンジ外のキーでは，7FH 7FH 7FH を送ってもよい。

シングル・ノート・チューニング・チェンジ（リアル・タイム）

シングル・ノート・チューニング・チェンジ・メッセージ（リアルタイム・サブ ID 番号 #1=08）は，その楽器のチューニング・メモリーを即座に書き換えるものである。この変更は，ただちに効果を発揮すべきであり，メッセージが受信された時点で発音しているノートに対して，不自然に聴こえないように変化させるべきである。

F0 7F <device ID> 08 02 tt ll [kk xx yy zz] F7

F0 7F	**リアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー**
<device ID>	**ターゲットとなる機器の ID**
08	**サブ ID 番号 #1 ＝ MIDI チューニング・スタンダード**
02	**サブ ID 番号 #2 ＝ノート・チェンジ**
tt	**チューニング・プログラム番号**
ll	**チェンジの数（1チェンジ ＝ [kk xx yy zz] の1セット）**
[kk	**MIDI ノート番号**
xx yy zz]	**そのキーの周波数データ（'ll'回繰り返される）**
F7	**EOX**

このメッセージは，転送効率を最大化するために，(強制ではないが)複数のチェンジをひとつのメッセージ中に入れることも可能である。後に続くチェンジの数は，ll バイトで示され，メッセージのトータルの長さは8+ (ll ×4) バイトである。

128 の MIDI ノート番号の全レンジを使用しない楽器の場合，演奏できないノートに関するデータを，受信時に無視すべきである。

このメッセージは，アクティブでない(バックグラウンドの)チューニングを変更するのにも使用できる。また，メーカーの判断により，ある操作を実行することで，楽器から送信されてもかまわない(例えば，ひとつかそれ以上のキーを押さえながら，フロント・パネルの“センド・シングル・ノート・チューニング”ボタンを押すなど)。

チューニング・プログラムの変更

RPN(レジスタード・パラメーター・ナンバー)によって，その楽器に記憶されているチューニングのいずれかを選び，“現在の”すなわちアクティブなチューニングとする。複数のマイクロ・チューニングを記憶できる楽器がこのメッセージを受信したら，“現在の”チューニングを，指定されたチューニングに即座に変更すべきである。この変更はただちに効果を発揮し，メッセージが受信された時点で発音中のノートがあった場合，発音停止やリセット，再トリガー，音の途切れなどなしに，変更が行われなければならない。

楽器において，MIDI プログラム・チェンジ同様，どのような方法でチューニング・プログラムの変更を実現するかを規定するものではない。例えば，レイヤーされているサウンドや，マルチティンバーのサウンドが，異なるチューニングにアサインされていて，2つ以上のチューニングが有効な場合,メーカーがこのメッセージをどのように解釈するかは任意である。複数のチューニングを区別するには,ベーシック・チャンネルが有効であろう。また，音域別にチューニング・プログラムが設定されることもあるだろう。

このメッセージは，RPN(レジスタード・パラメーター・ナンバー)の送信に続き，データ・エントリー，データ・インクリメント，あるいはデータ・デクリメント・メッセージが送られる。以下に例を示す(ランニング・ステータスが使用されている)。

Bn 64 03 65 00 06 tt　　(データ・エントリー)
Bn 64 03 65 00 60 7F　　(データ・インクリメント)
Bn 64 03 65 00 61 7F　　(データ・デクリメント)
n　　= ベーシック・チャンネル
tt　　= チューニング・プログラム番号 (1～128)

同様に，チューニング・バンク・チェンジの RPN も以下のように送信される。

Bn 64 04 65 00 06 tt　　(データ・エントリー)
Bn 64 04 65 00 60 7F　　(データ・インクリメント)
Bn 64 04 65 00 61 7F　　(データ・デクリメント)
n　　= ベーシック・チャンネル
tt　　= チューニング・バンク番号 (1～128)

最大限の互換性のために，チューニングのバンク番号は，バンク・セレクト(コントローラー #00)とは別に管理される。しかし，楽器の中には，チューニング・バンクとパッチ・パラメーターのバンクをいっしょに(例えば ROM カートリッジに)記憶させて，この2つをリンクさせたいと考える場合もある。

楽器が，それに相当するプログラムやバンクを持たない，チューニング・プログラム番号やバンク番号を受信した場合，そのメッセージを無視すべきである。共通で使用できるチューニングとプログラム番号の標準的なマップは，現時点では提唱されていない。

追記事項

リアルタイム・チューニング・チェンジへの即時対応が，いつも望ましいとは限らない。状況によっては，チューニング・チェンジが次のノートから有効となり，発音中のノートには影響しない場合の方がよいこともある。しかしまた，チューニング・チェンジが，この規格で規定されているように，即時に対応し，発音中のノートに対しても，その発音を途絶えさせることなく有効にすべき状況もある。

楽器が，後者のように即時対応できる機能を有する場合，前者の状況に対応する，何らかの回避策を持つことは可能であろう。しかし，その逆は真ではない。このため，この規格は発音中のノートにただちに効果をもたらすチューニング・チェンジを求めている。しかしながらメーカーは，楽器内部において“即時 / 次のノート・オンから”の切り換えをするオプションを用意してもよい。

シングル・ノート・チューニングは，演奏中の使用を意図したものである。これを実現するためには，次の2つが求められる。

1）チューニング・テーブルの一時的なコピーに必要な RAM

2）ノートのピッチをスムーズに変化させることができる演算能力

シングル・ノート・チューニング・メッセージを認識するためには，現在のチューニング・プログラムのコピーを，RAM に保持しておく必要がある。マルチティンバー音源では，個々のティンバーにひとつのコピーを持つ可能性がある。ハイエンドの楽器では，8Kを上回る RAM を，ティンバーごとに持つ必要があろう。しかし安価な楽器では，ひとつの RAM テーブルを持ち，それをベーシック・チャンネルのみ有効とするか，あるいはすべてのティンバーが同じチューニングを共有するようにしてもかまわない。ユーザー・マニュアルで説明されていれば，これらの方法でも容認される。

GM システム・メッセージ

GM(サブ ID 番号 #1 = 09)のために用意された, ノンリアルタイム・システム・エクスクルーシブ・メッセージである。このメッセージは, GM以外の動作モードを持つ音源において, GMモードへのオン／オフ切り換えを行う。

GMシステム・オン

F0 7E <device ID> 09 01 F7

F0 7E	**ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー**
<device ID>	**ターゲットとなる機器の ID(7F "all-call" を推奨)**
09	**サブ ID 番号 #1 =GMメッセージ**
01	**サブ ID 番号 #2 =GMオン**
F7	**EOX**

GMシステム・オフ

F0 7E <device ID> 09 02 F7

F0 7E	**ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー**
<device ID>	**ターゲットとなる機器の ID(7F "all-call" を推奨)**
09	**サブ ID 番号 #1 =GMメッセージ**
02	**サブ ID 番号 #2 =GMオフ**
F7	**EOX**

MTCフル・メッセージ, ユーザー・ビット, リアルタイム・キューイング

クォーター・フレーム・メッセージ(コモン・メッセージ)は, システムの基本動作を取り扱うが, 早送りや巻き戻し, または特定時間への位置決めや, キュー等に使用するには適していない。これは, このメッセージを高速かつ連続的に送信すると, MIDI データ・ラインを必要以上に混雑させたり, ラインの容量を越えてしまうからである。このような場合, 1フレームの時刻情報全部をひとつのメッセージで送信できるフル・メッセージを使用する。フル・メッセージを送信した後, メカニカルな機器がその点までシャトル(またはオート・ロケート)するまでタイムコード・ジェネレーターは一時休止し, その後クォーター・フレーム・メッセージを送信して動作を再開する。

ユニバーサル・システム・エクスクルーシブのリアルタイムのサブ ID 番号 #01 は, MTCフル・メッセージ, MTCユーザー・ビットに, リアルタイムのサブ ID 番号 #05 は MIDI キューイングに使用される。

詳細はMTC(MIDI タイムコード)の項を参照。

MIDI ショー・コントロール

MIDI ショー・コントロールの目的は, MIDI システムに劇場やライヴ・パフォーマンス, マルチメディアやオーディオ・ビジュアル, そして, それらに類する環境において, 専用のインテリジェントなコントロール機器を MIDI によって制御することである。その応用範囲は, 照明コントローラーに GO や STOP, RESUME を指示するという単純なものだけでなく, 数多くのコントローラーを同期させた, 大きなシステムとの複雑なコミュニケーションにまで及ぶ。

MIDI ショー・コントロールでは, ユニバーサル・システム・エクスクルーシブのリアルタイムのサブ ID 番号 #02 を使用する。

詳細はMSC(MIDI ショー・コントロール)の項を参照。

記譜情報（ノーテーション・インフォメーション）

リアルタイム・ユニバーサル・システム・エクスクルーシブのサブ ID 番号 #1(03)は，リアルタイムで記譜上の情報を伝えるのに使用される。

このメッセージには，小節マーカー，拍子記号（ディレイド），そして拍子記号（即時）が含まれる。

小節マーカー（バー・マーカー）

小節マーカー・メッセージは，次に受信される MIDI クロックが新しい小節の最初のクロックであることを指定する。

メッセージのフォーマットは，以下の通りである。

F0 7F <device ID> 03 01 aa aa F7

F0 7F	**リアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー**
<device ID>	**ターゲットとなる装置の ID（デフォルト =7F“all-call”）**
03	**サブ ID 番号 #1 ＝ノーテーション・インフォメーション**
01	**サブ ID 番号 #2 ＝小節マーカー**
aa aa	**小節番号（LSB から）**
	[00 40] 走行していない
	[01 40] - [00 00] カウント・イン
	[01 00] - [7E 3F] ソング中の小節番号
	[7F 3F] 走行中（小節番号は不明）
F7	**EOX**

小節番号の付け方は，マイナスの最大の数を“走行していない”フラグとして使用する。マイナスの数からゼロまではカウント・イン小節で，ゼロがカウント・インの最後の小節である（カウント・インが1小節だけの場合は，マイナスの数を取り扱う必要はなく，ゼロからプラス方向にカウントすればよい）。小節数はプラスへと増加するが，使用可能なプラスの最大の数値は，“走行中だが，小節番号は不明”（あるいは，小節番号が8Kを越えている）として予約されている。

MIDI クロック（F8H）が送られている場合，この小節番号は次に受信された F8 で有効となる。MTC が送られていて MIDI クロックが送られていない場合は，この小節番号は次に受信された F1H xxH で有効となる。またこれは，受信してすぐに表示してもよい（ドラム・マシンやシーケンサーが一時停止している状態で，新しいソング・ポジションにロケートされた場合など）。

このメッセージは，低レベルな同期とは異なる記譜上の同期及び情報の伝達を意図したものであって，他の MIDI のタイミング情報を伝えるメッセージに代わるものとして捉えるべきではない。

小節マーカー・メッセージは，即時または次の小節からの選択ができる（拍子記号のような）他のノーテーション・メッセージにとって重要である。後者の場合，他のメッセージは前の小節中の任意の時点で，あらかじめ送っておくことができる。これにより，ある小節の最後の F8H/F1H xxH と，次の小節の最初の F8H/F1H xxH との間に十分な余裕を持たせられ，動作の遅れを最小限にしてくれる。小節ごとに小節マーカー・メッセージが送られてくれば，受信側は現在どの小節を演奏しているのかを正確に知るために，MIDI クロックを数える必要はない。

したがって，動作の遅れやふらつき，メッセージの入れ替わりの可能性を防ぐために，小節番号は前の小節の最後の F8H，あるいは，F1H xxH の直後に送られることが強く薦められる（MIDI マージャーも，次の F8H より遅れないように，このメッセージに気を配る必要があるだろう）。

拍子記号

拍子記号メッセージは，受信側の機器に新しい拍子を伝えるのに使用される。イミディエイト（即時）とディレイド（遅延）のふたつのフォームがある。イミディエイト・フォーム（サブ ID 番号 #2 = 02H［bit6=reset］）は，受信と同時に（あるいは，スレーブとして MIDI に同期している場合は，次に受信される MIDI クロックで）効力を発揮する。

ディレイド・フォーム（サブ ID 番号 #2=42H［bit6 = set］）は，次の小節マーカー・メッセージの受信と同時に効力を発揮する。しかし，それは即時に表示されてもかまわない。

拍子記号（イミディエイト）

F0 7F <device ID> 03 02 ln nn dd bb cc bb [nn dd...] F7

F0 7F	リアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー
<device ID>	ターゲットとなる装置の ID（デフォルト =7F“all-call”）
03	サブ ID 番号 #1 ＝ノーテーション・インフォメーション
02	サブ ID 番号 #2 ＝拍子記号（イミディエイト）
ln	後に続くデータ・バイトの数
nn	拍子記号の分子
dd	拍子記号の分母（2のマイナスの累乗）
cc	ひとつのメトロノーム・クリック中の MIDI クロックの数
bb	MIDI 4分音符における記譜された 32 分音符の数
[nn dd...]	同じ小節中で混在する拍子記号を定義する拍子記号の追加ペア
F7	EOX

拍子記号（ディレイド）

F0 7F <device ID> 03 42 ln nn dd bb cc bb [nn dd...] F7

F0 7F	リアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー
<device ID>	ターゲットとなる装置の ID（デフォルト =7F“all-call”）
03	サブ ID 番号 #1= ノーテーション・インフォメーション
42	サブ ID 番号 #2= 拍子記号（ディレイド）
ln	後に続くデータ・バイトの数
nn	拍子記号の分子
dd	拍子記号の分母（2のマイナスの累乗）
cc	ひとつのメトロノーム・クリック中の MIDI クロックの数
bb	MIDI 4分音符における記譜された 32 分音符の数
[nn dd...]	同じ小節中で混在する拍子記号を定義する拍子記号の追加ペア
F7	EOX

[nn dd...] における追加データは，常にペアでなくてはならない。指定される追加拍子記号がない場合には，ln（データの長さ）は4である。これは，その小節中に存在する臨時の拍子記号ごとに，2の倍数として増加する。

このデータ・フォーマットは，スタンダード MIDI ファイルの拍子記号メタ・イベント（FFH 58H）と同じで，これに複合する拍子記号のための追加のバイトを加えたものである。複合する拍子記号のためのバイトは，末尾に追加されたので，メタ・イベントの先頭のバイトを崩さずに，このメッセージのフォーマットにマッチさせることが可能である。

次の小節の拍子記号が何なのか，前もって知らせるための負担は，送信側が持つ。受信側はクロックをカウントして，それをデコードする必要はない。表示機能を持つ機器が（それが MIDI クロックをサポートしていなくても），容易に有益な情報をユーザーに提供できるように，ノーテーション系のメッセージの解釈は，可能な限り受信側の負担が軽くなるように考慮されている。

デバイス・コントロール

マスター・ボリュームとマスター・バランス

これらのメッセージは,ステレオ出力を持つ機器における音量コントロール及び音量バランスのコントロールを行う。主にGM音源（例えばマスター・ボリュームは，音源モジュールの全レイヤーを同時にフェード・アウトさせる）での使用を意図したものであるが，さまざまなところで広く利用されるであろう。

これらのメッセージはリアルタイム・ユニバーサル・エクスクルーシブ（サブ ID 番号 #1 = 04）であり，MIDI チャンネルではなくデバイス ID を指定する。チャンネルごとのコントロールは，チャンネル・ボリューム（旧メイン・ボリューム）（cc#07）及びバランス（cc#08）を使用する。

マスター・ボリューム

F0 7F <device ID> 04 01 vv vv F7

F0 7F	**リアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー**
<device ID>	**ターゲットとなる装置の ID（デフォルト =7F“all-call”）**
04	**サブ ID 番号 #1 =デバイス・コントロール**
01	**サブ ID 番号 #2 =マスター・ボリューム**
vv vv	**ボリューム（LSB から）**
	00 00 =ボリューム・オフ
F7	**EOX**

マスター・バランス

F0 7F <device ID> 04 02 vv vv F7

F0 7F	**リアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー**
<device ID>	**ターゲットとなる装置の ID（デフォルト =7F“all-call”）**
04	**サブ ID 番号 #1 =デバイス・コントロール**
02	**サブ ID 番号 #2 =マスター・バランス**
vv vv	**バランス（LSB から，00 =オフ）**
	00 00 =左
	7F 7F =右
F7	**EOX**

これらのメッセージに正しく対応するには，機器は3つのボリュームと2つのバランスをトラックごとに持たなければならない。

1. デバイス ID による受信（パネル上で設定した ID に一致した場合，初期状態はフル・ボリューム）
2. all-call(7FH）による受信
3. チャンネル・メッセージの受信

MIDI マシン・コントロール

MIDI マシン・コントロールは，MIDI のシステムが，従来からあるオーディオ・レコーディングやプロダクションのシステムと通信し，これらを制御できるようにするための，汎用プロトコルである。その応用範囲としては，1台のテープレコーダーにプレイ，ストップ，ファースト・フォワード，リワインドなどを指示できる単純なインターフェースから，オーディオやビデオのレコーダや，デジタル・レコーディング・システム，さらにはシーケンサーなどによる，タイムコードをベースにした，大規模な同期システムとの複雑な通信にまで渡ろう。

MIDI マシン・コントロールでは2つのユニバーサル・リアルタイム・システム・エクスクルーシブ・メッセージが使われ，ひとつはコマンド（コントローラーからコントロールしたい装置へ送られる），もうひとつはレスポンス（コントロールされる装置からコントローラーへ送られる）である。（サブ ID 番号 #1 = 06,07）

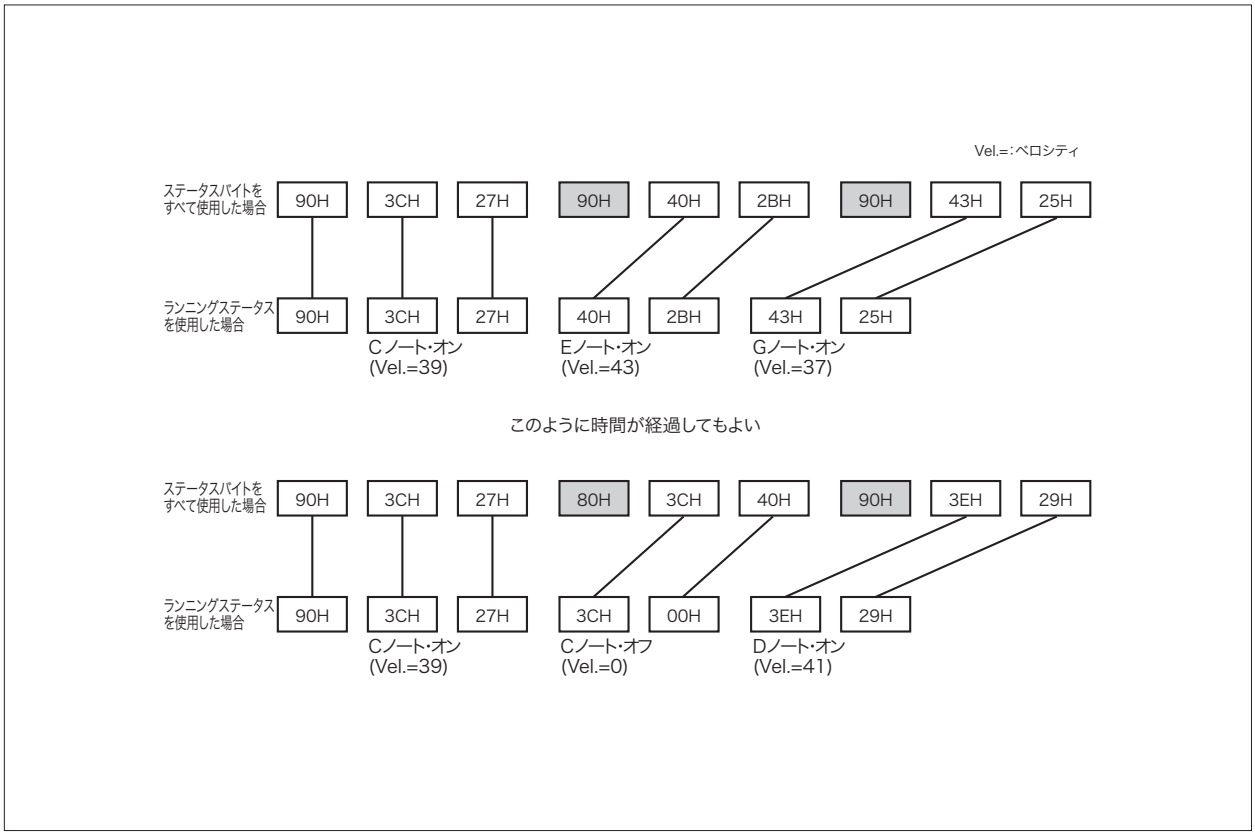
補足説明と運用上の注意

ランニング・ステータス

ランニング・ステータスとは，ステータス・バイトが直前のメッセージのそれと同じ値である時，そのステータス・バイトを省略してデータ・バイトのみを送ることで，送信時間を短縮し，実際の演奏からの MIDI 送信データの遅れを軽減するためのものである。受信側は，ステータス・バイトが省略されたデータ・バイトのみのメッセージを受信した時は，最後に受信したステータス・バイトを補う必要がある。

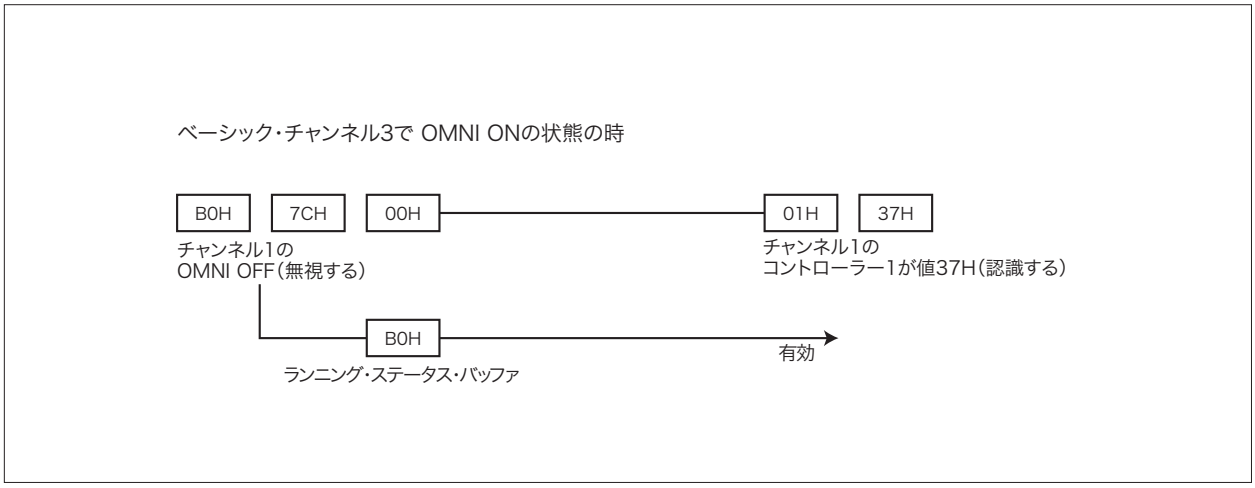
例えば，通常のノート・オン・メッセージは，ノート・オン・ステータス・バイト(9nH)，ノート番号，ベロシティ値の3バイトになるが，ランニング・ステータスを用いれば，同じチャンネルのノート・オンが続く場合には，2音目以降はステータス・バイトを省略してノート番号とベロシティ値のみを送信することができる。

ランニング・ステータスは，特にノート・オンとコンティニュアス・コントローラーとによるデータに有効である。ノート・オフをノート・オンのステータスのもとでベロシティ値0で送ると，さらに効果がある。



この例ではノート・オン・メッセージを取り上げているが，モード・メッセージやコントロール・チェンジ・メッセージについても，同様にしてデータ量を減らすことができる。

認識しないモード・メッセージも，ランニング・ステータス・バッファには保存されなければならない。例えば，オムニ・オンであってもモード・メッセージはベーシック・チャンネル以外では認識しないが，その直後にコントロール・チェンジがステータスなしで送られてくれば，オムニ・オンのモードであるから，認識するのが当然である。



受信側では，送信側でランニング・ステータスのルールを適用することを考慮に入れて，常に受信した最後のステータスを保持していなければならない。また，そのステータスのもとでは，データが何バイトでひとつのメッセージになるかを見ている必要がある。そのために，受信側にランニング・ステータス・バッファを設け，その処理を次のようにするのがよい。

1. 電源オン時にクリアする。
2. チャンネル・メッセージのステータスを受信した時，そのステータスを書き込む。
3. システム・エクスクルーシブ，システム・コモンのステータスを受信した時，クリアする。
4. システム・リアルタイム・メッセージを受信した時，何もしない。
5. ランニング・ステータス・バッファがクリア状態の時は，データ・バイトを無視する。

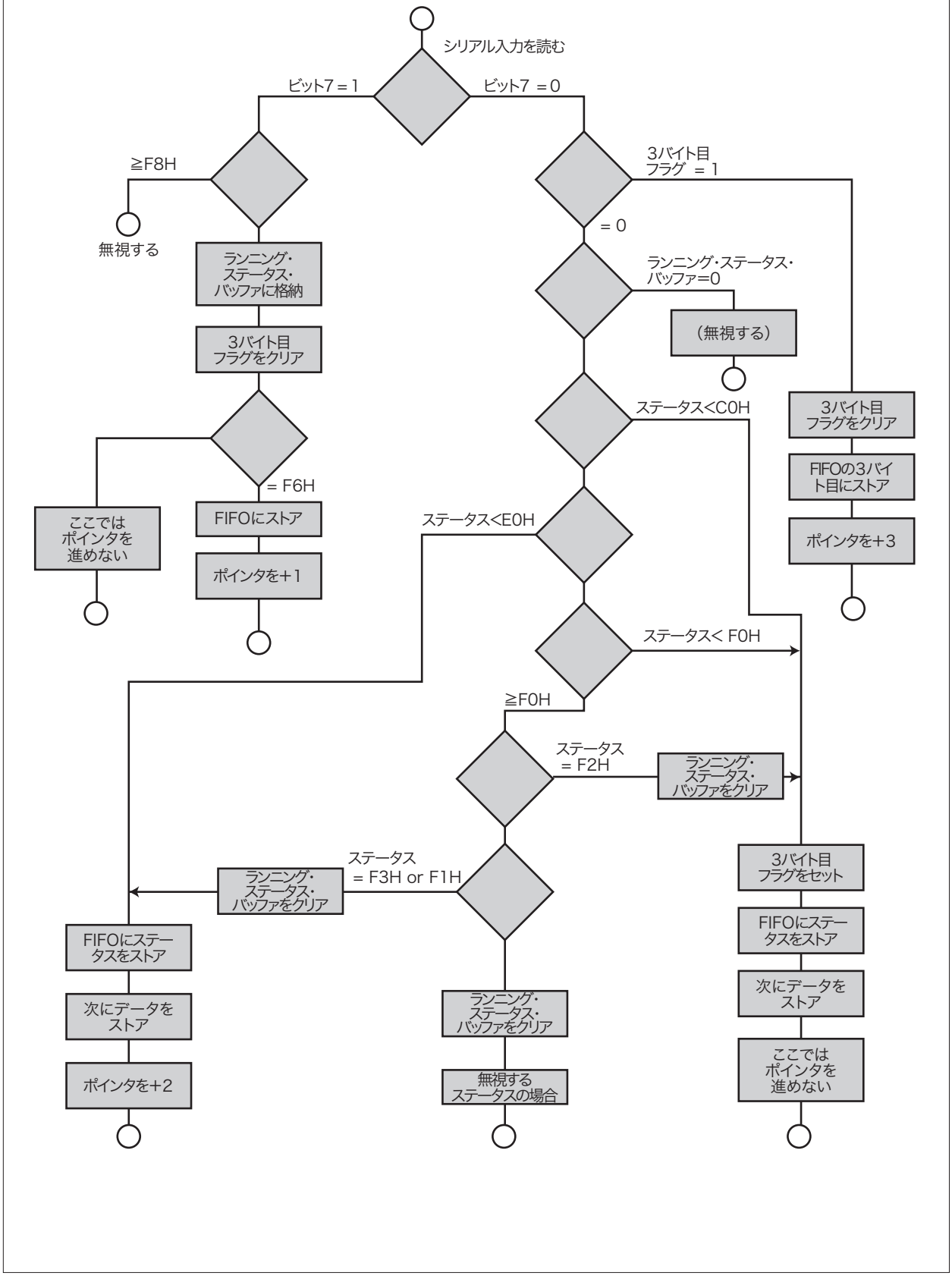
システム･コモン･メッセージには，現在未定義のステータスが2個（F4H，F5H）あるが，もしこれらを受信した時には，ランニング･ステータス・バッファをクリアするようにした方がよい。また，システム・リアルタイム・メッセージの中にも未定義が2個（F9H，FDH）あるが，これらは無視し，ランニング・ステータス・バッファはクリアしない方がよい。

ランニング・ステータスを使用している場合，受信側が送信側の電源オンより後に接続された時は，次のステータス・バイトが受信されるまで発音しないことがあるので，送信側は数秒に 1 回はステータス・バイトを送信することが望ましい。

まとめ

送信側は，ランニング･ステータスを使っても使わなくてもよい。また，ノート･オフは，ノート･オンのステータスでベロシティ0で送っても，ノート・オフのステータスを使ってもよい。

受信側は，チャンネル・メッセージのステータスを省略して送信されて来ることを十分考慮しなければならない。次の図は，受信側に設けられたランニング・ステータス・バッファに関する MIDI 受信処理の一例を示す。


シリアル入力を読む
ビット7 = 1
ビット7 = 0
≧F8H
無視する
ランニング・
ステータス・
バッファに格納
3バイト目
フラグをクリア
= F6H
ここでは
ポインタを
進めない
FIFOにストア
ポインタを+1
3バイト目
フラグ = 1
= 0
ランニング・ステータス・
バッファ=0
（無視する）
ステータス<C0H
3バイト目
フラグをクリア
FIFOの3バイ
ト目にストア
ポインタを+3
ステータス<E0H
ステータス< F0H
≧F0H
ステータス
= F2H
ランニング・
ステータス・
バッファをクリア
ステータス
= F3H or F1H
ランニング・
ステータス・
バッファをクリア
FIFOにステー
タスをストア
次にデータを
ストア
ポインタを+2
ランニング・
ステータス・
バッファをクリア
無視する
ステータスの場合
3バイト目
フラグをセット
FIFOにステー
タスをストア
次にデータを
ストア
ここでは
ポインタを
進めない

ノート・オン／ノート・オフの数

同一チャンネルで同じノート番号のオンを2回受信した場合，同じボイスに割り当てるか別のボイスに割り当てるか，あるいは他の方法を取るかは，受信側の選択に任されている。一方送信側では，同一チャンネルで同じノート番号のオンを2回送った後には，オフも2回送らなければならない。

< 参考 >

ここでは，ノート・オンの数とノート・オフの数とを一致させることを要求されている。

ポリ・モードでのアサイン

ポリ・モードでは，複数のノート・オンが送られてきて，それらを認識した時，どのようにボイスに割り当てて発音させるかは，特に規定されていない。受信側で処理可能なボイス数以上のノート・オンに対する処理はどのようにしてもよい。また，受信したノート・オンと自己の鍵盤操作などによるオンとの優先順位は，どのようにしてもよい。

自己の鍵盤操作による発音と，受信したノート・オン／ノート・オフのメッセージによる発音とを，区別することは重要である。

モード切換時のオール・ノート・オフ処理

受信側でオムニ・オン／オムニ・オフ，モノ／ポリを切り換える時には，オール・ノート・オフを行って，その後の確実な動作を期する必要がある。ただし，その時オフにするボイスは，MIDI IN のメッセージによってオンとなったものだけとして，自己の鍵盤によるオンのボイスは，そのままにする方がよい。

MIDI マージ時のオール・ノート・オフの扱い

シーケンサーが MIDI IN で受信したメッセージと，自己のシーケンス・データに基づいたメッセージとを混合して送信している時，あるいは，MIDI マージ装置が複数の MIDI IN からのデータを混合して送信している時には，受信したオール・ノート・オフをそのまま送ると不都合な場合があるので，注意しなければならない。

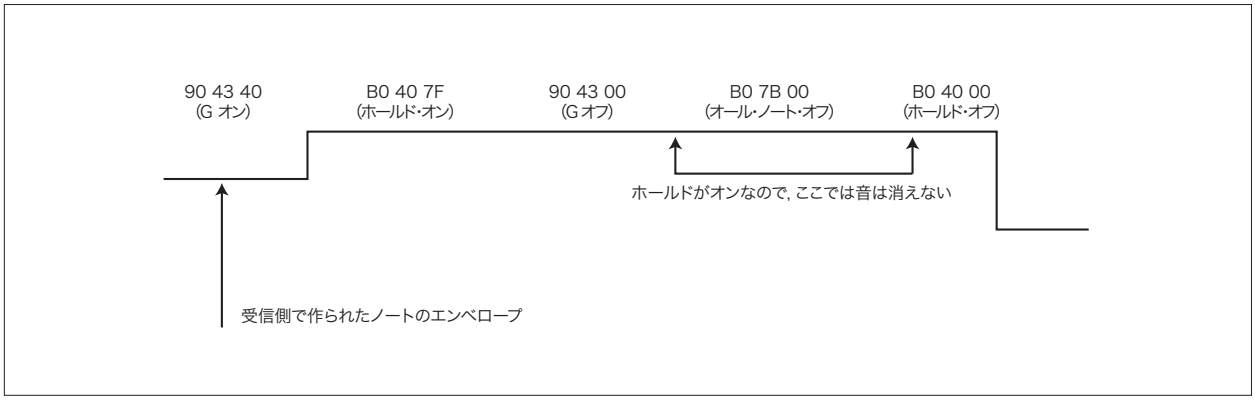
シーケンサーがオール・ノート・オフを受信しても，自己のシーケンスによってそのチャンネルのノートがオンになっている時には，そのオール・ノート・オフを送信してはならない。また，自己のシーケンス・データの中にオール・ノート・オフがあっても，そのチャンネルでノート・オンが受信されている間は，そのオール・ノート・オフを送信してはならない。また MIDI マージ装置には，オール・ノート・オフの送信を禁止できるような，フィルター機能が必要である。

第2バイトが 124 以上のチャンネル・モード・メッセージにも，オール・ノート・オフの機能があるので，同様の扱いをしなければならない。

ホールドとオール・ノート・オフとの関係

ホールド, ダンパー・ペダル, サステイン, セカンド・リリース・スイッチなどと称するスイッチのコントロール・チェンジを認識する受信側では, そのスイッチがオンになるメッセージを受信した後に通常のノート・オフを受けても, ボイス発音はまだオン状態が継続している。この状態の時に, チャンネル・モード・メッセージ 123(7BH)のオール・ノート・オフを受信してそれを認識しても, それは各ノート・オフを補佐しているだけであるから, ホールドによってオン状態が継続しているボイスをオフすべきではない。この時, 受信側は後にホールドがオフになった時に, ただちにオフになる潜在能力を保持しておくことが望ましい。

同様に, エンベロープの都合でリリース・タイムが長い時, オール・ノート・オフを認識することによって, 急激に音を消すような処理をすべきではない。



ホールド・ペダルについての補足

ホールド, ダンパー・ペダル, サステイン, セカンド・リリース・スイッチなどと称するスイッチは, コントロール番号 64(40H)を使用する。ホールド・ペダルの正しい動作は, エンベロープのサステイン・レベルを維持することである。“ホールド2(Hold 2)”スイッチが, コントロール番号 69(45H)として定義されている。これは, 他のホールド機能(例えば“フリーズ”, エンベロープ等がその時の状態で凍結される)のため, あるいは, 2つの異なったホールド機能を同時に実現するためなどに用いられる。

コントロール番号 66(42H)の“ソステヌート(コード・ホールド)”は, ペダルを踏んだ時に弾かれていた音だけを, ホールドするものである。

MIDI 受信の優先

MIDI では, 送信側は受信側の事情を無視して, 一方的に一定の転送レート(31.25Kbit/sec)でデータを送ってくる。受信側はこれを欠落のないように受信し, 処理しなければならない。通信以外の処理との優先関係に留意し, MIDI データが無視されたり, 誤って処理されたりすることがないようにすることが重要である。処理しきれないで捨てられるデータが発生すると, それによって誤動作することになるが, そのような場合でも, 常に安全な側に処理されるように配慮することが望ましい。

オムニの解除

自己のパネル操作, その他受信側で操作できる適切な方法によって, 受信モードをオムニ・オフにできることが望ましい。

シーケンサーのベーシック・チャンネル

MIDIのモードは,主として受信側がボイス･メッセージを処理して発音させることを主眼にして定められている。MIDIのシーケンサーは，MIDIのボイス・メッセージの時間経過を記録するための装置で，受信されたMIDIメッセージを相対的に同じ時間間隔で，別の時刻に再生するものであると考えられる。“別の時刻に”という点を除けば，送信のための結線と同じで，それ自体ボイス・メッセージによって仕事をしないと考えれば,そのシーケンサー自体は必ずしもチャンネル･モードを認識して,自らそのモードになる必要はない。したがって通常の機器のように，ベーシック・チャンネルがなくてもよいと考えられる。

もちろん，ベーシック・チャンネルを持ち，モード・メッセージを認識し，そのモードによって認識できるチャンネル番号などを変化させてもかまわない。

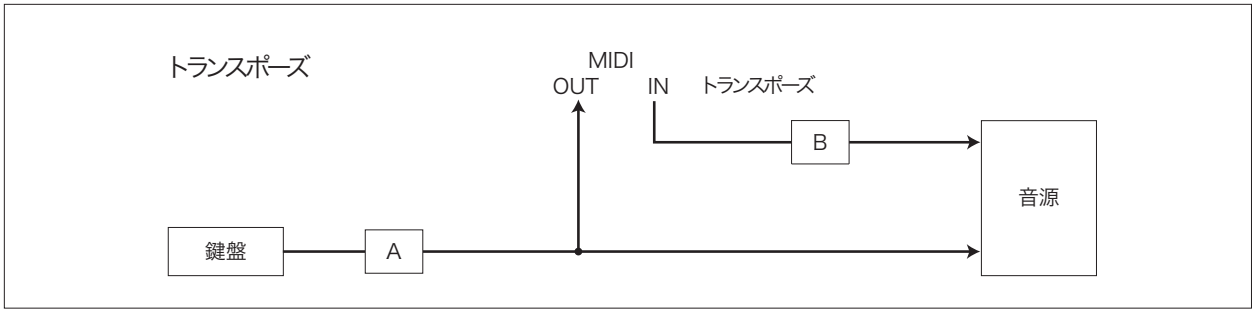
トランスポーズ

ノート・オン／ノート・オフ・メッセージのノート番号は，原則的に中央Cが60と定められている。この“中央C”とは，ピアノの中央部Cの鍵であり，楽譜上は高音部の下第一線のCを意味する。したがって，鍵盤の範囲によってはちょうど中央にはならないこともある。

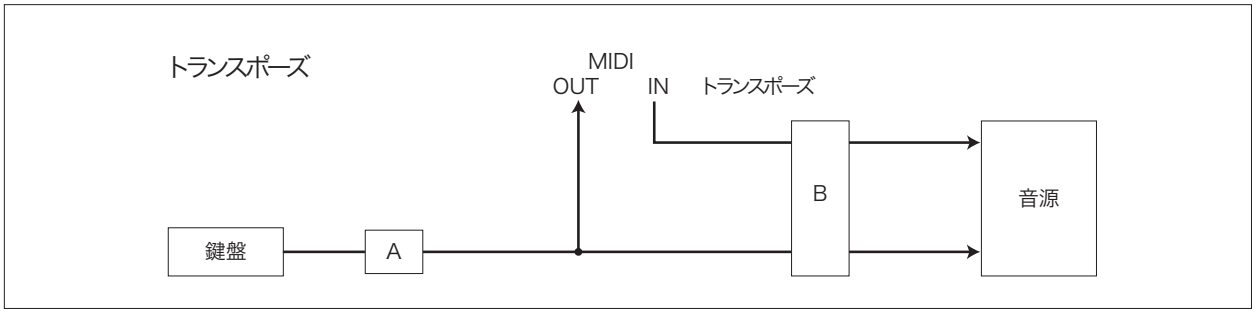
簡易な演奏を得る目的でクロマチックのキー･トランスポーズを行う場合は，中央C以外のキーで60を得ることになってもよい。この場合,トランスポーズをしない時との区別がパネル上ではっきりしている方が使用しやすいであろう。

＜日本語版注＞

受信側で上記以外の目的のために,トランスポーズをクロマチック，オクターブなどで行うことも自由である。ただし，次の図の構成のように，ひとつの楽器で，同一の操作でAB部の各トランスポーズが同じように変化するようにすると混乱を招くであろう。



2つのトランスポーズを持つ場合は，次のような構成も考えられる。この場合,トランズポーズAとBとは独立して変化し，両方の信号ともトランスポーズBを通過する。



受信側のボイスが発音できる音域の範囲外のノート番号に対しては，認識しなくてもよい。認識する場合は，オクターブずつシフトするのが普通である。

MIDI インプリメンテーション・チャートの作成法

標準 MIDI インプリメンテーション・チャートは，送受信機能の早見表として各製品ごとに作成される。ユーザーは，その機器でどのようなメッセージや機能が使用可能かを，容易に知ることができる。このチャートは，すべての MIDI 製品の取扱い説明書に含まれていなければならない。このチャートによって，利用者がふたつの MIDI 機器を接続しようとする時に，どのようなメッセージが相互に転送できるかを知ることができる。

例えば，機器Aの MIDI OUT を機器Bの MIDI IN に接続した時には，Aのチャートの送信側の記述部分をBのチャートの受信部分の横に並べて，各項目を比較すればよい。この目的のために，それぞれの機器のチャートは，少なくとも各行の縦方向の寸法は同じでなければならない。

全体の構成

1. 最初の[]には，その製品の名称を，例えば[16 音ポリフォニック・シンセサイザー]，[シーケンサー]などのように書く。

2. Model の項目には，その製品のモデル名（番号）を書き，必要に応じて，バージョン番号や，表作成の日付（または発売日）などを書く。

3. 表の内容は，4つのコラムに別れている。最初のコラムはそれぞれの機能など，次の2つはその機能に関する送信／受信の情報，4 番目は備考欄になっていて特記事項を記入する。送信，受信，備考間の線の位置は，見本を原則とするが，送信のファンクション切り替えが多い場合など，必要に応じて適宜左右に移動させることはさしつかえない。

機能の記述

1. ベーシック・チャンネル（Basic Channel）

電源 ON 時（Default）: 電源オン時にアサインされるチャンネル番号を書く。もし，チャンネル番号が電源オフ時にも記憶されていて，特定されない時は，1-16 と書き，右側の備考（Remarks）欄に記憶される（Memorized）などと書く。

設定可能（Changed）: チャンネル番号が機器のフロント・パネル操作などで変更可能なものについては，その可変範囲を書く。変更できない時は×と記す。

2. モード（Mode）

電源 ON 時（Default）: 送信部及び受信部が電源オン時にとるモードを 1 ～ 4 で記す。表の下部に 1 ～ 4 の意味が示してある。

メッセージ（Message）: 送信または認識可能なモード・メッセージを記入する。OMNI ON ／ OFF, MONO ON, POLY 等。

MONO ON, POLY ON はそれぞれ，MONO, POLY と略記してもよい。

代用（Altered）: 受信側でとり得ないモードがある時，どのモード・メッセージを受信した時どのモードに置き換えられるかを表す。例えば，モノ・オンを受け付けられない機器かそのメッセージを受信した時，オムニ・オンになる場合は，次のように記入する。

MONO > Mode 1 または，MONO ON > OMNI ON または，MONO > OMNI

また，モノ・モード ・メッセージの第3バイトMによって，異なる時は，MONO（M = 1）> Mode 3 または MONO（M ≠ 1）> Mode 1 などと書く。送信側には書く欄がない。

3. ノート・ナンバー(Note Number)

ノート・ナンバー: 送信または認識できるノート番号の全範囲を 36-96, 0-127 などと書く。

音域(True Voice): 認識側(受信側)のみ, 実際に発音する範囲を書く。例えば 0-127 まで認識しても, 実際の発音が 88 鍵のピアノの範囲であれば, ここには 21-108 と書かれるであろう。

4. ベロシティ(Velocity)

ノート・オン(Note on): ○または×を記す。送信側が○の場合はその右側の余白にベロシティのとり得る範囲を書いてもよい。例えば, 9n(または 9nH), v=1-127 など。ここの 9n はステータスで, nはチャンネル番号を表す部分である。

ノート・オフ(Note off): ノート・オン欄と同じように書く。送信欄には○でも×でもどのようにして送信するかを書く。例えば, 9n,v=0 とか, 8n,v=64, または, 8n,v=0-127 などである。

5. アフタータッチ(After Touch)

キー別(keys): 各ノートごとのアフタータッチについて○か×かを記す。

チャンネル別(ch's): 各チャンネル毎のアフタータッチについて○か×かを記す。

6. ピッチ・ベンド(Pitch Bend)

ベロシティと同様に○または×を記入する。必数に応じて有効ビット数や, 認識してピッチが変化する範囲を, 9 bit reso(resolution の略), 0-12 semi(semi tone= 半音の略)のように書く。

7. コントロール・チェンジ(Control Change)

送受信できるコントロール番号をいちばん左の欄の右端に書き, 送受信の欄には○か×を記す。備考(Remarks)欄には, 認識したコントロール番号をどのように利用しているかを, モジュレーション, ホールド, ポルタメントなどと書く。送信されるコントロール番号がどのような操作子によるかは, 送信欄の余白に書く。

コントロール番号をすべて記入するため, スペースが開けてあるが, 番号が多くてこの欄に書けない時は, 送受信できる番号か, 番号の範囲を 3, 7, 10 とか 64-75 のように書いて, それぞれの機能の記述は省略する。

8. プログラム・チェンジ(Program Change)

プログラム・チェンジ機能の有無を○か×かで記入する。○の場合には右端の余白にその番号の範囲を 0-15 のように書く。
設定可能範囲(True #(Number))認識の欄のみ, 実際にその番号通りに割り当てられる範囲を 0-15 のように書く。

9. エクスクルーシブ(System Exclusive)

システム・エクスクルーシブ・メッセージ送受信機能の有無を○×で記入する。○の場合は, 備考(Remarks)欄にその用途を書き, その機器のシステム・エクスクルーシブの詳細を, 別ページで付け加えること。

10. コモン(System Common)

システム・コモン・メッセージ送受信機能の有無を○×で記入する。下記のように略記されている。

ソング・ポジション(Song Pos) =ソング・ポジション・ポインター
ソング・セレクト(Song Sel) =ソング・セレクト
チューン(Tune) =チューン・リクエスト

<日本語版注>

ソング・セレクトについては, その範囲を余白に書く。

11. リアルタイム(System Real Time)

システム・リアルタイム・メッセージ送受信機能の有無を○×で記入する。

クロック(Clock): タイミング・クロックについて○か×を記入する。

コマンド(Commands): スタート, コンティニュー, ストップについて, それぞれ○か×を記す。
＜日本語版注＞

これらについて送信または認識のための条件があれば右側余白や備考(Remarks)欄に記入する。例えば, クロックは, 通常は内部に同期していて, ある操作をすれば, MIDI クロックに同期するようになっていることか多いが, その時には○(MIDI mode)などと書く。

12. その他(Aux Messages)

補助的なメッセージの送受信機能の有無を○×で記入する。

ローカル(Local)ON ／ OFF: ローカル・オン／オフ(モード・メッセージで第 2 バイトが 122)。

オール・ノート・オフ : 送信欄には, すべてのノートかオフになった時に送信するメッセージがある場合のみ○を記す。モードをかえる目的でパネル操作その他により送信されるモード・メッセージ(124 以上)については, それらがあっても○としない。認識欄には, モード・メッセージで第 2 バイトが 124 以上のものについて, オール・ノート・オフの機能を認識する場合には, 認識欄の右側余白か備考(Remarks)欄にその第 2 バイトを書く。(例 123-127)

アクティブ・センシング(Active Sense): アクティブ・センシング(FEH)。

リセット(Reset): システム・リセット(FFH)。

13. 備考(Notes)

その機器特有の事項で, それまでの表では表せなかったものについて特記する。例えば, 電源オン時に送信するモード ・メッセージの種類, あるファンクション・スイッチを操作した時に送るメッセージの種類や変化するモード, 受信したメッセージを送信する時にはその条件, 特殊用途のメッセージ, オール・ノート・オフを送信する条件などである。

＜日本語版注＞

追加機能について

リセット・オール・コントローラー このメッセージを送信または認識する場合は, コントロール・チェンジの欄の最下行にその旨を記入する。

MTC クォーター・フレーム このメッセージを送信または認識する場合は, 下の備考の欄にその旨を記入する。

Model

MIDI インプリメンテーション・チャート

Date：
Version：

ファンクション		送信	受信	備考
ベーシック チャンネル	電源 ON 時 設定可能			
モード	電源 ON 時 メッセージ 代用			
ノート ナンバー	音域			
ベロシティ	ノート・オン ノート・オフ			
アフター タッチ	キー別 チャンネル別			
ピッチ・ベンド				
コントロール チェンジ				
プログラム チェンジ	設定可能範囲			
システム・エクスクルーシブ				
コモン	ソング・ポジション ソング・セレクト チューン			
リアル タイム	クロック コマンド			
その他	ローカル ON/OFF オール・ノート・オフ アクティブ・センシング リセット			
備考				

モード1：オムニ・オン, ポリ　　モード2：オムニ・オン, モノ　　○：あり

モード3：オムニ・オフ, ポリ　　モード4：オムニ・オフ, モノ　　×：なし

Table I

ステータス・バイト

ステータス 16 進表示	ステータス 2 進表示	データ・バイト数	説明
チャンネル・ボイス・メッセージ			
8nH	1000nnnn	2	ノート・オフ
9nH	1001nnnn	2	ノート・オン（ベロシティ 0：ノート・オフ）
AnH	1010nnnn	2	ポリフォニック・キー・プレッシャー（アフタータッチ）
BnH	1011nnnn	2	コントロール・チェンジ
CnH	1100nnnn	1	プログラム・チェンジ
DnH	1101nnnn	1	チャンネル・プレッシャー（アフタータッチ）
EnH	1110nnnn	2	ピッチ・ベンド・チェンジ
チャンネル・モード・メッセージ			
BnH	1011nnnn （01111xxx）	2	チャンネル・モード選択
システム・メッセージ			
F0H	11110000	*****	システム・エクスクルーシブ
FsH	11110sss	0 to 2	システム・コモン
FtH	11111ttt	0	システム・リアルタイム

＜備考＞

nnnn ： チャンネル番号（1 ～ 16）を N とした時の N-1 すなわち，0000 がチャンネル 1，0001 がチャンネル 2，…，1111 がチャンネル 16。

*****： 0iiiiiii，データ，…，EOX（iiiiiii：ID コード）

sss： 1 から 7 の値（001=1，…，111=7）

ttt： 0 から 7 の値（000=0，…，111=7）

xxx： チャンネル・モード・メッセージは，コントロール・チェンジ・メッセージ（BnH）と同じステータス・バイトをとる。二者の区別は，1番目のデータ・バイトの値で行う。120（01111000）～ 127（01111111）が，チャンネル・モード・メッセージである。

Table Ⅱ

チャンネル・ボイス・メッセージ

ステータス		データ・バイト	データ・バイト 説明
16 進表示	2 進表示		
8nH	1000nnnn	0kkkkkkk	ノート・オフ
		0vvvvvvv	vvvvvvv：ノート・オフベロシティ
9nH	1001nnnn	0kkkkkkk	ノート・オン
		0vvvvvvv	vvvvvvv ≠ 0：ベロシティ
			vvvvvvv ＝ 0：ノート・オフ
AnH	1010nnnn	0kkkkkkk	ポリフォニック・キー・プレッシャー（アフタータッチ）
		0vvvvvvv	vvvvvvv：プレッシャー値
BnH	1011nnnn	0ccccccc	コントロール・チェンジ（Table Ⅲ参照）
		0vvvvvvv	ccccccc：コントロール番号（0 ～ 119）
			vvvvvvv：コントロール値
			ccccccc ＝ 120 ～ 127:チャンネル･モード･メッセージ（Table Ⅳ参照）
CnH	1100nnnn	0ppppppp	プログラム・チェンジ
			ppppppp：プログラム番号（0-127）
DnH	1101nnnn	0vvvvvvv	チャンネル・プレッシャー（アフタータッチ）
			vvvvvvv：プレッシャー値
EnH	1110nnnn	0lllllll	ピッチ・ベンド・チェンジ
		0hhhhhhh	lllllll：LSB
			hhhhhhh：MSB

＜備考＞

1. nnnn：　チャンネル番号（1 ～ 16）を N とした時の N-1
   すなわち，0000 がチャンネル 1，0001 がチャンネル 2，…，1111 がチャンネル 16
2. kkkkkkk：　ノート番号（0 ～ 127）
3. vvvvvvv：　キー・ベロシティ
4. コンティニュアス・コントローラーの値は，上位 7 ビット（MSB），下位 7 ビット（LSB）ずつのバイトに分けられる。分解能が 7 ビットで十分な場合は，MSB だけを送り LSB を送らなくても良い。さらに，細かい分解能が変化した場合には，LSB だけを送っても良い。

Table Ⅲ

コントロール番号

コントロール番号（第2バイト値）		コントロール機能
10 進表示	16 進表示	
0	00H	バンク・セレクト（MSB）
1	01H	モジュレーション・ホイールまたはレバー
2	02H	ブレス・コントローラー
3	03H	（未定義：予約済）
4	04H	フット・コントローラー
5	05H	ポルタメント・タイム
6	06H	データ・エントリー（MSB）
7	07H	チャンネル・ボリューム（旧メイン・ボリューム）
8	08H	バランス
9	09H	（未定義：予約済）
10	0AH	パン（パンポット）
11	0BH	エクスプレッション・コントローラー
12	0CH	エフェクト・コントロール 1
13	0DH	エフェクト・コントロール 2
14-15	0EH-0FH	（未定義：予約済）
16-19	10H-13H	汎用コントローラー 1 ～4
20-31	14H-1FH	（未定義：予約済）
32	20H	バンク・セレクト（LSB）
33-63	21H-3FH	コントロール番号 1-31 対応の LSB
64	40H	ホールド（ダンパー・ペダル, サステイン）
65	41H	ポルタメント・オン／オフ
66	42H	ソステヌート
67	43H	ソフトペダル
68	44H	レガート・フット・スイッチ
69	45H	ホールド 2
70	46H	サウンド・コントローラー 1（デフォルト：サウンド・バリエーション）
71	47H	サウンド･コントローラー 2（デフォルト:ティンバー／ハーモニック･インテンシティ）
72	48H	サウンド・コントローラー 3（デフォルト：リリース・タイム）
73	49H	サウンド・コントローラー 4（デフォルト：アタック・タイム）
74	4AH	サウンド・コントローラー 5（デフォルト：ブライトネス）
75-79	4BH-4FH	サウンド・コントローラー 6 ～ 10（デフォルト：なし）
80-83	50H-53H	汎用コントローラー（番号5～8）
84	54H	ポルタメント・コントロール
85-90	55H-5AH	（未定義：予約済）
91	5BH	エフェクト 1 デプス（外部エフェクト・デプス）
92	5CH	エフェクト 2 デプス（トレモロ・デプス）
93	5DH	エフェクト 3 デプス（コーラス・デプス）
94	5EH	エフェクト 4 デプス（セレステ・デプス）
95	5FH	エフェクト 5 デプス（フェイザー・デプス）
96	60H	データ・インクリメント
97	61H	データ・デクリメント
98	62H	ノンレジスタード・パラメーター番号（LSB）
99	63H	ノンレジスタード・パラメーター番号（MSB）
100	64H	レジスタード・パラメーター番号（LSB）
101	65H	レジスタード・パラメ [ター番号（MSB）
102-119	66H-77H	（未定義：予約済）
120-127	78-7FH	チャンネル・モード・メッセージ

Table Ⅲ a

レジスタード・パラメーター番号

パラメーター番号		機能
LSB	MSB	
00H	00H	ピッチ・ベンド・センシティビティ
01H	00H	ファイン・チューン
02H	00H	コース・チューン
03H	00H	チューニング・プログラム・セレクト
04H	00H	チューニング・バンク・セレクト
7FH	7FH	ヌル

Table Ⅳ

チャンネル・モード・メッセージ

ステータス 16 進表示　　2 進表示	データ・バイト	説明
BnH　　1011nnnn	0ccccccc 0vvvvvvv	モード・メッセージ ccccccc = 120 : オール・サウンド・オフ vvvvvvv = 0 ccccccc = 121 : リセット オール コントローラー vvvvvvv = 0 ccccccc = 122 : ローカル・コントロール vvvvvvv = 0, ローカル・コントロール・オフ vvvvvvv = 127, ローカル・コントロール・オン ccccccc = 123 : オール・ノート・オフ vvvvvvv = 0 ccccccc = 124 : オムニ・オフ（オール・ノート・オフ） vvvvvvv = 0 ccccccc = 125 : オムニ・オン（オール・ノート・オフ） vvvvvvv = 0 ccccccc = 126 : モノ・モード・オン（ポリ・モード・オフ）（オール・ノート・オフ） vvvvvvv = M, オムニ・オフ時有効で, Mはチャンネル数（1 ～ 16）。 vvvvvvv = 0, オムニ・オフ時有効で, (16-N ＋ 1)が最大割当て可能なチャンネル数 ccccccc = 127 : ポリ・モード・オン（モノ・モード・オフ）（オール・ノート・オフ） vvvvvvv = 0

＜備考＞

1. nnnn :　　ベーシック・チャンネル番号（1 ～ 16）を N とした時の N-1
   すなわち, 0000 がチャンネル 1, 0001 がチャンネル 2, …, 1111 がチャンネル 16
2. ccccccc :　　コントロール番号（120 ～ 127）
3. vvvvvvv :　　コントロール値

Table V

システム・コモン・メッセージ

ステータス		データ・バイト	説明
16 進表示	2 進表示		
F1H	11110001	0nnndddd	MIDI タイムコード・クォーター・フレーム
			nnn : メッセージ・タイプ
			dddd : 値
F2H	11110010		ソング・ポジション・ポインター
		0lllllll	lllllll : LSB
		0hhhhhhh	hhhhhhh : MSB
F3H	11110011	0sssssss	ソング・セレクト
			sssssss : ソング番号
F4H	11110100		(未定義 : 予約済)
F5H	11110101		(未定義 : 予約済)
F6H	11110110	なし	チューン・リクエスト
F7H	11110111	なし	EOX (End Of eXclusive)

Table Ⅵ

システム・リアルタイム・メッセージ

ステータス		データ・バイト	説明
16 進表示	2 進表示		
F8H	11111000	なし	タイミング・クロック
F9H	11111001		(未定義 : 予約済)
FAH	11111010	なし	スタート
FBH	11111011	なし	コンティニュー
FCH	11111100	なし	ストップ
FDH	11111101		(未定義 : 予約済)
FEH	11111110	なし	アクティブ・センシング
FFH	11111111	なし	システム・リセット

Table Ⅶ

システム・エクスクルーシブ・メッセージ

ステータス 10 進表示　16 進表示	データ・バイト	説明
F0H　11110000		システム・エクスクルーシブ・ステータス
	0iiiiiii	ID コード（備考 1. 参照） （00-7CH）メーカー ID （7DH）非営利目的 （7EH）ノンリアルタイム・システム・エクスクルーシブ （7FH）リアルタイム・システム・エクスクルーシブ
	0ddddddd ・ ・ ・ ・ 0ddddddd	データ・バイトの長さは任意（最上位ビットは 0）
F7H　11110111		EOX（End Of eXclusive）

＜備考＞

1. 0iiiiiii：

A）ID コード（0-127）。ID が0の場合は，次に来る2バイトが ID の拡張用として使用される。

B）7DH は学校，研究所などの営利を目的としない団体のため使用される。営利目的には使用しない。

C）7EH（ノンリアルタイム）及び 7FH（リアルタイム）は，MIDI 規格の拡張として使用される。Table Ⅶ a を参照。

2. 0ddddddd：

システム・エクスクルーシブのステータス・バイトと，EOX との間のすべてのバイトにおける最上位ビットは，0 でなければならない。ID コードが認識されて実行されるか否かにかかわらず，システム･エクスクルーシブのメッセージ中に，リアルタイム･メッセージ以外のメッセージをさしはさんではならない。

Table VII a

ユニバーサル システム・エクスクルーシブ ID 番号

サブ ID 番号 1	サブ ID 番号 2	説明
ノンリアルタイム (7EH)		
(すべて 16 進表示)	(すべて 16 進表示)	
00	未使用	
01	(not used)	サンプル・ダンプ・ヘッダー
02	(not used)	サンプル・ダンプ・パケット
03	(not used)	サンプル・ダンプ・リクエスト
04	nn	MIDI タイムコード (MTC)
	00	スペシャル
	01	パンチ・イン・ポイント
	02	パンチ・アウト・ポイント
	03	デリート・パンチ・イン・ポイント
	04	デリート・パンチ・アウト・ポイント
	05	イベント・スタート・ポイント
	06	イベント・ストップ・ポイント
	07	イベント・スタート・ポイント (追加情報あり)
	08	イベント・ストップ・ポイント (追加情報あり)
	09	デリート・イベント・スタート・ポイント
	0A	デリート・イベント・ストップ・ポイント
	0B	キュー・ポイント
	0C	キュー・ポイント (追加情報あり)
	0D	デリート・キュー・ポイント
	0E	追加情報中のイベント・ネーム
05	nn	サンプル・ダンプ拡張
	01	ループ・ポイント・トランスミッション
	02	ループ・ポイント・リクエスト
06	nn	一般情報
	01	アイデンティティ・リクエスト
	02	アイデンティティ・リプライ
07	nn	ファイル・ダンプ
	01	ヘッダー
	02	データ・パケット
	03	リクエスト
08	nn	MIDI チューニング・スタンダード
	00	バルク・チューニング・ダンプ・リクエスト
	01	バルク・チューニング・ダンプ
09	nn	GM
	01	GM システム・オン
	02	GM システム・オフ
7B		ファイルの終わり (EOF)
7C		ウェイト
7D		キャンセル
7E		否定応答 (NAK)
7F		肯定応答 (ACK)

サブ ID 番号 1	サブ ID 番号 2	説明
リアルタイム（7FH）		
（すべて 16 進表示）	（すべて 16 進表示）	
00		未使用
01	nn	MIDI タイムコード（MTC）
	01	フル・メッセージ
	02	ユーザー・ビット
02	nn	MIDI ショー・コントロール (MSC）
	00	MSC Extensions
	01-7F	MSC Commands
		（詳細は MSC 参照）
03	nn	記譜情報（ノーテーション・インフォメーション）
	01	バー・マーカー（小節マーカー）
	02	拍子記号（イミディエイト）
	42	拍子記号（ディレイド）
04	nn	デバイス制御
	01	マスター・ボリューム
	02	マスター・バランス
05	nn	リアルタイム MTC キューイング
	01	スペシャル
	01	パンチ・イン・ポイント
	02	パンチ・アウト・ポイント
	03	（Reserved）
	04	（Reserved）
	05	イベント・スタート・ポイント
	06	イベント・ストップ・ポイント
	07	イベント・スタート・ポイント（追加情報あり）
	08	イベント・ストップ・ポイント（追加情報あり）
	09	（Reserved）
	0A	（Reserved）
	0B	キュー・ポイント
	0C	キュー・ポイント（追加情報あり）
	0D	（Reserved）
	0E	追加情報中のイベント・ネーム
06	nn	MIDI マシン・コントロール・コマンド
	00-7F	MMC Commands（詳細は MMC 参照）
07	nn	MIDI マシン・コントロール・レスポンス
	00-7F	MMC Commands（詳細は MMC 参照）
08	nn	MIDI チューニング・スタンダード
	02	シングル・ノート・チューニング・チェンジ

＜備考＞

1. リアルタイム及びノンリアルタイム・ユニバーサル・エクスクルーシブ・メッセージの基本フォームは同じ。
   F0H ＜ ID 番号＞＜デバイス ID ＞＜サブ ID 番号 #1 ＞＜サブ ID 番号 #2 ＞ F7H

2. さらに詳しい内容は，MTC，MSC，MMC のドキュメントを参照。

Table Ⅷ b

システム・エクスクルーシブ・マニュファクチャーID表(すべて 16 進表示)

ID 番号	メーカー
American Group	
01	Sequential
02	IDP
03	Voyetra/Octave-Plateau
04	Moog
05	Passport Designs
06	Lexicon
07	Kurzweil
08	Fender
09	Gulbransen
0A	AKG Acoustics
0B	Voyce Music
0C	Waveframe Corp
0D	ADA Signal Processors
0E	Garfield Electronics
0F	Ensoniq
10	Oberheim
11	Apple Computer
12	Grey Matter Response
13	Digidesign
14	Palm Tree Instruments
15	JL Cooper Electronics
16	Lowrey
17	Adams-Smith
18	Emu Systems
19	Harmony Systems
1A	ART
1B	Baldwin
1C	Eventide
1D	Inventronics
1F	Clarity
00 00 01	Time Warner Interactive
00 00 07	Digital Music Corp.
00 00 08	IOTA Systems
00 00 09	New England Digital
00 00 0A	Artisyn
00 00 0B	IVL Technologies
00 00 0C	Southern Music System
00 00 0D	Lake Butler Sound Company
00 00 0E	Alesis
00 00 10	DOD Electronics
00 00 11	Studer-Editech
00 00 14	Perfect FretWorks
00 00 15	KAT
00 00 16	Opcode
00 00 17	Rane corp.
00 00 18	Anadi Inc.
00 00 19	KMX
00 00 1A	Allen & Heath Brenell
00 00 1B	Peavey Electronics
00 00 1C	360 Systems
00 00 1D	Spectrum Design and Development
00 00 1E	Marquis Music

ID 番号	メーカー
00 00 1F	Zeta Systems
00 00 20	Axxes
00 00 21	Orban
00 00 24	KTI
00 00 25	Breakaway Technologies
00 00 26	CAE
00 00 29	Rocktron Corp
00 00 2A	PianoDisc
00 00 2B	Cannon Research Group
00 00 2D	Rogers Instrument Corp.
00 00 2E	Blue Sky Logic
00 00 2F	Encore Electronics
00 00 30	Uptown
00 00 31	Voce
00 00 32	CTI Audio,Inc.(Music. Intel Dev.)
00 00 33	S&S Reseach
00 00 34	Broderbund Software,Inc
00 00 35	Allen Organ Co.
00 00 37	Music Quest
00 00 38	APHEX
00 00 39	Gallien Krueger
00 00 3A	IBM
00 00 3C	Hotz Instruments Technologies
00 00 3D	ETA Lighting
00 00 3E	NSI Corporation
00 00 3F	Ad Lib,Inc
00 00 40	Richmond Sound Design
00 00 41	Microsoft
00 00 42	The Software Toolworks
00 00 43	Niche/RJMG
00 00 44	Intone
00 00 47	GT Electronics/Groove Tubes
00 00 48	InterMIDI,Inc.
00 00 49	Timeline Vista
00 00 4A	Mesa Boogie
00 00 4C	Sequoia Development
00 00 4D	Studio Electronics
00 00 4E	Euphonix
00 00 4F	InterMIDI
00 00 50	MIDI Solutions
00 00 51	3DO Company
00 00 52	Lightwave Research
00 00 53	Micro-W
00 00 54	Spectral Synthesis
00 00 55	Lone Wolf
00 00 56	Studio Technologies
00 00 57	Peterson EMP
00 00 58	Atari
00 00 59	Marion Systems
00 00 5A	Design Event
00 00 5B	Winjammer Software
00 00 5C	AT&T Bell Labs

ID 番号	メーカー
00 00 5E	Symetrix
00 00 5F	MIDI the World
00 00 60	Desper Products
00 00 61	Micros 'N MIDI
00 00 62	Accordians Intl
00 00 63	EuPhonics
00 00 64	Musonix
00 00 65	Turtle Beach Systems
00 00 66	Markie Design
00 00 67	Compuserve
00 00 68	BES Technologies
00 00 69	QRS Music Rolls
00 00 6A	P G Music
00 00 6B	Sierra Semiconductor
00 00 6C	EpiGraf Audio Visual
00 00 6D	Electronics Deiversified
00 00 6E	Tune1000
00 00 6F	Advanced Micro Devices
00 00 70	Mediamation
00 00 71	Sabine Music
00 00 72	Woog Labs
00 00 73	Micropolis
00 00 74	Ta Horng Musical Inst.
00 00 75	eTek (formerly Force)
00 00 76	Electrovoice
00 00 77	Midisoft
00 00 78	Q-Sound Labs.
00 00 79	Westrex
00 00 7A	NVidia
00 00 7B	ESS Technology
00 00 7C	Mediatrix Peripherals
00 00 7D	Brooktree
00 00 7E	Otari
00 00 7F	Key Electronics
00 01 01	Crystalake Multimedia
00 00 02	Crystal Semiconductor
00 00 03	Rockwell Semiconductor

European Group

ID 番号	メーカー
20	Passac
21	SIEL
22	Synthaxe
24	Hohner
25	Twister
26	Solton
27	Jellinghaus MS
28	Southworth Music System
29	PPG
2A	JEN
2B	SSL Limited
2C	Audio Veritreib
2F	Elka
30	Dynacord
31	Viscount
33	Clavia Digital Instruments
34	Audio Architecture
35	GeneralMusic Corp.
39	Soundcraft Electronics
3B	Werist
3C	Avab Electronik Ab
3D	Digigram
3E	Waldorf Electronics
3F	Qasimidi
00 20 00	Dream
00 20 01	Strand Lighting
00 20 02	Amek Systems
00 20 04	Bohm Electronic
00 20 06	Trident Audio
00 20 07	Real World Studio
00 20 09	Yes Technology
00 20 0A	Audiomatica
00 20 0B	Bontempi/Farfisa
00 20 0C	F.B.T. Elettronica
00 20 0D	Miditemp
00 20 0E	LA Audio (Larking Audio)
00 20 0F	Zero 88 Lighting Limited
00 20 10	Micon Audio Electronics GmbH
00 20 11	Forefront Technology
00 20 13	Kenton Electronics
00 20 15	ADB
00 20 16	Marshall Products
00 20 17	DDA
00 20 18	BSS
00 20 19	MA Lighting Technology
00 20 1A	Fater
00 20 1B	QSC Audio
00 20 1C	Artisan Classic Organ
00 20 1D	Orla Spa
00 20 1E	Pinnacle Audio
00 20 1F	TC Electonics
00 20 20	Doepfer Musikelectronik
00 20 21	Creative Technology Pte
00 20 22	Minami/Seiyddo
00 20 23	Goldstar
00 20 24	Midisoft s.a.s di M Cima
00 20 25	Samick
00 20 26	Penny and Giles
00 20 27	Acorn Computer
00 20 28	LSC Electronics
00 20 29	Novation EMS
00 20 2A	Samkyung Mechartronics
00 20 2B	Medeli Electronics
00 20 2C	Charlie Lab
00 20 2D	Blue Chup Music Tech
00 20 2E	BEE OH Corp

ID 番号	メーカー

Japanese Group（1998 年 12 月）

40	株式会社河合楽器製作所
41	ローランド株式会社
42	株式会社コルグ
43	ヤマハ株式会社
44	カシオ計算機株式会社
46	株式会社カミヤスタジオ
47	赤井電機株式会社
48	日本ビクター株式会社
49	（明創社）
4A	（星野楽器）
4B	富士通株式会社
4C	ソニー株式会社
4D	（日伸音波研究所）
4E	ティアック株式会社
50	松下電器産業株式会社
51	フォステクス株式会社
52	株式会社ズーム
53	（緑電子工業株式会社）
54	松下通信工業株式会社
55	株式会社鈴木楽器製作所
56	不二音響株式会社
57	株式会社音響総合研究所

Table VIII

追加（Recommended Practice）

タイトル	説明
MIDI タイムコード	RP004/RP008
MIDI ショー・コントロール 1.1	RP002/RP014
MIDI マシン・コントロール	RP013
スタンダード MIDI ファイル	RP001
GM システム・レベル1	RP003
記譜情報（ノーテーション・インフォメーション）	RP05/RP06
ファイル・ダンプ	RP09

3. MIDI タイムコード

MIDI タイムコード
RP004/RP008

Copyright ©1998 社団法人　音楽電子事業協会

MIDIタイムコード

MIDI タイムコードには，機器の同期用として，クォーターフレーム及びフルというふたつの基本メッセージ及び SMPTE のユーザー・ビット用のオプション・メッセージが設けられている。

＜日本語版注＞

NTSC ビデオ(及びそれに対応するタイムコード)の物理レートは29.97 フレーム / 秒である。物理的に 30 フレーム / 秒であるビデオ（タイムコード）も存在する。一般に「30 フレーム / 秒」という表記は，「30 と 29.97 を区別した上での 30」という意味の場合と，「両者の総称」である場合があるので，注意を要する。この章「MIDI タイムコード」での意味は「総称」である。

クォーター・フレーム・メッセージ

クォーター・フレーム・メッセージが使用されるのは，システムが動作している間のみである。これまでになじみのあるPPQNや MIDI クロックに似ているが，いくつかの点で重要な相違がある。

名前からわかるように，これらのメッセージの分解能は非常に高い。例えば，秒あたりのフレームを 30 とすると，クォーター・フレーム・メッセージの数は1秒につき 120 となる。この場合，最大待ち時間は 8.3ms である（秒あたり 30 フレームとして）。機器によってはこれ以上の精度も期待できる（クォーター・フレームの間を補間すれば）。クォーター・フレーム・メッセージはふたつの目的に使用できる。ひとつは，システムの基本タイミング・パルスとして，もうひとつはカレント SMPTE タイムを送るためである。

クォーター・フレーム・メッセージは，8つのメッセージからなるグループの集合と考えられる。これらのグループがそれぞれ，SMPTE タイムの時，分，秒，フレームのいずれかを分担して定義する。ひとつのタイムコード・メッセージには，8つのクォーター・フレームが必要なので，SMPTE タイムは2フレームごとに更新される。各クォーター・フレーム・メッセージは，2バイトで構成される。最初のバイト F1H で，これはクォーター・フレーム・システム・コモン・バイトである。2つ目のバイトは，メッセージ・タイプ（0から7まで）を表す3ビットと，時間（時，分，秒，フレーム）を表す4ビット（日本語版注：BCD ではなく，2進数表現の上位 / 下位4ビット）からなる。

クォーター・フレーム・メッセージ（2バイト）

F1 <message>

F1	**システム・コモン・ステータス・バイト**
<message>	**0nnn dddd**
	nnn：メッセージ・タイプ
	0 = フレーム・カウント下位 4 ビット
	1 = フレーム・カウント上位 4 ビット
	2 = 秒カウント下位4ビット
	3 = 秒カウント上位4ビット
	4 = 分カウント下位4ビット
	5 = 分カウント上位4ビット
	6 = 時刻カウント下位4ビット
	7 = 時刻カウント上位4ビットと SMPTE タイプ
	dddd：このメッセージ・タイプで表す4ビットのバイナリィ・データ

前記カウントの下位及び上位4ビットが組み合わせられると，これらのビット・フィールドは，次のように割り当てられる。

フレーム・カウント：xxx yyyyy

xxx	**未定義，将来のための予備。** **送信側はこれらのビットを 0 にセットして送信すること。また，受信側は，これらのビットを無視すること。**
yyyyy	**フレーム番号（0-29）**

秒カウント：xx yyyyyy

xx	**未定義，将来のための予備。** **送信側はこれらのビットを 0 にして送信すること。また受信側は，このビットを無視すること。**
yyyyyy	**秒カウント（0-59）**

分カウント：xx yyyyyy

xx	**未定義，将来のための予備。** **送信側はこれらのビットを 0 にして送信すること。また受信側は，このビットを無視すること。**
yyyyyy	**分カウント（0-59）**

時カウント：x yy zzzzz

x	**未定義，将来のための予備。** **送信側はこのビットを 0 にして送信すること。また受信側は，このビットを無視すること。**
yy	**タイムコード・タイプ：** **0 = 24 フレーム / 秒** **1 = 25 フレーム / 秒** **2 = 30 フレーム / 秒（ドロップ・フレーム）** **3 = 30 フレーム / 秒（ノンドロップ）**
zzzzz	**時カウント（0-23）**

クォーター・フレーム・メッセージの運用

タイムコードが前進方向の場合，MIDI タイムコードを発生する機器は，クォーター・フレーム・メッセージを次の順序で4分の1フレームの時間間隔で送信する。

F1 0X
F1 1X
F1 2X
F1 3X
F1 4X
F1 5X
F1 6X
F1 7X

以後このシーケンスは，8メッセージを2フレーム（8クォーター・フレーム）ごとに，自動的に繰り返す。逆方向の場合，クォーター・フレーム・メッセージも F1H 7XH から F1H 0XH へと逆順で送信される。この場合も，最低8クォーター・フレーム・メッセージが送信されなければならない。F1H 0XH 及び F1H 4XH は，常にフレームの境界を示す。

SMPTE の実際の時間は，8個のクォーター・フレーム・メッセージによって定まるものであるから，読取り側が8個のメッセージを最初から最後まで完全に読み終わるまでタイミング・クロックが完成しない。読取り側がいつオンラインとなるかによって異なるが，2から4フレームを要する。

早送り，巻き戻し，またはシャトル・モードの時，タイムコード・ジェネレーターはクォーター・フレーム・メッセージの送信を停止し，最終目的に到達した時点でフル・メッセージのみを送信する。その後，他の機器がその点までシャトルする間停止し，プレイ・モードが再開されると，クォーター・フレーム・メッセージの送信を再開する。フル・メッセージの後の最初のクォーター・フレーム・メッセージを受信した時点から，タイムはランしているものと見なされる。

高速のシャトル・モードでは，クォーター・フレーム・メッセージを連続で送信してはならない。MIDI のデータ・ラインを不必要に混雑させてしまうからである。長いシャトル中に機器のタイムコードを更新したい場合は，フル・メッセージを必要に応じて送信すればよい。

フレーム境界点では，クォーター・フレーム・メッセージ F1H 0XH（フレーム・カウント下位4ビット）を送信しなければならない。フレーム・カウントで示されるフレーム番号が，その境界から開始するフレーム番号となる。これは，通常の SMPTE LTC（Longitudunal Time Code）（日本語版注：最新の SMPTE では Linear Time Code）の規格に準じたもので，80 ビット・メッセージのビット 00 の到着と,そのビットが指示するフレームの開始は正確に一致する。8メッセージの送信には2フレームを要するので,SMPTE タイムコードは，各8メッセージごとに2フレーム増加する。

これを別の面から説明する。クォーター・フレーム・メッセージの最後のメッセージ（F1 7X）が到着してタイムが完全に構成されると，それはすでに2フレーム分遅い情報となる。したがって，受信側では2フレームを加算してディスプレイできるような内部オフセットを保持しなければならない。特異に思われようが，SMPTE では通常このような受信方式が行われ，また同様に，逆進の場合（タイムコードを逆方向に走らせる）の混乱を防いでいる（8つクォーター・フレーム・メッセージを逆順で受信している時，F1H 0XH メッセージはやはりそれが表現するフレームの境界上にある）。

各クォーター・フレーム・メッセージ番号（0～7）は，2フレーム・シーケンス中の8つのクォーター・フレームのどれであるかを示している。例えば，メッセージ 0（F1 0X）は，シーケンス中のフレーム番号 #1 の最初のクォーター・フレームであることを示す。メッセージ4（F1 4X）は，フレーム番号 #2 の最初のクォーター・フレームであることを示している。読み取り側がこれらのメッセージを降順で受信した場合,タイムコードが逆方向に送られていることがわかる。また,読み取り側がいつオンラインとなっても,その時点で受け取っているクォー

ター・フレームが，2フレーム・シーケンス中のどれにあたるかを正確に知ることができる。

＜日本語版注＞

この段落で言う「タイムコード・リーダー」とは，レコーダー等から来たSMPTEタイムコードを読み取り，MTCに変換して出力する装置のことである。

MTCが正確に解釈されるようにするのは，MTCを出力する側であるタイムコード・リーダーの責任である。したがってリーダーは，タイムコード･ロックが得られるまで，必要なだけの時間をかけ，また同期が外れるまで，このロックを保持しなければならない。各クォーター・フレーム・メッセージから相対的なクォーター・フレーム・カウントを知ることはできるが，適切なロックを保証するためには，8メッセージ･シーケンス（2フレーム）が終了した段階で実際のタイムコードを確認する必要がある。同期が外れた場合，送信側はNAKメッセージを送らねばならない。受信側は，これを「テープ停止」と解釈し，持続中の音，その他が残っていればそれをオフにしなければならない。

＜日本語版注＞

このような場合に適応できるMIDIのNAKメッセージは定義されていない。なお，MMC（MIDIマシン・コントロール）のメッセージでは，タイムコードのステータスを表現することができる。

例題として，タイムが01:37:52:16（秒あたり30フレーム，ノンドロップ）であると仮定する。タイムは下位桁から上位桁へと送信されるので，最初の2クォーター・フレーム・メッセージは，データ16（フレーム）を含んでいる。2番目の2つはデータ52（秒），3番目のふたつは37（分），そして最後の2つは1（時間及びSMPTEタイプ）を表している。各タイム・フィールド内では，バイナリィ・データがどのように2つの4ビットに分配されているかは，クォーター・フレーム・メッセージの解説欄で説明されている。本規格では（単純なBCDと異なり），SMPTEタイプのエンコード用（及び将来の使用に備えた）のビットを残してある。

さて，上記例の01:37:52:16を正確な16進に置き換えながら，クォーター・フレーム・フォーマットに変換してみる。

F1 00
F1 11　10H = 10進16

F1 24
F1 33　34H = 10進52

F1 45
F1 52　25H = 10進37

F1 61
F1 76　01H = 10進01（SMPTEタイプは30フレーム / ノンドロップ）

＜備考＞

SMPTEタイプ（2進で11）は，ビット5と6にエンコードされて送信されるので，送信値は“6”となる。

SMPTEタイプが24，30ドロップ・フレーム，及び30ノンドロップ・フレームの場合，フレーム・カウントは常に偶数となる。SMPTEタイプが25では，8メッセージ・シーケンスがどのフレーム・カウントから開始したかによって，偶数にも奇数にもなりうる。この場合，MIDIタイムコードのフレーム・カウントが，どこで秒ごとに偶数と奇数に交代するかを知ることが可能である。

MIDIタイムコードがMIDIの帯域に占める割合は非常に小さい。最も速いSMPTEタイムレートでも秒あたり30フレームである。本仕様では，フレームあたり4メッセージ，つまり8.333msごとに2バイト・メッセージ（640 μs）を送信することになっている。これがMIDI帯域幅に占める割合は7.68％で，妥当と思われる数値である。また，通常のMIDIタイムコード・システムでは，一般のMIDIと同一バスを同時に使用することは，まずあり得ないと考えられる。

＜備考＞

VITC 信号を SMPTE-MIDI 変換回路を通じて MIDI タイムコードを駆動した場合，MIDI タイムコードを期待通り2フレーム進めることはできない。1フレームだけか，またはまったく進まないかもしれない。また，偶数のフレーム・カウントは奇数のフレーム・カウントに変わってしまうかもしれない。同期を取るためには，SMPTE の最初の4ビット（少なくとも）が受信されるまで MTC を受信しないことである。これによって，フレームが進んだかどうかを判断することができる。

フル・メッセージ

クォーター・フレーム・メッセージは，システムの基本動作を取り扱うが，早送りや巻き戻し，または特定時間への位置決めや，キュー等に使用するには適していない。これは，このメッセージを高速かつ連続的に送信すると，MIDI データ・ラインを必要以上に混雑させたり，ラインの容量を越えてしまったりするからである。このような場合には，1フレームの時刻情報全部を単一メッセージにエンコードしてしまうフル・メッセージを使用する。

フル・メッセージを送信してしまうと，タイムコード・ジェネレーターは，メカニカルな機器がその点までシャトル（またはオート・ロケート）するまでの間，休止することができる。その後，クォーター・フレーム・メッセージを送信して動作を再開する。

フル・メッセージ（10バイト）

F0 7F <device ID> 01 01 hr mn sc fr F7

F0 7F	**リアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー**
<device ID>	**7F（システム全体を意図されたメッセージ）**
01	**サブ ID 番号＃ 1 ＝ MIDI タイムコード**
01	**サブ ID 番号＃ 2 ＝フル・タイムコード・メッセージ**
hr	**時間とタイプ：0 yy zzzzz** yy：**タイプ** **00 ＝ 24 フレーム／秒** **01 ＝ 25 フレーム／秒** **10 ＝ 30 フレーム／秒（ドロップ・フレーム）** **11 ＝ 30 フレーム／秒（ノンドロップ）** zzzzz：**時間 (00 ～ 23)**
mn	**分 (00 ～ 59)**
sc	**秒 (00 ～ 59)**
fr	**フレーム (00 ～ 29)**
F7	**EOX**

フル・メッセージ以後の，最初のクォーター・フレーム・メッセージを受信した瞬間から，時間は進行しているものとみなされる。

ユーザー・ビット

”ユーザー・ビット”は，特殊機能用として SMPTE が制定した 32 ビットである。機能は適用例によって異なるし，またその目的に合わせて設計された機器からでしかプログラムすることはできない。最大4文字,または8数字を書き込むことができる。使用例としては，日付またはリール番号をテープに追加する，等である。ユーザー・ビットがテープなどの走行中に変化するような使われ方は少ない。

ユーザー・ビット・メッセージ（15 バイト）

F0 7F <device ID> 01 02 u1 u2 u3 u4 u5 u6 u7 u8 u9 F7

F0 7F	**リアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー**
<device ID>	**7F（システム全体を意図されたメッセージ）**
01	**サブ ID 番号＃ 1 ＝ MIDI タイムコード**
02	**サブ ID 番号＃ 2 ＝ユーザー・ビット・メッセージ**
u1	0000aaaa
u2	0000bbbb
u3	0000cccc
u4	0000dddd
u5	0000eeee
u6	0000ffff
u7	0000gggg
u8	0000hhhh
u9	000000ji
F7	**EOX**

u1 から u8 までのメッセージ・バイトは，SMPTE/EBU バイナリィ・グループの1から8にそれぞれ相当している。バイト u9 は，SMPTE/EBU バイナリィ・グループ・フラッグ・ビットを含み，そこではjは SMPTE タイムコードのビット 59（EBU ビット 43）に，i は SMPTE タイムコードのビット 43（EBU ビット 27）に相当している。

バイナリィ・グループのニブル1- 8が，8ビットの情報を表現するのに使用される場合は，それらは以下の順で，4つの8ビット・キャラクターとしてまとめられるべきである（hhhhgggg ffffeeee ddddcccc bbbbaaaa）。それらが BCD 形式（このような使い方も多い）でタイムコードを表現するのに使用される場合には，フレームの一の位はグループ1に，そして時の十の位はグループ8になる。正しく表示するためには，グループ8でスタートし，グループ1で終わる。8ビットキャラクターの場合と同じ，hhhhgggg ffffeeee ddddcccc bbbbaaaa の順である。

このメッセージは，同一ラインの機器に対し，ユーザー・ビットの転送が必要な場合には，いつでも送信することができる。ユーザー・ビット・メッセージは，MIDI タイムコード・コンバーターにより，いつでも送信することができる。また，モードに関係なく使用できる。

＜注意＞

このメッセージは，SMPTE タイムコードが読まれる方法をより正確に反映するために，1991 年 11 月に再定義された。オリジナルのバージョンでは，次のように指定してあった。8ビット・コードは，aaaabbbb ccccdddd eeeeffff gggghhhh ii の4個の8ビット文字と1個の2ビット・コードにデコードされる。

MIDIキューイング

＜日本語版注＞

MIDI キューイングは，レコーダーやシーケンサーに，自動パンチ・イン／アウト等の動作を行わせるための，設定方法を提供する。この規格は，MMC（MIDI マシンコントロール）が登場する以前に制定された。MIDI キューイングの機能には,MMC によって実現可能なものもある。MMC と RP の「付録D」に,両者の比較や,MIDI キューイング・メッセージを MMC コマンドで置き換える方法などが述べられている。

MIDI キューイングは，システム内の個別の機器に指示するのに，セットアップ・メッセージを使用する（個別の機器とは，MTR，VTR，エフェクト・ジェネレーター，MIDI シーケンサー等を指す）。

128 のイベント・タイプのうち，現在定義済みのものは次の通りである。

セットアップ・メッセージ（13 バイト＋追加情報）

F0 7E <device ID> 04 <sub-ID#2> hr mn sc fr ff sl sm <additional info.> F7

F0 7E	ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー
<device ID>	ユニットのデバイス ID
04	サブ ID 番号＃ 1 ＝ MIDI タイムコード
<sub-ID#2>	セットアップ・タイプ
00	スペシャル
01	パンチ・イン・ポイント
02	パンチ・アウト・ポイント
03	デリート・パンチ・イン・ポイント
04	デリート・パンチ・アウト・ポイント
05	イベント・スタート・ポイント
06	イベント・ストップ・ポイント
07	イベント・スタート・ポイント（追加情報あり）
08	イベント・ストップ・ポイント（追加情報あり）
09	デリート・イベント・スタート・ポイント
0A	デリート・イベント・ストップ・ポイント
0B	キュー・ポイント
0C	キュー・ポイント（追加情報あり）
0D	デリート・キュー・ポイント
0E	追加情報中のイベント・ネーム
hr	時刻とタイプ：0 yy zzzzz
	yy：　タイプ
	00 ＝ 24 フレーム / 秒
	01 ＝ 25 フレーム / 秒
	10 ＝ 30 フレーム / 秒（ドロップ・フレーム）
	11 ＝ 30 フレーム / 秒（ノンドロップ）
	zzzzz：　時 (00 ～ 23)
mn	分 (00 ～ 59)
sc	秒 (00 ～ 59)
fr	フレーム (00 ～ 29)
ff	フラクショナル・フレーム (00 ～ 99)
sl　sm	イベント番号 / スペシャルのタイプ（LSB から）
<add. info.>	追加情報
F7	EOX

MTC キューイング・セットアップ・タイプ解説

00　スペシャルとは，トラック，サウンド，プログラム，シーケンス等の個別のものではなく，ユニット全体に影響を与えるセットアップ情報をいう。この場合，フォーマット中の sl sm はスペシャルのタイプを表す。6つが定義済み。タイプ 01 00 から 04 00 までは，イベント・タイム・フィールドを無視していることに注意。

00 00　タイムコード・オフセットとは，各ユニットにおける相対的なタイムコード・オフセットをいう。例えば，同時に再生される音楽と映像が別々の時間に制作され，絶対時間コードの位置も異なるといったことは，充分考えられる。したがって，どちらか一方をオフセットにして，タイミングを合わせる必要がある。システム全体で唯一のマスター・タイムコードが存在するようにすれば，各ユニットごとにひとつのオフセットを持てばよい。

01 00　イネーブル・イベント・リストは，適当な MTC または SMPTE タイムが発生すると，そのリスト中のイベントを実行することを可能にする。

02 00　ディスエーブル・イベント・リストは，あるユニットのリスト中のイベントを実行できなくすること。ただし，リストから消去してしまうわけではない。複雑なシステムでは，多数のイベントが同時に進行するが，特定のイベントを強調するため，他のイベントをミュートする必要も生じる。そのような時に MTC のイベント・マスターが使用する。

03 00　クリア・イベント・リストは，あるユニットのイベント・リスト全部を消去してしまうこと。

04 00　システム・ストップとは，ユニットが停止してもよいタイムをいう。これはイベント・ストップと組み合わされていないイベント・スタートや，リールの最後に来てもまだ回転しているテープ・マシン等を保護するたものもの。

05 00　イベント・リスト・リクエストは，マスターが MTC 周辺機器へ送信する。これを受け取った周辺機器は，自己のデバイス ID 番号（チャンネル番号）が一致している場合，イベント・リスト・リクエスト・メッセージで指定された SMPTE タイム以降の自己のキュー・リストすべてをセットアップ・メッセージの形式で送信する。

01/02　パンチ・インとパンチ・アウトは，あるユニットのレコード・モードのオン，オフをいう。イベント番号は，レコードすべきトラック。複数のパンチ・イン / パンチ・アウト・ポイントは（及びその他下記のイベント・タイプも同様），複数のセットアップ・メッセージを，それぞれ異なる時間情報を付けて送信することによって指定することができる。

03/04　デリート・パンチ・イン / アウトは，キュー・リストからタイム及びイベントの番号の一致したところを削除する。

05/06　イベント・スタート / ストップは，あるイベントの走行またはプレイバックを指す。また，連続する一連のイベントの開始または停止を意味することもある。イベント番号は目的のスレーブのどのイベントをプレイすべきかを指示する。あるイベント（特定サンプルのプレイバック，自動コンソールのフェーディング等）が一連のキュー・リスト中で何回も起こることはあり得ることであるが，これらは，異なったスタート及びストップ・タイムの同じイベント番号で表すことができる。

07/08　追加情報ありイベント・スタート / ストップとは，イベント（上記例のような）に関する追加パラメーターが，タイムと EOX の間のセットアップ・メッセージ中で送信されることをいう。パラメーターの内容は，後に解説があるが，エフェクターのパラメーターであったり，サウンド・エフェクトの音量レベルであったりする。

09/0A　デリート・イベント・スタート / ストップは，タイム及びイベント番号の一致したイベント（追加情報あり，またはなしの両タイプ）をキュー・リストから削除する。

0B　キュー・ポイントとは，個々のイベントの発生をいう。例えば，サウンド・エフェクトの”hit”ポイントやエディットの基準点をマークする，等である。キュー番号は，特別な瞬時動作，例えばワンショット・サウンド・イベント等を指示するのに使用することも可能である(非連続的なイベントの連続的なものは,スタート / ストップで取り扱う)。あるキュー・リストの中で,同じキューが何回も出てくることもあり得るが，この場合は，同一キュー番号で表示する。ただし，そのタイムはそれぞれ異なる。

0C　追加情報ありキュー・ポイントは，追加情報ありイベント・スタート / ストップとまったく同じで，ただイベントがスタート / ストップ・ポイントを表すのでなく，キュー・ポイントを表す点が異なるだけである。

0D　デリート・キュー・ポイントは，キュー・リストからタイム及びイベント番号の一致したキュー・イベント（追加情報あり，またはなしの両タイプ）を削除する。

0E　追加情報におけるイベント・ネームは，単にイベント番号に名称を付けることで，整理，記録の便をはかるためのもの。アディショナル・インフォメーション（追加情報）の解説参照。

イベント・タイム

SMPTE/MIDI タイムコードで,イベント発生タイミングを決定する。時間分解能は,ビット処理やその他の方法で,サブ・フレームまでの分解能を有するユニットの場合，1/100 フレームまで可能となる。

イベント番号

14 ビットで，上で説明したイベント・タイプを 16,384 まで，個々にアドレスすることができる。“sl”は下位7ビット，“sm”は上位7ビット。

アディショナル・インフォメーション（追加情報）

追加情報は4ビットの MIDI データで，下位ビットが先行する。ただし，セットアップ・タイプ 0E は例外で，4ビット，下位ビット先行の ASCII である。ASCII ラインを開始する場合は，CRとLFを ASCII コードで送信する。CR単独ではキャリッジ・リターンのみ，またLF単独ではライン・フィードとしての機能しか果たさない。

例えば MIDI ノート・オン・メッセージ “91 46 7F”は，4ビット構成で “01 09 06 04 0F 07”として送信される。この方法によれば，いかなる機器もこの情報をデコードすることが可能である。機器に固有のメッセージは，4ビット構成にした MIDI システム・エクスクルーシブ・メッセージで送信すべきである。

リアルタイム MIDI キューイング・セットアップ・メッセージ（8バイト＋追加情報）

F0 7F <device ID> 05 <sub-ID #2> sl sm <additional info.> F7

F0 7F	リアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー
<device ID>	ターゲットとなる装置のID
05	サブ ID 番号＃ 1 ＝ MIDI タイムコード・キューイング・メッセージ
<sub-ID #2>	セットアップ・タイプ
00	スペシャル
	イベント番号 ＝ 04 00= システム・ストップ（他はすべて予約済み）
01	パンチ・イン・ポイント
02	パンチ・アウト・ポイント
03	（予約済み）
04	（予約済み）
05	イベント・スタート・ポイント
06	イベント・ストップ・ポイント
07	イベント・スタート・ポイント（追加情報あり）
08	イベント・ストップ・ポイント（追加情報あり）
09	（予約済み）
0A	（予約済み）
0B	キュー・ポイント
0C	キュー・ポイント（追加情報あり）
0D	（予約済み）
0E	追加情報中のイベント・ネーム
sl, sm	イベント番号 / スペシャルのタイプ（LSB ファースト）
<add. info.>	前項の規定のように4ビット化
F7	EOX

リアルタイム MTC キューイングは，基本的にはノンリアルタイム MTC キューイング・メッセージをリアルタイム・メッセージの領域に移したものである。

時刻フィールドが抜かされていることに注意されたい。このメッセージにおいては，実行時刻は“このメッセージを受信したらすぐ”である。削除メッセージすべてと，スペシャル・メッセージの多くは除外されている。それらはキュー・リストを外部から編集することを意図したもので，リアルタイム・レスポンスには必要とされない。

それ以外のすべてのメッセージのフォーマットと定義は，ノンリアルタイムにおけるものと同じままで，1対1で対応している。

より詳しい情報は，ノンリアルタイム MTC の定義（前述）を参照のこと。

潜在する諸問題

MTCが定義される以前に作られた MIDI マージャーは，F1 メッセージの後に何バイト続くかがわからないために，正しく取り扱えないという問題の起こる可能性がある。しかし，通常の MIDI タイムコード・システムにおいては，MIDI タイムコードが同時に起こる他の MIDI 信号とマージされなければならないような応用方法は予想されない。

現在は想定していない状況や設定に対応するために，セットアップ・タイプその他を付加する余裕が十分にあることを，心にとめていただきたい。

各 MTC 周辺機器は，電源投入時にその MIDI メーカーのシステム・エクスクルーシブ ID 番号をデフォルトのデバイス ID とすることが推奨される。この番号を，ユーザーがそのデバイスのフロント・パネルから変更できるようになっているのが望ましいのは明らかである。そうすれば，同じメーカーからのいくつかの周辺機器が，同一の MTC システムの中で唯一のIDを持つことができる。

多くの場合，デバイスIDはチャンネル番号のように機能する。それにより，ユニバーサル・システム・エクスクルーシブ・メッセージを個々の機器宛に送ることができる。よって，チャンネル番号とデバイス ID のデフォルト値とするのも，良いことである。しかし，大きなシステムでは，マスター・コントローラーやコンピューターは，同じ MIDI チャンネルの異なる機器に個別にアドレスすることを望むかもしれない，したがって，デバイスIDはそのような問題を解決するために，ユーザーが変更可能になっているべきである。また，デバイスが電源投入時に，前回電源を切られた時の状態に復帰するようになっていてもよい。

MTC 信号経路概説

タイムコードを発生するマスター（例えば，SMPTE シンクロナイザー内蔵のマルチトラック・テープデッキ）と MIDI タイムコード・コンバーター間のデータ転送は，常に SMPTE タイムコードで行われる。

MIDI タイムコード・コンバーターからマスター・コントロール / キュー・シート（キュー・シートならびに MTC を装備したテープデッキまたはミキシング・コンソールを含む）へのデータ転送は，常に MIDI タイムコードで行われる。特定の MIDI タイムコードは，下記の説明のように，カレント動作モードによって異なる。

プレイ・モード

マスター・タイムコード源（テープデッキ）は，通常のプレイ・モードで，スピードは一定または可変。MIDI タイムコード・コンバーターは，クォーター・フレーム・メッセージを “F1 0X” から “F1 7X” まで昇順でマスター・コントロール / キュー・シートへ転送する。テープ・マシンが反転モードでも動作可能なものであれば，クォーター・フレーム・メッセージは “F1 7X” から ”F1 0X” へと逆順で送られる。

キュー・モード

マスター・タイムコード・ソースは，手動でゆさぶられたり，キューされており，かつテープはまだ再生ヘッドに接触しているので，キューやレビューでテープの内容を聴くことができる。MIDI タイムコード・コンバーターは，クォーター・フレーム・メッセージをマスター・コントロール / キュー・シートへ転送する。もしテープがフォワードで再生中であれば，クォーター・フレーム・メッセージは，昇順で送られ，”F1 0X” で始まって ”F1 7X” で終わる。反転モードで再生されている場合，クォーター・フレーム・メッセージは “F1 7X” から ”F1 0X” へと逆順で送られる。

キュー・モードでは，テープは手で動かされているので，テープの方向が急激，頻繁に変わる。クォーター・フレーム・メッセージも，それに応じてシーケンスの方向を切り替えねばならない。

ファースト・フォワード／リワインド・モード

このモードでは,テープは早送りまたは早巻き戻しとなっており,再生ヘッドには接触していない。テープ内容についてのキューは行われない。本モードはサーチ・モードのひとつなので，マスター・コントロール / キュー・シートの同期は，プレイ・モードやキュー・モードの時ほど重要ではない。したがって，MIDI タイムコード・コンバーターは，キュー・シートに対し，フル・メッセージのみを必要に応じて送信すればよい。これは，マスターの位置を大まかに示すインジケーターとなる。フル・メッセージが示した SMPTE タイムは，次にクォーター・フレーム・メッセージを受信した時に有効となる。

シャトル・モード

ファースト・フォワード／リワインド・モードの別名。

参考規格

SMPTE　12 M（ANSI　V98. 12 M -1981）

4. スタンダード MIDI ファイル 1.0

スタンダード MIDI ファイル 1.0
RP001

Copyright ©1998 社団法人　音楽電子事業協会

スタンダード MIDI ファイル 1.0

序文

このドキュメントは,MIDI ファイルの仕様のアウトラインを述べたものである。MIDI ファイルの目的は,同一あるいは異なるコンピューターにおける異なるプログラム間で, 時間情報を持った MIDI データのやり取りの方法を提供することである。「コンパクトな表現」を目標のひとつにしているので, ディスク・ベースのファイル・フォーマットとしては適しているが, シーケンサー・プログラムがメモリーに素早くアクセスするために, 不適切な面もある(MIDI ファイルは, 読み込み中あるいは書き出し中に, 素早いアクセスが可能なフォーマットに, コンバートするのは容易である)。各シーケンサー・プログラムのファイル・フォーマットを, このフォーマットに置き換えることも可能であるが, それを意図したものではない。

MIDI ファイルは, 各イベントに時間情報を伴った, ひとつかそれ以上の MIDI ストリームを含んでいる。ソング, シーケンス, トラック構造, テンポ, 拍子記号情報は, すべてサポートされ, トラック名や他の記述情報を, MIDI データとともにストアすることもできる。マルチトラックとマルチ・シーケンスをサポートしており, マルチトラックで構成されたシーケンサーのデータを, 別のシーケンサーに移動しようとする場合にも, このフォーマットを使うことができる。

この仕様は, MIDI ファイルで使用される8ビットのバイナリィ・データ・ストリームを定義している。データはバイナリィ・ファイルにストアすることはもちろん, 4ビット化や MIDI 送信のための7ビット化, 16 進法の ASCII への変換, あるいは印刷可能なテキスト・ファイルに変換することも可能である。8ビットのストリーム中に何があるかを規定しているのであって, どのように MIDI ファイルを MIDI 経由で送信するかを規定するものではない。MIDI ファイルの送信は,汎用的なファイルを送信するために開発されたプロトコル(ファイル・ダンプ・スタンダード)を使用する。

シーケンス , トラック , チャンク（ファイル・ブロックの構造）

取り決め

このドキュメントでは，ビット0はバイトの最下位ビット，ビット7は最上位ビットを意味している。

MIDI ファイルの中には，可変長表記と呼ばれる形式で表現される数がある。これらの数値表記は，上位ビットから順に1バイトあたり7ビットで表現される。最後のバイトを除くすべてのバイトにはビット7がセットされ，最後のバイトはビット7がクリアされる。したがって，ある数が 0 から 127 の場合はそのまま1バイトで表現される。

以下は，可変長表記として表現される数のいくつかの例である。

数値（16 進）	可変長表現（16 進）
00000000	00
00000040	40
0000007F	7F
00000080	81 00
00002000	C0 00
00003FFF	FF 7F
00004000	81 80 00
00100000	C0 80 00
001FFFFF	FF FF 7F
00200000	81 80 80 00
08000000	C0 80 80 00
0FFFFFFF	FF FF FF 7F

この表記で表現できる最大の数は，数値を可変長化するためのルーチンの中で 32 ビットに収まる，0FFFFFFF である。理論的にはもっと大きい数も可能であるが，500（拍 / 分）という速いテンポにおいて，2 × 10^8 個の 96 分音符は3日間分にあたり，デルタ・タイムとしては十分な長さである。

ファイル

いかなるファイル・システムに対しても MIDI ファイルは，単に8ビットのバイトの連続である。Macintosh においては，このバイト・ストリームは（ファイル・タイプ “Midi” の）ファイルのデータ・フォークに，あるいは（データ・タイプ “Midi” として）クリップボードにストアされる。他のコンピューターでは，ファイル中に8ビット・バイト・ストリームとしてストアされる。コンピューターにとってのネーミングやストレージの取り決めは，それぞれ必要に応じて定義される。

チャンク

MIDI ファイルは，いくつかのチャンクで構成されている。各チャンクは4文字の“タイプ”と 32 ビットの“レングス”を持っている。レングスは，そのチャンク中のバイトの数である。この構造は，将来におけるチャンクの拡張を可能にする。新たなチャンク・タイプが導入される以前に書かれたプログラムは，容易にそのチャンクを無視できる。プログラムは，未知のチャンクがある可能性を予期し，それらは存在しないかのように取り扱うべきである。

各チャンクは，4つの ASCII 文字から始まる。その後に 32 ビットのレングスが最上位バイトから順に続く(6のレングスは 00 00 00 06 とストアされる)。レングスは，その後に続くデータのバイトの数を示す(タイプとレングスの8バイトは含まれない)。したがって，レングス“6”を持ったチャンクは，実際には，ディスク・ファイル中に 14 バイトを占有する。

チャンクの構造は，エレクトロニック・アーツ社の IFF フォーマットで使用されている構造に類似しており，ここに記述されているチャンクは，容易に IFF ファイル中に配置可能である。MIDI ファイル自身は，IFF ファイルではない。ネスティッド・チャンクをまったく含まないし，強制的に偶数バイト長に限定してはいない。IFF ファイルへの変換は，奇数長のチャンクに付け足しをして，すべてをフォーム・チャンクの中に入れ込むことで，容易に行える。

MIDI ファイルは，ヘッダー・チャンクとトラック・チャンクの2つのタイプのチャンクを含む。ヘッダー・チャンクは，MIDI ファイル全体に直接関係する最小限の情報を提供する。そしてトラック・チャンクは，最大 16MIDI チャンネルまでの情報を含むであろう MIDI データの，連続的なストリームである。マルチトラック，マルチ MIDI 出力，パターン，シーケンス，そしてソングの概念は，すべていくつかのトラック・チャンクを使用することでインプリメントできる。

MIDI ファイルは，常にヘッダー・チャンクで始まり，その後にひとつかそれ以上のトラック・チャンクが続く。

```
MThd <length of header data>
<header data>
MTrk <length of track data>
<track data>
MTrk <length of track data>
<track data>
...
```

チャンクの説明

ヘッダー・チャンク

ファイル先頭のヘッダー・チャンクには，ファイル中のデータについてのいくつかの基本的な情報がストアされる。以下に，チャンクの構文を示す。

```
<Header Chunk> = <chunk type> <length> <format> <ntrks> <division>
```

前述の通り，<chunk type> は4つの ASCII キャラクター‘MThd’であり，<length> には6が 32 ビット表現で最上位バイトからストアされる。

<length> に続くデータ・セクションは，上位バイトが最初にストアされた，3つの 16 ビットのワードで構成される。

最初のワード <format> は，そのファイルの全般的な構成を指定する。<format> では次の3つの値を指定できる。

0　　ただ1つのマルチ・チャンネルのトラックを含む

1　　1つのシーケンスの，1つかそれ以上の同時トラック（あるいは MIDI アウトプット）を含む。

2　　1つかそれ以上の進行上独立したシングル・トラック・パターンを含む。

これらのフォーマットについての詳細な情報は，後述する。

次のワード <ntrks> は，そのファイル中のトラック・チャンクの数である。フォーマット0のファイルでは，常に1である。

第3のワード <division> は，デルタ・タイムの意味を指定する。これにはふたつのフォーマットがある。ひとつは記譜上のタイム（拍数をベースにしたタイム）であり，もうひとつは，タイムコードをベースにしたタイムである。

0	4分音符あたりのティック数	
1	SMPTE フォーマットの負数	フレームあたりのティック数

15 14　　8 7　　0

<division> のビット 15 がゼロの場合，14 から 0 までのビットは4分音符の長さになるデルタ・タイム のティック数である。例えば，<division> が 96 の場合，そのファイル中の2つのイベント間の8分音符分の時間間隔は 48 となる。

<division> のビット 15 が1の場合，ファイル中のデルタ・タイムは，SMPTE や MIDI タイムコードと一致する方法で，秒の分数に相当する。14 から 8 までのビットは，4つの標準的な SMPTE と MIDI タイムコードのフォーマットに相応して，－ 24，－ 25，－ 29，あるいは－ 30 の4つの値のうちのひとつ（－ 29 は 30 ドロップ・フレームに相当）を含み，秒ごとのフレーム数を表す。これらのマイナスの値は，2の補数形式でストアされる。第2バイト（プラスでストアされる）は，フレーム内の分解能である。典型的な値は 4（MIDI タイムコード分解能），8，10，80（ビット分解能），あるいは 100 であろう。この方式は，タイムコード・ベースの正確な分解能を可能にするが，25 フレーム / 秒でフレームごとの分解能を 40 に指定することによって，トラックをミリセカンド・ベースにすることも可能にする。ファイル中のイベントが 30 フレーム・タイムコードのビット分解能でストアされている場合，<division> のワードは E250H である。

フォーマット 0，1，2

フォーマット0ファイルは，ヘッダー・チャンクとそれに続くひとつのトラック・チャンクを持つ。これは，最も互換性の高いデータ表現である。シンセサイザーを演奏させる単純なシングル・トラック・プレーヤーにとって有用であるが，基本的にはミキサーやサウンド・エフェクト・ボックスなどのために用意された。たとえトラック・ベースのシーケンサー・プログラムであっても，単純なプログラムでも再生できるよう，フォーマット0を作成できるようにすることが望ましい。

フォーマット1あるいは2のファイルは，ヘッダー・チャンクと，それに続くひとつ以上のトラック・チャンクを持つ。いくつかの同時進行するトラックをサポートするプログラムは，縦方向にトラックが並んだファーマット1の形式で，セーブおよびリードできるべきである。いくつかの独立したパターンをサポートするプログラムでは，トラックを横割にして扱うフォーマット2の形式で，データをセーブおよびリードできるべきである。これらの最小限の対応を行うことで，最大限の互換性が保証される。

ソング・ポジション・ポインターとタイミング・クロックを使用して，コンピューターと SMPTE シンクロナイザーを同期させる MIDI システムでは，テンポ・マップ（トラックの始めから終わりまでのテンポを記述。拍子記号を含む場合もあるので，小節番号が明らかになることもある）は，一般的にコンピューター上に作成される。テンポ・マップをシンクロナイザーと共に使用するには，それをコンピューターから転送する必要がある。シンクロナイザーが，テンポ情報を MIDI ファイルから容易に抽出できるよう，テンポ情報は，常に最初の MTrk チャンクにストアされるべきである。フォーマット0のファイルでは，テンポはトラック中のいたるところにばらまかれており，テンポ・マップ・リーダーは，その間にあるイベントを無視しなければならない。フォーマット1では，テンポ・マップは第1トラックにストアされなければならない。フォーマット1を使用できない場合，テンポ情報のみを有するフォーマット0のファイルを作る機能を持つことは，テンポ・マップ・リーダーに対して親切である。

すべての MIDI ファイルは，テンポと拍子の指定を持つべきである。指定されていない場合，拍子は 4/4 に，テンポは 120BPM（毎分 120 拍）と解釈される。フォーマット0では，これらのメタ・イベントは，唯一のマルチチャンネル・トラックの先頭にあるべきである。フォーマット1において，これらのメタ・イベントは第1トラックに含まれるべきである。フォーマット2は，時間的に独立したパターンであり，それぞれに少なくとも初期設定となる拍子記号とテンポ情報を含むべきである。

他の構造をサポートするため，他のフォーマット ID を定義することになる場合もある。未知のフォーマット ID に遭遇したプログラムは，それでもなおそのプログラムがそのファイルから発見する他の MTrk チャンクを，ユーザーがそれらの意味をくみ取り，他の何らかの構造にアレンジできる場合は，妥当ならば，フォーマット 1 あるいは 2 として読み込んでもかまわない。同様に，将来もっと多くのパラメーターが MTrk チャンクに付け加えられる場合もある。たとえレングスが 6 より長かったとしても，それを読み込んで尊重することは重要である。

トラック・チャンク

トラック・チャンク（タイプ MTrk）は，実際のソング・データがストアされるところである。各トラック・チャンクは，単純にデルタ・タイムの値が先行する MIDI イベント（及び非 MIDI イベント）のストリームである。トラック・チャンクのフォーマット（以下に記述される）は，MIDI ファイルの3つのフォーマット（0，1，2：上述の「ヘッダー・チャンク」を参照）すべてについて，完全に同一である。

以下は，MTrk チャンクの構文である（+ は “ひとつかそれ以上” を意味する。少なくともひとつの MTrk イベントが存在しなければならない）。

```
<Track Chunk> = <chunk type> <length> <MTrk event>+
```

MTrk イベントの構文はたいへん単純である。

```
<MTrk event> = <delta-time> <event>
```

<delta-time> は，可変長表記としてストアされる。これは，次に続くイベントまでの時間間隔を表す。あるトラックの最初のイベントが，トラックの開始と同時に起こる場合，あるいは2つのイベントが同時に起こる場合，デルタ・タイムはゼロである。デルタ・タイムは，(ゼロであっても)常に存在する。デルタ・タイムは，ヘッダー・チャンク中で指定されているように，ビート(拍)(あるいは，SMPTE タイムでのトラックのレコーディングでは秒)を分割した単位である。

```
<event> = <MIDI event> | <sysex event> | <meta-event>
```

<MIDI event> は，何らかの MIDI チャンネル・メッセージである。ランニング・ステータスを使うことができる。すなわち，先行しているイベントが同じステータスの MIDI チャンネル・メッセージである場合，ステータス・バイトを省略してもかまわない。各 MTrk チャンク中の最初のイベントは，ステータスを指定しなければならない。デルタ・タイム自体はイベントではなく，MTrk イベントの構文の一部分である。ランニング・ステータスは，デルタ・タイムを越えて生起することに注意。

<sysex event> は，MIDI システム・エクスクルーシブ・メッセージを複数のパケットで指定する時，あるいは任意のバイトを送信するための“エスケープ”として使用される。通常の完全なシステム・エクスクルーシブ・メッセージは，MIDI ファイルにおいてこのようにストアされる。

```
F0 <length> <F0 の後に送信されるデータ >
```

<length> は，可変長表記としてストアされる。それは，F0 及び <length> 自身を含まない，その後に続くバイト数である。例えば，送信されるメッセージ“F0 43 12 00 07 F7”は，MIDI ファイルにおいては“F0 05 43 12 00 07 F7”とストアされる。MIDI ファイルの読み出し時にメッセージ全体を読んだことがわかるように，末尾に F7 を含む必要がある。

SysEx イベントには，F0 が送信されるべきという条件を伴わないもうひとつのフォームが用意されている。これは，システム・リアルタイム・メッセージ，ソング・ポインターやソング・セレクト，MIDI タイムコードなど，規定されていないものを送信するための“エスケープ”として使用される。これは F7 コードを使用する。

```
F7 <length> <all bytes to be transmitted>
```

シンセサイザー・メーカーによっては，システム・エクスクルーシブ・メッセージを複数の小さなパケットとして送信するように指定していることがある。各パケットは，構文的にはシステム・エクスクルーシブ・メッセージ全体の一部分にすぎないが，それらが送信されるタイミングが重要である。F0 と F7 の SysEx イベントは，構文的に完全なシステム・エクスクルーシブ・メッセージを，タイミングを調整したパケットに分割するために，いっしょに使用される場合もある。

F0　SysEx イベントは，シリーズ中の最初のパケットで使用され，それは F0 メッセージが送信されるべきメッセージである。F7 SysEx イベントは最初に F0 を送らない，残りのパケットで使用される(もちろん，F7 はシステム・エクスクルーシブ・メッセージの一部分とはみなされない)。

システム・エクスクルーシブ・メッセージは，現実の装置がそれを送らなかった場合でも，構文上，常に F7 で終了しなければならない。そうすれば，MIDI ファイル中の次のイベントを待ち受けることなしに，SysEx メッセージ全体の末尾に到達したのがわかる。ひとつの F0 SysEx イベント中で完結している場合，最後のバイトは F7 でなければならない。パケットに分割されている場合は，最後のパケットの最後のバイトは F7 でなければならない。また，ひとつのマルチパケット・システム・エクスクルーシブ・メッセージのパケット間には，いかなる送信可能な MIDI イベントもあってはならない。この原則を次に例示する。

以下は，マルチパケット・システム・エクスクルーシブ・メッセージの例である。バイト“F0 43 12 00”が送られ，次に 200 ティックのディレイがあり，次にバイト“43 12 00 43 12 00”が続き，次に 100 ティックのディレイが続き，次にバイト“43 12 00 F7”が続く場合，MIDI ファイル中では以下のようになる。

F0 03 43 12 00
81 48 **200tick デルタ・タイム**
F7 06 43 12 00 43 12 00
64 **100tick デルタ・タイム**
F7 04 43 12 00 F7

MIDI ファイルを読む時に，マルチパケット・システム・エクスクルーシブ・メッセージ・シーケンスを構成する，先行する F0 システム・エクスクルーシブ・イベントなしに F7 システム・エクスクルーシブ・イベントが現れた場合，F7 イベントが“エスケープ”として使用されていることが推定される。この場合，F7 の送信が必要とされない限り，F7 で終わる必要はない。

<meta-event> は，次の構文で，このフォーマットあるいはシーケンサーに対して有効な非 MIDI 情報を指定する。

FF <type> <length> < データ >

すべてのメタ・イベントは，FF で始まり，イベント・タイプのバイト（常に 128 未満）を持ち，可変長表記としてストアされたデータのレングスとデータそのものを持つ。データがまったくない場合，レングスは0である。チャンクの場合と同様，現存するプログラムにとっては未知のメタ・イベントが将来登場する場合もあるだろう。したがってプログラムは，未知のメタ・イベントを適切に無視しなければならないし，それらに出会うことを当然のこととして予期すべきである。プログラムは，認識したメタ・イベントのレングスを決して無視してはならないし，それが予期していたより大きくても，正常に動作しなければならない。その場合プログラムは，存知の部分以降はすべて無視すべきである。ただしプログラムは，メタ・イベントの末尾に，それ固有の情報を決して付加してはならない。

SysEx イベントとメタ・イベントは，その時点で有効になっているすべてのランニング・ステータスを無効にする。ランニング・ステータスは，これらのメッセージには適用されないので，使用してはいけない。

メタ・イベント

ここには，いくつかのメタ・イベントが定義されている。すべてのプログラムがすべてのメタ・イベントをサポートする必要はない。

以下のメタ・イベントそれぞれに対する構文の解説では，イベントのパラメーターを記述するのに，次の規約を使用する。各イベントを始めるFF，各イベントのタイプ，そして可変長でないイベントではレングスは直接16進で与えられている。ddやseといった小文字2文字のニーモニック的な表記は，8ビットの値を表す。wwwwといった4つの同一の小文字は，上位バイトが先にストアされた16ビットの値，tttttといった6つの同一の小文字は，最上位バイトからストアされた24ビットの値を表す。lenという表記は，メタ・イベント構文のレングス部分，すなわち，可変長表記としてストアされた数値で，そのメタ・イベント中でいくつのデータ・バイトがその後に続くのかを指定する。text及びdataという表記は，lengthで指定されたバイト数の（テキスト）データを示す。

一般的に，トラック中の同時に発生するメタ・イベントは，いかなる順序でもかまわない。コピーライト・イベントが使用される場合，目にとまりやすいように，ファイル中で可能な限り始めの方に置かれるべきである。シーケンス番号とシーケンス／トラック・ネーム・イベントが存在する場合は，タイム0になければならない。エンド・オブ・トラック・イベントは，トラック中の最後のイベントとして置かれていなければならない。

定義済みのメタ・イベントとしては，以下のものがある。

FF 00 02 ssss　　シーケンス番号

このイベントは，トラック先頭のあらゆるゼロでないデルタ・タイム及びあらゆる送信可能なMIDIイベントの前に置かれ，シーケンスの番号を指定する。フォーマット2のMIDIファイルでは，“ソング”シーケンスがキュー・メッセージを使用して，パターンを参照するための“パターン”識別番号をとして利用する。IDナンバーが省略される場合，ファイル中のシーケンスの順序がデフォルトとして使用される。フォーマット0あるいは1のMIDIファイルでは，ひとつのシーケンス番号のみを含み，この数値は最初の（あるいは唯一の）トラックに含まれるべきである。いくつかのマルチトラック・シーケンスの転送が必要な場合，それぞれが異なるシーケンス番号を持った，複数フォーマット1のファイルを用いなければならない。

FF 01 len text　　テキスト・イベント

内容もテキスト長も任意であるテキストのイベントである。意図されたオーケストレーションなど，ユーザーがそこに置きたいと思う情報を，トラック･ネームと共にトラックの先頭の方に置くのは良い考えである。またテキスト･イベントは，トラック中のあるタイミングに，歌詞，あるいはキュー・ポイントの記述として使用される場合もある。このイベント中のテキストは，最大限の互換性のために，プリント可能なASCIIキャラクターを使うべきである。しかしながら，拡張キャラクター・セットをサポートする，同じコンピューターの異なるプログラム間でのファイルのやり取りに，高位ビットを使用するキャラクター・コードが使用される場合もある。非ASCIIキャラクターをサポートしないコンピューター上のプログラムは，これらのキャラクターを無視すべきである。

メタ・イベント・タイプ01から0Fまでは，さまざまなタイプのテキスト・イベントのために予約されており，それぞれはテキスト・イベントの仕様（上述）に合致するが，その使用目的は異なる。

FF 02 len text　　コピーライト表示

プリント可能なASCIIテキストとしてのコピーライト表示である。その告知は(C)というキャラクター,コピーライトの取得年,そしてコピーライト所有者を含むべきである。同じMIDIファイル中にいくつかの曲がある場合，すべてのコピーライト表示をファイルの先頭にまとめて置くべきである。このイベントはタイム0における，第1トラック・チャンク中の最初のイベントであるべきである。

FF 03 len text　　**シーケンス／トラック・ネーム**

フォーマット0のトラック中，あるいはフォーマット1のファイルの第1トラック中の場合，シーケンスのネームを表す。それ以外の場合は，トラックのネームを表す。

FF 04 len text　　**インストゥルメント・ネーム**

そのトラックで使用される楽器のタイプを記述する。その記述がどの MIDI チャンネルに当てはまるかを指定するために，MIDI プリフィックス・メタ・イベントと共に使用される場合もある。あるいは，チャンネルをテキスト中に記述してもよい。

FF 05 len text　　**歌詞**

歌われる歌詞を記述する。一般的に，各音節はそのイベントのタイミングで始まる個別のリリック・イベントである。

FF 06 len text　　**マーカー**

通常フォーマット0のトラック，あるいはフォーマット1ファイルの第1トラック中に置く。リハーサル・マークやセクション・ネーム（“First Verse”など）といったシーケンス中のそのポイントのネームである。

FF 07 len text　　**キュー・ポイント**

楽曲スコア中のそのポイントにおいて，フィルムやビデオ・スクリーンやステージで起こりつつある何かの記述（“Car crashes into house”，“curtain opens”，“she slaps his face”など）。

FF 20 01 cc　　**MIDI チャンネル・プリフィックス**

このイベント中に含まれる MIDI チャンネル（0 ～ 15）は，システム・エクスクルーシブとメタ・イベントを含むその後に続くすべてのイベントに，ある MIDI チャンネルを結びつけるために使用することができる。このチャンネルは，次に続く通常の MIDI イベント（チャンネルを含む），あるいは次の MIDI チャンネル・プリフィックス・メタ・イベントまで有効である。MIDI チャンネルがトラックと対応している場合，このメッセージによって，複数のトラックをフォーマット 0 のファイルに詰め込んだ時に，非 MIDI データとトラックの関係を保持することができる。

FF 2F 00　　**エンド・オブ・トラック**

このイベントの使用は必須である。トラックのループや連結に必要なトラックの，正確なエンディング・ポイント及び正確なレングスを指定するために含まれている。

FF 51 03 tttttt　　**セット・テンポ（MIDI 4分音符あたりのマイクロ秒）**

このイベントは，テンポ・チェンジを示す。“4分音符あたりのマイクロセカンド”は，言い換えれば “MIDI クロックあたりのマイクロ秒の 24 倍”である。拍／時の代わりに時／拍としてテンポを表現することで，SMPTE タイムコードや MIDI タイムコードのような，タイムベース同期プロトコルとの完全に正確な長期間の同期を可能にする。このテンポ分解能によって表される値の精度は，120（拍／分）の4分の曲が，曲の末尾で 500 μ s 以内の正確さであることを可能にする。理想的には，これらのイベントは MIDI クロックがロケートできる場所にのみ置かれるべきである。そうすることにより，他の同期装置との互換性を保証，あるいは少なくともその可能性を増加させることができ，このフォーマットにストアされた拍子記号／テンポ・マップを，容易に他の装置に転換できるようになる。

FF 54 05 hr mn se fr ff **SMPTE オフセット**

このイベントが存在する場合，トラック・チャンクがスタートすべき SMPTE タイムを指定する。それはトラックの先頭，すなわちゼロでないデルタ・タイム，そしてあらゆる送信可能な MIDI イベントの前に存在すべきである。MIDI タイムコードの場合とまったく同様に，hr には SMPTE フォーマットを含めてエンコードしなければならない。フォーマット1のファイルにおいて，SMPTE オフセットはテンポ・マップと共にストアされなければならず，他のトラックでは，どこに置いても何の意味も持たない。ff フィールドの単位は，たとえデルタ・タイムに異なるフレーム・ディビジョンを指定している SMPTE ベースのトラックであったとしても，フレームの 1/100 である。

FF 58 04 nn dd cc bb　　**拍子記号**

拍子記号は，4つの数値として表される。nn と dd は拍子記号の分子と分母を表わす。分母は，2のマイナスの累乗である。例えば2は4分音符を表現し，3は8分音符を表現する。cc パラメーターは，ひとつのメトロノーム・クリック中の MIDI クロックの数を表す。bb パラメーターは,ひとつの MIDI 4分音符（24MIDI クロック）中で表記される 32 分音符の数を表す。これは,MIDI での4分音符（24 クロック）を他の音符に換算したり，記譜するように指定できるプログラムが，すでに多数存在するために付け加えられた。

したがって，メトロノームが8分音符3つごとにクリックし，MIDI クロックが4分音符ごとに 24 クロック，1小節が 72 クロックである完全な 6/8 拍子のイベントは，以下のようになる。

FF 58 04 06 03 24 08

これは，6/8 拍子（8は2の3乗なので 06 03），付点4分音符につき 36MIDI クロック（24H），そして MIDI 4分音符ごとに 32 分音符8つ表記されることを示す。

FF 59 02 sf mi　　**調号**

sf = -7 : 7 フラット
sf = -1 : 1 フラット
sf = 0 : C調
sf = 1 : 1 シャープ
sf = 7 : 7 シャープ
mi = 0 : 長調
mi = 1 : 短調

FF 7F len data　　**シーケンサー固有のメタ・イベント**

特定のシーケンサーに固有の機能のために，このイベントを使用することができる。最初のデータは，メーカーの ID（これらは1バイトあるいは，最初のバイトが 00 の場合は3バイト）である。MIDI システム・エクスクルーシブのように，このメタ・イベントの使用を定義するメーカーは,他のメーカーにその使用方法がわかるように公表すべきである。このフォーマットは互換性を与えるためのものなので，このイベントを使用するのは，これをその唯一のファイル・フォーマットとして使用するシーケンサーであろう。すべての機能を含む固有のフォーマットを持つシーケンサーは，このフォーマットを使用する場合，標準的な使い方から逸脱しないようにすべきであろう。

プログラム断片と MIDI ファイルの例

以下は MIDI ファイルにおいて可変長表記の数値を read したり write したりするいくつかのルーチンである。これらは，C言語において getc と putc を使用して記述されている。8ビットのキャラクターを infile/outfile によってファイルから読み出し，ファイルへ書き込む。

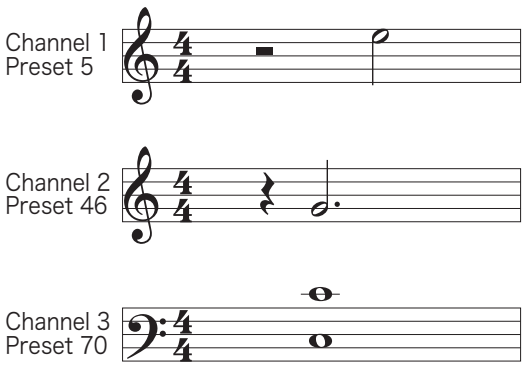
```
WriteVarLen (value)
        register long value;
{
        register long buffer;

        buffer = value & 0x7f;
        while ((value >>= 7) > 0)
        {
                buffer <<= 8;
                buffer |= 0x80;
                buffer += (value & 0x7f);
        }
        while (TRUE)
        {
                putc(buffer,outfile);
                if (buffer & 0x80)  buffer >>= 8;
                else                break;
        }
}

unsigned long ReadVarLen ()
{
        register unsigned long value;
        register unsigned char c;

        if ((value = getc(infile)) & 0x80)
        {
                value &= 0x7f;
                do
                {
                        value = (value <<7)+((c=getc(infile))&0x7f);
                } while (c & 0x80);
        }
        return (value);
}
```

例として，次の楽譜の抜粋に対する MIDI ファイルを以下に示す。最初に，フォーマット0のファイルで，すべてのインフォメーションを混合して示す。次に，フォーマット1のファイルで，すべてのデータを4つのトラックに分けて示す。ひとつは，テンポと拍子記号用であり，3つは音符用である。4分音符につき 96 ティックの分解能が使用される。4/4 の拍子記号と 120 のテンポは，省略時のデフォルトであるが，明示的に指定される。



この例に表現される MIDI ストリームの内容は，以下に分析される通りである。

デルタ・タイム（10 進）	イベント・コード（16 進）	その他のデータ（10 進）	コメント
0	FF 58	04 04 02 24 08	4 バイト（4/4 拍子，24MIDI クロック / クリック ８ 32 分音符 /24 MIDI クロック）
0	FF 51	03 500000	3 バイト，4分音符につき 500,000 μ 秒
0	C0	5	Ch.1，プログラム・チェンジ 5
0	C1	46	Ch.2，プログラム・チェンジ 46
0	C2	70	Ch.3，プログラム・チェンジ 70
0	92	48 96	Ch.3 Note On #48，フォルテ
0	92	60 96	Ch.3 Note On #60，フォルテ
96	91	67 64	Ch.2 Note On #67，メゾフォルテ
96	90	76 32	Ch.1 Note On #76，ピアノ
192	82	48 64	Ch.3 Note Off #48,standard
0	82	60 64	Ch.3 Note Off #60,standard
0	81	67 64	Ch.2 Note Off #67,standard
0	80	76 64	Ch.1 Note Off #76,standard
0	FF 2F	00	エンド・オブ・トラック

16 進におけるフォーマット0の MIDI ファイルの内容全体は，以下の通りである。

最初に，ヘッダー・チャンク。

4D 54 68 64	MThd
00 00 00 06	チャンク・レングス
00 00	フォーマット0
00 01	トラック数＝１
00 60	4分音符につき96

次に，トラック・チャンク。そのヘッダーと，後に続くイベント（ところどころにランニング・ステータスが使用されていることに注意）。

	4D 54 72 6B	**MTrk**
	00 00 00 3B	**チャンク・レングス（59）**
デルタ・タイム	**イベント**	**コメント**
00	FF 58 04 04 02 18 08	**拍子記号**
00	FF 51 03 07 A1 20	**テンポ**
00	C0 05	
00	C1 2E	
00	C2 46	
00	92 30 60	
00	3C 60	**ランニング・ステータス**
60	91 43 40	
60	90 4C 20	
81 40	82 30 40	**2バイトのデルタ・タイム**
00	3C 40	**ランニング・ステータス**
00	81 43 40	
00	80 4C 40	
00	FF 2F 00	**エンド・オブ・トラック**

このファイルの，フォーマット1による表現は，若干異なっている。

最初に，ヘッダー・チャンク。

	4D 54 68 64	**MThd**
	00 00 00 06	**チャンク・レングス**
	00 01	**フォーマット1**
	00 04	**トラック数＝ 4**
	00 60	**4分音符につき96**

次に，拍子記号 / テンポ・トラックのトラック・チャンク。そのヘッダー，後に続くイベント。

	4D 54 72 6B	**MTrk**
	00 00 00 14	**チャンク・レングス（20）**
デルタ・タイム	**イベント**	**コメント**
00	FF 58 04 04 02 18 08	**拍子記号**
00	FF 51 03 07 A1 20	**テンポ**
83 00	FF 2F 00	**エンド・オブ・トラック**

次に，最初の音楽トラックのトラック・チャンク。note on/off ランニング・ステータスのための MIDI の取り決めが，この例で使用されている。

	4D 54 72 6B	**MTrk**
	00 00 00 10	**チャンク・レングス（16）**
デルタ・タイム	**イベント**	**コメント**
00	C0 05	
81 40	90 4C 20	
81 40	4C 00	**ランニング・ステータス（note on, vel＝ 0）**
00	FF 2F 00	**エンド・オブ・トラック**

次に，第2の音楽トラックのトラック・チャンク。

4D 54 72 6B	**MT rk**
00 00 00 0F	**チャンク・レングス（15）**

デルタ・タイム	イベント	コメント
00	C1 2E	
60	91 43 40	
82 20	43 00	**ランニング・ステータス**
00	FF 2F 00	**エンド・オブ・トラック**

次に，第3の音楽トラックのトラック・チャンク。

4D 54 72 6B	**MTrk**
00 00 00 15	**チャンク・レングス（21）**

デルタ・タイム	イベント	コメント
00	C2 46	
00	92 30 60	
00	3C 60	**ランニング・ステータス**
83 00	30 00	**2バイトのデルタ・タイム，ランニング・ステータス**
00	3C 00	**ランニング・ステータス**
00	FF 2F 00	**エンド・オブ・トラック**

5. GMシステム・レベル1

GM システム・レベル1
RP003

Copyright ©1998 社団法人　音楽電子事業協会

この規定は，所定の用途のための MIDI 音源を定義する“General　MIDI（以後GMと称す）システム”に要求される MIDI 構成の規定概要を述べるものである。GMシステムは，MIDI シンセサイザー間における高いレベルの互換性を提供し，他のGM音源用に作成されたソング（MIDI データ形式）を再生することを可能にする。

GM製品は，高い互換性と使いやすさで，音楽，民生機器，エンターテイメントのマーケットにおいて，幅広く応用されることを意図している。

背景

ユーザーが MIDI シンセサイザーで MIDI データを再生しようとした場合，GM がなければ，どんなシンセサイザーを使うか，その性能がどんなものかによってかなり違った結果となる。MIDI データは，最初に意図した通り正確に聴かせるためには，特定のシンセサイザーやドラム・マシンのために，特別に用意されなければならなかった。

例えば，チャンネル1のプログラム1で送信される MIDI ノート・メッセージによって演奏されるサウンドは，個々のシンセサイザー・メーカーによって決定される。しかしながらGM以外のシンセサイザー間においては，プログラム・ナンバーと期待される音色の間には，たいていの場合ほとんど類似性はない。他にもピッチ・ベンド・レンジの可変範囲や，オクターブ設定，あるいはドラム・ノートのマッピング等も同様である。

プロフェッショナル・ユーザーにとって，この多様性は素晴らしいことであるが，消費者や音楽制作者にとってはやっかいなものになりうる。したがって,多くの MIDI シンセサイザーで演奏できる MIDI データを作り出すことは,過去においては事実上不可能であった。データは，メーカーや機種を特定して作られなければならなかった。このことは，MIDI データの使い方を特定の楽器に，あるいは良くてもせいぜい特定メーカーの楽器に限定してきたのである。

このような問題の解決を阻んでいたものは,あるシンセサイザーに組み込まれていることをあてにできるような“最小限の MIDI 構成”，すなわち一連の MIDI 機能に関して，オリジナルの MIDI 規格には規定されていないことである。ある特定の MIDI 機器は，その MIDI ケーブルの反対側にどのような MIDI 機器が接続されているかをまったく関知しないし，現在に至るまで，メーカーや音楽制作者がリファレンスとして使うことのできる，業界標準の最小構成はなかったのである。

解決策

GMシステムは，この問題に対する解決策である。最小のボイス数，サウンド配置，ドラム・ノート・マッピング，オクターブ設定，ピッチ・ベンド・レンジ，そしてコントローラーの用法を記述し，それによって，所定のシンセサイザー・モジュールにおいて期待される一連機能を定義している。このモードは，CD規格をサポートしているすべての装置に見られる“Compact　Disk”のロゴのように，楽器に付けられたロゴによって識別される。

GMは，共通の“ベーシックな状態”を提供するために，シンセサイザーが入ったり出たりすることのできるモードである。ハイエンドの製品は，おそらくGM以外のモードをサポートするであろうし，また，GMによって制限を受けるべきではない。GMは，さらに進んだ応用や継続的改善のために，さらなる拡張（上位レベル）に向かっても開かれている。

GM音源の必要条件

音源方式：

・メーカーによる

ボイス数：

最低限

1）メロディーとリズム・サウンドの両方に，完全にダイナミックにアロケートされる 24 ボイスが同時に利用可能。

2）あるいは，メロディー用にダイナミックにアロケートされる 16 ボイス+リズム音用に8ボイス。

サポートされる MIDI チャンネル：

・16 の MIDI チャンネルすべて

・各チャンネルで演奏可能なボイス（ポリフォニック）数は可変

・各チャンネルは異なる楽器音（音色）を演奏可能

・鍵盤に配列されたリズム音は，常にチャンネル 10

楽器：

・楽器音用に“GMサウンド・セット”（表2を参照のこと）に従う最小 128 のプリセット（MIDI プログラム・ナンバー）

・“GMパーカッション・マップ”（表3を参照のこと）にしたがう最小 47 のプリセット・リズム・サウンド（音）。

GM音源推奨ハードウェア

・マスター・ボリューム・コントロール

・MIDI IN 端子（OUT と THRU 端子はオプション）

・オーディオ出力（ステレオ）+ヘッドフォン端子

GMプロトコル・インプリメンテーション必要条件

ノート・オン／ノート・オフ

・オクターブ設定：中央C＝ MIDI キー（ノート番号）60(3CH)

・リズム音を含むすべてのボイスは，ベロシティに対応

・ボイスはダイナミックにアロケートされる

コントロール・チェンジ

コントロール番号	機能
1	モジュレーション
7	ボリューム
10	パン
11	エクスプレッション
64	サスティン
121	リセット・オール・コントローラー
123	オール・ノート・オフ

レジスタード・パラメーター番号（RPN）	機能
0	ピッチ・ベンド・センシティビティ
1	ファイン・チューニング
2	コース・チューニング

チャンネル・メッセージ

・チャンネル・プレッシャー（アフタータッチ）

・ピッチ・ベンド（デフォルト・レンジ＝±2半音）

デフォルト・セッティング

・ベンド＝“0”，ボリューム（0-127）＝“100”，コントローラーは通常状態

GMシステム・メッセージ

前述の，すでに定義された MIDI メッセージに加えて，音源で（それが2つ以上のオペレーション・モードを持つ場合を想定して），GMをオン / オフするためのノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・メッセージが定義されている。

GMシステム・オン：F0 7E <device ID> 09 01 F7

F0 7E	**ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー**
<device ID>	**対象とするデバイスのID（推奨7F ： ブロードキャスト）**
09	**サブ ID 番号＃ 1 ＝GMメッセージ**
01	**サブ ID 番号＃ 2 ＝GMオン**
F7	**EOX**

GMシステム・オフ：F0 7E <device ID> 09 02 F7

F0 7E	**ノンリアルタイム・ユニバーサル・システム・エクスクルーシブ・ヘッダー**
<device ID>	**対象とするデバイスの ID（推奨 7F: ブロードキャスト）**
09	**サブ ID 番号＃ 1 ＝GM メッセージ**
02	**サブ ID 番号＃ 2 ＝GM オフ**
F7	EOX

表1　GMサウンド・セット・グルーピング

（チャンネル 10 を除く全 MIDI チャンネル）

プログラム番号	楽器グループ	プログラム番号	楽器グループ
1 - 8	ピアノ	65 - 72	リード
9 - 16	クロマチック・パーカッション（音階付き打楽器）	73 - 80	パイプ
17 - 24	オルガン	81 - 88	シンセ・リード
25 - 32	ギター	89 - 96	シンセ・パッド
33 - 40	ベース	97 -104	シンセ・エフェクト
41 - 48	ストリングス	105 -112	エスニック
49 - 56	アンサンブル	113 -120	パーカッシブ
57 - 64	ブラス	121 -128	サウンド・エフェクト

表２　GMサウンド・セット

（MIDI プログラム番号1～ 128；チャンネル 10 を除く全 MIDI チャンネル）

プログラム番号	楽器	プログラム番号	楽器	プログラム番号	楽器	プログラム番号	楽器
1.	アコースティック・グランド・ピアノ	33.	アコースティック・ベース	65.	ソプラノ・サックス	97.	FX1（雨）
2.	ブライト・アコースティック・ピアノ	34.	エレクトリック・ベース（フィンガー）	66.	アルト・サックス	98.	FX2（サウンドトラック）
3.	エレクトリック・グランド・ピアノ	35.	エレクトリック・ベース（ピック）	67.	テナー・サックス	99.	FX3（クリスタル）
4.	ホンキー・トンク・ピアノ	36.	フレットレス・ベース	68.	バリトン・サックス	100.	FX4（アトモスフィア）
5.	エレクトリック・ピアノ1	37.	スラップ・ベース1	69.	オーボエ	101.	FX5（ブライトネス）
6.	エレクトリック・ピアノ2	38.	スラップ・ベース2	70.	イングリッシュ・ホルン	102.	FX6（ゴブリン）
7.	ハープシコード	39.	シンセ・ベース1	71.	バスーン	103.	FX7（エコー）
8.	クラビ	40.	シンセ・ベース2	72.	クラリネット	104.	FX8（SF）
9.	チェレスタ	41.	バイオリン	73.	ピッコロ	105.	シタール
10.	グロッケン	42.	ビオラ	74.	フルート	106.	バンジョー
11.	ミュージック・ボックス（オルゴール）	43.	チェロ	75.	リコーダー	107.	三味線
12.	ビブラフォン	44.	コントラバス	76.	パン・フルート	108.	琴
13.	マリンバ	45.	トレモロ・ストリングス	77.	ボトル・ブロウ	109.	カリンバ
14.	シロフォン	56.	ピチカート・ストリングス	78.	尺八	110.	バグ・パイプ
15.	チューブラ・ベル	57.	オーケストラ・ハープ	79.	ホイッスル（口笛）	111.	フィドル
16.	ダルシマー	58.	ティンパニー	80.	オカリナ	112.	シャナイ
17.	ドローバー・オルガン	59.	ストリング・アンサンブル1	81.	リード1（矩形波）	113.	ティンカ・ベル
18.	パーカッシブ・オルガン	50.	ストリング・アンサンブル2	82.	リード2（鋸歯状波）	114.	アゴゴ
19.	ロック・オルガン	51.	シンセ・ストリングス1	83.	リード3（calliope 蒸気オルガン）	115.	スティール・ドラム
20.	チャーチ・オルガン	52.	シンセ・ストリングス2	84.	リード4（chiff）	116.	ウッド・ブロック
21.	リード・オルガン	53.	ボイス（アー）	85.	リード5（charang）	117.	太鼓
22.	アコーディオン	54.	ボイス（ウー）	86.	リード6（ボイス）	118.	メロディック・タム
23.	ハーモニカ	55.	シンセ・ボイス	87.	リード7（5度）	119.	シンセ・ドラム
24.	タンゴ・アコーディオン	56.	オーケストラ・ヒット	88.	リード8（ベース＋リード）	120.	リバース・シンバル
25.	アコースティック・ギター（ナイロン）	57.	トランペット	89.	パッド1（ニュー・エイジ）	121.	ギター・フレット・ノイズ
26.	アコースティック・ギター（スティール）	58.	トロンボーン	90.	パッド2（ウォーム）	122.	ブレス・ノイズ
27.	エレクトリック・ギター（ジャズ）	59.	チューバ	91.	パッド3（ポリシンセ）	123.	海辺
28.	エレクトリック・ギター（クリーン）	60.	ミュート・トランペット	92.	パッド4（クワイア）	124.	鳥のさえずり
29.	エレクトリック・ギター（ミュート）	61.	フレンチ・ホルン	93.	パッド5（bowed）	125.	電話のベル
30.	オーバードライブ・ギター	62.	ブラス・セクション	94.	パッド6（メタリック）	26.	ヘリコプター
31.	ディストーション・ギター	63.	シンセ・ブラス1	95.	パッド7（halo）	127.	拍手喝采
32.	ギター・ハーモニクス	64.	シンセ・ブラス2	96.	パッド8（スウィープ）	128.	ガン・ショット

表３　GMパーカッション・マップ

（チャンネル 10）

MIDI キー	ドラム・サウンド	MIDI キー	ドラム・サウンド	MIDI キー	ドラム・サウンド
35	アコースティック・バス・ドラム	51	ライド・シンバル1	67	ハイ・アゴゴ
36	バス・ドラム1	52	チャイニーズ・シンバル	68	ロー・アゴゴ
37	サイド・スティック	53	ライド・ベル	69	カバサ
38	アコースティック・スネア	54	タンバリン	70	マラカス
39	ハンド・クラップ	55	スプラッシュ・シンバル	71	ショート・ホイッスル
40	エレクトリック・スネア	56	カウベル	72	ロング・ホイッスル
41	ロー・フロア・タム	57	クラッシュ・シンバル2	73	ショート・ギロ
42	クローズド・ハイハット	58	ビブラスラップ	74	ロング・ギロ
43	ハイ・フロア・タム	59	ライド・シンバル2	75	クラベス
44	ペダル・ハイハット	60	ハイ・ボンゴ	76	ハイ・ウッド・ブロック
45	ロー・タム	61	ロー・ボンゴ	77	ロー・ウッド・ブロック
46	オープン・ハイハット	62	ミュート・ハイ・コンガ	78	ミュート・クイーカ
47	ロー・ミッド・タム	63	オープン・ハイ・コンガ	79	オープン・クイーカ
48	ハイ・ミッド・タム	64	ロー・コンガ	80	ミュート・トライアングル
49	クラッシュ・シンバル1	65	ハイ・ティンバレス	81	オープン・トライアングル
50	ハイ・タム	66	ロー・ティンバレス		

GMサウンド・セット

音楽制作者にとって，オリジナルの MIDI 規格で最も欲求不満を感じさせた部分のひとつは，サウンドの定義が欠けている点だった。例えば，「この楽器ではピアノ・サウンドはどこだ（プログラム・ナンバーは何番だ）？」というように。その解決策はGMシステムに規定された「サウンド・セットに対するプログラム・チェンジ・ナンバー」にある。

このマッピングは，GMシステム内で動作している間に効力を発することのみを要求されており，GMモード以外の場合は，メーカーがどのようにサウンドを編成しようとも自由である。要するに，GMシステム内で動作している間のみ，このマップは効力を発し，他のモードにおいては，メーカーは自由に音色を配置できる。

GMサウンド・セット（楽器音とリズム音のマップ）は，表2と3に示されている。このマッピングは，GMシステムのもとでサウンド選択用の MIDI プログラム・チェンジ・ナンバーを記述している。楽器は，これらのGMプログラム・ナンバーを，それ自身の内部配列に割当てなおしてもよい。MIDI プログラム・ナンバーは，演奏中にリアルタイムで変更できる。

GM音色の定義

GMは音源方式を規定しない。各メーカーは，各プリセットに的確な音色を選ぶにあたって，それ自身の考えや個人的な美学を自由に表現すべきである。また，シンセ・リード，シンセ・パッド，シンセ・エフェクトそれぞれの後にある括弧書きの名前は，指針として書かれたものである。

したがって，たくさんのGMサウンド・モジュールに対してソング再生の一貫性を促進するために，GMの曲データ制作者と楽器メーカーのためのガイドライン一式が示されることになろう。

GMパフォーマンス・ノート

すべての楽器において，モジュレーション・ホイール（コントロール番号1）は，サウンドの特性を最も自然な（期待される）方法，すなわち LFO の深さ，音色の変更，より鋭いサウンドの追加などで変化させるだろう。

現在のところ，MMA と AMEI の間でペンディングになっている MIDI メッセージが他にもあるが，それはGMレベル2の一部となるであろう。

適用のルール

AMEIとMMAは，製品がこの規格にしたがうことを示すために，以下のロゴ・デザインを承認している。

音源にとってGMは，GM用に作られたいかなるスコアをもユーザーが手を加えることなしに再生させようとするものである。このことは，GM音源が各セクションに記述されたすべての機能を，ユーザーによる操作を必要とせずに，サポートしなければならないということを意味している。これらの必要条件を満たす製品のみがGMロゴを持つべきである。

シーケンサーやノーテーション・プログラム，ゲームあるいはMIDIミュージックを作ったり，再生したりする他のアプリケーションのようなソフトウェアにも，その製品がGM互換の音源と共に使われる場合に必要なGMデータのパフォーマンスを妨げない限り，GMロゴを表示してよい。例えば，再生する際にユーザーが異なるサウンドを選ぶこともできるソフトウェアは，GMサウンドのリストを備えるべきである。加えて，GM互換のいかなるソフトウェアも，MIDIファイルの中に含まれる，あるいはMIDI経由で演奏されたすべてのコントローラーの設定や，その他の必要なメッセージを修正なしで適切に再生しなければならない。

GMロゴ・バリエーション

ロゴは，申請して認可契約することによってAMEI／MMAから入手可能である。契約へ，製品や梱包そしてマーケティング素材にGMロゴを適用するための条件，及び制限を規定する。詳細は現在のライセンス契約を参照されたい。

GMシステム・ロゴ

このバージョンのロゴは，GMシステム・レベル1規格に従う音源，アプリケーション・ソフトウェア（ゲーム，シーケンサーなど），そしてスコア（MIDIデータ）に適用される。

GMサウンド・セット

このバージョンのロゴは，ある特定の音源を，GMシステム・レベル1互換にするためのサウンド・セット（サンプルやパッチ）に表示するためのものである。

6. MIDI ショー・コントロール 1.1

MIDI ショー・コントロール 1.1
RP002/0014

Copyright ©1998 社団法人　音楽電子事業協会

MIDI ショー・コントロール（MSC）1.1

1. 序文

MIDI ショー・コントロール（MSC）の目的は，劇場やライヴ・パフォーマンス，マルチメディアやオーディオ・ビジュアルといったものに類する，専用のインテリジェントなコントロール機器と通信し，MIDI によってそれらを制御することである。

その応用範囲は，ひとつの照明コントローラーに GO や STOP，RESUME を指示するといった，単純なインターフェースから，あらゆる種類のパフォーマンス・テクノロジーを駆使した数多くのコントローラーを利用する，大規模な同期システムとの複雑な通信にまでおよぶだろう。

コマンド群は，現在存在しているコンピューター・メモリーの照明コントロールやサウンド・コントロール，そしてショー・コントロール・システムのコマンド構成をモデルにしている。これは，MSCの仕様と専用のコントローラー間のコマンド変換を，同一の操作原則に基づくことによって，比較的容易なものにするためである。一方で，この変換はテーブルの参照以上のものを伴い，データの仕様やコミニュケーションの細部では，少なからぬバリエーションが見られる。MSCは，同一のあるいは類似のオペレーションを持つ装置と，容易に通信を行うことを意図したものである。

2. 全体構成

2.1 ユニバーサル・システム・エクスクルーシブ・フォーマット

MSCは，単一のリアルタイム・ユニバーサル・システム・エクスクルーシブID番号（サブ ID#1=02H）を，すべてのショー・コマンドで使用する。

ライヴ・パフォーマンス・コントロールの背後にある考え方は，個々の被制御装置の故障は，他の被制御装置のコミュニケーションを阻害すべきではないということである。この原則は，オープン・ループあるいはクローズド・ループのどちらにも当てはまることであろう。

オープン・ループの場合，被制御装置から制御装置へのコマンド・レスポンスは規定されていないし，必要とされない。オープン・ループによる制御は，最も経済的な手法であり，また MIDI を取り扱う上での基本である。MIDI ショー・コントロールは，他のチャンネル・メッセージやシステム・メッセージと同様に扱うことができるオープン・ループをサポートしている。

逆に，クローズド・ループは，被制御装置からの応答を期待する。クローズド・ループを行うには，よりインテリジェンな装置が必要であり，多くのメッセージを使用するが，制御装置と被制御装置のより正確なコミュニケーションを可能にする。クローズド・ループには，MIDI ショー・コントロールの 4.5 と 6 に記述されている「2フェイズ・コミット・プロトコル」を使用する。

このドキュメントでは，特に触れられていない限り，送信される数値はすべて 16 進数で表されている。また，“MSC”という頭文字は，MIDI ショー・コントロールのサブ ID#1（=02H）を示すために使われる。ショー・コントロール・メッセージのフォーマットは，以下の通りである。

```
F0 7F <device_ID> < MSC > <command_format> <command> <data> F7
```

＜備考＞

1. ひとつの SysEx で，2つ以上のコマンドは送信されない。
2. ショー・コントロール・メッセージの総バイト数は，128 を越えてはならなない。
3. SysEx メッセージは，すべての情報を送信したら，即座に F7H を送らなければならない。

2.2 装置の識別

<device ID> は，常に被制御装置の ID 番号である。

複数のコマンドが，同時にひとつの装置宛に送られることはよくある。例えば，2つの照明卓に GO とコマンドする場合は，以下のように送信する。

```
F0 7F <device_ID=1> < MSC > <command_format= 照明 > <GO> F7
F0 7F <device_ID=2> < MSC > <command_format= 照明 > <GO> F7
```

<device_ID> 値

00 ～ 6F	個々の ID
70 ～ 7E	グループ ID　1 ～ 15（オプション）
7F	ブロードキャスト（All-call ID）

すべての装置は，個々の ID と“All-call（7FH）”ID の両方に応答しなければならない。グループ・アドレッシング・モードは，オプションである。ある装置が,ひとつ以上のメーカー ID と,ひとつ以上のグループ ID に応答する場合もある。メッセージが認識される前に，メッセージの <device_ID> と <command_format> の両方が，被制御装置のデバイス ID とコマンド・フォーマットに一致しなければならない。

同じコマンド･フォーマットに応答する2つの独立した装置が,同じデバイス ID に応答するように設定されている場合,ひとつのメッセージを送信すれば，両方が応答する。“All-call”デバイス ID（7FH）は，同じコマンド・フォーマットの装置に対して（command_format = all-types と共に使用された場合は，すべての装置に対して），同一のコマンドをシステム全体にブロードキャストするために使用される（後述の4. 1を参照）。

<device_ID> バイトを認識する前に，SysEx が適切なMSCコマンドのフォーマットであるかを知るために，< MSC > と <command_format>（両方とも <device_ID> の後に続く）を見る必要がある。

典型的なシステムは,少なくともひとつの制御装置と,それに接続されたひとつ以上の被制御装置から構成されている。同じ装置が，同時に制御装置と被制御装置の両方であることも可能である。これはその装置が MSC コマンドのトランスレーター,インタープリター，あるいはコンバーターとして動作する場合である。この場合，その装置はMSCコマンドのトランスレーターやインタープリター，あるいはコンバーターのように動作する。あるコマンドを受信したら，そこでプログラムされている指示に従って，似たような，あるいは異なるコマンドを送信する場合もある。

複数の制御装置の送信がマージされ，ひとつ以上の被制御装置に分配されることもある。

オプションではあるが，被制御装置は要求される結果を得るために，それ自身によって必要とされるMSCコマンドを（MIDI OUT から）送信できてもかまわない。この時，装置は妥当なMSCコマンドを送信するが，制御装置として動作する必要のない場合もある。これは，制御装置が制御されているパフォーマンスに必要とされるメッセージのデータベースを効率よく制作したりエディットできるように，（MIDI IN を通じて）妥当なMSCメッセージをとらえる機能を持っている場合に有効である。この場合，被制御装置は制御装置に対して送信を行うが，それはパフォーマンス中にメッセージをとらえてストアし，再送信する目的のためである。

被制御装置によるMSCコマンドの送信によって可能となるもうひとつの応用方法は，同種の複数の装置のスレーブ化である。例えば，照明卓で GO ボタンを押した時にMSCコマンドの“GO”を送信した場合，MSCコマンドに従う他の照明卓がこの MIDI メッセージを受信すれば,同様に GO するだろう。このように,多くの被制御装置が,制御装置として動作することもできる別の被制御装置によって，制御される場合もある。相互に接続された場合でも，制御装置から被制御装置への一般的な呼応と同じである。

2.3　コマンド・フォーマット

コマンド・フォーマットは，制御装置から被制御装置へのメッセージ・バイトであり，コマンド・フォーマットごとに，それに続くコマンド・バイトのフォーマットが決められている。各コマンド・フォーマットは 01H ～ 7FH のフォーマット・コードで，その後ろに決められたコマンド・バイトを伴っていなければならない（コマンド・フォーマットの 00H は，拡張のために予約されている。また現在のところ，すべてのコードが定義されているわけではない）。

2.4　コマンド

コマンドは，制御装置から被制御装置へのメッセージ・バイトである。コマンドは 01H ～ 7FH のコマンド・コードで，総メッセージ長が 128 バイトを越えない，ひとつ以上のデータ・バイトをその後ろに伴っている場合もある。コマンド 00H は，拡張のために予約されており，また現在のところすべてのコードが定義されているわけではない）。

2.5　拡張セット

コマンド・フォーマット 00H とコマンド 00H は，2つの拡張セットのために予約されている。

00 01	**第1次拡張レベルの最初のコマンド・フォーマットかコマンド**
00 00 01	**第2次拡張レベルの最初のコマンド・フォーマットかコマンド**

現時点では，拡張機能はいっさい定義されていない。しかし，MSCが将来拡張された場合に対応するため，コマンド・フォーマットやコマンド・フィールドを見る時は，常に拡張のチェックをしなければならない。

2.6　データ長

送信されるデータ･バイト数に関する唯一の制限は，メッセージ･バイトの総数が 128 を越えてはならないことである。送信されるメッセージの，実際のデータ・フォーマットは，装置のメーカーによって定義されている。このことは，制御装置（あるいは制御装置のプログラマー）は，装置の正確なデータ・フォーマットを知らなければならないことを意味する。この情報は，メーカーや装置特有のものであろうから，すべてのメーカーは，しっかりした MIDI SysEx インプリメンテーション・ドキュメントを公表すべきである。

この仕様は，とてもローコストな照明卓から，複雑なオーディオ / ビデオ・マルチメディアを含む大規模なものまで，機器とメーカー側のひじょうに幅広い多種多様なニーズに対応できることを意図している。このため，単純な装置で使用されるデータ・フォーマットは，多くの機能を持つ装置で使用されるものよりも,ずっと短く複雑ではないだろう。そこでデータは,最も包括的な情報が最初に送られる。そうでないデータは，包括的でないデータを送信していることを示すために，各データ・バイトのグループ間にヌル・キャラクターの区切り文字をはさんで送信される。例えば単純な被制御装置では，基本的なデータのみを見て，残りは放棄するだろう。

例えば，複雑な被制御装置は “235.32.7.8.654” のように，数多くの小数点で細かく表されたサブセクションを持ち，キュー番号を扱うことができたりする。この制御装置が，キュー番号を “xxx.x” という形式でしか処理できない単純な装置へ送信した場合，その装置は，これらのデータを無視するか，あるいは “235.3.” というキュー番号として，妥当と思われる方法で対応することができる。

さらには，送信されたキュー番号のデータがキュー 235.3 を呼び出したとして，その後に区切り文字とキュー・リスト 36.6 を指定するデータが続き,さらにキュー･パス 59 の指定が区切り文字と共に続く場合もある。複数のキュー･リストはサポートするが,複数のキュー･パスはサポートしていない装置の場合，キュー・リスト 36.6（あるいは 36）のキュー 235.3 を処理し，キュー・パスのデータは無視して，単に現在の，あるいはデフォルトのキュー・パスを使用するだろう。

逆の状況を見てみると，すべてのキュー・データを処理できる装置に，単純なキュー番号データが送信された場合，そのキュー番号は，現在の，あるいはデフォルトのキュー・リストやキュー・パスを使用して処理されるだろう。

3. 標準仕様

データは，しばしば何らかの形式のキュー番号指定を含んでいるのでキュー番号と，それに関連するデータ送信の“標準的”な仕様には一貫性を持たせてあり，個々の詳細データ仕様の説明では省かれている（セクション5）。

3.1 キュー番号

キュー番号がデータとして送られる場合，追加情報フィールドに完全な“キュー番号”の記述が含まれる場合と，そうでない場合がある。キュー番号は，Q_number に加え，Q_list と Q_path が送られる場合がある。Q_list は，現在有効なキュー・リストのうちのどのリストに，その Q_number が置かれたり操作されたりするかを規定する。Q_path は，すべての利用可能なキュー保存メディアの中の，どの有効キュー・パスの Q_number を使用するかを規定する。データは，これらの情報フィールドを，以下の順序で含んでいる。

```
<Q_number> 00 <Q_list> 00 <Q_path> F7
```

上記のように，それぞれの独立したフィールド間に，00H という区切り文字バイトが，前のフィールドの終わりと次のフィールドの始まりを示すために置かれる。以下のもののみを送ることも可能である。

```
<Q_number> F7  または  <Q_number> 00 <Q_list> F7
```

被制御装置は，たとえ Q_number や Q_list，Q_path のデータがまったく送られてこないとしても，ひとつより多い区切り文字バイトのセットを受け入れ可能でなければならない。データは常に F7H で終了する。

Q_number や Q_list，Q_path は，サブセクションを正確に表すのに使用される ASCII 小数点キャラクター（2EH）と共に，ASCII ナンバー0～9（30H ～ 39H とエンコードされる）として示される。上記の例では，キュー 235.6，リスト 36.6，パス 59 は，次のような 16 進数データとして表されるだろう。

```
32 33 35 2E 36 00 33 36 2E 36 00 35 39 F7
```

小数点は，少なくとも1桁の数によって区切られなければならないが，被制御装置は，2つ以上の小数点が連なっている場合のエラーに対応しなければならない。MSC メッセージの長さが 128 バイトを超えてはいけないことを除けば，サブセクションを正確に表すのに，いくつの小数点が使用されてもかまわないし，各サブセクションに何桁の数字が使用されてもかまわない。

Q_list（あるいは Q_path）をサポートしない装置は，Q_number（あるいは Q_list）データの後にある 00H バイトを見つけたら，そこから F7H までのすべてのデータを放棄しなければならない。同様に，サブセクションを正確に表す小数点の数や，サブセクションにおける数字の桁数，あるいは何らかのフィールドにおいて受信されたキャラクターの総数をサポートしていない装置は，予測できる論理的な方法で，受信されたデータを取り扱わなければならない。

Q_list および Q_path，またはその一方をサポートする被制御装置で，これらのフィールドが Q_number と共に送られてこなかった場合，通常現在の，あるいはデフォルトのキュー・リストとキュー・パスを使用する。

照明への応用のために，キュー・リストは，オプションでキューが対応する再生，あるいはサブマスター・コントロール（0 ～ 127）を定義する。

すべてのメーカーは，上記の状況に対して自社製品の応答を明確かつ簡潔な記述で記載することを強く推奨する。

3.2　タイムコード・ナンバー

データは，しばしば何らかの形式の時刻情報を含んでいる。時刻の送信の“標準的”な仕様には一貫性があり，個々の詳細データ仕様の説明では省かれている。

MSCのタイムコードと，ユーザー・ビットの仕様は，MIDI タイムコードと MIDI キューイングで使用されるフォーマットと完全に一致しており，また MIDI マシン・コントロール 1.0 において提唱された，標準タイムコード・フォーマットとも同一である。いくつかの拡張フラグが追加されているが，それらは MIDI タイムコード / キューイング環境で使用された場合，常にゼロと認識されるので，完全にそのまま使用できるように定義されている。

3.2.1　標準タイムコード（タイプ {ff} と {st}）

これはタイムコード仕様の完全な形式で，常にきっかり5バイトのデータを含んでいる。

タイムコードのサブフレームには，ふたつの形式が定義されている。

第1の形式（ラベル {ff} が付けられたもの）は，まさしく MIDI キューイング仕様に記述されているサブフレーム・データ，すなわち 1/100 フレーム単位の小数フレームを含むものである。

第2の形式（ラベル {st} が付けられたもの）は，サブフレームに代わって，タイムコードのステータス・データがその場所に置き換えられたものである。例えば，テープレコーダーからデータを読んでいる時，これが記録されているタイムコード・データなのか，それとも単に高速巻き取り中のタコメーター・パルスによって更新されるタイム・データなのかを知ることができる。この場合，タイムコード進行中の場合と同様，サブフレーム・データは，手に入れることもタイミングよく送信することも困難であり，実際は役に立たない。

```
hr mn sc fr (ff!st)
hr  =  時刻とタイプ      :0 tt hhhhh
                tt  =  タイムコード・タイプ（ビット・フォーマット）
                       00  =  24フレーム
                       01  =  25フレーム
                       10  =  30ドロップ・フレーム
                       11  =  30フレーム
                hhhhh  =  時間 0 ～ 23（00H ～ 17H とエンコードされる）
mn  =  分        :0 c mmmmmm
                c  =  カラー・フレーム・ビット（タイムコード・ストリームの中のビットからコピーされる）
                       0  =  ノンカラー・フレーム
                       1  =  カラー・フレームド・コード
                mmmmmm  =  分 0 ～ 59（00H ～ 3BH とエンコードされる）
sc  =  秒        :0 k ssssss
                       k  =  予約済み（ゼロがセットされなければならない）
                ssssss  =  秒 0 ～ 59（00H ～ 3BH とエンコードされる）
fr  =  フレーム，第5バイト標識，及び符号 :0 g i fffff
                g  =  符号ビット
                       0  =  プラス
                       1  =  マイナス（符号付きタイムコードが許される場合）
```

i = 最終バイト標識ビット
0 = サブフレーム
1 = ステータス
fffff = フレーム 0 ～ 29 (00H ～ 1DH とエンコードされる)

最終バイト・ビット＝サブフレーム(i=0)の場合:
ff = フラクショナル・フレーム:0 bbbbbbb (0 ～ 99, 00H ～ 63H とエンコードされる)

最終バイト・ビット＝ステータス(i=1)の場合:
st = コード・ステータス・ビット・マップ:0 e v d xxxx
e = 推測コード・フラッグ・ビット
0 = 通常のタイムコード
1 = タコ, またはコントロール・トラックにより更新されたコード。
v = 不正なコード・ビット(e=1 の場合は無視)
0 = 妥当
1 = 不正(エラーあるいは現存しない)
d = ビデオ・フィールド・アイデンティフィケーション・ビット
0 = このフレームにフィールド情報はない
1 = 4あるいは8フィールド・ビデオ・シーケンスの最初のフレーム
xxxx = 予約されたビット(0000 をセットする)

<ドロップ・フレームの注意>

1. タイムコード・データの受信では, 書かれているデータのドロップ・フレーム, あるいはノンドロップ・フレーム・ステータスは, 被制御装置のステータス(すなわち装置自身からのタイムコード)が優先される場合がある。例えば, SET_CLOCK データがノンドロップ・フレーム・ナンバーと共にロードされ, 被制御装置のタイムコードがドロップ・フレームである場合, SET_CLOCK データは変換されることなく, 単にドロップフレーム・ナンバーとして解釈される。

2. さらに, 上記の SET_CLOCK ナンバーが取り得ないドロップ・フレーム・ナンバー(例えば 00:22:00:00)と共にロードされた場合は, 次の妥当なナンバーが使用される(この場合には, 00:22:00:02)。

3. オフセット, すなわち単なる2つのタイムコード間の時間差の計算では, ナンバーのひとつ, あるいは両方がドロップ・フレームの場合, 混乱を引き起こしうる。この仕様ではオフセットの計算の前に, まずドロップ・フレーム・ナンバーをノンドロップ・フレームに変換しておかなければならない。そうすれば, オフセットの計算結果は, ノンドロップ・フレームとして得られる。

   ドロップ・フレームからノンドロップ・フレームに変換するには, リファレンス・ポイント 00:00:00:00 から "ドロップ" されてきたフレームのナンバーを引く。例えば, ドロップ・フレーム・ナンバー 00:22:00:02 を, ノンドロップ・フレームに変換するには, 40 フレームを引いて, 00:21:58:22 である。40 という数は, 01 から 09, 11 から 19, 21 と 22 の各分ごとに2フレームが "ドロップ" されたことを意味している(メーカーによっては, すべての内部タイムコードをリファレンス・ポイント 00:00:00:00 からの単純なフレーム量としてストアする方を好むこともあるだろう。これは計算の複雑さを軽減するが, 変換はすべての入力と出力の段階で必要となる)。

4. インデックス・リスト

4.1. コマンド・フォーマット

コマンド・フォーマットは，一般，特殊，そしてオール・タイプというカテゴリーに分類される。一般コマンド・フォーマットは，照明の場合（01H）を除いて，下位4ビットが0である。そして，特殊コマンド・フォーマットの上位4ビットは，一般コマンド・フォーマットのそれと一致しており，一般コマンド・フォーマットの機能する範囲をさらに細かく制限している。

コマンド・フォーマットのオール・タイプ（7FH）は，同じデバイス ID を持つ装置に対して（あるいは device_ID=All-call と共に使用された場合は，すべての装置に対して）（前述の 2.2 を参照），同一のコマンドをシステム全体にブロードキャストするために使用する。例えば，オール・コールのデバイス ID と共に，オール・タイプのコマンド・フォーマットを使用すれば，ひとつのメッセージで，システムをまるごとリセットできる。

被制御装置は通常，オール・タイプの他には，ただひとつのコマンド・フォーマットにのみ反応する。複雑な装置では，時として2つ以上の動作を同時に行う場合があり，2つ以上のコマンド・フォーマットに反応する。制御装置は通常，すべてのコマンド・フォーマットにおけるコマンドを生成し，送信できなくてはならない。そうでなくては，有用性が限られてしまうだろう。

MSCには，潜在的に危険で生命を脅かすような，さまざまなパフォーマンス・プロセスの可能性が考えられるが，この仕様の意図は，コマンド・フォーマットやコマンドのタイプによって，従来のように手動で行われるよりもはるかに正確で厳密な，要求される結果をきちんともたらすようなコントロールを可能にすることである。このような条件下で，MSCを使用する主な利点を以下に挙げる。

1. **キューイングにおけるエラーの可能性が少ない。デジタル制御は，繰り返し動作させることが多い場合，圧倒的に信頼できる。とりわけ，疲労していたり未熟な舞台係よりは，はるかにそうである。**
2. **タイミングが正確である。さらに，コンピューターによるデジタル制御は，進行のタイミングが終始一貫して正確であり，またマニュアル・コントロールでも，人間のオペレーターと寸分違わない正確さを出すことができる。**

この仕様は，危険を伴う機器が使用されている場合に必要とされるようなものや，安全策として重大な意味を持つものに取って代わるものではない。緊急停止，放すと電気の切れるスイッチ，確認を可能にするコントロールなど，さまざまな安全装置を，最大限の安全のために用意すべきである。

接近センサー，ガス感知器，赤外線カメラ，そして動作感知器のような自動安全装置を，最大限の安全のために使用すべきである。MSCは危険な機器に，いつ GO したら大丈夫かを伝えるようには意図されていない。それはただ，安全なパフォーマンスのために，すべての条件が許容範囲内で申し分ない場合に，何が要求されているのかを知らせることを意図しているのみである。いかなる時にも，適切に設計された安全システムと訓練された安全要員だけが，諸条件が許容範囲内で申し分ないかどうかをはっきりさせることができるのである。

2フェイズ・コミットは，コミュニケーションのエラーを無くすことができ，ショー・コントロール・システムの安全性を高めてくれる。しかし，これも規格のパラメーターに従って実行されるものであり，上記の安全に関する注意に代わるものではない。

16 進	コマンド・フォーマット
00	拡張のために予約されている
01	Lightning（照明）　[一般]
02	Moving Lights（ムービング・ライト）
03	Color Changers（カラー・ライト）
04	Strobes（ストロボ）
05	Lasers（レーザー）
06	Chasers（チェイサー）
10	Sound（音響）　[一般]
11	Music（音楽）
12	CD Players（CD プレーヤー）
13	EPROM Playback（ROM 再生）
14	Audio Tape Machines（テープレコーダー）
15	Intercoms（インターコム）
16	Amplifiers（アンプ）
17	Audio Effects Devices（エフェクト装置）
18	Equalizers（イコライザー）
20	Machinery（舞台装置）[一般]
21	Rigging（仕掛け）
22	Flys（フライ）
23	Lifts（昇降機）
24	Turnrtables（ターン・テーブル）
25	Trusses
26	Robots（ロボット）
27	Animation（アニメーション）
28	Floats（フロート）
29	Breakways（ブレイクアウェイ）
2A	Barges（バージ）
30	Video（ビデオ）　[一般]
31	Video Tape Machines（VTR）
32	Video Cassette Machines（ビデオ・カセット）
33	Video Disk Players（ビデオ・ディスク）
34	Video Switchers（スイッチャー）
35	Video Effects（エフェクト）
36	Video Character Generatrors（キャラジェネ）
37	Video Still Stores（ストロボ）
38	Vidoe Monitors（モニター）

16 進	コマンド・フォーマット
40	Projection（プロジェクター）　[一般]
41	Film Projectors（フィルム）
42	Slide Projectors（スライド）
43	Video Projectors（ビデオ）
44	Dissolvers（ディゾルバー）
45	Shutter Controls（シャッター・コントロール）
50	Process Control（プロセス・コントロール）[一般]
51	Hydraulic Oil（ハイドローリック [水圧]・オイル）
52	H_2O（水）
53	CO_2（二酸化炭素）
54	Compressed Air（圧縮空気）
55	Natural Gas（ガス）
56	Fog（霧）
57	Smoke（煙）
58	Cracked Haze
60	Pyro（火）　[一般]
61	Fireworks（花火）
62	Explosions（爆発）
63	Flame（炎）
64	Smoke pots（スモーク・ポット）
7F	All-types（オール・タイプ）

4.2. 推奨される最小限のセット

MSCは，コマンド・フォーマットを受信する装置において，必ず装備しなければならないコマンドやデータの最小限のセットを規定していない。

しかし，異なるメーカーの制御装置と被制御装置の間をインターフェースさせる負担を軽減するために，推奨されるコマンドとデータの最小限のセットが4つのレベルで設定されている。被制御装置が，推奨される最小限のセットに従うのなら，その装置を確実に操作できる制御装置の設計を簡単なものとしてくれる。

現在定義されている推奨される最小限のセットを，以下に記述する。

1. 単純な被制御装置　　タイムコードなし　基本的なデータのみ
2. タイムコードなし　　完全なデータ適応性
3. 完全なタイムコード　　完全なデータ適応性
4.2フェイズ・コミット体系　（セクション 4.5 及び 6 参照）

各コマンドやデータが，どのレベルの推奨される最小限のセットに属するかは，インデックス・リストの右端の欄に示されている。

推奨される最小限のセットは，いかなる意味においても，装置がサポートする操作の範囲を制限するものではない。それは，“lowest common denominator”の精神に基づくものである。

4.3. 一般コマンド

以下のコマンドは，現在の照明システムにおける基本的な動作ばかりであり，おそらく一般的な意味で，すべての演劇用ショー・コントロール・システムに当てはまる。被制御装置が，これらのコマンドすべてに対応することは，必要条件ではないが，強く推奨される。

16 進	コマンド	データ・バイトの数	推奨される最小限のセット
00	拡張用に予約されている		
01	GO	可変	123
02	STOP	可変	123
03	RESUME	可変	123
04	TIMED_GO	可変	23
05	LOAD	可変	23
06	SET	4 or 9	23
07	FIRE	1	23
08	ALL_OFF	0	23
09	RESTORE	0	23
0A	RESET	0	23
0B	GO_OFF	可変	23

4.4. サウンド・コマンド

上記したコマンドに加え，次のコマンドは，現在のコンピューター制御による音響システムの動作としては基本的なものであり，ショー・コントロール・システムよって幅広く使用されている。制御装置は，これらのコマンドの送信をサポートすることが推奨される。

16 進	コマンド	データ・バイトの数	推奨される最小限のセット
10	GO/JAM_CLOCK	可変	3
11	STANDBY_+	可変	23
12	STANDBY_-	可変	23
13	SEQUENCE_+	可変	23
14	SEQUENCE_-	可変	23
15	START_CLOCK	可変	3
16	STOP_CLOCK	可変	3
17	ZERO_CLOCK	可変	3
18	SET_CLOCK	可変	3
19	MTC_CHASE_ON	可変	3
1A	MTC_CHASE_OFF	可変	3
1B	OPEN_CUE_LIST	可変	23
1C	CLOSE_CUE_LIST	可変	23
1D	OPEN_CUE_PATH	可変	23
1E	CLOSE_CUE_PATH	可変	23

4.5 2フェイズ・コミット・コマンド

以下のコマンドは，MIDI ショー・コントロールに2フェイズ・コミット（2PC）を拡張するためのものである。これには，MIDI ショー・コントロールで使われるいかなるフォームも必要としない。これは基本的な MIDI ショー・コントロールに，データの確認とエラー検出を付け加えるものである。拡張コマンドの使用が求められる状況としては，ショーが完全にモニターされ，中央コントロールされているような状況や，あるいは安全性のために，ショー・コントロールにさらなるチェックや代理機能などを必要とする場合である。

セクション6には，この規格で使用される2フェイズ・コミットの方法が記述されている。セクション5における2フェイズ・コミット・メッセージの記述を読む前に，セクション6に目を通すことを推奨する。

＜備考＞

2フェイズ・コミットでは，双方向のコミュニケーションを必要とする。被制御装置の MIDI OUT と制御装置の MIDI IN を MIDI でつないでおく必要がある。

16 進	コマンド	データ・バイトの数	推奨される最小限のセット
20	STANDBY	可変	4
21	STANDING_BY	可変	4
22	GO_2PC	可変	4
23	COMPLETE	可変	4
24	CANCEL	可変	4
25	CANCELLED	6	4
26	ABORT	6	4

5. コマンドとデータの詳細

00 拡張用のためにリザーブされている

01 GO

01	ゴー
<Q_number>	オプション（Q_list が送られた場合は必要）
00	区切り文字
<Q_list>	オプション（Q_path が送られた場合は必要）
00	区切り文字
<Q_path>	オプション

キューの実行あるいはフェード・インを行う。実行時刻は，被制御装置のキューによって決められる。キュー番号が指定されていない場合は，数字順で次のキューが GO される。キュー番号が指定されている場合は，そのキューが GO する。キューの実行が完了するまで“run”状態とする。制御装置から実行時刻の指定をしたい場合は，TIMED_GO を使用する。

複数のキュー・リストを持つ装置では，キュー番号が指定されていない場合，有効な Q_list にある，数字順で次のキューが（同一番号の付いたものはすべて）GO する。Q_number が，Q_list なしに送信された場合は，有効なキュー・リストにある，キュー番号と同一の番号のすべてのキューが GO する。

02 STOP

02	ストップ
<Q_number>	オプション（Q_list が送られた場合は必要）
00	区切り文字
<Q_list>	オプション（Q_path が送られた場合は必要）
00	区切り文字
<Q_path>	オプション

現在実行中のキューを停止させる。キュー番号が指定されていない場合は，すべての実行中の動作を STOP する。キュー番号が指定されている場合は，そのキューで実行された動作のみが STOP し，他はすべて変更されない。

03 RESUME

03	リジューム
<Q_number>	オプション（Q_list が送られた場合は必要）
00	区切り文字
<Q_list>	オプション（Q_path が送られた場合は必要）
00	区切り文字
<Q_path>	オプション

STOP されたキューによる動作を再開する。キュー番号が指定されていない場合は，すべての STOP された動作を RESUME する。キュー番号が指定されている場合には，そのキューで実行され STOP された動作のみが RESUME し，他はすべて変更されない。

04 TIMED_GO

04	タイムド・ゴー
hr mn sc fr ff	標準タイムコード仕様
<Q_number>	オプション（Q_list が送られた場合は必要）
00	区切り文字
<Q_list>	オプション（Q_path が送られた場合は必要）
00	区切り文字
<Q_path>	オプション

時刻を指定してキューの実行やフェード･インを行う。キュー番号が指定されていない場合は,数字順で次のキューが GO する。キュー番号が指定されている場合は，そのキューが GO する。キューの実行が完了するまで“run”状態とする。

時刻は，サブフレーム (ff タイプ) を持つ標準タイムコード仕様であり，即時から 24 時間の実行時間を設定できる。被制御装置が TIMED_GO をサポートしていない場合，時刻データは無視しても，キュー番号は通常通りの処理をして，TIMED_GO の代わりに GO しなくてはならない。要求された実行時間が，デフォルトのキュー・タイムであった場合には，TIMED_GO の代わりに GO が送られなくてはならない。

複数のキュー・リストを持つ被制御装置への規則は，前述の GO と同様である。

05 LOAD

05	ロード
<Q_number>	必要
00	区切り文字
<Q_list>	オプション（Q_path が送られた場合は必要）
00	区切り文字
<Q_path>	オプション

キューをスタンバイ・ポジションに置く。キュー番号が指定されていなければならない。LOAD は，要求されるキューへのアクセスに時間を要する場合に有効である。LOAD を前もって送信することで，キューを即座に GO することができる。

複数のキュー･リストを持った被制御装置において，Q_number が Q_list なしに送信された場合には，有効なキュー･リストにある，キュー番号と同一番号のすべてのキューをスタンバイにする。

06 SET

06	セット
cc cc	ジェネリック・コントロール・ナンバー（LSB から）
vv vv	ジェネリック・コントロール・バリュー（LSB から）
hr mn sc fr ff	標準タイムコード仕様（オプション）

照明のための標準的なジェネリック・コントロール・ナンバー		192-223	Speed controllers for the 32 playbacks
0-127	Sub masters	224-255	Chase sequence masters
128-129	Masters of the first playback	256-287	Chase sequence speed masters
130-131	Masters of the second playback	510	Grand Master for all channels
...	(etc.)	511	General speed controller for all fades
190-191	Masters of the 32nd playback	512-1023	Individual channel levels

ジェネリック･コントロールを指示する。ジェネリック･コントロールとその値は，それぞれ 14 ビットの数値で設定される。被制御装置は，ジェネリック・コントロールの変数，属性，レート，レベル，モード，ファンクション，エフェクト，サブ・チャンネル，スイッチなど，SET によって値を送ることができる。オプションとして，ジェネリック・コントロールがその値に達する時刻を送信することができる。

時刻は，サブ・フレーム (ff タイプ) を持つ標準時刻仕様であり，即時から 24 時間の時刻を指定できる。被制御装置が SET における時刻をサポートしていない場合は，時刻データを無視すべきである。

07 FIRE

07	ファイアー
mm	マクロ番号

あらかじめプログラムされたキーボード・マクロを作動させる。マクロは7ビットの数で定義される。マクロ自身は，被制御装置でプログラムされるか，あるいは ASCII キュー・データ・フォーマットや，その装置に適用できる方法を使用して，MIDI のファイル・ダンプ機能によってロードされる。

08 ALL_OFF

08	オール・オフ

コントロールの設定を変えることなく，すべての機能とアウトプットをオフにする。RESTORE によって ALL_OFF 以前の状態へ回復する場合もある。

09 RESTORE

09	リストア

状態を，ALL_OFF の前の状態に回復する。

0A RESET

0A	リセット

すべての機能を，電源投入時の状態などの初期状態とする。すべての動作中のキューを終了させ，各キュー・リストにある最初のキューを，適切なスタンバイ･ポジションに置く。言い換えれば RESET は，いかなるコントロールの値をも勝手に変更することなく，ショーを STOP し，それをスタンバイにする。

RESET によって自動的に，無効になっているキュー・リストやキュー・パスを有効にするべきかどうかは，被制御装置のメーカー次第であるが，その装置の MIDI インプリメンテーションには，はっきりと述べられていなければならない。

0B GO_OFF

0B	ゴー・オフ
<Q_number>	オプション（Q_list が送られた場合は必要）
00	区切り文字
<Q_list>	オプション（Q_path が送られた場合は必要）
00	区切り文字
<Q_path>	オプション

キューの実行，あるいはフェード・アウトをする。実行時刻は，被制御装置で決めらる。

キュー番号が指定されていない場合は，現在実行中のキューの動作が GO_OFF する。キュー番号が指定されている場合には，指定されたキューで実行中の動作が GO_OFF する。

複数のキュー・リストを持つ装置において，キュー番号が指定されていない場合は，有効なキュー・リストにある，現在実行中のすべてのキューが GO_OFF する。キュー番号が Q_list なしに送信された場合は，有効なキュー・リストにあるキュー番号と同一番号のすべてのキューが GO_OFF する。

GO コマンドを受信しても自動的に現在のキューを新しいキューに入れ代えない装置との互換性を保つために,制御装置はオプションで，同時に GO_OFF コマンドを送信することを促すべきである。

10 GO/JAM_CLOCK

10	ゴー／ジャム・ロック
<Q_number>	オプション（Q_list が送られた場合は必要）
00	区切り文字
<Q_list>	オプション（Q_path が送られた場合は必要）
00	区切り文字
<Q_path>	オプション

あるキューが自動追従するキューの場合，クロック･タイムを Go Time に強制設定するのと同時に，そのキューを実行，あるいはフェード・インさせる。実行時刻は，コントロールされる装置のキューによって決められる。

キュー番号が指定されていない場合は，数字順で次のキューが GO し，適切なキュー･リストのクロックが，そのキューのタイムにジャムする。数字順で次のキューがマニュアル･キューの(すなわち,そのキューがそれを“Auto Follow(自動追従)”キューにする特定の“Go Time”と共にストアされていない場合には，GO/JAM_CLOCK コマンドは無視される。

キュー番号が指定されている場合は，そのキューが GO し，そのキューがマニュアル動作でない限り，適切なキュー・リストのクロックがそのキュー・タイムにジャムする（マニュアル動作の場合には何の変化も起こらない）。

複数のキュー・リストを持つ装置の動作は，前述にある GO と同様である。

11 STANDBY_+

11	スタンバイ・プラス
<Q_list>	オプション

現在スタンバイ中であるキューの後の，数字順で次のキューをスタンバイ・ポジションに置く。

Q_list の指定がない場合は，数字順で次のキューを含む有効なキュー・リストが使用される。2つ以上の有効なキュー・リストが，同一番号を持つ場合は，各キューはそれぞれのスタンバイ・ポジションに移動する。

Q_list が標準キュー番号形式で送信された場合は，指定されたキュー・リストにおける次のキューのみがスタンバイ・ポジションに移動する。

12 STANDBY_-

12 スタンバイ・マイナス
<Q_list> オプション
現在スタンバイ中であるキューの，数字順で前のキューを，スタンバイ・ポジションに置く。

Q_list が送信されなかった場合は，数字順で前のキューを含む有効なキュー・リストが使用される。2つ以上の有効なキュー・リストが同一番号を持つ場合には，各キューはそれぞれのスタンバイ・ポジションに移動する。

Q_list が標準キュー番号形式で送信された場合は，指定されたキュー・リストにおける前のキューのみがスタンバイ・ポジションに移動する。

13 SEQUENCE_+

13 シーケンス・プラス
<Q_list> オプション
現在スタンバイ中であるキューの後の，数字順で次の親キューをスタンバイ・ポジションに置く。

“親”は，最初の小数点の前のキューの番号の整数値(“最も重要な数”)を引用する。例えば，キュー 29.324.98.7 がスタンバイ中で，その後に続くキューが 29.325，29.4，29.7，29.9.876，36.7，36.7.832，36.8，37.，そして 37.1，である場合には，SEQUENCE_+ によってキュー 36.7 がロードされ，スタンバイする。

Q_list が送信されなかった場合は，親シーケンス(同じ親を持つキューの集合)における次のキューを含む有効なキュー・リストが使用される。2つ以上の有効なキュー・リストが同一番号を有している場合は，それらのキューがそれぞれのスタンバイ・ポジションに移動する。

Q_list が標準形式で送信された場合は，指定されたキュー・リストにおける次の親キューのみがスタンバイ・ポジションに移動する。

14 SEQUENCE_-

14 シーケンス・マイナス
<Q_list> オプション
現在スタンバイ中のキューの前の，数字順で最も低い番号のキューをスタンバイ・ポジションに置く。

“親”は，最初の小数点の前のキュー番号の整数値(“最も重要な数”)を引用する。例えば，キュー 37.4.72.18.5 がスタンバイ中で，その前に先立つキューが 29.325，29.4，29.7，29.9.876，36.7，36.7.832，36.8，37.，そして 37.1，である場合には，SEQUENCE_- によってキュー 36.7 がロードされ，スタンバイする。

Q_list が送信されなかった場合は，前の親シーケンス(同じ親を持つキューの集合)を含む有効なキュー・リストが使用される。2つ以上の有効なキュー・リストが前の親シーケンスにおける同一の最も低い番号の親キューを有している場合には，それらのキューがそれぞれのスタンバイ・ポジションに移動する。

Q_list が標準フォームで送信された場合は，指定されたキュー・リストの前のシーケンスにおける最初の親キューのみがスタンバイ・ポジションに移動する。

15 START CLOCK

15	スタート・クロック
<Q_list>	オプション

自動追従のためのクロック・タイマーをスタートさせる。クロックがすでに走行している場合は，何の変化も起こらない。クロックは，それが STOP されていた間に，それが含んでいた時刻からカウントを続ける。

Q_list が送信されなかった場合は，すべての有効なキュー・リストのクロックが同時にスタートする。

Q_list が標準フォームで送信された場合は，そのキュー・リストのクロックのみがスタートする。

16 STOP CLOCK

16	ストップ・クロック
<Q_list>	オプション

自動追従のためのクロック・タイマーを停止させる。クロックがすでに停止している場合には，何の変化も起こらない。クロックが停止している間，STOP コマンドを受信した瞬間に，それが含んでいた時刻の値を保持する。

Q_list が送信されなかった場合，すべての有効なキュー・リストのクロックが停止する。

Q_list が標準フォームで送信された場合，そのキュー・リストのクロックのみ停止する。

17 ZERO_CLOCK

17	ゼロ・クロック
<Q_list>	オプション

自動追従のためのクロック･タイマーを，それが走行しているいないに関わらず，00:00:00:00.00 をセットする。クロックがすでに停止し，ゼロになっている場合には何の変化も起こらない。ZERO_CLOCK は，クロックの走行ステータスには影響を及ぼさない。

Q_list が送信されなかった場合，すべての有効なキュー・リストのクロックが，ゼロになる。

Q_list が標準フォームで送信された場合，そのキュー・リストのクロックがゼロになる。

18 SET_CLOCK

18	セット・クロック
hr mn sc fr ff	標準時刻仕様
<Q_list>	オプション

自動追従のためのクロック・タイマーを，それが走行しているいないに関わらず，送信された標準タイムと等しい値に設定する。SET_CLOCK は，クロックの走行ステータスには影響を及ぼさない。

Q_list が送信されなかった場合は，すべての有効なキュー・リストのクロックが，同時にセットされる。

Q_list が標準形式で送信された場合は，そのキュー・リストのクロックのみがセットされる。

19 MTC_CHASE_ON

19　　MTC チェイス・オン

<Q_list>　　オプション

自動追従のためのクロック・タイマーに入ってくる MIDI タイムコードへのチェイスを開始する。このコマンドが受信された時，何の MTC も受信されていない場合には，クロックは MTC が受信されるまで，現在実行中または STOP の状態に留まり，MTC が受信された時点でクロックは，MTC と同じ時刻を連続的に示す。MTC が断続的になる場合は，クロックは受信された最後の妥当な MTC メッセージの値を表示し続ける。

Q_list が送信されない場合は，すべての有効なキュー・リストのクロックが同時にチェイスする。Q_list が標準形式で送信された場合は，そのキュー・リストのクロックのみがチェイスする。

1A MTC_CHASE_OFF

1A　　MTC チェイス・オフ

<Q_list>　　オプション

自動追従にためのクロック・タイマーに入ってくる MIDI タイムコードにチェイスすることを中止させる。MTC_CHASE_OF が受信された時，クロックは MTC_CHASE_ON が受信された瞬間におけるオペレーティング状態に従って，走行中あるいは停止の状態に戻る。

MTC_CHASE_OFF は，クロックの時刻を変更しない。すなわち，クロックが停止している場合は，受信された最後の妥当な MTC メッセージの値（あるいは単にクロック・レジスターの中の最も近い時刻）を保持する。クロックが走行している場合は，そのレジスターの中の最も近い時刻からカウントを続ける。

Q_list が送信されなかった場合は，すべての有効なキュー・リストのクロックが，同時にチェイスを停止する。

Q_list が標準形式で送信された場合は，そのキュー・リストのクロックのみがチェイスを停止する。

1B OPEN_CUE_LIST

1B　　オープン・キュー・リスト

<Q_list>　　必要とされる

指定のキュー・リストを，すべてのMSCコマンドに対して利用可能にし，そのリストに含まれるキューを，現在のショーに含める。

OPEN_CUE_LIST が受信された時は，指定されたキュー・リストが有効になり，そのリスト中のキューへのアクセスが可能となる。標準フォームの Q_list が送信されなければならない。

指定されたキュー・リストがすでにオープンしていたり，あるいはそれが存在しない場合には，何の変化も起こらない。

1C CLOSE_CUE_LIST

1C　　クローズ・キュー・リスト

<Q_list>　　必要とされる

指定のキュー・リストを，すべてのMSCコマンドに対し利用不可能にし，そのリストに含まれるキューを，現在のショーから除外する。

CLOSE_CUE_LIST が受信されると，指定されたキューリストが無効になり，そのリスト中のキューは，アクセスできなくなる。しかし，リスト中のキューの状態は変化しない。標準形式の Q_list が送信されなければならない。

指定されたキュー・リストがすでに閉じられていたり，あるいはそれが存在しない場合には，何の変化も起こらない。

1D OPEN_CUE_PATH

1D	オープン・キュー・パス
<Q_path>	必要とされる

指定のキュー・パスを，すべてのMSCコマンドに対して，そして同様にすべてのショー・コントロールにおけるキュー・パスにアクセスする必要のあるものに対して，利用可能にする。

OPEN_CUE_PATH が受信されると，そこで指定されたキュー・パスは有効となり，そのパスにあるキューは，被制御装置によってアクセスできる。標準形式のキュー・パスが送信されなければならない。

指定したキュー・パスがすでにオープンしていたり，あるいはそれが存在しない場合には，何の変化も起こらない。

1E CLOSE_CUE_PATH

1E	クローズ・キュー・パス
<Q_path>	必要

指定のキュー・パスを，すべてのMSCコマンドに対して，そして同様にすべてのショー・コントロールにおけるキュー・パスにアクセスする必要のあるものに対して，利用不可能にする。

CLOSE_CUE_PATH が受信されると，そこで指定されたキュー・パスは無効となり，そのパスにあるキューは，被制御装置によってアクセスできなくなる。標準形式のキュー・パスが送信されなければならない。

指定したキュー・パスがすでに閉じられていたり，あるいはそれが存在しない場合には，何の変化も起こらない。

20 STANDBY

20	スタンバイ
cc cc	チェックサム（LSB から）
nn nn	シーケンス番号（LSB から）
d1 d2	データ（4つの7ビット・キュー・データの値）
d3 d4	（セクション 6.8 を参照）
<Q_number>	必要とされる
00	区切り文字
<Q_list>	オプション（Q_path が送られた場合は必要）
00	区切り文字
<Q_path>	オプション

通常，制御装置が送信するメッセージで，特定のキューをスタンバイ状態にする。このメッセージを受信した装置は，STANDING_BY または ABORT のどちらかで応答しなければならない。何らかの理由で，装置が指定されたキューを実行する準備ができない場合，ABORT メッセージが返される。

STANDBY メッセージ中の d1，d2，d3，d4 の値は，実行したいキューについての追加情報である。これを受信した装置がこれらの値を使用するかどうかは，オプションである。ただし制御装置は，これらの値を STANDBY メッセージ中に常に含めて送る必要があり，d1 ～ d4 の値が不明な時は，ゼロが送られる。

STANDBY メッセージ中で送られる d1 ～ d4 の値は，続いて送られる GO_2PC メッセージにあるキューの情報と一致していなければならない。これが一致しない場合,GO_2PC メッセージを受信した装置は,“invalid dn cue date value（無効なキューデータ値）”ステータス・コードを含む ABORT メッセージを返すだろう。d1 ～ d4 の値についての追加情報と使用例については，セクション 6.8 を参照のこと。

STANDBY メッセージを受信した装置は，2秒以内に STANDING_BY メッセージを返す必要がある。制御装置は，この時間内に STANDING_BY メッセージが返されなかった場合，中断と判断する。

21 STANDING_BY

21	スタンディング・バイ
cc cc	チェックサム（LSB から）
nn nn	シーケンス番号（LSB から）
hr mn sc fr ff cue	キューを実行するのに必要とされる最大時間（標準時刻仕様）
<Q_number>	オプション（Q_list が送られた場合は必要）
00	区切り文字
<Q_list>	オプション（Q_path が送られた場合は必要）
00	区切り文字
<Q_path>	オプション

通常，被制御装置から送信される。指定されたキューを実行する準備ができていることを表す。
＜備考＞　キューは，キュー番号，キュー・リスト，そしてキュー・パスが指定された BSTANDBY メッセージ中にある d1, d2, d3, d4 は，そのキューの実行方法に影響する場合もあるが，それらはキューを特定するものではない。

“実行の準備ができている”とは，指定されたキューが被制御装置に存在し，それがメモリー中にあるか，あるいは即時に実行する準備が完全にできており，装置がそのキューによって指令される動作を完全に実行できることを意味する。指定されたキューの準備ができたことは，返された STANDBY メッセージのシーケンス番号が一致したことによって確定する。オプションとして，キューがシーケンス番号とキュー番号，キュー・リスト，キュー・パスによって識別されてもかまわない。いずれも STANDBY メッセージに含まれる値と完全に一致していなければならない。そうでなければ，ABORT メッセージで応答することになる。

STANDING_BY メッセージは，そのキューを実行するのに必要とされる最大時間を含んでいる。制御装置はこの時間によって，被制御装置がそのキューの実行に失敗していないかを確認できるだろう。最大時間は，実際に必要とされる時間よりもずっと大きくしておいてもかまわないが,小さくしてはいけない。そのキューを実行するにあたり,オペレーターによるなんらかの操作を必要とする場合，最大時間は，そのオペレーターの操作に必要とされる時間を考慮したものでなければならない。

22 GO_2PC

22	GO_2PC
cc cc	チェックサム（LSB から）
nn nn	シーケンス番号（LSB から）
d1 d2	データ（4つの7ビット・キュー・データの値）
d3 d4	（セクション 6.8 を参照）
<Q_number>	必要
00	区切り文字
<Q_list>	オプション（Q_path が送られた場合は必要）
00	区切り文字
<Q_path>	オプション

通常，制御装置から送信される。キューの実行を開始する。

＜備考＞　キューは，Q_number，Q_list，Q_path で指定される。GO_2PC メッセージ中にある d1，d2，d3，d4 は，そのキューの実行方法に影響する場合もあるが，それらはキューを特定するものではない。

キューの実行が完了したら，被制御装置は COMPLETE メッセージを返す。何らかの理由でそのキューの実行前，あるいは実行中にキューを終了させなければならない事態が発生した場合，装置はただちに ABORT メッセージを返す。

被制御装置は，GO_2PC を受信する前に受信された STANDBY メッセージに対し，STANDING_BY メッセージで応答する。そのメッセージにおける Q_number 等のフィールドは,GO_2PC メッセージ中のそれと一致していなければならない。そうでない場合，被制御装置は GO_2PC メッセージに対して“not standing　by（準備は完了していない）”ステータスを含む ABORT メッセージで応答しなければならない。

GO_2PC メッセージ中の d1，d2，d3，d4 の値は，実行したいキューについての追加情報である。これを受信した装置がこれらの値を使用するかどうかは，オプションである。ただし制御装置は，これらの値を GO_2PC メッセージ中に常に含め送る必要があり，d1 ～ d4 の値が不明な時は，ゼロが送られる。

メッセージ中で送られる d1 ～ d4 の値は，先行する STANDBY メッセージ中で送られるそれらの値と一致しなければならない。これが一致しない場合は，被制御装置は GO_2PC メッセージに対して“invalid dn cue date value（無効なキュー・データ）”ステータスを含む ABORT メッセージで応答するだろう。d1 ～ d4 の値についての追加情報と，使用例についてはセクション 6.8 を参照のこと。

被制御装置は,先行する STANDBY と STANDING_BY のやり取りをいつまでも記憶しておく必要はない。しかし,被制御装置は，これらのやり取りを記憶していなかったために“not standing　by（準備は完了していない）”ABORT を送信することが極力ないように，やり取りを十分記憶できるべきである。各メーカーは，装置が記憶できる STANDBY と STANDING_BY の記録数を公表すべきである。記憶された STANDBY と STANDING_BY のやり取りは，GO_2PC の受信時にクリアされる。そのキューが再実行される時も，新たな STANDBY と STANDING_BY のやり取りをしなければならない。

被制御装置は，前もって制御装置に送った STANDING_BY メッセージによって，キューを実行するのに必要とされる最大時間を知らせる。この時間内に被制御装置はキューを処理し，COMPLETE あるいは ABORT メッセージを返す。制御装置は，時間内にどちらも受信できなかった場合，ABORT メッセージが返されたかのごとく処理する。

23　COMPLETE

23	コンプリート（完了）
cc cc	チェックサム（LSB から）
nn nn	シーケンス番号（LSB から）
<Q_number>	オプション（Q_list が送られた場合は必要）
00	区切り文字
<Q_list>	オプション（Q_path が送られた場合は必要）
00	区切り文字
<Q_path>	オプション

通常，被制御装置から送信される。指定されたキューの実行を完了したことを示す。

＜備考＞　キューは，Q_number，Q_list，Q_path で指定される。GO_2PC メッセージ中にある d1，d2，d3，d4 は，そのキューの実行方法に影響する場合もあるが，それらはキューを特定するものではない。

キューが処理されたことは，GO_2PC メッセージで送られたシーケンス番号が返されたことにより，認識されなければならない。オプションとして，キューはシーケンス番号とキュー番号，キュー・リスト，キュー・パスの両方で指定されていてもかまわない。両方が用い

られる場合，それらは GO_2PC メッセージ中のそれらの値と完全に一致していなければならない。そうでなければ，この応答は ABORT メッセージとして取り扱われる。

24 CANCEL

24	キャンセル
cc cc	チェックサム（LSB から）
nn nn	シーケンス番号（LSB から）
<Q_number>	必要とされる
OO	区切り文字
<Q_list>	オプション（Q_path が送られた場合は必要）
OO	区切り文字
<Q_path>	オプション

通常,制御装置が送信する。Q_number 等のフィールドによって指定されたキューは中断されるべきであることを示す。CANCEL メッセージを受信する前には，少なくとも STANDBY メッセージが（おそらく GO_2PC メッセージも）受信されていなければならない。そうでなければ，被制御装置は“not standing　by（準備は完了していない）”ステータスを含む CANCELLED メッセージで応答しなければならない。

STANDBY と STANDING_BY のやり取りが行われていた場合（GO_2PC メッセージはなし），そのやり取りは単純に忘れられる。そのキューを再実行する際は，新たな STANDBY と STANDING_BY のやり取りが必要となる。

GO_2PC によってキューが実行され，それに対する COMPLETE が送られていないの場合には，以下の可能性がある。

1.	完了	通常の完了にまで至る
2.	ポーズ	停止して，再実行を待つ
3.	終了	再実行の可能性なしで，停止する
4.	リバース	GO_2PC メッセージの実行が開始される直前の状態に戻される

これらの4つのどの動作が取られるかは，被制御装置と CANCEL メッセージが受信された状況による。装置によっては，常に4つの動作のひとつを行うものもあれば,プログラムに従って動作を選ぶ装置もあるだろう。各メーカーは自社の装置が CANCEL メッセージを受信した場合に取る動作を明らかにすべきである。

キャンセルされたキューを完了させる場合，その装置は“completing（完了中）”ステータスを含む CANCELLED メッセージと COMPLETE メッセージの両方を送信しなければならない。CANCELLED メッセージは，CANCEL メッセージが来たら即時に返され，COMPLETE メッセージは，そのキューが実際に完了した時に送られる。

CANCEL メッセージによって中断されたメッセージは,STANDBY と STANDING_BY,GO_2PC メッセージをやり取りすることで，再開できる。それ以外の場合でも，STANDBY と STANDING_BY，GO_2PC メッセージのやり取りによって，そのキューが再実行されることになるだろう。

CANCEL メッセージを受信したら，CANCELLED メッセージ，あるいは ABORT メッセージで応答しなければならない。ABORT メッセージを送り返すのは,CANCEL メッセージ中にチェックサム･エラーが発見された場合のみである。それ以外の場合は，すべて CANCELLED メッセージで応答しなければならない。CANCEL メッセージに対する応答が2秒以内になかった場合は，ABORT（中断）と見なされるだろう。

25 CANCELLED

25	キャンセルド
cc cc	チェックサム（LSB から）
s1 s2	ステータス・コード（ステータス =(s1 × 4)+(s2 × 512)）
nn nn	シーケンス番号（LSB から）

Cancelled メッセージのための標準ステータス・コード

80 04	completing
80 08	paused
80 0C	terminated
80 10	reversed
80 24	not standing by
80 28	manual override in progress

通常，被制御装置から送信される。CANCEL メッセージが受け入れられたこと，またはそれが不適切だったことを示す。ステータス・コードは，そのキューがどう処理されたかを示す。有効なステータス・コードは，“not standing by（非準備中）”，“manual override in progress（手動優先進行中）”，“completing（完了中）”，“paused（中断されている）”，“terminated（終了）”，そして“reversed（リバース）”である。

ステータス・コード“not standing by”は，先の STANDBY と STANDING_BY のやり取りにおいて，Q_number 等によってキューが指定されたとして，これが記憶されもしないし，実行もされていないことを示す。この状態が起きる理由は明らかだが，それ以外として，被制御装置が CANCELLED メッセージを受信する前に，GO_2PC メッセージの実行をすでに終了してしまった場合がある。

“completing”，“paused”，“terminated”，そして“”reversed”ステータス・コードは，それぞれ CANCEL メッセージ後のキューの処置を示す。CANCEL メッセージを送信する前に GO_2PC メッセージが受信されなかった場合は，CANCELLED メッセージ中のステータス・コードは，常に“”terminated”である。

“manual override in progress”ステータス・コードは，装置を操作するオペレーターがキューの実行コントロールを引き継いだことを示す。装置は，オペレーターが手動でキュー動作を開始した場合は，すべての MIDI ショー・コントロール・メッセージを無視するように設計されるだろう。これに関する追加情報は，セクション 6.4.5 を参照。

CANCEL メッセージ中のシーケンス番号は，送信したキャンセルに対する応答であることを確認するために使用される。この時，オプションのパラメーターは認識されない。

26 ABORT

26	アボート（中断）
cc cc	チェックサム（LSB から）
s1 s2	ステータス・コード（ステータス =(s1 × 4)+(s2 × 512)）
nn nn	シーケンス番号（LSB から）

ABORT メッセージのための標準ステータス・コード

00 00	unknown/undefined error
80 00	checksum error
80 20	*timeout

```
80 24        not standing by
80 28        manual override initiated
80 30        manual override in progress
80 40        deadman interlock not established
80 44        required safety interlock not established
80 50        unknown <Q_number>
80 54        unknown <Q_list>
80 58        unknown <Q_path>
80 5C        too many cues active
80 60        cue out of sequence
80 64        invalid d1 cue data value
80 68        invalid d2 cue data value
80 6C        invalid d3 cue data value
80 70        invalid d4 cue data value
80 90        manual cueing of playback medium required
80 A0        power failure in controlled device subsystem
80 B0        reading new show cues from disk
10 04        (meaning is dependent on Command_Format)
10 08        (meaning is dependent on Command_Format)
10 0C        (meaning is dependent on Command_Format)
10 10        (meaning is dependent on Command_Format)
10 14        (meaning is dependent on Command_Format)
10 18        (meaning is dependent on Command_Format)
10 1C        (meaning is dependent on Command_Format)
11 04        (meaning is dependent on Command_Format)
11 08        (meaning is dependent on Command_Format)
12 04        (meaning is dependent on Command_Format)
```

通常，被制御装置から送信される。STANDBY，GO_2PC，あるいは CANCEL メッセージの実行に失敗したことを示す。ステータス・コードは，メッセージが実行できなかったいちばんの理由を示す。メッセージ中にあるシーケンス番号は，STANDBY の ABORT や GO_2PC，あるいは CANCEL メッセージが，送ったメッセージに対する応答なのかを確認するために使用される。

< 備考 > これに加え，そのメッセージが実行できなかったもっと軽微な理由がある場合もある。したがって，ステータス・コードによって報告されたエラー状態を正しただけでは，実行を可能にするには十分でない場合もある。

ステータス・コードの深刻さは，その対処のしやすさに関係する。例えば，“deadman interlock not established（自動安全連動装置が設置されていない）”ステータス・コードは，“motor failure（モーター故障）”ステータス・コードよりも深刻ではない。前者の場合，すぐに人が対処することが可能であるが，後者の状態を正すには，モーターの修理が必要かもしれない。

“manual override in progress（手動優先進行中）”ステータス・コードは，装置を操作するオペレーターがキューのコントロールを引き継いだことを示す。装置は，オペレーターが手動でキュー動作を開始した場合，すべての MIDI ショー・コントロール・メッセージを無視するように設計されるだろう。これに関する追加情報については，セクション 6.4.5 を参照。

6. 2フェイズ・コミットの詳細

2フェズ・コミット(2PC)は，メッセージ処理の方法論である。2フェイズ・コミットは，コミュニケーションに2つの異なったフェイズを含んでいる。最初のフェイズでは，すべての関係者は何がなされるべきかについて合意する。第2のフェイズでは，すべての関係者は前もって合意された行動を開始し，その結果を報告する。

2フェイズ・コミットの方法論は，演劇における vocal cue calling system において採用されているキュー出しの方法にとてもよく似ている。最初のフェイズは standby(スタンバイ)フェイズである。第2のフェイズは，GO(作動)フェイズである。

6.1 制御装置と被制御装置

2フェイズ・コミットによる送信では，制御装置は STANDBY，GO_2PC，そして CANCEL メッセージを送る。被制御装置は，STANDING_BY，COMPLETE，CANCELLED，そして ABORT メッセージを送る。

6.2 人間による操作

この仕様では，2PCによる制御装置と被制御装置のところに，通常，人間のオペレーターが付いているものとしている。この役割を勤める人は，“ローカル・オペレーター”と呼ばれる。オペレーターは，単に動作をモニターしているだけという場合もある。あるいはオペレーターは，作動の安全確認の役割を果たす場合もある。オペレーターはまた，セクション 6.4.5 で記述されるように，被制御装置を手動操作する場合もある。

おそらくオペレーターは，何らかのコンソールや他のユーザー・インターフェースを経由して，各自の操作を行う。このインターフェースは，“ローカル・オペレーター・インターフェース”と呼ばれる。

6.3 2フェイズ・コミットと他の MSC メッセージとの関係

2フェイズ・コミット(2PC)と他の MIDI ショー・コントロール (MSC) メッセージとでは，いくつかの大きな違いがある。これらの違いを理解することは，製品を首尾よく作る上で重要である。さらにこの違いは，ひとつの装置で両方のメッセージ・セットをサポートすべきではないことを強く示唆するものである。したがって，この違いを理解することは，どのような装置にしろ，それに使用すべき最良のメッセージ・セットを選ぶために重要である。

MSC と 2PC の違いは，基本となる原理からある。

・MSC は動的で即時性を持つ：これを今実行しなさい，このように……というように。

・2PC は計画され，台本が書かれている：このシーケンスを立案どおりにやりなさい，他の方法はだめというように。

MSC は台本のないコンサートや，照明や音響，そしてビデオなどのような安全への配慮を考えなくもよいメディアに適している。これに対し 2PC は，きっちりと台本の書かれたショーや，機械仕掛けや発煙筒，プロセス・コントロールといった安全への配慮が必要なメディアに，よりふさわしい。

MSC メッセージ・セットは，被制御装置が何らかの機械的，あるいは電気的アクションを引き起こす，多くのメッセージを含んでいる。例えば，GO，SET，FIRE，STOP，そして RESUME のように。このアクションをメッセージの大きな集団は，多種多様な被制御装置全体にわたって一貫性のあるビヘイビア(振る舞い)をもって，要求された様々な結果すべてとコミュニケーションをとる必要がある。適切なビヘイビアは，MSC 規格に定義された各アクション・メッセージとの明快な一致を通じて保証される。

2PC メッセージ・セットは，GO_2PC という唯一のメッセージしか持たない。GO_2PC メッセージによってどのような結果が生み出されるのかは，被制御装置にストアされているキューと，その装置がキューの中身をどのように解釈するかによって決まる。望む動作をさせるには，適切な装置を選び，それらに正しい結果を生み出すためのキューを入力しておかなければならない。さらにまた制御装置は，すべての被制御装置のキューを適切にコーディネートして，2PC メッセージを発行しなければならない。

MSC の利点としては，リクエストされた動作に対する応答が素早いことと，システム構成がシンプルなことである。次の2つの理由から,素早い応答が可能である。第1に,MSC は動作させるのにただ1つのメッセージしか必要としないが,2PC は少なくとも3つのメッセージを必要とする。第2に 2PC は，動作のリクエストと動作開始までの間に，最大2秒のディレイを含んでいる。

2PC の利点としては，次の2つが挙げられる。第1に，2PC は複数の被制御装置の動作を非常に高いレベルでコーディネートすることができる。第2に，2PC はプロトコル中に組み込まれたエラー発見と回復操作が可能なことだ。

6.4　2フェイズ・コミット・メッセージ・シーケンス

6.4.1 通常のメッセージ・シーケンス

いかなるエラーもない場合，2フェイズ・コミット・プロトコルでひとつのキューを実行するには，4つのメッセージが必要である。以下の表は，メッセージとその順番，各メッセージの送信側と受信側，そしてそのメッセージの目的を示している。第1と第2のメッセージは「セクション 6」における“スタンバイ”フェイズ，第3と第4のメッセージは“GO”フェイズである。

順	メッセージ	送信側／受信側	目的
1st	STANDBY	制御装置	被制御装置に，あるキューが実行されようとしていることを知らせる（キュー実行シーケンスを開始）。
2nd	STANDING_BY	被制御装置	制御装置に，被制御装置が準備ができていて，先行する STANDBY メッセージ中に記述されたキューを実行可能であることを知らせる（制御装置に，そのキューを実行するのに必要とされる時間をも知らせる）。
3rd	GO_2PC	制御装置	被制御装置に，先行する STANDBY と STANDING_BY メッセージのペア中で，指定したキューの実行開始を指示する。
4th	COMPLETE	被制御装置	制御装置に，被制御装置が先行する GO_2PC メッセージ中で指定されたキューの実行を完了したことを知らせる（キュー実行シーケンスを終了させる）。

ひとつの制御装置が複数の被制御装置に，制御装置によって適切であると考えられるキュー番号の順で，STANDBY と GO_2PC メッセージを送ってもかまわない。しかし，ひとつの被制御装置に送られるメッセージのキュー番号の順には，配慮が必要である。被制御装置は，実行するキューに特定の配列を必要とする場合もある。この順を守らなかった場合，被制御装置は，ひとつかそれ以上の STANDBY メッセージに“cue out of sequence（シーケンス外のキュー）”ステータス・コードを含む ABORT メッセージを返すことになろう。

制御装置は被制御装置からの STANDING_BY，COMPLETE，あるいは CANCELLED の各メッセージを受信する順序を想定して動作してはならない。それらの順序は個々のキューの実行時間によって決定されるものであるため，そのような想定は特に COMPLETE メッセージにおいて矛盾を生じる。被制御装置は，以下に述べられている2秒間のタイム・アウト（時間切れ）ルールが守られる限り，いかなる順でも STANDING_BY と COMPLETE を自由に送ることができる。

STANDBY メッセージは，通常 GO_2PC メッセージを送る少なくとも2秒前に送られるべきである。被制御装置がタイム・アウト（時間切れ）時間である2秒まで，STANDING_BY メッセージでの応答に時間を必要とすることも考えられる。GO_2PC メッセージの2秒前に STANDING_BY メッセージを送るのが，常に可能とは限らない。しかし，制御装置の設計者は“2秒スタンバイ”ルールを守れなかった場合のリスクを考えておかなければならない。できるだけこのルールを守ることに努め，そして守れなかった場合の備えも必要である。

6.4.2 レスポンスの時間切れ

制御装置は，STANDBY メッセージの送信後，指定したキューの実行時間の 125％以内に COMPLETE メッセージを受信できない場合，タイム・アウト処理を行う。上乗せされた 25% は，制御装置と被制御装置におけるタイミングの誤差を考慮したものだ。

被制御装置から制御装置への応答メッセージのタイム・アウトは，2フェイズ・コミットにおける重要な要素である。STANDBY または CANCEL メッセージの送信後，2秒以内に STANDING_BY や COMPLETE メッセージが受信できない場合は，タイム・アウトとして処理する。

タイム・アウトと見なされる時は，被制御装置が働かなくなっている（少なくとも2フェイズ・コミット・プロトコルに関しては）ことを制御装置に知らせる。タイム・アウトと見なした制御装置は，あたかも被制御装置から制御装置に ABORT メッセージが返されたかのように取り扱わなければならない。

タイム・アウトを内部表現するために，“タイム・アウト”ステータス・コードが定義されている。送信される ABORT メッセージには“タイム・アウト”ステータス・コードは含まれないが，制御装置は，タイム・アウトの状態を，“タイム・アウト”ステータス・コードを含む ABORT メッセージを内部的に生成してもかまわない。

6.4.3　異常（例外的）な状態の取り扱い

ABORT，CANCEL，そして CANCELLED という2フェイズ・コミット・メッセージは，異常な状態が起こった場合にのみ使用される。以下の表は，これらのメッセージの目的，送信側，そして受信側を記述している。

メッセージ	送信側	目的
ABORT	被制御装置	制御装置に異常を通知する
CANCEL	制御装置	被制御装置に，先行する指示を放棄するように指示
CANCELLED	被制御装置	先行する指示を放棄することを確認

被制御装置が異常（何か通常のキューの実行を妨げるもの）を発見した時はいつでも，ABORT メッセージを使用して，制御装置にその状態を報告する。その異常を表すステータス・コードに加え，ABORT メッセージは，異常な状態になったことで適切に実行できなくなった STANDBY，あるいは GO_2PC メッセージのシーケンス番号を含んでいる。

＜備考＞

被制御装置は，STANDBY，GO_2PC，CANCEL メッセージに対する応答以外に，ABORT メッセージを送ってはいけない。もっと広い視野から眺めてみれば，異常な状態は，その存在によって適切なキューの実行が妨げられるまでは明らかにならないのである。しかし，被制御装置は，例外的な状態が発見された瞬間においてもなおローカルな動作を開始するかもしれない。

場合によって制御装置は，ABORT メッセージに対して，結果として ABORT メッセージとなるメッセージを再送信する場合もあるだろう。これはセクション 6.4.4 に述べられている。そうでなければ制御装置は，ABORT メッセージに対して，オペレーターが読むための情報を表示する。オペレーターへのメッセージ・テキストは，ABORT メッセージ中に含まれるステータス・コードに基づく。加えて，CANCEL メッセージがすべての関係するキューに送られる。CANCEL メッセージは，STANDBY メッセージが送られているすべてのキューと，それに対して GO_2PC メッセージが送られてしまっているすべてのキューに送られる。

CANCEL メッセージの全体への送信は，現在アクティブなすべての被制御装置に対し，異常があることやショーのキュー・シーケンスが特別の回復フェイズに入ろうとしていることを知らせる。この情報は重要である。例えば，通常はシーケンス外のキューの実行を許可しない装置が，CANCEL メッセージの受信後はそれを許可する場合があるかもしれない。

時として ABORT メッセージは，特殊な回復フェイズの開始を示さない。例えば，セクション 6.9.1 のエレクトリック・アイ・コントロール装置によって送られた ABORT メッセージは，期待されたイベントがまだ起こっていないことを示すだけである。このような場合，制御装置は ABORT メッセージの受信に伴って，CANCEL メッセージを全体へ送信することはない。

問題とされる動作がキャンセルされた後，その動作が次に取るべき方向としては，いくつかの可能性がある。最も単純な可能性は，被制御装置を受け持つオペレーターの人が介入して，可能な動作をさせることである。もっと高度な制御装置では，ABORT（中断）状態が報告された場合，自動的に（あるいは手動で）稼動される非常事態用のキューを作動させるかもしれない。

2フェイズ・コミット・プロトコルの重要なエラー処理

1. **異常な状態の存在と，何かその状態の特質についてのこと（ステータス・コードによって）を知らせる ABORT メッセージの提供**
2. **関与するすべての被制御装置に，誤りのないキュー実行シーケンスからの逸脱を通知する CANCEL メッセージの使用**
3. **あらかじめ定義されたステータス・コードの大きなセットの値。これらの定義は，制御装置が，その問題からの回復の責任を最も持っているであろう制御装置のオペレーターに対して，その問題の情報を表示する。**

制御装置からの CANCEL メッセージ送信は，STANDBY メッセージによって準備されているキューと，すでに GO_2PC メッセージを受けて実行中のキューに向けらる場合がある。CANCEL メッセージ中で指定されたキューが，STANDBY メッセージによって準備されたものの場合，被制御装置は，STANDBY メッセージを受信したことを単純に忘れる。しかし，CANCEL メッセージがすでに実行中のキューに対する場合は，事態ははるかに複雑である。

キューの実行はすでに進行中なので，取りうるシャット・ダウンの選択肢の数は劇的に増加する。なすべき最も確実な正しいことを選ぶのは，はるかに困難になる。結局のところ，2フェイズ･コミットで被制御装置の設計者は，すでにその実行が開始されているキューの，完了，ポーズ，終結，そしてリバースのどれを選べばよいのかを，慎重に熟慮しなければならない。さらに，動作をキャンセルするかどうかを，装置のオペレーターの手に委ねるということも検討されるべきである。

現在実行中のキューが，何かを安全なところに動かしているのであれば，おそらくそのキューを完了させることが正しい選択である。現在実行中のキューが，何かはっきりしない状態へ動かしているのなら，おそらくそのキューは中断，終了，あるいはリバースされるべきである。3つの中でどれが最良なのかは，機器とパフォーマンス状況の詳細によって左右される。

動作中のキューのキャンセルが，機械的にどのように取り扱われるかは，2フェイズ･コミットの方法論の重要な安全構成要素である。それぞれの状況は異なっているので，絶対的な必要条件をこの規格の中に割り当てることは不可能である。この規格にできることは，可能な限り多くの選択肢を提供することだけである。それにもまして，装置の設計者は，動作中のキューのキャンセルには特別な注意を払わなければならない。

6.4.4　メッセージの再試行をともなう異常な状態の取り扱い

ある状態下では，制御装置はオペレーターの操作を求める代わりに，異常という結果に終わったメッセージの送信をリトライ（再試行）することを選ぶ場合もある。メッセージのリトライが有効であろう最もよくあるケースは，“checksum error（チェックサム･エラー）”ステータス･コードを伴う ABORT メッセージである。再送信されたメッセージが正しく届く見込みは十分にある。他の状態でも，再送信によって回復可能な場合もある。制御装置の設計者は，自分が適切であると考えるなら，有効と思われる回数だけリトライすることを選択

してかまわない。しかし，リトライが実行される場合は次のようになされなければならない。

いかなる状況においても被制御装置は，メッセージの再送信を異常な状態の回復戦術としてはならない。

制御装置は，ABORT メッセージで返されたシーケンス番号で送ったメッセージを再送信することで，ABORT メッセージに対してリトライをしてかまわない。制御装置は，論理的に考えてそのような変更によって，被制御装置による処理の成功の可能性が向上するならば，再送信の際にそのメッセージを改めてもかまわない。例えば，制御装置は不明の <Q_list> ステータス・コードを伴う ABORT（中断）メッセージに対する再送信を試みるかもしれない。この再送信のロジックには，再送信されるメッセージから <Q_list> データをはずすことが含まれている。

制御装置はまた，“タイムアウト（時間切れ）”エラーからの回復として，あるいは ABORT（中断）メッセージとしての“チェックサム・エラー”（セクション 6.5 参照）を伴う到着中のメッセージの取り扱いの一部として，再送信を使用してもかまわない。しかしながら，制御装置には，被制御装置に最後のメッセージの再送信を依頼する方法はない。したがって，制御装置は被制御装置のキューをクリアし，再設定を行わなければならない。

制御装置はこれを，被制御装置からの STANDING_BY メッセージによって知り，それに対してまだ GO_2PC が送られていないあらゆるキューについて，CANCEL メッセージを送ることによって行う。さらに，制御装置は，たった今キャンセルされたすべてのキューに対して STANDBY メッセージを再び送る。この再送信のプロセス中に送られるメッセージのどれであっても，再びエラーを引き起こした場合には，再送信のプロセスは失敗と考えなければならず，オペレーターによる操作が必要になる。

“被制御装置からの悪いメッセージ”による再試行アルゴリズムでは，すでに GO_2PC が送られているキューをクリアすることはできない。そうすることは，キューの実行を停止させることを意味するだろう。これは，オペレーターの操作によらなければならない。

6.4.5. マニュアル・オーバーライド [手動優先] プロセッシング [処理]

被制御装置は，ローカル・オペレーターが手動でキュー動作を開始した時には，すべての2フェイズ・コミット MIDI ショー・コントロール・メッセージを無視するように設計されるであろう（“ローカル・オペレーター”についてはセクション 6.2 に記述されている）。この状態は，発効中には “manual override in progress” ステータス・コードによって示される。この方式で設計されている被制御装置は，すべての MSC_2PC メッセージを無視しなければならない。あるメッセージを選択的に無視して他のメッセージを無視しないと，制御装置のシステム故障が引き起こされる可能性がある。

マニュアル（手動優先）モードでは，すべての STANDBY と GO_2PC メッセージに対し，“manual override in progress（手動優先進行中）”ステータス・コードを含む ABORT メッセージによって応答しなければならない。すべての CANCEL メッセージに対しても，“manual override in progress（手動優先進行中）”ステータス・コードを含む CANCELLED メッセージによって応答しなければならない。これは，被制御装置を操作しているオペレーターがマニュアル・オーバーライド（手動優先）状態を解除するまで続く。

オペレーターが，GO_2PC メッセージによって開始されたキューの実行中にマニュアル・モードにした場合，“manual override initiated（手動優先が開始される）” ステータスを含む ABORT メッセージが，オペレーターの操作時に送られるべきである。ABORT メッセージを制御装置に送っておかないと，キューの実行をタイム・アウト（時間切れ）にしてしまうかもしれない。“manual override initiated” ステータスは，被制御装置がマニュアル動作モードへ変更されたことを示す。これは，進行中の状態の継続を示す “manual override in progress（手動優先進行中）”とは異なっている。制御装置は，そのオペレーターに適切に通知すべきである。

6.4.6. メッセージの待ち受け

2フェイズ・コミット MIDI ショー・コントロールのプロトコルは，被制御装置がメッセージを待ち受けることのないように設計される。被制御装置は，単にメッセージを受け取り，そのメッセージが要求する動作を実行し，その応答メッセージを送り返す。被制御装置は，

GO_2PC メッセージを正しく処理するために，最近の STANDBY と STANDING_BY のやり取りを記憶しておく必要がある。けれども厳密に言えば，被制御装置は，前もって STANDBY メッセージを受信しているので，GO_2PC メッセージを待ち続けているわけではない。

待ち受けのない2フェイズ・コミットによって，被制御装置が準備に入ることができるという利点は，特筆に値する。被制御装置は，決してメッセージを待ち受けしないので，常にオペレーターからの手動操作を受け入れることができる。したがって，万一制御装置が機能しなくなったとしても，個々の装置をその場所で手動操作すれば，パフォーマンスを継続できる（手動については 6.4.5 に記述）。

制御装置は，被制御装置からの応答（STANDING_BY，COMPLETE，CANCELLED メッセージ）を待ち受けなければならないが，次の2つの処理をすれば，待ち受けが制御装置をハングさせる（入力に対して応答不能にする）ことはない。

1. 待ち受けるべき時間は，限られている。そのリミットは，STANDING_BY と CANCELLED メッセージでは2秒である。また，COMPLETE メッセージの待ち受け時間のリミットは可変である（先行する STANDING_BY メッセージの内容に基づく）。

2. 制御装置は，送信順が不確定なたくさんのメッセージを同時に待ち受けなければならない。このため制御装置は，リミットの範囲内で応答を待っているメッセージを追跡し，そして応答受信の失敗を見つけ出す，何らかのアルゴリズムを持たなければならない。

6.5 チェックサム

それぞれの2フェイズ・コミット・メッセージは，2バイト（MIDI の 14 データ・ビットにおいて 16 ビット）のチェックサムを含んでいる。チェックサムを計算するために，メッセージのコマンド・フォーマット，コマンド，データ部分が2バイト値の配列として扱われる。メッセージのコマンド・フォーマット，コマンド，データ部分が奇数バイトであれば，チェックサムの計算のために，値ゼロの追加バイトがメッセージに付加される。チェックサムを計算する前に，チェックサム自身をゼロにしておく。

2バイト値の配列における合計が計算される。オーバーフローは無視される。次に，<device_ID> バイトが合計に付け足される。最後に，その合計が一定不変の 7F 7F（10 進で 32,639）に付け足される。この論理的操作は，チェックサムの値を MIDI における送信に適したものとする。結果として生じた値が，そのメッセージのチェックサムである。

メッセージを受信したらチェックサムを再計算し，それを受信したチェックサムの値と比較することによって，受信したメッセージが正しいかを検証する。被制御装置でチェックサムの比較が一致しなかった場合，“checksum error（チェックサム・エラー）” ステータス・コードを伴う ABORT メッセージを返す。制御装置でチェックサムの比較が失敗した場合には，それはあたかも ABORT（中断）メッセージが受信されたかのように続行する。このことは，セクション 6.4.4. で記述されたように，メッセージ送信の再試行を意味する場合もある。

6.6 シーケンス番号

シーケンス番号は，14 ビット（2MIDI バイト）の符号ナシのバイナリィである。STANDBY，GO_2PC 及び CANCEL メッセージは，1～16,383 のシーケンス番号を持つ（シーケンス番号のゼロは，将来の拡張のために予約されている）。被制御装置は，その STANDING_BY，COMPLETE，CANCELLED，あるいは ABORT メッセージが指定するキューを受信したことの確認のために，STANDBY，GO_2PC あるいは CANCEL メッセージによって，受信したシーケンス番号を返す。

オプションだが，被制御装置はシーケンス番号に加えて，STANDING_BY と COMPLETE メッセージ中に，キュー番号情報を含んでもかまわない。しかし，この情報はそのメッセージの安全チェックとしてのみ使用される。

このようにシーケンス番号を使用することで，2フェイズ・コミット処理は，とても単純な装置（安全装置センサーのような）にも適応することができる。例えば，ガス・ディテクターはキューの概念をまったく持っていないかもしれない。STANDBY メッセージが受信された時はいつも，それは単にガスをチェックする。ガスが発見されない場合には STANDING_BY メッセージで応答し，ガスが発見された場合には ABORT メッセージで応答する。それは受信したメッセージ中の <Q_number> 等のフィールドは無視し，到着するシーケンス番号を応答メッセージにただ単にコピーする。

さらに，シーケンス番号は，制御装置における応答の追跡処理を簡素化する。

6.7　ステータス・コード

ステータス・コードは，ABORT と CANCEL メッセージ中に現れる。ステータス・コードは，2バイトの符号ナシのバイナリィである。ステータス・コードが送信される際,MIDI に合うように,ステータス・コードの値の下位の2ビットは常にゼロでなければならない。したがって，ステータス・コードの最小値は4，最大値は FF FCH（10 進の 65,532）である。MIDI メッセージの記述において，ステータス・コードは s1 s2 と表示している。符号ナシ整数値を用いるので，ステータス・コードを s1 と s2 の値に変換する方法と，それを戻す方法は，以下の通りである。

```
s1 = (status_code/4)&7F
s2 = (status_code/512)&7F
status_code = (s1 × 4)+(s2 × 512)
```

それぞれのステータス・コードの値は，エラーの状態，またはキャンセルされたキューの状態を表している。ステータス・コードには，3つの範囲がある。最初の範囲は，すべてのコマンド・フォーマットの値に共通であり（コマンド・フォーマットの値ついてはセクション 4.1 を参照のこと），ここのステータス・コードは，すべてのタイプの MIDI ショー・コントロールに当てはまる。CANCELLED メッセージ中で返されるステータス・コードは，すべて最初のステータス・コードの範囲に入る。

2番目のステータス・コード範囲は，ABORT メッセージ中に現れるコマンド・フォーマットの値として使用される。これらのステータス・コードの意味は，装置のタイプによって異なる。例えば，ステータス・コード “10 08” は，プロセス・コントロール系の装置（コマンド・フォーマット 50 ～ 5F）が受信した場合には “water low（水位低下）” を意味するが，サウンド系の装置（コマンド・フォーマット 10 ～ 1F）が受信した場合，“10 08 ” は “amplifier failure（アンプの故障）” を意味する。

3つ目のステータス・コード範囲では，コマンド・フォーマットとメーカーの両方によって決められるものである。

これらのステータス・コードの正確な意味は，それを送った装置のタイプとメーカーの両方によって決まる。各メーカーは，自社の装置で使用する3つ目のステータス・コード範囲に設定したすべてのステータス・コードについて，情報を公表しなければならない。

以下の表は，ステータス・コードの範囲を要約したものである。

範囲	内容
00 04 -- 0F FC	コマンド・フォーマットとメーカーによって決められるステータス・コード（値は 1023 まで）
10 00 -- 7F FC	コマンド・フォーマットによって決められるステータス・コード（値は 7,168 まで）
80 00 -- FF FC	コマンド・フォーマットから独立したステータス・コード（値は 8,192 まで）
00 00	未定義のステータス・コード（不明のエラー状態）

＜備考＞

ステータス・コードは，符号付きの値として扱われた場合，マイナスの値なので，コマンド・フォーマットの中から容易に発見できる。また，ゼロのステータス・コードは不明のエラー状態や他のステータス・コードの値がどれも当てはまらないエラー状態を示すためにリザーブされている。

メーカーが決めらる範囲のステータス・コードは，各メーカーが自らがふさわしいと思う方法で自由に使用できる。しかし，最大の互換性が得られるのは，メーカーによって決められたステータス・コードが無い場合であろう。またメーカーは，ステータス・コードの範囲内にステータス・コードの値を定義する方法が用意されている。

以下の表は，コマンド・フォーマットから独立しているステータス・コードを記載したものである。メッセージの欄には，どのメッセージがそのステータスコードを含むレスポンスを生成できるのかを示す（S=STANDBY,G=GO_2PC,そして C=CANCEL）。最初の表は，CANCELLED メッセージに関するものである。

CANCELLED メッセージのステータス・コード

16 進	メッセージ	内容
80 04	--C	completing（完了中）
80 08	--C	paused（ポーズ）
80 0C	--C	terminated（終了）
80 10	--C	reversed（リバース）
80 24	--C	not standing by（非準備中）
80 28	--C	manual override in progress（手動優先進行中）

次のステータス・コード表は，ABORT メッセージに関するものである。

＜備考＞

CANCEL メッセージに対する ABORT メッセージに含めることができるステータス・コードは，“unknown/un-defined error（不明 / 不定義のエラー）”と“checksum error（チェックサム・エラー）”のみである。“timeout（タイム・アウト）”ステータス・コードは，実際のメッセージ送信では出現しないが，制御装置の内部設計を単純化するために用意された。他のキャンセル状態はすべて，CANCELLED メッセージで送られる。

ABORT メッセージのステータス・コード

16 進	メッセージ	内容
00 00	SGC	unknown/undefined error（不明 / 不定義エラー）
80 00	SGC	checksum error（チェックサム・エラー）
80 20	sgc	timeout（タイム・アウト（時間切れ））※
80 24	-G-	not standing by（非準備中）
80 28	SG-	manual override initiated（手動優先が開始される）
80 30	-G-	manual override in progress（手動優先進行中）
80 40	SG-	deadman interlock not established（自動安全連動装置が設置されていない）
80 44	SG-	required safety interlock not established（必要とされる安全装置がない）
80 50	S--	unknown <Q_number>
80 54	S--	unknown <Q_list>
80 58	S--	unknown <Q_path>
80 5C	SG-	too many cues active（有効なキューが過剰）
80 60	S--	cue out of sequence（シーケンス外のキュー）
80 64	SG-	invalid d1 cue data value（無効な d1 キュー・データの値）
80 68	SG-	invalid d2 cue data value（無効な d2 キュー・データの値）
80 6C	SG-	invalid d3 cue data value（無効な d3 キュー・データの値）
80 70	SG-	invalid d4 cue data value（無効な d4 キュー・データの値）
80 90	S--	manual cueing of playback medium required（手動キューイングが必要）
80 A0	SG-	power failure in controlled device subsystem（装置の停電）
80 B0	SG-	reading new show cues from disk（新しいショー・キューを読み込み中）

※制御装置内で使用し，送信されない。

ステータス・コードが存在するからといって，それを使用しなければいけないわけではない。ステータス・コードの使用は，セクション5における詳細コマンドとデータの記述，あるいはセクション6～ 6.6 における2フェイズ・コミット全般での解説において指定されている場合にのみ必要である。ここにステータス・コードの値を記載したのは，装置の設計者がこのプロトコルを実現する場合に，選ぶことができるように用意したものだ。またこのリストは，制御装置を操作する人に文字として情報を与える際のリファレンスとなる。

例えば，装置によっては“cue out of sequence（シーケンス外のキュー）”ステータス・コードを使用しないという場合もある。あるいは別の装置では，それによって特別なキュー・セットを実行するという場合もある。そのような被制御装置は，キューの連続についてのルールが破られた場合，“cue out of sequence（シーケンス外のキュー）”ステータス・コードを返すだけだろう。

“manual override in progress（手動優先進行中）”ステータス・コードを使用する際は，特別な注意を払わなければならない。オペレーターが手動でキュー動作を開始した時，被制御装置はすべての2フェイズ・コミット MIDI ショー・コントロール・メッセージを無視するようにしなければならない。この状態は，“manual override in progress（手動優先進行中）”ステータス・コードによって示される。被制御装置があるメッセージだけを選択的に無視するようなことをすると，制御装置のシステム故障を引き起こす可能性がある。手動優先の処理について詳細は，セクション 6.4.5 を参照。

次の表は，コマンド・フォーマットの値ごとのすべての制御装置のメーカーにあてはまるステータス・コードを記載している。メッセージ欄は，どのメッセージがそのステータス・コードを含むレスポンスを生成できるのかを示す（S=STANDBY，そして G=GO_2PC）。

16 進表示	メッセージ	内容
01 と 0F の間のコマンド・フォーマット用（照明）		
10 04	SG	position motor failure（ポジション・モーターの故障）
10 08	SG	scroller motor failure（スクロール・モーターの故障）
10 0C	SG	strobe not charged（ストロボがチャージされていない）
10 10	SG	laser safety interlock not established（レーザーの安全連動装置がない）
10 と 1F の間のコマンド・フォーマット用（サウンド）		
10 04	SG	amplifier failure（アンプの故障）
10 08	SG	amplifier overload（アンプのオーバーロード）
20 と 2F の間のコマンド・フォーマット用（機械装置）		
10 04	SG	motor failure（モーターの故障）
10 08	SG	（scroller）limit switch inhibiting movement（制限スイッチが動きを妨げている）
10 0C	SG	unequal movement in multiple section system(複数セクションの不均衡な動き）
10 10	SG	servo failure（サーボの故障）
30 と 3F の間のコマンド・フォーマット用（ビデオ）		
10 04	SG	sync lost（シンクが失われた）
10 08	SG	time code lost（タイムコードが失われた）
40 と 4F の間のコマンド・フォーマット用（プロジェクション）		
10 04	SG	film tension lost（フィルムのテンションが失われた）
10 08	SG	lamp failure（ランプの故障）
50 と 5F の間のコマンド・フォーマット用（プロセス・コントロール）		
10 04	SG	hydraulic oil low（水圧オイル低下）
10 08	SG	water low（水位低下）
10 0C	SG	carbon dioxide low（二酸化炭素（炭酸ガス）低下）
10 10	SG	excess gas detected（過剰なガス（気体）が発見された）
10 14	SG	gas pilot out（ガスの種火が消えている）
10 18	SG	improper gas ignition conditions(windy)（適切でないガス点火条件（強風））
10 1C	SG	smoke/fog fluid low（スモーク（煙）/ フォッグ（霧）流動体低下）
11 04	SG	invalid switch number（無効なスイッチ番号）
11 08	SG	latch setting system inoperative（ラッチ・システムが稼動していない）
12 04	SG	burned out cue light（キュー・ライトの焼失）
60 と 6F の間のコマンド・フォーマット用（花火（発煙筒））		
10 04	SG	charge not loaded（チャージされていない）
10 08	SG	atmospheric conditions prohibit discharge（コンディションにより発射を禁止）

6.8. キュー・データの値(d1, d2, d3, d4)

STANDBY と GO_2PC メッセージは，両方とも4つの7ビットのキュー・データを含んでいる。これらのデータの値は，被制御装置によって実際に使用されるか否かにかかわらず，すべての STANDBY と GO_2PC メッセージ中に存在していなければならない。正しい d1 ～ d4 の値が不明な場合には，STANDBY あるいは GO_2PC メッセージ中の d1 ～ d4 の値は，ゼロでなければならない。

被制御装置は，d1, d2, d3, d4 の値によって，キューが実行されるべき方法についての情報が制御装置側に多少なりあることがわかる。これらの値がどのように使用されるかは，このセクションで後述している。d1 ～ d4 の値は，キュー番号の付け方やその識別に関係するものではない。したがって，d1 ～ d4 の値が異なる2つのキューを被制御装置が同時に実行するような場合，それらのキューには異なるキュー番号が与えられる。

d1 ～ d4 の値の扱い方は，それぞれの装置の設計者によって決められるが，以下のどちらかでなければならない。

1) セクション 6.8.1 から 6.8.2 までの中に記述された共通使用方法フォームのひとつに従う。
2) 被制御装置についてのドキュメンテーション中に明確に記述する。

選択は1)の方が望ましい。共通使用方法フォームは，被制御装置の多くのメーカーにとって適切なものとするために，必要に応じて更新されるだろう。

被制御装置が，正しくない d1, d2, d3, d4 の値を受信した場合は，“invalid dn cue data value(無効な dn キュー・データの値)”ステータス・コードを含む ABORT メッセージで応答しなければならない。したがって，d1 の値が正しくなかったら，“invalid d1 cue data value(無効な d1 キュー・データの値)”ステータス・コードを含む ABORT メッセージを返す。

d1, d2, d3, d4 の値を使用しない装置は，使用しない値についての正当性をチェックする必要はない。例えば，被制御装置が d2 と d3 を使用している(しかし d1 と d4 は使用しない)としよう。その装置は，受信するすべての d2 と d3 の値の正当性をチェックしなければならない。しかし，d1 と d4 の値は無視する。d1 ～ d4 までの値をどれも使用しない装置は，それらをすべて無視しなければならない。

エラーが d1 ～ d4 の値のどれかひとつというわけではない場合，ABORT メッセージは，その最も低いナンバーに付けられたデータのステータスコードを含めなければならない。d3 と d4 の値が正しくなかった場合，“invalid d3 cue data value(無効な d3 キュー・データの値)”ステータス・コードを含む ABORT メッセージが送られる。

6.8.1 GO-TO レベルの照明卓の d1 と d2 の使用法

“Go On, Go Off”で作動する照明卓は，gl を“Go Level”として使用する(ここでは gl=d1+(d2*128))。gl=255 である STANDBY と GO_2PC のシーケンスは，Go On に相当する。gl=0 である STANDBY と GO_2PC のシーケンスは，Go Off に相当する。gl は，0と 255 の間の値を取ることもできる。それらは，指定されたキューが gl で設定されたレベルまで GO するべきであることを示す。

例えば gl=128 は,Go Cue To 50% に相当する。0 から 255 の範囲ではない gl の値はどれも現時点では認められていない。“invalid d1 cue data value(無効な d1 キュー・データの値)”ステータス・コードを含む ABORT メッセージで応答しなければならない。256 とそれ以上の gl の値は，将来における拡張のためにリザーブされている。

6.8.2　マルチ・スイッチ＆キュー・ライトの d1，d2，d3 の使用法

マルチ･スイッチのテストとセット･ボックスは，sn をスイッチ番号として使用する（ここでは sn=d1+(d2*128)）。さらにこれらのボックスは，d3 をスイッチ・タイプとして使用する。

既知のスイッチ・タイプの値は：

0	リザーブされている
1	クローズ・スイッチの番号が付けられた sn
2	オープン・スイッチの番号が付けられた sn
3	クローズされたスイッチ番号 sn のテスト （オープンの場合には “deadman interlock not established” を伴い，ABORT）
4	オープンのスイッチ番号 sn のテスト （クローズされている場合には “deadman interlock not established” を伴い，ABORT）
10	ラッチ番号が付けられた sn をリセット
11	ラッチ番号が付けられた sn をテスト （ラッチされていない場合には “deadman interlock not established” を伴い，ABORT）
20	キュー・ライト番号が付けられた sn をオペレート

セクション 6.9 の例におけるエレクトリック・アイは，これらのスイッチ・ボックスのひとつの番号が付けられたラッチを使用してテストされるだろう。しかしその例は，特別で独特なハードウェアを想定している。

6.9　2フェイズ・コミットの処理例

次のセクションは，2フェイズ・コミットにおける処理の方法である。最初に，基本的なエラーのないメッセージのやり取りが記述されている。それから，いくつかのエラー状態と，その処理方法を述べる。

このセクションは，これに先立って記述された2フェイズ･コミット･プロトコルを利用する人に，ガイダンスを提供することを意図している。したがってこれは，厳密にはプロトコル定義の一部ではない。このセクションにある記述は，単なる「例」にすぎない。前述された2フェイズ・コミット・プロトコルを利用する際，もっとよい方法があれば，それを使用するのは自由である。

以下の例は，あらかじめ設定されたあるキューに基づくものである。このキューには，動力で動くターンテーブルの動作が含まれるとしよう。キューの開始時に舞台の奥にあるセットを表に出すために，180 度回転しなければならないが，ターンテーブルが 180 度回転するには 30 秒かかる。ターンテーブル上にあるセット舞台の前面には，動力で動くフライ・システムによって持ち上げられる吊り下ろしがある。ターンテーブルとフライ・システムは，両方とも MIDI の被制御装置として操作されているとしよう。それらは両方とも独立したオペレーター卓を有している。

キューが開始可能となる前に，女優はターンテーブルの外に出なくてはならない。どちらの装置のオペレーターも，外に出るのを見ることができない状況である。それゆえこのシステムは，女優が外に出る順路にライト・ビームを遮るように設計されたエレクトリック・アイ装置を設置した。フライやターンテーブルが動き始める前に，エレクトリック・アイは女優が外に出たことを感知しておかなければならない。

もちろん，照明とサウンドのキューもある。ひとつのサウンド・キューは，女優がエレクトリック・アイ・ビームを遮った時に始まり，2番目のサウンド・キューは，ターンテーブルがその回転の半分（90 度）に達した時に始まる。ライト・キューのひとつは，フライが上昇を開始したときに始まり，別のライト・キューはターンテーブルが回転を開始したときに始まり，最後のキューはターンテーブルがその 180 度の回転を完了した時に始まる。

これらの操作は，すべて制御装置である MIDI ショー・コントロール・コンピューターによって進行される。各機能は，きっかけとなるイベントが制御装置の MIDI ショー・コントロール・コンピューターが感知できるようにいくつかの構成要素に分割される。例えば，ターンテーブルの回転は，単一の 180 度の回転ではなく，ふたつの 90 度の回転に分割される。

以下の表は，前述のキューの詳細なキュー・リスト例である。

2フェイズ・コミットにおけるキューの例

制御装置	キュー	記述
エレクトリック・アイ	EE-6	女優がエレクトリック・アイ・ビームを遮る
サウンド	S-109	最初のサウンド・キュー
フライ	F-28	ライン 12 を 10 フィートに上げる（クリア・セット）
ライト	L-118	最初のライト・キュー
ターンテーブル	TT-34	90 度回転
ライト	L-118.1	第2のライト・キュー
フライ	F-28.1	ライン 12 を上限まで上げる
ターンテーブル	TT-34.1	90 度回転
サウンド	S-110	第2のサウンド・キュー
ライト	L-119	第3のライト・キュー

このキューが起こるシーケンスは：

1. EE-6 が起こるのを待つ
2. S-109 を GO
3. 3秒待つ（女優にはける時間を与えるため）
4. F-28 と L-118 を GO
5. F-28 の完了を待つ
6. F-28.1，TT-34，そして L-118.1 を GO
7. TT-34 の完了を待つ
8. TT-34.1 と S-110 を GO
9. TT-34.1 の完了を待つ
10. L-119 を GO

L-119，TT-34.1 といった名称を使用したのは，便宜のためである。MIDI メッセージ中で送信されるキュー番号は，それらを含まないだろう。例えば，TT-34.1 の MIDI メッセージは，<command_format>=24 と <Q_number>=34.1 と表され，そして L-119 は，<command_format>=01 と <Q_number>=119 と表されるだろう。

このシーケンスは，概念的には大きなテーマ･パークの自動化されたパフォーマンス･アトラクションに基づいている。しかし，2フェイズ･コミット MIDI ショー・コントロールの機能を説明するために，変更が加えられている。このシーケンスが非現実的なものに見えたとしたら，これは実際の状況ではなく，MIDI ショー・コントロール2フェイズ・コミットを説明するものと思っていただきたい。

6.9.1 基本的なメッセージのやり取り

さて，上述の基本的なキュー・シーケンスが2フェイズ・コミット MIDI ショー・コントロール・メッセージでどのように表現されるのか考えてみよう。ここでは，エラー状態はいっさい起こらないものとする。イベントのシーケンスは，次の表のようになる。

時間は表の上から下へと経過する。表中の最初の欄は秒で表示されている。この欄の数字は，そのイベントが起こるおおよその時間を，1/1000 秒単位で表している。

エラーのない2フェイズ・コミットのキューの実行例

秒	コマンド	フォーマット	キュー	シーケンス	ステータス	備考（下を参照）
00.000	STANDBY	10(S)	109	001		A
00.001	STANDBY	22(F)	28	002		
00.002	STANDBY	01(L)	118	003		
00.003	STANDBY	24(TT)	34	004		
00.004	STANDBY	01(L)	118.1	005		
00.005	STANDBY	22(F)	28.1	006		
00.010	STANDING_BY			001		B
00.011	STANDING_BY			002		
00.012	STANDING_BY			006		
00.013	STANDING_BY			003		
00.014	STANDING_BY			004		
00.015	STANDING_BY			005		
05.000	STANDBY	5F(EE)	6	007		
05.010	ABORT 007				80 40	C
05.500	STANDBY	5F(EE)	6	008		
05.510	ABORT 008				80 40	C
06.000	STANDBY	5F(EE)	6	009		
06.010	STANDING_BY			009		D
06.020	GO_2PC	10(S)	109	010		
07.500	COMPLETE			010		
09.000	GO_2PC	22(F)	28	011		E
09.001	GO_2PC	01(L)	118	012		
09.500	GO_2PC	22(F)	28.1	013		F
10.000	COMPLETE			012		
11.000	COMPLETE			011		
11.001	GO_2PC	24(TT)	34	014		G
11.002	GO_2PC	01(L)	118.1	015		
11.003	STANDBY	24(TT)	34.1	016		
11.004	STANDBY	10(S)	110	017		
11.005	STANDBY	01(L)	119	018		
11.015	STANDING_BY		016			
11.016	STANDING_BY		017			
11.017	STANDING_BY		018			
16.000	COMPLETE			015		
21.000	COMPLETE			013		H
24.000	GO_2PC	24(TT)	34.1	019		I
26.000	COMPLETE			014		J
26.001	GO_2PC	10(S)	110	020		
34.000	COMPLETE			020		
41.000	COMPLETE			019		K
41.001	GO_2PC	01(L)	119	021		
44.000	COMPLETE			021		

< 備考 >

A. ターンテーブルが 90 度の回転を完了する前に GO する,すべてのキューに STANDBY メッセージを送る。これは, あらゆるキューの実行前に, 2秒のタイム・アウト時間を完全に取ることができるよう, 十分に早く行わなければならない。

B. STANDING_BY メッセージが受信される。STANDING_BY メッセージが適用されるキューを識別するのに,シーケンス番号のみが使用される。また, STANDING_BY メッセージは, STANDBY メッセージが送られたのと同じ順で送られてこない場合もある。この例においてフライ装置は, 他のメッセージによって隔てられている STANDBY メッセージを受け取ったにもかかわらず, ふたつの STANDING_BY メッセージをたて続けに受け

取っている。

C. 女優がエレクトリック・アイ・ビームを遮っていないので，エレクトリック・アイの STANDBY は ABORT。

D. エレクトリック・アイの STANDING_BY メッセージは，女優がエレクトリック・アイ・ビームを遮ったことを示す。（厳密な MIDI 2フェイズ・コミットを完全に行うためには，EE-6 に CANCEL を送らなければならない。しかし，これは被制御装置が必要としていないので，不要である。）

E. 秒の欄に注目してほしい。06.010 における STANDING_BY の受信と，09.000 におけるフライとライトのキューの開始の間には，おおよそ3秒のディレイがある。

F. フライ・キュー 28.1 の GO_2PC メッセージは，キュー 28 の COMPLETE メッセージが受信される前に送られる。フライの制御装置はこれを，ライン 12 はキュー 28 の COMPLETE メッセージが送られる間も動き続けるべきだということを指示するものだと解釈する。

G. フライ・キュー 28 が COMPLETE を報告したら，ターンテーブル・キュー 34 とライトのキュー 118.1 が開始されうる。この時点で，残っているキューの STANDBY メッセージのやり取りも行われる。上の備考Aで述べられたのと同じタイム・アウトに対する配慮がここでも適用される。

H. この時点で，ライン 12 はフライ完了する。

I. ターンテーブル・キュー 34.1 の GO_2PC メッセージは，キュー 34 の COMPLETE メッセージが受信される前に送られる。フライ装置のように，ターンテーブル装置はこれを，キュー 34 の COMPLETE メッセージが送られる間も動きは続けるべきだということを指示するものと解釈する。

J. ターンテーブル・キュー 34 が COMPLETE を報告したら，サウンド・キュー 110 が開始されうる。

K. ターンテーブル・キュー 34.1 が COMPLETE を報告したら，ライト・キュー 119 が開始されうる。

6.9.2 早期に発見されたエラー状態

ここでは，6.9.1 で記述された基本的なキュー・シーケンスに，いくらかのエラー状態を想定してみよう。第1に，キュー・シーケンス全体の実行を妨げるエラー状態を考えていただきたい。フライ・システムをコントロールしているコンピューターが，ライン 12 のウインチ・モーターのひとつに問題を発見したとしよう。この時，STANDING_BY メッセージの代わりに ABORT メッセージが返される。制御装置は，前もって STANDBY メッセージを送ったすべてのキューに対して，CANCEL メッセージを送ることになるだろう。

メッセージ・シーケンスは次のようになる：

秒	コマンド	フォーマット	キュー	シーケンス	ステータス	備考（下を参照）
00.000	STANDBY	10(S)	109	001		
00.001	STANDBY	22(F)	28	002		
00.002	STANDBY	01(L)	118	003		
00.003	STANDBY	24(TT)	34	004		
00.004	STANDBY	01(L)	118.1	005		
00.005	STANDBY	22(F)	28.1	006		
00.010	STANDING_BY			001		
00.011	ABORT			002	10 04	A
00.012	CANCEL	10(S)	109	007		B
00.013	ABORT			006	10 04	
00.014	CANCEL	01(L)	118	008		
00.015	STANDING_BY			003		
00.016	CANCEL	24(TT)	34	009		
00.017	CANCEL	01(L)	118.1	010		
00.018	CANCEL	22(F)	28.1	011		
00.030	CANCELLED			007	80 0C	C
00.031	CANCELLED			009	80 0C	
00.032	CANCELLED			010	80 0C	
00.033	CANCELLED			008	80 0C	
00.034	CANCELLED			011	80 24	D

< 備考 >

A. 最初のフライ・キューが “motor failure（モーターの故障）” ステータス・コードを伴う ABORT メッセージを返す。このステータスは，テキストに変換され，制御装置のオペレーターに向けて表示される。

B. すべての被制御装置の，まだ実行が完了していないすべてのキューに，CANCEL メッセージが送られる。これは，セクション 6.4 に記述された2フェイズ・コミットのエラー回復原則に従うものである。個々の被制御装置は，いまだ STANDBY メッセージを処理中なので，STANDING_BY メッセージと CANCEL メッセージが交錯する。

C. CANCELLED メッセージが到着し始める。STANDING_BY メッセージのように，CANCELLED メッセージは CANCEL メッセージが送られたのと同じ順で受信される必要はない。キャンセルされたキューは，実行を開始していないので，CANCELLED ステータスは “terminated” である。

D. 制御装置は，完璧を期するため，たとえ 00.013 秒の時点で F-28.1 への ABORT メッセージが受信されていても，制御装置はそのキューに CANCEL メッセージを送る。フライ装置は，すでに F-28.1 がスタンディング・バイだったことを “忘れて” いるため，その CANCELLED メッセージは “not standing by” ステータス・コードを含む。

この仮定された状況において，幸いなのは MIDI ショー・コントロール2フェイズ・コミット・プロトコルが動作していないフライ・システム・モーターを発見し，報告したことである。ターンテーブルの回転開始は，自動的に停止された。ターンテーブル上のセットは，回転して表に現れず，吊りものをズタズタにしないだろう。MIDI ショー・コントロールを管理していたステージ・マネージャーは，変化した状況に合わせなくてはならないし，フライ・システムのオペレーターは，不良なライン 12 のウインチ・モーターに対処するために，迅速に行動しなければならくなる。

6.9.3 キューの実行中に発見されたエラー

ひとつかそれ以上のキューの実行中に発見されたエラー状態は，基本的には前のセクションで示されたのと同じ方法で処理される。すべてのペンディング中の，あるいは実行中の動作は，CANCEL メッセージを使用して終了させられる。大きな違いは，いくつかの動作がすでに進行中なので，取りうるシャット・ダウン方法の選択肢がたくさんあることである。なすべき最も確実で正しいことを選ぶのは，はるかに困難となる。

結局のところ，2フェイズ・コミットで制御される装置の設計者は，すでにその実行が開始されているキューの完了，ポーズ，終結，リバースのどれを選べばよいのかを，慎重に熟慮しなければならない。例えば，基本的なキュー・シーケンスの例において，ターンテーブルを動かしているモーターが，ターンテーブルが 45 度回転したところで故障したとしよう。ABORT メッセージは，6.9.1 の表のおおよそ 18.000 秒の時点で送られることになるだろう。

CANCEL メッセージは，スタンディング・バイ中の2つのメッセージ，S-110 と L-119 に送られるだろう。問題は，その時点ではまだ完了していない F-28 への CANCEL メッセージの取り扱いである。ライン 12 は送り出されつつあるので，なすべき最も安全なことはおそらく F-28 を完了させること，つまり CANCELLED メッセージ中に “completing” ステータスを返し，そして最終的に COMPLETE メッセージを返すことである。しかし，ライン 12 が巻き取られている場合には，F-28 は完了させるべきではない。その状況では，キューをポーズ，終結，あるいはリバースすることが妥当な選択である。

7. MIDI マシン・コントロール 1.0

MIDI マシン・コントロール 1.0
RP013
Copyright ©1998 社団法人　音楽電子事業協会

MIDI マシン・コントロール

目　　次

1　はじめに

MIDI マシン･コントロールとは，MIDI のシステムが，先ずは従来からあるオーディオ･レコーディングやプロダクションのシステムと通信し，これらを制御することができるようにするための，汎用のプロトコルである。その応用範囲としては，1台のテープレコーダーにプレイ，ストップ，ファースト･フォワード，リワインドなどを指示できる単純なインターフェースから，オーディオやビデオのレコーダーや，ディジタル･レコーディング・システム，さらにはシーケンサーなどによる，タイムコードをベースにした大規模な同期システムとの複雑な通信にまでおよぶだろう。この MIDI マシン・コントロールのプロトコルは，将来においてもかなりの程度までの拡張が実現可能であって，多岐にわたる多くのオーディオやビジュアル，各種メディアの混合した装置群なども，こうして単一の汎用的な制御の傘の下に統合されるであろう。

コマンドとレスポンスのセットは，ES バス規格中のオーディオ・テープレコーダー部分をモデルとしている。その意図するところは，MIDI マシン・コントロールの仕様と ES バス規格とが，操作法上で同じ方針に基づくことによって，両者のコマンドが比較的素直に置き換えられるようにすることにある。他方では，このコマンド翻訳は，単なるテーブルの参照で済むものではなく，データ仕様や他の通信の詳細ではかなりの相異点が出て来るであろうことを前提としている。本質的には，MIDI マシン・コントロールは，ES バス規格に規定されているものと同じ操作セットを実行するよう設計された装置と，容易に情報交換のできることを目指している。

ES バスや他の制御プロトコルと大きく異なる点として，MIDI マシン・コントロールにおいては，制御する装置が自分の制御している装置に関して熟知していることを必要としていない。比較的簡単なアプリケーションにおいては，制御装置はその設計者にとって「手頃な」と思えた範囲のコマンド・セットを用意するのであって，制御される側の装置がそのインテリジェンスを生かして，受け取ったコマンドに対する最も適切な応答の仕方を決めるのである。同時に，他方では MIDI マシン・コントロールによれば，より複雑な制御装置も設計可能なのであって，アプリケーションの範囲は，まったくの基礎的なものから完全なプロ仕様のものにまで拡がるものと期待されているのである。

数値表現上の約束

本書においては他に断りの無い限り，すべての数値量は 16 進表現によるものとする。すべてのビット･フィールドは，最上位ビット（MSB）を先頭に置いた形式で表わすものとする。

2　全体構造

ユニバーサル・システム・エクスクルーシブ・フォーマット

MIDI マシン・コントロールは，2個のユニバーサル・リアルタイム・システム・エクスクルーシブ ID ナンバー（サブ ID#1）を用いる。ひとつはコマンド（制御する装置［コントローラー］から制御される装置への送信）用であり，もうひとつはレスポンス（制御される装置から制御する装置への送信）用である。

本書を通じて，マシン・コントロール・コマンドとマシン・コントロール・レスポンスのサブ ID#1 を表現するのに，それぞれ”mcc”と”mcr”を使用するものとする。この結果としてのリアルタイム・システム・エクスクルーシブは以下のようになる。

F0 7F <device_ID> <mcc> <commands...> F7
F0 7F <device_ID> <mcr> <responses...> F7

＜備考＞

1. 2つ以上のコマンド（またはレスポンス）を，ひとつのシステム・エクスクルーシブで送ることができる。
2. ひとつの <commands...> または <responses...> フィールドの総バイト数は，48 を越えてはならない。
3. システム・エクスクルーシブは，常にその時点で準備されている情報をすべて送信し終えしだい，速やかに F7 で閉じなくてはならない。
4. <mcc> と <mcr> の実際の値は，それぞれ 16 進の 06 と 07 である。

＜日本語版注＞

5. JMSC 版の MIDI 規格 1.0(Document Ver. 4.1)においては,システム･エクスクルーシブ･メッセージの第3項は＜チャンネル番号 channel ＞となっている。本規格は, ここを <device_ID> と読み換える最近の MMA での案に従っている。内容に変わりはない。

オープン・ループとクローズド・ループ

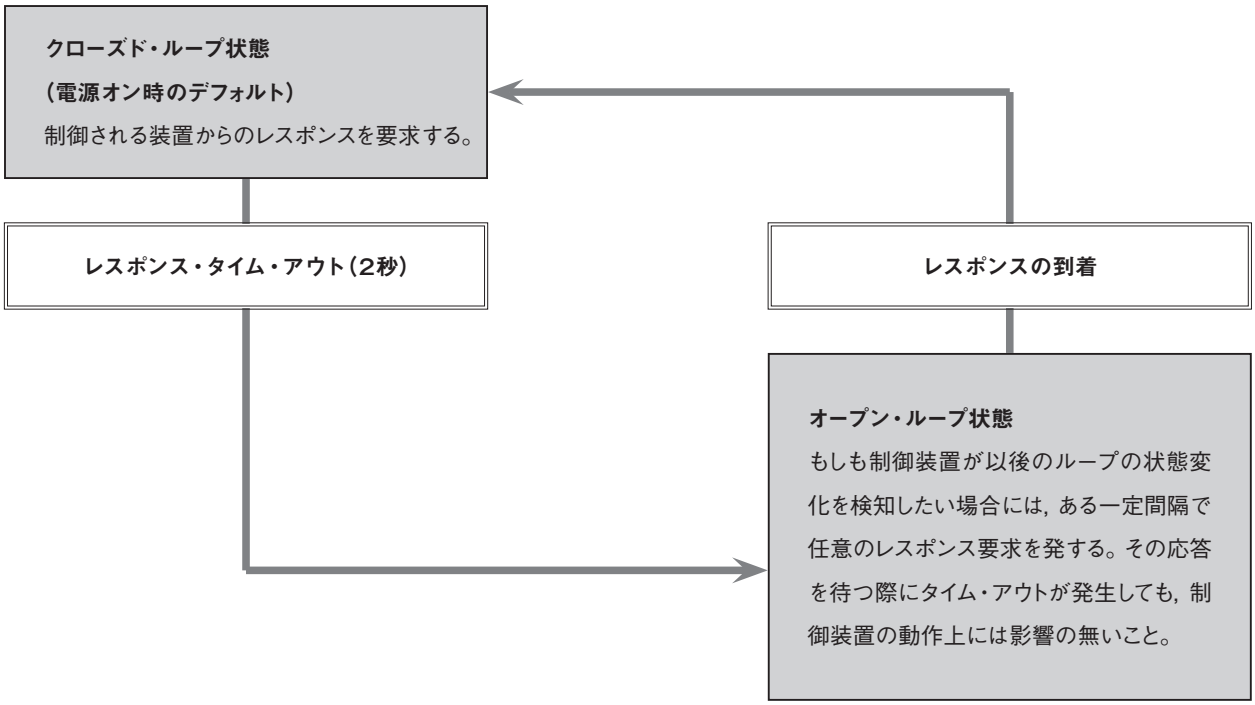
MIDI マシン・コントロールは, オープン・ループとクローズド・ループのどちらのシステムでも動作することを, 明確に意図して設計されている。

オープン・ループ： 1本のケーブルが, 制御する装置[コントローラー]の MIDI OUT と, 制御される装置の MIDI IN とを結んでいるもの。

クローズド・ループ： もう1本のケーブルが, 制御される装置の MIDI OUT から, 制御する装置の MIDI IN へ戻して結ばれており, 全二重(フル・デュプレックス)方式の通信を可能にしているもの。

コントローラーは, 電源立ち上がり時にはクローズド・ループを想定している必要がある。もしもレスポンスを期待するコマンドを発した後に, 2秒以内に応答の無かった場合には, ループはオープンであると見なすことができる。

コントローラー内でのこの2つの状態の切り替えは, 次図のように表すことができる。



ハンドシェイク

データのフロー制御は必要に応じて, 2つの簡単なメッセージで行なわれる。

WAIT ：「送信を保留せよ。当方のバッファはフル状態に近付きつつあり, 等々」

RESUME ：「送信を再開せよ。当方のバッファは再度受信可能状態となれり。」

それぞれのメッセージは,これ以外のメッセージを含まない専用のシステム･エクスクルーシブで送信しなければならない。コントローラーからのハンドシェイク・メッセージは, 全員呼び出し(all-call)アドレスに向けて送られるのに対して, 制御される装置からのハンドシェイク・レスポンスは, この装置自身の ID 番号によって他と区別できることになるであろう(次節,「装置の認識」を参照のこと)。WAIT と RESUME のメッセージで起こり得る4つのケースを次に示す。

制御する装置から制御される装置へ

F0 7F <all_call=7F><mcc><WAIT> F7

F0 7F <all_call=7F><mcc><RESUME> F7

制御される装置から制御する装置へ

F0 7F <device_ID><mcr><WAIT> F7

F0 7F <device_ID><mcr><RESUME> F7

WAIT のハンドシェイクが間違いなく行なわれるためには，それぞれの MMC 装置内では MIDI の受信バッファとして，ある程度の最小限のサイズが必要となる。付録E,「受信バッファ・サイズの決定」を参照のこと。

装置の認識（デバイス ID）

<device_ID> は文脈により，宛先アドレスか発信元アドレスのいずれかを指す。

コマンド　：　<device_ID> は宛先（デスティネーション）装置のアドレス

レスポンス：　<device_ID> は発信元（ソース）装置のアドレス

コマンド文字列はほとんどの場合，一度には1台の装置を宛先とする。例えば，2台の装置にプレイを指示するには，以下のように送信する。

F0 7F <device_ID= 装置1> <mcc> <PLAY> F7

F0 7F <device_ID= 装置2> <mcc> <PLAY> F7

コマンドにはグループ・デバイス ID が使える。上の例は次のようにも送信できる。

F0 7F <device_ID= グループ1> <mcc> <PLAY> F7

ここの「グループ1」とは，装置1と装置2とからなる。

コマンドでは全員呼び出し（all-call）デバイス ID（7F）も使用することができ，システム全体への「放送 broadcasts 」に役に立つ。

レスポンスの情報列では，以上とは違って，常に1台の装置にのみ対応付けられている。

（a）コントローラーが，デバイス ID をその（レスポンス・メッセージの）発信元として認識できること，及び（b）制御される装置がグループ・デバイス ID を認識できるようになっていることの2点は，他のユニバーサル・システム・エクスクルーシブのインプリメンテーション中には見られない，MIDI マシン・コントロールに特有な要請事項となっている。

構文解析ルーチンはしたがって，受信したシステム・エクスクルーシブ中にマシン・コントロール・メッセージが含まれているかどうかを最初に判断するためには，<device_ID> バイトを解釈し終えるよりも先に，その次に来る＜サブ ID#1 ＞を調べる必要がある。

代表的なシステムは，1台のコントローラーと，これに結ばれた1台あるいはそれ以上の制御される装置とで構成されよう。制御される装置複数台からのレスポンスは，どこかでマージしなくてはならなが，なるべくならコントローラー自身の側で，複数の MIDI IN を用いてするのが良い。たいていの場合には，外部に設けた MIDI マージ装置で充分に役に立ちそうであるが，MIDI の帯域いっぱいにまで使われている場合には,WAIT ハンドシェイクの開始やメッセージ伝達上での遅れが何らかの問題を起こすことが考えられる（付録E「受信バッファ・サイズの決定」を参照されたい）。

勧められることではないが，複数のコントローラーからのコマンド出力がマージされて複数の制御される装置に送られ，さらには複数の装置からのレスポンスがマージされて,複数のコントローラー群に送り返されるといったケースが考えられる。すべてのコントローラーは，すべての制御される装置からのすべてのレスポンスを受け取ることになる可能性があるので,個々のコントローラーは,実は他のコントローラーが要求した装置からのレスポンスでも受ける用意ができていることが重要となる。

複数のコントローラーがこのように結ばれている場合には，確実なエラー処理は犠牲にならざるを得ない場合が考えられる。したがって，エラーの検出機能が真っ先にインプリメントされるものとすれば，個々のコントローラーごとに，時にはエラー検出をしないようにすることができるようにするための，何らかの方法を用意しておく必要がある。エラー処理の詳細に関しては，情報フィールド COMMAND ERROR の項を参照のこと。

＜日本語版注＞

複数のコントローラーが，ある制御される装置からのエラー内容を READ しようとした場合，レスポンスには要求元のアドレス情報がないので，複数回送り返されるエラー報告はどれが誰の分ともわからず，リセットのタイミングともからんで，2回目以降の内容には不確実性が伴うことが考えられる（<COMMAND ERROR> の項参照）。

コマンド

コマンドとは，制御する装置から制御される装置，あるいは1グループをなす制御される装置群へのメッセージである。それぞれのコマンドは，16 進で 01 から 77 の間のコマンド・コードを有し，その後ろにひとつないし数個のデータ・バイトを伴う場合がある(コマンド 00 は拡張用に，また 78 から 7F まではハンドシェィク用に予約されている)。

レスポンス

レスポンスとは，制御される装置から制御する装置へのメッセージであり，通常はコマンドに対する応答として送信される。

制御される装置の内部では，概念的にコントローラーからアクセスされるデータは，情報フィールド(または内部レジスター)と呼ばれる配列内に常に用意されている。例えば，ある装置の現在のタイムコードは，その SELECTED TIME CODE と呼ばれる5バイトの情報フィールドを見ればわかり，またそのタイムコード・ジェネレーターの動作モードは，3バイトのフィールド GENERATOR SET UP の中から知られよう。

それぞれの情報フィールドは，16 進で 01 から 77 の間の名前を持っている(00 は拡張用に，また 78 から 7F まではハンドシェィク用に予約されている)。

ほとんどのレスポンスは，単純に情報フィールド名と，それに続くそのフィールドに含まれているデータとから構成されている。

READ と WRITE のコマンドが，コントローラーにとって，制御される装置の持っている情報フィールドにアクセスする主要な手段となっている(各情報フィールドの説明では，それが「読み出し専用(“read only”)」であるか「読み出し／書き込み可(“read/write”able)」であるかを示している)。

例えば，<SELECTED TIME CODE> がある情報フィールドの 16 進名を表している場合，これに対する READ コマンドとそのレスポンスは：

コマンド　：　F0 7F <device_ID> <mcc> <READ> <count=01> <SELECTED TIME CODE> F7

レスポンス：　F0 7F <device_ID> <mcr> <SELECTED TIME CODE> hr mn sc fr st F7

情報フィールド GENERATOR SET UP への WRITE は：

コマンド　：　F0 7F <device_ID> <mcc> <WRITE> <count=05>
<GENERATOR SET UP> <count=03> <data1> <data2> <data3>
F7

レスポンス：　要求されていない

＜備考＞

ある情報フィールドが「読み出し／書き込み可」としてリストに載っている場合には，READ コマンドによって当該装置の実際の状態を反映するデータが戻される。これは WRITE コマンドによって最後に書き込まれたデータに対応している場合もあれば，していない場合もある。

拡張セット

コマンド 00 とレスポンス／情報フィールド名 00 は，それぞれ2レベルの拡張セット用に確保されている。

00　01　　第1次拡張レベルの最初のコマンドまたは情報フィールド

00　00　01　　第2次拡張レベルの最初のコマンド，または情報フィールド

現時点で拡張機能は，未だ定義されていない。しかしながら，将来での MIDI マシン・コントロールの拡張に備えて，構文解析ルーチンは常に，データの流れの中でコマンドやレスポンス／情報フィールドの名前が来るべき箇所ではどこでも，この拡張形でないかどうかチェックしなければならない。

データ長

1個のシステム・エクスクルーシブで複数のマシン・コントロール・メッセージ（すなわち，複数のコマンドまたはレスポンス／情報フィールド）を送信することができ，また，メッセージ・セットは将来拡張可能でなければならないので，受信する装置はいずれも，いかなる受信メッセージについても，それがその装置にとって既知であるかどうかに関わらず，その長さを判断できるようになっている必要がある。

そこで，多くのコマンドや情報フィールドは，データのバイト数を示す情報を含んでいるが，その他残りのものについては，バスの（特にレスポンス用の MIDI ラインでの）帯域巾を無駄にしないために，それぞれの名前を示すバイトの 16 進数値自体が，暗黙的にその長さを含む方式が取られている。

コマンド

00	拡張用に予約されている
01～3F	0データ・バイト
40～77	可変長データであり，<count> バイトがデータ部の先頭に置かれる
78～7F	0データ（ハンドシェイク用）

レスポンス／情報フィールド

00	拡張用に予約されている
01～1F	5データ・バイト（標準タイムコード・フィールド）
20～3F	2データ・バイト（短形式タイムコード・フィールド）
40～77	可変長データであり，<count> バイトがデータ部の先頭に置かれる
78～7F	0データ（ハンドシェイク用）

＜備考＞

1. 拡張セットもこの同じフォーマットに従う。例えば，拡張レスポンス00 27 には2個のデータ・バイトが続くことになる。
2. 可変長データは，<count> <data …> の形式を取る。ただし <count> は，このカウント・バイト自体は含まない。
3. 可変長フィールドは，本規格の将来のバージョンにおいてはその長さが延長される可能性はあり得るが，現時点で定義されている内容が変えられることはない。例えば，<count=04> <aa bb cc dd> は，必要となれば <count=07> <aa bb cc dd xx yy zz> と拡張されるかも知れないが，<aa bb cc dd> 部分の定義は変わらずに残るはずである。
4. 以下，本文において可変長フィールドは，3種類の異なるフォーマットで現われる。
   (i) あらかじめ決められた長さのもの。例えば，<count=03> <pp qq rr>。
   (ii) データ部分に含まれるコマンドや情報フィールドに拡張セットの用いられる可能性のあるために，長さが部分的にのみ決められているもの。例えば，<count=02+ext> <name1 name2>。
   (iii) 長さが調節可能なもの。例えば，<count=variable>。

セグメンテーション

ひとつのメッセージ（あるいはメッセージ列）が長すぎて，MMC システム・エクスクルーシブの定める最大データ・フィールド長（48 バイト）には収まらない場合が出て来るであろう。このようなメッセージはいくつかのセグメントに分割して，複数個のシステム・エクスクルーシブの形で，部分部分ごとに送信することができる。

このようにして受信されたメッセージは，あたかも1個のシステム・エクスクルーシブで，全部一度に届けられたかのように扱うものとする。

このセグメンテーション・プロセスをサポートするために，2つの特別なメッセージを用意する。すなわち，COMMAND SEGMENT と RESPONSE SEGMENT である。両者はそれ自身のデータ・フィールド中に，長いメッセージの中のあるセグメント部分をはめ込んで伝える働きを持つ。付け加えると，このフィールド中の先頭バイトには，セグメント番号を示す降順のカウント値が，「先頭」のセグメントの場

合にこれを示すフラグ(40H)と共に置かれている(受信中の装置は，これにより先頭のセグメントを調べて，この後にいくつのセグメントが送信されて来るかを確かめることができる)。最後のセグメントは常に，ダウン・カウント値が0にセットされている。

例えば，コマンド列 aa bb cc dd ee ff gg hh jj kk mm は，(ひとつの長いコマンドでも，あるいは短いコマンドいくつかのつながりでも良いのであるが)通常，次のように送信されるであろう。

F0 7F <device_ID> <mcc> aa bb cc dd ee ff gg hh jj kk mm F7

セグメンテーションを用いて，これとまったく同じ結果が得られる。

F0 7F <device_ID> <mcc> <COMMAND SEGMENT> <count=05> 42 aa bb cc dd F7
F0 7F <device_ID> <mcc> <COMMAND SEGMENT> <count=05> 01 ee ff gg hh F7
F0 7F <device_ID> <mcc> <COMMAND SEGMENT> <count=04> 00 jj kk mm F7

エラー処理

READ あるいは UPDATE コマンドが，当該装置のサポートしていない情報フィールドに置かれるデータを要求した場合には，いつでも制御される装置はコントローラーに対して，メッセージ RESPONSE ERROR を送信する。こうすることでコントローラーは，常に自分の出したいかなるデータ要求に対しても，少なくとも1個のレスポンスを受け取ることとなる。すなわち，要求したデータか，あるいは RESPONSE ERROR メッセージのいずれかを必ず受信するのである。

さらに詳しくは，READ，UPDATE 及び RESPONSE ERROR についての解説部分で説明されている。

レスポンス・エラーとは違って，コマンド・エラーに関しては，情報フィールド COMMAND ERROR 及び COMMAND ERROR LEVEL，これに加えてハンドシェイク・メッセージの COMMAND ERROR RESET で扱われる。コマンド・エラーの処理をサポートする場合には，これら3個のメッセージを一体として備えなければならない。

制御される装置は，その無指定時(デフォルト)の状態においてはすべてのコマンド・エラーは無視し，受信したコマンドのうちエラーの心配のないものだけに対して処理を続けるよう努めるものとする。この方法は，“オープン・ループ”やその他の極めて基本的なシステムに対しては，最も適したものと言える。

しかしながら，さらに“インテリジェントな”コントローラーにあっては，COMMAND ERROR LEVEL 情報フィールドに WRITE することで，定義されているコマンド・エラーのいくつかか，あるいはすべてを有効(イネーブル)にすることができる。“イネーブル”とされたエラーの起こった場合にはいつでも，制御される装置はすべてのコマンド処理を停止し，COMMAND ERROR メッセージをコントローラーに送り返すことで，エラーの詳細とこれを引き起こしたコマンドに関する参考情報を伝えるものとする。これに続いてコントローラーは，COMMAND ERROR RESET を出すことによって，通常のオペレーションを再開可能にしなければならない。

コマンド・メッセージの構文

<command message> ::= F0 7F <destination> <mcc> <command string> F7

<destination> ::= <device address> | <group address> | <all call address>
<device address> ::= 00 | 01 | ... |7E
<group address> ::= 00 | 01 | ... |7E
<all call address> ::= 7F
<mcc> ::= 06 (MMC コマンド・サブ ID 番号 #1)

<command string> ::= <command> | <command string> <command>

<command> ::= <command_code_0>
| <command_code_variable> <count> <command data>

| <handshake>

<command_code_0> ::= 01 | 02 | ... |3F | 00 <command_code_0>
<command_code_variable> ::= 40| 41 | ... | 77 | 00<command_code_variable>
<handshake> ::= 78 | 79 | ... | 7F | 00 <handshake>

レスポンス・メッセージの構文

<response message> ::= F0 7F <source> <mcr> <response string> F7

<source> ::= <device address>
<device address> ::= 00 | 01 | ... | 7E

<mcr> ::= 07 (MMC レスポンス・サブ ID 番号 #1)

<response string> ::= <response> | <response string> <response>

<response> ::= <info_field_name_5> <standard 5-byte time code data>
| <info_field_name_2> <2-byte short time code data>
| <info_field_name_variable> <count> <info_field_data>
| <handshake>

<info_field_name_5> ::= 01 | 02 | ... | 1F | 00 <info_field_name_5>
<info_field_name_2> ::= 20 | 21 | ... | 3F | 00 <info_field_name_2>
<info_field_name_variable> ::= 40 | 41 | ... | 77 | 00 <info_field_name_variable>
<handshake> ::= 78 | 79 | ... | 7F | 00 <handshake>

3　標準仕様

標準タイムコード（タイプ {ff} 及び {st}）

これはタイムコード仕様の“完全”形式であり，常にきっかり5バイトのデータを含んでいる。

タイムコードのサブフレーム・データには，2つの形式が定義されている。

第1の形式（{ff} と標識のついたもの）は，MIDI キューイング仕様で説明されている通りのサブフレーム・データ，すなわち 100 分の1フレーム単位で計った小数フレームを含んでいる。

第2の形式（{st} と標識のついたもの）は，サブフレームの代わりにここをタイムコードのステータス・データで置き換える。例えばテープからのタイムコード・データを処理する際には，本物のタイムコード・データを受け取っているのか，それとも高速巻取中のテープ・トランスポートのタコメーターからのパルスで更新された，単なるタイム・データなのかがわかるとしばしば便利である。

個々の MMC タイムコード情報フィールド内に設定されているすべてのステータス・ビットについて，その厳密な使用法に関しては，付録Bの「タイムコード・ステータスの運用細則」をも参照されたい。

hr mn sc fr {ff|st}

hr = 時刻とタイプ：　0 tt hhhhh

tt = フレーム・タイプ

00 = 24 フレーム

01 = 25 フレーム

10 = 30 ドロップ・フレーム

11 = 30（ノンドロップ）フレーム

hhhhh = 時（10進の 0 ～ 23, 16 進の 00 ～ 17）

＜日本語版注＞

上の「30 ドロップ」や「30 ノンドロップ」という表現は，「30 と 29.97 とを区別した上での 30」という意味ではなく，「30 と 29.97 との総称」である。

mn = 分：　0 c mmmmmm

c = カラー・フレーム・フラグ（タイムコード中の該当ビットの写し）

0 = カラー・フレーム適用せず

1 = カラー・フレーム適用あり

mmmmmm = 分（10進の 0 ～ 59, 16 進の 00 ～ 3B）

sc = 秒：　0 k ssssss

k =「ブランク」ビット

0 = 正常時

1 = タイムコード・データは未だ1度もこの情報フィールドに読み込まれていない。（すなわち，パワー・オンか，MMC RESET されたまま）タイムコードの数値部分はすべてゼロとすること。

ssssss = 秒（10進の 0 ～ 59, 16 進の 00 ～ 3B）

fr = フレーム，第5バイト標識，及び符号：　0 g i fffff

g = 符号

0 = プラス

1 = マイナス（符号付きタイムコードの許される場合）

i = 最終バイト標識

0 = サブフレーム

1 = ステータス

fffff = フレーム（10進の 0 ～ 29,16 進の 00 ～ 1D）

最終バイトがサブフレームの場合（i = 0）

ff = 小数フレーム：0 bbbbbbb（10進の 0-99,16 進の 00-63）

最終バイトがステータスの場合 （i = 1）

st = タイムコードのステータス : 0 e v d n xxx

e = 推測（estimated）コード・フラグ

0 = 通常のタイムコード

1 = タコまたはコントロール・トラックにより更新されたコード

v = 不正（invalid）コード（e = 1 の場合は無視のこと）

0 = 現タイムコード値は，本装置内での正当性（validity）チェックに合格

1 = 現タイムコード値は，正当性が確認されていない

d = ビデオ・フィールド1標識

0 = 本フレームにフィールド情報は含まれない

1 = ビデオ用4／8フィールド・シーケンスの先頭フレーム

n = タイムコード不在（“no time code”）フラグ

0 = タイムコード・リーダの入力部でタイムコードが検出されている

1 = パワー・オンあるいは MMC RESET 後，タイムコードの入力無し

xxx = 予約されている。000 でなければならない。

標準短形式タイムコード

この短縮形のフォーマットは，繰り返し応答のモード（UPDATE コマンドを参照のこと）で用いられ，この場合制御装置は，制御される装置に対してある情報フィールドからのデータを，その内容が変わりしだいいつでも送信するように指示する。この種の伝送では，その大半は何らかの形でタイムコードを含んでおり，さらにその大部分は，直前の内容と比べるとフレーム部分のみが違っているはずある。言い換えれば，ひとたび最初のタイムコードの値が送られた後では，時分秒のデータはそのいずれかに変化のあった時に限り（すなわち1秒ごとに）送りさえすればよい。この“短形式”タイムコードは，したがって，フレームとサブフレームのデータのみを含むのであり，“完全”形式でのフレームとサブフレームの部分とまったく同等になっている。

fr {st|ff}

この「短形式」の大きな利点は，レスポンス・ラインの帯域巾を節約できる点である。

＜備考＞

1. コード 01-1FH の範囲に置かれた5バイト長タイムコードの情報フィールド名すべてに対して，これに対応した2バイトの“短形式”タイムコードを持つ同じ名前のフィールドが，21 ～ 3FH の領域に用意されている。例えば，06H は GENERATOR TIME CODE であり，26H は Short GENERATOR TIME CODE である。
2. “短形式”のものは，後の「レスポンス，情報フィールド詳説」編で個々には取り上げられていないが，そのフォーマットについては“標準”形式についての説明から容易に類推できるであろう。

標準ユーザー・ビット

u1 u2 u3 u4 u5 u6 u7 u8 u9

u1 = バイナリィ・グループ 1:　　0000 aaaa

u2 = バイナリィ・グループ 2:　　0000 bbbb

u3 = バイナリィ・グループ 3:　　0000 cccc

u4 = バイナリィ・グループ 4:　　0000 dddd

u5 = バイナリィ・グループ 5:　　0000 eeee

u6 = バイナリィ・グループ 6:　　0000 ffff

u7 = バイナリィ・グループ 7:　　0000 gggg

u8 = バイナリィ・グループ 8:　　0000 hhhh

u9 = フラグ :　　0000 0tji

t =「2次」タイムコード・ビット

0 = 標準ユーザー・ビット

1 = ユーザー・ビットは「2次」タイムコードを含む。

j = バイナリィ・グループ・フラグ1（SMPTE の bit 59, EBU の bit 43）

i = バイナリィ・グループ・フラグ0（SMPTE の bit 43, EBU の bit 27）

＜備考＞

1. バイナリィ・グループ及びバイナリィ・グループ・フラグの定義については，SMPTE なり EBU なりの適当な規格書を参照されたい。
2. タイムコードは，時によりユーザー・ビット中に符号化されることがある（t=1）。その場合のタイムコードは，（必要なだけの種々のステータス・フラグを備えた）通常のタイムコード用の SMPTE/EBU で規定されている BCD 形式であり，次のように読み込まれる。

   aaaa = フレームの1の位
   bbbb = フレームの10の位
   cccc = 秒の1の位
   dddd = 秒の10の位
   eeee = 分の1の位
   ffff = 分の10の位
   gggg = 時の1の位
   hhhh = 時の10の位
3. もしもバイナリィ・グループ1～8の各4ビット（ニブル）を用いて8ビット情報が表されている場合には，4個の8ビット文字は次の順序で再編成すること。hhhhgggg ffffeeee ddddcccc bbbbaaaa
4. ユーザー・ビットの 10 進表示をする場合にも，その順序は同じく hhhhgggg ffffeeee ddddcccc bbbbaaaa とすること。

ドロップ・フレームの扱い

1. タイムコードを含む情報フィールドにデータを書き込む際には，そのデータがドロップ・フレームかノンドロップ・フレームかを示すステータスは，SELECTED TIME CODE のフレーム種別ステータスの方が優先することによって，無効になる場合がある。

   例えば，今あるテープレコーダーが，そのテープからドロップ・フレームのタイムコードを読んでいるとすれば，SELECTED TIME CODE フィールドはドロップ・フレームのステータスを示している。もしもここでコントローラーが，GP0/LOCATE POINT フィールドにノンドロップ・フレームの値を書き込んだ上で LOCATE to GP0 を実行させると，この GP0/LOCATE POINT 値は SELECTED TIME CODE に従ってドロップ・フレームであるものと解釈され，いかなる変換操作も行われることはない。

2. さらにまた，もしも上記の GP0/LOCATE POINT に，実際にはドロップ・フレームとしてはあり得ない（例えば 00:22:00:00 といった）値がセットされていた場合には，この直後に来る次の正当な時刻値が使われることになる（今の例では，00:22:00:02 に）。

3. オフセット，すなわち単純に2つのタイムコード間の数学的な差を計算する際に，片方あるいは両方の数値がドロップ・フレームであると，混乱した事態が起こり得る。この規格の目的とするところを考えて，オフセット計算を行う際には，ドロップ・フレームの数は計算に先立って，まずノンドロップ・フレームに変換するものとする。オフセット計算の結果は，したがって，ノンドロップ・フレームの量として表現されることになる。

   ドロップ・フレームからノンドロップ・フレームに変換するには，基準点の 00:00:00:00 から数えた“落とされる”フレームの総数を引くと良い。例えば，ドロップ・フレームの数 00:22:00:02 をノンドロップ・フレームに変換するには，40 フレームを引いて，結果は 00:21:58:22 となる。40 という数は，01 から 09，11 から 19 まで，それに 21，22 の各分ごとに2フレームが“落とされる”ことから出て来る。

標準スピード

この3バイト長のフォーマットは，VARIABLE PLAY，DEFERRED VARIABLE PLAY，RECORD STROBE VARIABLE，SEARCH，SHUTTLE の各コマンド，及び VELOCITY TALLY 情報フィールドで用いられる。この仕様は，いずれの制御される装置に対しても，その内部的なスピード制御や速度の検出に関して，特定の分解能を要請するものではない。

sh sm sl

sh　スピード値を表す仮の整数部分 : 0 g sss ppp

g　= 符号（1はリバース方向）

sss = 左シフトの数（下記参照）

ppp = プレイ・スピードの倍速値を表す整数の MSB 部分

sm　スピード値を表す仮の小数部分の MSB 側 : 0 qqqqqqq

sl　スピード値を表す仮の小数部分の LSB 側 : 0 rrrrrrr

左シフト・カウントとスピード値の対応

sss	2進表示		設定範囲（10進）	
	倍速指定整数部分	小数部分	整数部の範囲	小数部の分解能
000	ppp	. qqqqqqqrrrrrrr	0-7	1/16384
001	pppq	. qqqqqqrrrrrrr	0-15	1/8192
010	pppqq	. qqqqqrrrrrrr	0-31	1/4096
011	pppqqq	. qqqqrrrrrrr	0-63	1/2048
100	pppqqqq	. qqqrrrrrrr	0-127	1/1024
101	pppqqqqq	. qqrrrrrr	0-255	1/512
110	pppqqqqqq	. qrrrrrr	0-511	1/256
111	pppqqqqqqq	. rrrrrr	0-1023	1/128

標準トラック・ビット・マップ

この可変長フィールドには，制御される装置がサポートするオーディオなりビデオなりの各“トラック”ごとに1ビットが用意される。ビットの値が1であるのはアクティブ状態であることを表し，対して0は非アクティブ状態であることを示す。使用していないビットや予約されているビットはすべて，0にリセットしておかなければならない。標準トラック・ビット・マップは，システム・エクスクルーシブのセグメンテーションを使わない限りでは，論理的には最大 317 トラックまで拡張できる。

標準トラック・ビット・マップは，現時点では情報フィールド TRACK RECORD READY，TRACK RECORD STATUS，TRACK SYNC MONITOR，TRACK INPUT MONITOR，それに TRACK MUTE で用いられている。

レスポンスとして読み出される際には，制御される装置は標準トラック・ビット・マップをちょうど必要とされるバイト数分だけ送信する必要がある。レスポンス送信に含まれていないトラックについては，いずれも非アクティブ状態であると見なして，そのビットは0にリセットするものとする。標準トラック・ビット・マップは，常にその前にバイト・カウントを伴っているので，全トラックが非アクティブ状態である場合には，カウントを 00 としても良い。

WRITE コマンドによって書き込まれる場合には，送信中に含まれていないトラックについては，それぞれのビットを0にリセットのこと（トラックは非アクティブ状態となる）。全トラックをリセットしたい場合には，バイト・カウントを 00 としても良い。

MASKED WRITE コマンドを用いると，標準トラック・ビット・マップ中の特定ビット（複数）を修正することもできる。

r0 r1 r2 . . .

r0　　ビット・マップ 0:　0 gfedcba

a = ビデオ

b = 将来のために予約（0でなければならない）
c =（専用の） タイムコード・トラック
d = 補助トラックA（例えばアナログのガイド・トラック等）
e = 補助トラックB
f = トラック 1（ステレオ左／モノラル）
g = トラック 2（ステレオ右）

r1　　ビット・マップ 1:　0 nmlkjih

h = トラック 3
i = トラック 4
j = トラック 5
k = トラック 6
l = トラック 7
m = トラック 8
n = トラック 9

r2　ビット・マップ 2: トラック 10 ～ 16
r3　ビット・マップ 3: トラック 17 ～ 23
r4　ビット・マップ 4: トラック 24 ～ 30
r5　ビット・マップ 5: トラック 31 ～ 37
r6　ビット・マップ 6: トラック 38 ～ 44
r7　ビット・マップ 7: トラック 45 ～ 51
r8　ビット・マップ 8: トラック 52 ～ 58
r9　ビット・マップ 9: トラック 59 ～ 65

：
etc

＜日本語版注＞

第1データ・バイト r0 は，主としてビデオ・テープレコーダーでの各種トラックと，オーディオの2トラックをカバーしている。ビデオ・レコーダーでの制御対象としては，VITC のタイムコードをはじめ FM，PCM，あるいはその両方の組み合わせなどが考えられるが，これらの記録はいずれも，通常はビデオ信号の記録と完全に独立には制御できない（ビデオを記録せずに VITC だけを記録することはできない）。本情報フィールドは,それぞれ完全に独立にレコードのオン／オフ制御が可能な“トラック”を対象とするため,以上の要素に対しては別々のビットを与えることはしない。したがって，例えばビデオの記録時に同時に VITC を記録するかどうかは，別の情報フィールドで規定される（<VITC INSERT ENABLE> 参照）。他についても同様の考えによる。

トラックcは，したがって SMPTE/EBU のタイムコードを記録する専用のリニア・トラックであり，トラックd，eもまたこうした独立トラックに充てられる。

モーション・コントロール・ステート及びプロセス

モーション・コントロール・ステート（MCS）

PLAY，STOP，FAST FORWARD，あるいは REWIND などの基本的なトランスポート・コマンドは，いずれも制御される装置を新たな，相互に排他的な動作状態に移行させる。これらのコマンドはしたがって，まとめて“モーション・コントロール・ステート：Motion Control State”コマンドと名付ける。それぞれの MCS コマンドは新しいトランスポート状態へと移行させ,それまでのモーション・コントロール・ステートを止めさせる。

MCS コマンドが直接出され，これを受信した場合もまた，後に述べるように，進行中のモーション・コントロール・プロセス（MCP）は自動的に打ち切られる。（ロケート MCP の実行中に，DEFERRED PLAY や DEFERRED VARIABLE PLAY コマンドを受信した場合は例外となる。）

MCS コマンドは：　(i)このコマンド・セットで直接的に指示される。
or (ii)モーション・コントロール・プロセス（以下を参照）の実行中に，その各ステップとして間接的に出される。
or (iii)それ以外の，例えば装置自身の制御パネルから起動される。

モーション･コントロール･ステートの動きは,MOTION CONTROL TALLY 情報フィールド中の"直前に実行された MCS コマンド(ms)"バイト上にタリーされる。要求されたステートに当該装置が正しく移れたかどうかは,同情報フィールド中の"ステータス及び成功レベル(ss)"バイト上に示される。

モーション・コントロール・ステートのコマンドはすべて，インデックス・リスト中では"(MCS)"と，コマンド説明部では"(MCS コマンド)"と，それぞれ注記されている。

現時点で定義されている MCS コマンドは：
STOP
PAUSE
PLAY
DEFERRED PLAY
VARIABLE PLAY
DEFERRED VARIABLE PLAY
FAST FORWARD
REWIND
SEARCH
SHUTTLE
STEP
EJECT

モーション・コントロール・プロセス (MCP)

(LOCATE や CHASE といった)モーション・コントロール・プロセス(Motion Control Process)とは，制御される装置がそれ自身の持つモーション・コントロール・ステート・コマンド列を，自動的に出すことで所望の結果を得るようにする，優先的なコントロール・コマンドを言う。

モーション・コントロール・プロセスは互いに排他的であり，MCP コマンドによって指示される。MCP コマンドを受信すると，これより先に受信していた MCS コマンドは打ち切られる。

MCP コマンドは： (i)このコマンド・セットによって直接的に出される。
or (ii)それ以外の，例えば装置自身の制御パネルから起動される。

モーション・コントロール・プロセスの動きは，MOTION CONTROL TALLY 情報フィールド中の"直前に実行された MCP コマンド(mp)"バイト上にタリーされる。要求されたプロセスを当該装置が正しく実行できたかどうかは，同情報フィールド中の"ステータス及び成功レベル(ss)"バイト上に示される。

さらに，モーション・コントロール・プロセスの間は，自動的に出されるそれぞれのモーション・コントロール・ステートは，MOTION CONTROL TALLY 情報フィールド中に，前節に述べられている方式で，そのつど登録される。

モーション・コントロール・プロセス・コマンドは，すべてインデックス・リスト中では"(MCP)"と，コマンド説明部では"(MCP コマンド)"とそれぞれ注記されている。

現時点で定義されている MCP コマンドは：**LOCATE**
CHASE

4　インデックス・リスト

メッセージ・タイプ

本インデックス・リスト中のそれぞれのコマンドやレスポンス／情報フィールドには，以下のようなタイプ標識が付けられている。

Comm	**通信の支援用。例えば WAIT, RESUME, GROUP。**
Ctrl	**「トランスポート」の操作に直接関わる。例えば PLAY, STOP, TRACK RECORD READY 等。**
Evnt	**時刻指定のイベント・トリガー。**
Gen	**タイムコード・ジェネレーター用インターフェース。**
I/O	**READ, WRITE のコマンド，及びエラー処理等。**
Sync	**タイムコード同期用。「マスター」タイムコード・フィールドを含む。**
Math	**タイムコード計算用。**
MTC	**MIDI タイムコードの入出力制御。**
Proc	**プロセジャー（あらかじめ決めておくコマンド・シーケンス）の定義と実行。**
Time	**装置自身のタイムコードの流れに直接関連する情報フィールド。**

使用する省略形式

ATR	**オーディオ・テープレコーダー**
{ff}	**「サブフレーム」を含むタイムコード（3節，標準仕様参照）**
MCP	**モーション・コントロール・プロセス（Motion Control Process）＊**
MCS	**モーション・コントロール・ステート（Motion Control State）＊**
MMC	**MIDI マシン・コントロール**
r	**情報フィールドは読み出しのみ可（read only）**
R/W	**情報フィールドは読み出し／書き込み共に可**
{st}	**「ステータス」を含むタイムコード（3節，標準仕様参照）**
VTR	**ビデオ・テープレコーダー**

＊ インデックス・リスト中の各モーション・コントロール・コマンドには，（MCS）あるいは（MCP）のタグが付けられており，それぞれそのコマンドがモーション・コントロール・ステート，あるいはモーション・コントロール・プロセスのいずれかを起動させることを示している。これらの用語の説明については，「3 標準仕様」を参照のこと。

最小セットのガイドライン

MIDI マシン・コントロールにおいては，いずれの装置もが必ず装備していなくてはならないといった，コマンドやレスポンス／情報フィールドの絶対的な最小限のセットなるものは規定しない。

しかしながら，異なる状況の下で，どのコマンドやレスポンスが重要であろうかを理解するのを助けるために，コマンドとレスポンス／情報フィールドのガイドラインとしての最小セットが4個設定された。

#1　単純なトランスポート（タイムコード・リーダなし／オープン・ループの通信のみ）
#2　基本的なトランスポート（タイムコード・リーダなし／クローズド・ループの通信が可能）
#3　高度なトランスポート（タイムコード・リーダ有り／クローズド・ループの通信方式／イベント・トリガー機能付き／トラックごとに独立の記録制御）
#4　基本的なシンクロナイザー（クローズド・ループの通信方式）

最小セットのガイドラインは，いかなる意味においても，個々の装置の操作（オペレーション）の範囲を制限することは意図していない。ひたすら（ハードウェア，ソフトウェアの）技術者達が MMC について学び，そしておそらくは始めてこれをインプリメントしようとした際に，それを助けるために用意されたものである。

それぞれのコマンドやレスポンス／情報フィールドがどの最小セット・ガイドラインに割り当てられているかは，インデックス・リスト中の右端の欄に示されている。

＜日本語版注＞

日本語版においては，それぞれのコマンドやレスポンス／情報フィールドの解説部にも，タイトル行ごとに示されている。

SIGNATURE 情報フィールドについては，特にひと言記しておく必要があろう。制御される装置がサポートしているすべてのコマンドやレスポンス／情報フィールドは，このフィールドによって完全なビット・マップの形で示される。コントローラーは，対象とする装置の

SIGNATURE を調べることにより, サポートされている機能にぴったりとマッチするよう, それとのやり取りを適合させる場合も考えられる。この SIGNATURE フィールドを備えることは, したがって大いに推奨されている。制御される装置の製造者はまた, SIGNATURE 情報フィールドの説明の項に述べられているフォーマットに従って, その装置の signature を文書の形で明らかにすべきである。

コマンド・リスト

16 進	コマンド	タイプ	データ・バイト数	ガイドライン最小セット
00	reserved for extensions			1234
01	STOP(MCS)	Ctrl	—	1234
02	PLAY(MCS)	Ctrl	—	-234
03	DEFERRED PLAY(MCS)	Ctrl	—	1234
04	FAST FORWARD(MCS)	Ctrl	—	1234
05	REWIND(MCS)	Ctrl	—	1234
06	RECORD STROBE	Ctrl	—	1234
07	RECORD EXIT	Ctrl	—	1234
08	RECORD PAUSE	Ctrl	—	----
09	PAUSE(MCS)	Ctrl	—	----
0A	EJECT(MCS)	Ctrl	—	----
0B	CHASE(MCP)	Sync	—	---4
0C	COMMAND ERROR RESET	I/O	—	-234
0D	MMC RESET	Ctrl	—	1234
40	WRITE	I/O	n	1234
41	MASKED WRITE	I/O	n	--3-
42	READ	I/O	n	-234
43	UPDATE	I/O	n	-234
44	LOCATE(MCP)	Ctrl	n	1234
45	VARIABLE PLAY(MCS)	Ctrl	3	-234
46	SEARCH(MCS)	Ctrl	3	--34
47	SHUTTLE(MCS)	Ctrl	3	----
48	STEP(MCS)	Ctrl	1	----
49	ASSIGN SYSTEM MASTER	Sync	1	----
4A	GENERATOR COMMAND	Gen	1	----
4B	MIDI TIME CODE COMMAND	MTC	1	----
4C	MOVE	Math	2	1234
4D	ADD	Math	3	-234
4E	SUBTRACT	Math	3	-234
4F	DROP FRAME ADJUST	Math	1	--34
50	PROCEDURE	Proc	n	--34
51	EVENT	Evnt	n	--34
52	GROUP	Comm	n	-234
53	COMMAND SEGMENT	Comm	n	-234
54	DEFERRED VARIABLE PLAY(MCS)	Ctrl	3	-234
55	RECORD STROBE VARIABLE	Ctrl	3	----
7C	WAIT	Comm	—	-234
7F	RESUME	Comm	—	-234

レスポンス／情報フィールド・リスト

16 進	レスポンス／情報フィールド名	タイプ	バイト数	Read/Write	ガイドライン最小セット
00	reserved for extensions				----
01	SELECTED TIME CODE {st}	Time	5	RW	1234
02	SELECTED MASTER CODE {st}	Sync	5	r	---4
03	REQUESTED OFFSET {ff}	Sync	5	RW	---4
04	ACTUAL OFFSET {ff}	Sync	5	r	---4
05	LOCK DEVIATION {ff}	Sync	5	r	---4
06	GENERATOR TIME CODE {st}	Gen	5	RW	----
07	MIDI TIME CODE INPUT {st}	MTC	5	r	----
08	GP0/LOCATE POINT {ff}	Math	5	RW	1234
09	GP1 {ff}	Math	5	RW	-234
0A	GP2 {ff}	Math	5	RW	-234
0B	GP3 {ff}	Math	5	RW	-234
0C	GP4 {ff}	Math	5	RW	----
0D	GP5 {ff}	Math	5	RW	----
0E	GP6 {ff}	Math	5	RW	----
0F	GP7 {ff}	Math	5	RW	----
21 ～ 2F	01 ～ 0F の短形式		2	r	-234
40	SIGNATURE	I/O	n	r	-234
41	UPDATE RATE	I/O	1	RW	-234
42	RESPONSE ERROR	I/O	n	-	-234
43	COMMAND ERROR	I/O	n	r	-234
44	COMMAND ERROR LEVEL	I/O	1	RW	-234
45	TIME STANDARD	Time	1	RW	-234
46	SELECTED TIME CODE SOURCE	Time	1	RW	----
47	SELECTED TIME CODE USERBITS	Time	9	r	----
48	MOTION CONTROL TALLY	Ctrl	3	r	-234
49	VELOCITY TALLY	Ctrl	3	r	----
4A	STOP MODE	Ctrl	1	RW	----
4B	FAST MODE	Ctrl	1	RW	----
4C	RECORD MODE	Ctrl	1	RW	-234
4D	RECORD STATUS	Ctrl	1	r	-234
4E	TRACK RECORD STATUS	Ctrl	n	r	--3-
4F	TRACK RECORD READY	Ctrl	n	RW	--3-
50	GLOBAL MONITOR	Ctrl	1	RW	--3-
51	RECORD MONITOR	Ctrl	1	RW	----
52	TRACK SYNC MONITOR	Ctrl	n	RW	----
53	TRACK INPUT MONITOR	Ctrl	n	RW	----
54	STEP LENGTH	Ctrl	1	RW	----
55	PLAY SPEED REFERENCE	Ctrl	1	RW	-23-
56	FIXED SPEED	Ctrl	1	RW	----
57	LIFTER DEFEAT	Ctrl	1	RW	----
58	CONTROL DISABLE	Ctrl	1	RW	---4
59	RESOLVED PLAY MODE	Sync	1	RW	---4
5A	CHASE MODE	Sync	1	RW	---4
5B	GENERATOR COMMAND TALLY	Gen	2	r	----
5C	GENERATOR SET UP	Gen	3	RW	----
5D	GENERATOR USERBITS	Gen	9	RW	----

5E	MIDI TIME CODE COMMAND TALLY	MTC	2	r	----
5F	MIDI TIME CODE SET UP	MTC	1	RW	----
60	PROCEDURE RESPONSE	Proc	n	r	--34
61	EVENT RESPONSE	Evnt	n	r	--34
62	TRACK MUTE	Ctrl	n	RW	--3-
63	VITC INSERT ENABLE	Gen	3	RW	----
64	RESPONSE SEGMENT	Comm	n	-	-234
65	FAILURE	Ctrl	n	-	-234
7C	WAIT	Comm	-	-	-234
7F	RESUME	Comm	-	-	-234

コマンドと情報フィールドのタイプ別リスト

読み出し／書き込み(I/O)

Commands:

- 0C COMMAND ERROR RESET
- 40 WRITE
- 41 MASKED WRITE
- 42 READ
- 43 UPDATE

Information Fields:

- 40 SIGNATURE
- 41 UPDATE RATE
- 42 RESPONSE ERROR
- 43 COMMAND ERROR
- 44 COMMAND ERROR LEVEL

トランスポート制御(CTRL)

Commands:

- 01 STOP(MCS)
- 02 PLAY(MCS)
- 03 DEFERRED PLAY(MCS)
- 04 FAST FORWARD(MCS)
- 05 REWIND(MCS)
- 06 RECORD STROBE
- 07 RECORD EXIT
- 08 RECORD PAUSE
- 09 PAUSE(MCS)
- 0A EJECT(MCS)
- 0D MMC RESET
- 44 LOCATE(MCP)
- 45 VARIABLE PLAY(MCS)
- 46 SEARCH(MCS)
- 47 SHUTTLE(MCS)
- 48 STEP(MCS)
- 54 DEFERRED VARIABLE PLAY(MCS)
- 55 RECORD STROBE VARIABLE

Information Fields:

- 48 MOTION CONTROL TALLY
- 49 VELOCITY TALLY
- 4A STOP MODE
- 4B FAST MODE
- 4C RECORD MODE

4D RECORD STATUS
4E TRACK RECORD STATUS
4F TRACK RECORD READY
50 GLOBAL MONITOR
51 RECORD MONITOR
52 TRACK SYNC MONITOR
53 TRACK INPUT MONITOR
54 STEP LENGTH
55 PLAY SPEED REFERENCE
56 FIXED SPEED
57 LIFTER DEFEAT
58 CONTROL DISABLE
62 TRACK MUTE
65 FAILURE

個々のタイムコード(Time)

Information Fields:

01 SELECTED TIME CODE {st}
21 Short SELECTED TIME CODE {st}
45 TIME STANDARD
46 SELECTED TIME CODE SOURCE
47 SELECTED TIME CODE USERBITS

同期(Sync)

Commands:

0B CHASE(MCP)
49 ASSIGN SYSTEM MASTER

Information Fields:

02 SELECTED MASTER CODE {st}
03 REQUESTED OFFSET {ff}
04 ACTUAL OFFSET {ff}
05 LOCK DEVIATION {ff}
22 Short SELECTED MASTER CODE {st}
23 Short REQUESTED OFFSET {ff}
24 Short ACTUAL OFFSET {ff}
25 Short LOCK DEVIATION {ff}
59 RESOLVED PLAY MODE
5A CHASE MODE

タイムコード・ジェネレーター(Gen)

Commands:

4A GENERATOR COMMAND

Information Fields:

06 GENERATOR TIME CODE {st}
26 Short GENERATOR TIME CODE {st}
5B GENERATOR COMMAND TALLY
5C GENERATOR SET UP
5D GENERATOR USERBITS
63 VITC INSERT ENABLE

MIDIタイムコード(MTC)

Commands:

4B MIDI TIME CODE COMMAND

Information Fields:

07 MIDI TIME CODE INPUT {st}
27 Short MIDI TIME CODE INPUT {st}
5E MIDI TIME CODE COMMAND TALLY
5F MIDI TIME CODE SET UP

タイムコード計算（Math）

Commands:

4C MOVE
4D ADD
4E SUBTRACT
4F DROP FRAME ADJUST

Information Fields:

08 GP0/LOCATE POINT {ff}
09 GP1 {ff}
0A GP2 {ff}
0B GP3 {ff}
0C GP4 {ff}
0D GP5 {ff}
0E GP6 {ff}
0F GP7 {ff}
28 Short GP0/LOCATE POINT {ff}
29 Short GP1 {ff}
2A Short GP2 {ff}
2B Short GP3 {ff}
2C Short GP4 {ff}
2D Short GP5 {ff}
2E Short GP6 {ff}
2F Short GP7 {ff}

手続き（プロセジャー：Proc）

Commands:

50 PROCEDURE

Information Fields:

60 PROCEDURE RESPONSE

イベント・トリガー（Evnt）

Commands:

51 EVENT

Information Fields:

61 EVENT RESPONSE

通信（Comm）

Commands:

52 GROUP
53 COMMAND SEGMENT
7C WAIT
7F RESUME

Information Fields:

64 RESPONSE SEGMENT
7C WAIT
7F RESUME

5 コマンド詳説

制御する装置(コントローラー)から制御される装置へのメッセージ。

00 Reserved for extensions [1234]

拡張用にリザーブされている。

01 STOP (MCS コマンド) [1234]

できるだけ早く停止する。

モニター出力は, もしも STOP MODE 情報フィールドがサポートされている場合は, これに従う。

STOP 状態は, 他の MCS あるいは MCP コマンドを受け取ると取り消される。

レコーディング[リハーサル]中のトラックはレコード[リハーサル]を止める。

```
01          STOP
```

02 PLAY (MCS コマンド) [-234]

プレイバック・モードに入る。

PLAY 状態は, 他の MCS あるいは MCP コマンドを受け取ると取り消される。

```
02          PLAY
```

＜備考＞

レコーディング[リハーサル]中のトラックは, この PLAY コマンドを受け取っても, 自動的にレコード[リハーサル]を止めることはない。その動作が望まれる場合は <RECORD EXIT> <PLAY> と送る。

03 DEFERRED PLAY (MCS コマンド) [1234]

PLAY コマンドと同等だが, 装置がこの時, もしも LOCATE(MCP)の実行中であった場合には, ロケート動作が完了するまでは強引に PLAY 状態に入ることはないという点が異なる。

他の MCS あるいは MCP コマンドを受信すると, この DEFERRED(保留付き)PLAY 状態は取り消される。

ロケート動作の実行中にこのコマンドを受け取った場合は, MOTION CONTROL TALLY 情報フィールド中の "MCP 成功レベル" 部は, ロケートしている間はずっと, "Deferred Play を保留にして, ロケートを実行中" なる旨を示していなければならない。

ロケートが完了すると:

(i) 自動的に PLAY コマンド(MCS)が出さる。

(ii) MOTION CONTROL TALLY 中の "MCS コマンド" バイトは "PLAY" に変わる。

(iii) "MCP コマンド" バイトは "MCP は現在アクティブでない" に変えられ, それまでの "LOCATE" や "Deferred Play" 標識は共にクリアーされる。

装置が LOCATE コマンドを実行中でなかったり, サポートしていなかった場合には, このコマンドではただちにプレイバック・モードに入るべきである。

```
03          DEFERRED PLAY
```

＜備考＞

1. もしもある装置が1個だけプレイ・コマンドをサポートするのであれば，DEFERRED PLAY が望ましい。オープン・ループの場合を考慮すると，“ロケート完了”ステータスが得られないので制御装置にとっては，このコマンドの動作内容を他の方法で実現することはできない。ところが即実行タイプの PLAY の方は，STOP に続いて DEFERRED PLAY を出すことで置き換えようがあるのである。
2. レコーディング［リハーサル］中のトラックは，DEFERRED PLAY コマンドを受け取っても自動的にレコード［リハーサル］を止めることはない。その動作が望まれる場合は <RECORD EXIT> <DEFERRED PLAY> と送る。

04 FAST FORWARD（MCS コマンド） ［1234］

可能な範囲での最大スピードで前方向に早送りする。

モニター出力は，もしも FAST MODE 情報フィールドがサポートされている場合は，これに従う。

FAST FORWARD 状態は，他の MCS あるいは MCP コマンドを受け取ると取り消される。

レコーディング［リハーサル］中のトラックはレコード［リハーサル］を止める。

```
04          FAST FORWARD
```

05 REWIND（MCS コマンド） ［1234］

可能な範囲での最大スピードで逆方向に戻す。

モニター出力は，もしも FAST MODE 情報フィールドがサポートされている場合は，これに従う。

REWIND 状態は，他の MCS あるいは MCP コマンドを受け取ると取り消される。

レコーディング［リハーサル］中のトラックはレコード［リハーサル］を止める。

```
05          REWIND
```

06 RECORD STROBE ［1234］

制御される装置を RECORD MODE 情報フィールドの設定内容に応じて，レコード・モードなりリハーサル・モードに入れたり，あるいはこれを止めたりさせる。

RECORD STROBE コマンドは2種類の条件下でのみ効果を有する。

条件1： 制御される装置がすでにプレイ中

もし制御される装置がすでにプレイ中（すなわち，MOTION CONTROL TALLY 情報フィールド中の“直前に実行された MCS”が PLAY か VARIABLE PLAY）であれば，この RECORD STROBE コマンドは，この時点でレコード・レディ状態にセットされているトラックすべてをレコード［またはリハーサル］状態にし，現在はレコーディング中［リハーサル中］であるが，もはやレコード・レディにはセットされていないトラックに対しては，レコード［リハーサル］動作を止めさせる。 *1 *8 *9

条件2： 制御される装置が停止中

RECORD STROBE コマンドを受け取った時に，もし制御される装置が明確な STOP あるいは PAUSE コマンドの結果として完全に停止中であれば，（すなわち，［i］MOTION CONTROL TALLY 情報フィールド中の“直前に実行された MCS”が STOP か PAUSE であって，［ii］“MCS 成功レベル”が“完全に停止している”を示しており，かつ［iii］“直前に実行された MCP”が“MCP は現在アクティブで無い”になっている場合には）

(i) 自動的に PLAY（MCS）コマンドが出され，*3 *4

(ii) 装置の立ち上がりフェーズ中の適当な時点で，この時にレコード・レディ状態にセットされているトラックすべてに対しレコード［リハーサル］動作を開始させる。 *1 *5

06 RECORD STROBE

＜備考＞

*1. TRACK RECORD READY 情報フィールドを用いて，各トラックはレコード・レディ状態に入れたり外したり切り替えられる。

2. 制御される装置を MMC に準拠して新たに設計する際には，少なくともひとつのトラックがレコード・レディ状態となっているのでない限り，RECORD STROBE コマンドが来てもレコード［リハーサル］状態にはしないようにすることが推奨される。

しかしながら現行の非 MMC 装置の間では，（例えば，RECORD MODE 情報フィールドに適当な値をセットする等により）レコード［リハーサル］がイネーブルになっていると，レコード［リハーサル］のコマンドを受け取った際に，たとえその時いずれのトラックもレコード・レディとなっていなくても，レコード［リハーサル］動作を開始させるといったことは，ごく普通に見られる。このような動作は，MMC の下でこれからも許されるものとする。ただし，この結果得られるステータス上は，RECORD STATUS 情報フィールドを見ると“アクティブなトラックはない”のビットが立っていることで，正しく示されることを条件とする。

*3. 条件2の下で PLAY（MCS）コマンドを自動的に出すのは，STOP または PAUSE 状態がはっきりしている場合に限られる。これ以外の場合，RECORD STROBE コマンドは，プレイ・モードやプレイ・スピードに関していかなる関りも持たない。

*4. 同じく条件2の下で PLAY（MCS）コマンドを自動的に出すことそのものは，レコード・レディのトラックがあるかどうかや，RECORD MODE がどうなっているかには関わりなく行われる。

*5. RECORD-PAUSE 状態があることによって，RECORD STROBE の動作内容に影響することはない（PAUSE 及び RECORD PAUSE コマンドを参照のこと）。

6. RECORD MODE 情報フィールドをサポートしており,それが“VTR: Insert”や“VTR:Assemble”にセットされた装置において，上に概説した条件12の下でレコード［リハーサル］状態に移る際にはクリーンで正しい遷移過程を取ることが期待されている。（条件2の）STOP や PAUSE モードから立ち上がる際にはレコード［リハーサル］を始める前に,まずは装置の立ち上がりフェーズの完了するまで待つことが通常は必要となるであろう。

7. 制御される装置が，すでにプレイ状態になっているのでなければ完全なストップ状態でもない時に，RECORD STROBE コマンドを受け取った場合には，これを無視することとする。

*8. 条件1「制御される装置がすでにプレイ中」の下で RECORD STROBE が受け付けられるのに，PLAY や VARIABLE PLAY コマンドは必ずしも”成功した”状態にまでなっている必要はない。しかしながら,RECORD STROBE を受信した時に，もしもそのプレイ動作が未だ確立していなかった場合には，装置はその立ち上がりフェーズ中の適当な時点までレコード［リハーサル］に入るのを遅らせる必要があろう。

*9. さらにまた条件1の下では，MCP の実行中だからといって，レコード動作の妨げられることはないことにも注意されたい。

＜日本語版注＞

10. RECORD MODE のデフォルトが“Disabled”であるようなレコーダーに対して，MODE の設定をせずに本コマンドを送っても，RECORD 状態にはならない。この不具合は実際に発生している。

本コマンドの名前が示す意味は，「（RECORD 関係の設定― RECORD MODE や，TRACK RECORD READY ―はあらかじめ行っておいて，）設定を実際に適用するタイミングを与える」ということである。コントローラー側は，「ストローブに先立って必ず設定を行う」という動作にすることを推奨する。

MODE の設定をしないコントローラーが現に存在するので，レコーダ側は，「Record が可能な状態をデフォルトとする」または，「ユーザが RECORD MODE（あるいはそのデフォルト）を変えられる」という仕様にすることを推奨せざるを得ない。

11. 従来からある伝統的なテープ装置に対しては，ストップ状態にある時にコマンド列 <PLAY> <RECORD STROBE>，あるいは単独のコマンド <RECORD STROBE> のいずれを受け取った場合でも，通常のレコード［リハーサル］動作が正しく始まることが想定されている。前者には条件1と注8が，後者には条件2が適用され，まったく同等の効果を持つことが期待されている。これとは違って，電子的な補償手法などにより立ち上がりフェーズを実効的になくした装置においては，後者の単独コマンドであれば，停止位置直後からのレコード［リハーサル］が期待できるのに対し，前者の方式ではレコード［リハーサル］状態に変わる前に再生動作が実行され位置が移動し，その間の距離と時間が無視できない等の有意の差が生じ得る。これらは MMC の許容する“装置による違い”であり，コントローラーにはこうした事情に関する理解が期待されている。

07 RECORD EXIT [1234]

現在レコーディング中のトラックすべてについて，レコード動作をやめさせる。

```
07          RECORD EXIT
```

08 RECORD PAUSE [----]

すでに PAUSE モードになっている(すなわち，直前のモーション・コントロール・ステートが PAUSE)との条件の下で，制御される装置を RECORD PAUSE モードにする。*4

RECORD PAUSE コマンドによって実際にレコード動作が起動されることはないが，装置は迅速かつ滑らかにレコード・モードへ移行できるような状態に置かれる。レコード・レディにセットされているトラックの出力部は，すべてそれぞれの入力信号をモニターする状態に切り替えられている。

制御される装置がこの RECORD PAUSE コマンドをサポートする場合には，本項及び次の PAUSE コマンド自体の項に述べた意味で，RECORD PAUSE モードを完全にインプリメントしていなくてはならない。

制御される装置がこのコマンドをサポートしない場合は，決して RECORD PAUSE モードになってはならない。

ひとたび RECORD PAUSE 状態となると，続くコマンドの効果は以下の通り。

PAUSE	:変化なし
RECORD PAUSE	:変化なし
RECORD EXIT	:RECORD PAUSE 状態は取り消され，PAUSE 状態となる
RECORD STROBE	:スムースにレコード(PLAY かつ Record)を再開する *6
RECORD STROBE VARIABLE	:スムースにレコード(Vari PLAY かつ Record)を開始 *6
その他の MCS や MCP	:RECORD PAUSE モードを抜けた後，次の動作に移行 *7

RECORD PAUSE 状態は，RECORD STATUS 情報フィールドにタリーされる。

```
08          RECORD PAUSE
```

＜備考＞

1. RECORD PAUSE コマンドは，トランスポートを起動させることなしに，非記録の PAUSE 状態から RECORD PAUSE 状態に入れる唯一の方法である。
2. RECORD PAUSE コマンド単独では，PAUSE モードにすることはない。
3. コマンド列 <PAUSE> <RECORD PAUSE> は，必ず PAUSE 状態を生み出し，もしインプリメントされていれば，RECORD PAUSE 状態にする。

*4. RECORD PAUSE コマンドが受け付けられる条件として，PAUSE コマンドが“成功した”とのレベルで確立していることは，必ずしも要求されていない。したがって，<PAUSE> <RECORD PAUSE> のコマンド・シーケンスを受信した際に，PAUSE コマンドに伴ってすべての動作が停止するまでの間，RECORD PAUSE にするのを待たせるのは制御される装置の責任範囲である。RECORD PAUSE によって，実際にレコード動作が行われることは決してあってはならない。

5. モーション・コントロール・ステートが PAUSE になっていない場合には，制御される装置は RECORD PAUSE コマンドを無視して良い。

*6. RECORD STROBE 及び RECORD STROBE VARIABLE コマンドの定義部も参照のこと。

*7. このカテゴリーには PLAY, DEFERRED PLAY, VARIABLE PLAY, DEFERRED VARIABLE PLAY も含まれ，これらのいずれも，RECORD PAUSE に続いてレコード動作を始めることはない。

8. 現時点では，REHEARSE PAUSE モードはサポートされていない。

09 PAUSE（MCS コマンド） [----]

できるだけ早く停止する。

VTRやその他の映像機器は，「絵を伴った」状態で停止する。

トランスポートの機構部は，次の立ち上がり時間が最小となるような状態に置かれる。

PAUSE 状態は，他の MCS あるいは MCP コマンドを受け取ると取り消される。

もしも装置が RECORD PAUSE コマンドをサポートしていれば，その場合に限り，実際にレコードが行なわれている最中に PAUSE コマンドを受け取ると，RECORD PAUSE 状態となる。レコーディング中のトラックは，この例外を除いてはレコードを止める（RECORD PAUSE コマンドの説明を参照されたい）。リハーサル中のトラックは，常にリハーサル動作から抜ける。

09 PAUSE

＜備考＞

1. カセット・タイプのVTRに時折見かけるような，PAUSE コマンドが繰り返し出されると，そのつど PAUSE と PLAY が切り替わる動作は，このコマンドでは行なわないものとする。
2. VTRにおいては，テープがヘッドに絡み付いたりするのを防ぐために，ある決められたタイム・アウト時間を経過すると「絵の出ない」状態に切り替わることがある。ここで PAUSE コマンドが出された際には “絵の出る” 状態に戻るものとする。
3. PAUSE コマンドは，自動的な “スタンバイ・オン” または “レディ” といったコマンドを，こうした動作をサポートする装置に対しては，意味する。
4. そのままでは独立のポーズ機能をサポートしていない装置であっても，これを STOP で置き換えることにより，PAUSE コマンドを用意しているものとしても良い。ただしこの時，MOTION CONTROL TALLY はあくまで PAUSE でなければならない。

0A EJECT（MCS コマンド） [----]

取り外し可能なメディア（カセットやディスクなど）の装置において，トランスポートの機構部からメディアをイジェクトする。

MOTION CONTROL TALLY 情報フィールドでタリーとして用いられる場合には，EJECT はメディアのないこと，そしてこれはオペレーターによる介入なしには回復できない状態にあることを意味している。

レコーディング［リハーサル］中のトラックは，レコード［リハーサル］を止める。

0A EJECT

＜備考＞

EJECT は通常，これをコマンドとしてはサポートしていない装置においても，タリーとしては用いられることがある。例えば，リール・トゥー・リール・タイプの装置では，テープがリールの端まで巻き取られてしまった場合には，これを EJECT のタリーで示すべきである。

0B CHASE（MCP コマンド） [---4]

制御される装置が，SELECTED MASTER CODE に追従し，同期を確立し，これを維持するようにさせる。SELECTED MASTER CODE が PLAY モードにあることが検出されると，制御される装置はプレイに入り，同期するように努めなくてはならない。それ以外の場合には，制御される装置はひたすら，SELECTED MASTER CODE が PLAY モードに入った時に，できるだけ少ない時間で同期が得られるような位置に持って行くように努めるべきである。

同期状態は，制御される装置の SELECTED TIME CODE と SELECTED MASTER CODE との間で，REQUESTED OFFSET 情報フィールドに指定された，以下の式で定義される量のオフセットを保持しつつ，確立されるべきである。

REQUESTED OFFSET = SELECTED TIME CODE - SELECTED MASTER CODE

CHASE コマンドを受け取った時に，もしも REQUESTED OFFSET 情報フィールド中の「ブランク」ビットがセットされていた場合には，(すなわち，タイムコードの値が実際には未だ1度も読み込まれていない時には) "26タイムコードが来ていない" の COMMAND ERROR レスポンスを生成し，チェイス・ステータスは "失敗した" にセットするものとする。

制御される装置が，SELECTED TIME CODE なり SELECTED MASTER CODE なりのブランク・ビットをそれ自身の動作によっては取り除くことができない場合にも，上と同じエラーを報告すること。

CHASE MODE 情報フィールドは，実行すべき同期のタイプを規定する。

CHASE 状態は，他の MCS あるいは MCP コマンドを受け取ると取り消される。

```
0B          CHASE
```

0C　COMMAND ERROR RESET　[-234]

COMMAND ERROR 情報フィールド中の「エラー停止」フラグをリセットし，制御される装置内でのコマンドの処理が再開できるようにする。COMMAND ERROR 及び COMMAND ERROR LEVEL 情報フィールドの解説を参照。

```
0C          COMMAND ERROR RESET
```

0D　MMC RESET　[1234]

制御される装置内の MIDI マシン・コントロールに関わる通信チャネルを，電源オン時の状態にリセットし，あわせて：

(a) UPDATE リストを空にする。
(b) すべての PROCEDURE を削除する。
(c) キベての EVENT を削除する。
(d) すべての GROUP 指定を解除する。
(e) 「エラー停止」フラグも含めて COMMAND ERROR 情報フィールドをクリアーする。
(f) COMMAND ERROR LEVEL をゼロ("すべてのエラーをディスエーブルに")にする。
(g) すべてのタイムコード関連情報フィールド中のフラグを，付録Bに述べられているそのデフォルトにセットする。
(h) MOTION CONTROL TALLY 情報フィールド中の「MCP コマンド・タリー」バイトを 7F("現時点で有効になっている MCP はない")にリセットする。
(i) コマンド・システム・エクスクルーシブのセグメント復元の実行中であれば，これを打ち切る。

MMC に対応するすべての制御される装置は，この MMC RESET コマンドをサポートしていなければならない。

```
0D          MMC RESET
```

40　WRITE　[1234]

指定された情報フィールド(複数可)にデータを書き込む。

WRITE コマンドの後には，情報フィールド名とこれに続いてこの情報フィールドに書き込むべき完全なデータとからなる，1個ないしそれ以上のデータ列が置かれる。

情報フィールドのデータ・フォーマットは，「レスポンス，情報フィールド詳説」編で定義されれている通りでなくてはならない。

指定の情報フィールドは書き込み可能でなくてはならない。

40	**WRITE**
<count=variable>	**後に続く情報のバイト数**
<name>	**書き込み可能な情報フィールド名**
<data . .>	**この情報フィールド名によって規定されたフォーマット**
<name>	**必要に応じてさらに <name> <data . .> の組**
<data . .>	
:	
:	

41　MASKED　WRITE　[--3-]

ビット・マップ型の情報フィールド(備考 * 1参照)において，特定ビットだけを変えられるようにする。

この“マスクド・ライト”が可能であるためには，情報フィールドは <count> <data> 型のフォーマットであり，かつビット・マップは <count> バイトの直後から始まっていなければならない。

41	**MASKED WRITE**
<count=04+ext>	**後に続く情報のバイト数**
<name>	**マスク式の書き込み可能な情報フィールド名 *1**
<byte #>	**ビット・マップ中の目的とするバイトのバイト番号。バイト0は，マスク式の書き込み可能な情報フィールドの定義中で，<count> フィールドの後の先頭バイト**
<mask>	**このマスク中で1は，目的とするビット・マップ・バイト中で変えようとするビットの位置を示す。**
<data>	**目的ビット・マップ・バイトに置かれる新しいデータ**

＜備考＞

* 1.　現時点で定義されているマスク式の書き込みが可能な情報フィールドは，標準トラック・ビット・マップ形式を採っているものに限られている(「3 標準仕様」参照)。

2.　<mask>,<data> のいずれにおいても，「すべて1」を指定する値は言うまでもなく 7F である。

42　READ　[-234]

指定した情報フィールド(複数可)の内容を送信するよう，要求する。

制御される装置がその情報フィールドをサポートしていない場合には，RESPONSE ERROR メッセージが送り返される(RESPONSE ERROR 情報フィールドの説明参照)。

42	**READ**
<count=variable>	**バイト数(コマンドとカウントは含まない)**
<name>	**情報フィールド名のリスト**
:	
:	

43　UPDATE　[-234]

フォーマット1：UPDATE[BEGIN]

UPDATE[BEGIN]は，制御される装置に対してただちに次の動作を起こさせる。

(i)　指定した情報フィールド(複数可)の内容を送り返すこと。

(ii)　指定した情報フィールド(複数可)の名前を内部の「アップデート・リスト」に加えること。 *1

これに続けて，UPDATE RATE 情報フィールドで規定された時間以上の間隔を置いて，制御される装置は：

(iii)　内部「アップデート・リスト」に登録されている情報フィールドのいずれについても，その内容が変わり，かつ UPDATE コマンドの結果として送信した直前の内容とこれが異なりしだい，これを送信する。 *2　*3

制御される装置が指定された情報フィールドのいずれかをサポートしていない場合(あるいは定義されていなかったり「アクセス不可」の項目として定義されている場合)には，このエラーを生じさせている情報フィールド名を含んだ RESPONSE ERROR メッセージが生成される(同じ UPDATE[BEGIN]リクエスト中の問題ない項目に関しては，通常通り“アップデート・リスト”に加える)。RESPONSE ERROR メッセージは1回だけ送信され，繰り返して送られることはない。 *4

制御される装置においては,UPDATE コマンドの出されるごとに,UPDATE の対象となる情報フィールドの内部「リスト」は累積される。

リクエストされた情報フィールドがタイムコード・フィールドであった場合には，(直後に送る)最初の UPDATE レスポンスは，常に(5バイト長の)標準タイムコード・フォーマットを用い，これより後に続くレスポンスにおいては，適用できる限りは(2バイト長の)標準短形式タイムコード・フォーマットを用いる(「3 標準仕様－標準短形式タイムコード」の項参照)。

この「リスト」から項目を除くには，UPDATE[END]を用いる。

MMC RESET コマンドで，この「リスト」は完全にクリアーされ，すべての UPDATE レスポンスも止まる。

43	**UPDATE[BEGIN]**
<count=variable>	**バイト数(コマンドとカウントは含まない)**
00	**“BEGIN”のサブコマンド**
<name>	**情報フィールドの名前のリスト　*5**
:	
:	

＜備考＞

* 1　もしも新たにリクエストのあった情報フィールド名が，すでに内部「アップデート・リスト」中に含まれていた場合には，期待されている通りにこの情報フィールドの内容は，ただちに1回送信される。内部の「リスト」は変わらない。
* 2.　もしもある情報フィールドの値が，最後の UPDATE 送信以後に2回以上変わっていた場合には，最新のデータだけが送り返される。
* 3.　もしもある情報フィールドの値が,最後の UPDATE 送信以後に2回以上変わっていた場合であっても,最後に送った値と同じに戻った場合には，UPDATE レスポンスを改めて送る必要はない。
* 4.　RESPONSE ERROR 情報フィールドの説明の項も参照のこと。
* 5.　タイムコード情報フィールド名は，標準タイムコードのもの(01 ～ 1F)か，対応する標準短形式タイムコードのもの(21 ～ 3F)のいずれかで指定される。どちらの場合であっても，結果の UPDATE 動作には違いは生じない。

フォーマット2：UPDATE[END]

制御される装置は，指定された情報フィールド（複数可）をそのアップデート・リストから取り除き，その内容を自動的に送り続けることを止めなければならない。

指定の名前が現在のアップデート・リスト中にない場合でも，エラーは生じない。

43	**UPDATE[END]**
<count=variable>	**バイト数（コマンドとカウントは含まない）**
01	**“END”のサブコマンド**
<name>	**（取り除く）情報フィールドの名前のリスト**
:	このリスト中のどこかに 7F のある場合は，すべてのアップデート動作を停止する。

＜備考＞

1. MMC RESET コマンドは常に，アップデート・リストを空にし，すべてのアップデート動作を停止させる。
2. [BEGIN]か[END]以外のサブコマンドを指定した UPDATE コマンドは，“規定のサブコマンドでない”の COMMAND ERROR レスポンスを生じるものとする。

44 LOCATE（MCP コマンド） [1234]

フォーマット1：LOCATE[I/F]

制御される装置を，SELECTED TIME CODE の値を基準にして，指定の情報フィールドにセットされているタイムコード位置にまで送る。 *1 *2

もしも指定された情報フィールドの“ブランク”ビットがセットされていたら（すなわち，タイムコードの値が未だ1度も読み込まれていない場合は），“26：タイムコードが来ていない”の COMMAND ERROR が返され，MOTION CONTROL TALLY 情報フィールド中の“MCP 成功レベル”は“失敗した”にセットされる。

DEFERRED PLAY（MCS）と DEFERRED VARIABLE PLAY（MCS）コマンドの場合を除いては，LOCATE[I/F]動作は，他の MCS あるいは MCP コマンドを受け取ると取り消される。

44	**LOCATE[I/F]**
<count=02+ext>	**バイト数（拡張セットを用いなければ count=02）**
00	**“I/F”サブコマンド**
<name>	**有効な情報フィールド名**
	00 = 拡張のためリザーブされている
	08 = GP0/LOCATE POINT
	09 = GP1
	0A = GP2
	0B = GP3
	0C = GP4
	0D = GP5
	0E = GP6
	0F = GP7

フォーマット2：LOCATE[TARGET]

制御される装置を，SELECTED TIME CODE の値を基準にして，コマンド・データ中に指定されたタイムコード位置にまで送る。

DEFERRED PLAY（MCS）と DEFERRED VARIABLE PLAY（MCS）コマンドの場合を除いては，LOCATE［TARGET］動作は，他の MCS あるいは MCP コマンドを受け取ると取り消される。

44	**LOCATE[TARGET]**

<count=06>	**バイト数**
01	**“TARGET”サブコマンド**
hr mn sc fr ff	**サブフレーム付き標準時刻仕様（{ff} タイプ）**

＜備考＞

*1. LOCATE[IF]コマンドのためには，GP0 から GP7 までのうち少なくとも1個の汎用レジスターのサポートされている必要がある。

*2. ひとたび LOCATE[IF]が開始すると，指定の情報フィールドの内容がこの後どのように変えられようとも，ロケートの実行に影響を与えることはない。言い替えれば，ロケート・ポイントの時刻値は，LOCATE コマンドの受信された時に指定の汎用レジスターから読み取られ，これとは別の，特に規定はしない内部のロケート・ポイント・レジスターに移される。

3. サブフレーム精度でのロケートのできない装置にあっては，ロケート・ポイント指定中のサブフレーム・バイトは無視すること。

4. ロケート動作の完了時には，制御される装置が PAUSE コマンドをサポートしている場合には，この PAUSE 状態になるべきである。そうでない場合は，ロケートは通常の STOP コマンドで終了し，できればモニターについては，STOP MODE 情報フィールドの指定に従うものとする。

5. DEFERRED PLAY 及び DEFERRED VARIABLE PLAY コマンドの説明の項も参照のこと。

6. [I/F]や[TARGET]以外のサブコマンドを指定した LOCATE コマンドに対しては，“規定のサブコマンドでない”の COMMAND ERROR が発生する。

45 VARIABLE PLAY（MCS コマンド） [-234]

指定された方向とスピードの可変速プレイ・モードに滑らかに入る。

もしも要求されたスピードの値が，制御される装置の可能な範囲を越える場合には，「できる範囲での最も近いスピード」で動作するものとする。

VARIABLE PLAY は，他の MCS あるいは MCP コマンドを受け取ると取り消される。

45	**VARIABLE PLAY**
<count=03>	**バイト数**
sh sm sl	**標準スピード仕様**

＜備考＞

レコーディング[リハーサル]中のトラックは，この VARIABLE PLAY コマンドを受け取っても自動的にレコード[リハーサル]を止めることはない。その動作が望まれる場合は <RECORD EXIT> <VARIABLE PLAY> と送る

46 SEARCH（MCS コマンド） [--34]

制御される装置を，指定の方向へ指定の早さで進める。

モニター出力は可能でなければならないが，その品質は，記録されている素材が認識できる程度を満たしていさえすれば良い。

もしも要求されたスピードの値が制御される装置の可能な範囲を越える場合には，「できる範囲での最も近いスピード」で動作するものとする。極端な場合には，それぞれの方向にひとつの固定速でしかサーチのできない装置であっても，SEARCH コマンドをサポートしていると言ってもかまわない。

サーチ状態は，他の MCS あるいは MCP コマンドを受け取ると取り消される。

レコーディング中[リハーサル]のトラックはレコード[リハーサル]を止める。

46	**SEARCH**
<count=03>	**バイト数**
sh sm sl	**標準スピード仕様**

＜備考＞

絵と音の両出力をもつ装置においては，SEARCH の間は絵のモニターだけが要求されている。音の同時モニターについては，装置の製造者の裁量範囲に属する。

47 SHUTTLE（MCS コマンド） [----]

制御される装置を，指定の方向へ指定の速さで，必ずしも音や絵の再生は伴わずに進める。

もしも要求されたスピードの値が制御される装置の可能な範囲を越える場合には，「できる範囲での最も近いスピード」で動作するものとする。

シャトル状態は，他の MCS あるいは MCP コマンドを受け取ると取り消される。

レコーディング［リハーサル］中のトラックはレコード［リハーサル］を止める。

47	**SHUTTLE**
<count=03>	**バイト数**
sh sm sl	**標準スピード仕様**

48 STEP（MCS コマンド） [----]

制御される装置を，現在位置に対して前方向なり後ろ方向へ指定の距離だけ進める。

STEP 以外の他の MCS あるいは，MCP コマンドが次に来るまでは，同じコマンドが続くと次々に累積される。

ビジュアル装置においては，STEP 動作の間は少なくとも絵のモニターは可能でなければならず（音は任意とする），また STEP 動作の終了後も絵は持続していなければならない。（PAUSE モードと同様）オーディオ装置は，STEP 中は音のモニターができなければならず，また装置がディジタルの“ルーピング”機能を持つ場合には，STEP の完了後もループ動作を続けるべきである。

シーケンサーは STEP の間，MIDI 出力をイネーブルとすべきであり，また STEP の完了時にノートをオフにすることは避けるべきである。

いずれの場合にも，他の MCS や MCP コマンドの受信によって STEP モードの解除された後は，モニター機能は“通常”状態に復帰するものとする。

STEP 移動する距離は STEP LENGTH 情報フィールドで定義される単位長で測るものとする。無指定（ディフォルト）時の単位は1ビデオ・フィールド（1/2 フレーム）である。

48	**STEP**
<count=01>	**バイト数**
<steps>	**STEP LENGTH の倍数 : 0 g ssssss**
	g = 符号（1はリバース方向）
	ssssss = 量

49 ASSIGN SYSTEM MASTER [----]

ほとんどの“チェイス”同期システムでは，“マスター”装置を決める必要がある。そうすることで他の装置は，この装置からのタイムコードに追従し，同期することになる。

システム・マスターのアサイン（割り当て）を行うことは，この“マスター”装置のタイムコードをシステム内で適宜分配することになる。このタイムコードを受け取った装置では通常，その SELECTED MASTER CODE フィールド内へ，直接にあるい

は間接的にこれを取り込む。この分配の仕方やオペレーションの方法に関してこのコマンドでは特定の方法は指定しない。

マスター・タイムコードの流れを割り付けることはまた，基準時刻を定めることになるので，これを基にシステム全体のすべてのイベントに付いて，一貫した時間上で位置決めすることが可能となる。

ASSIGN SYSTEM MASTER コマンドに対する対応の仕方は，次の真理値表による。

自分の <device_ID> は本コマンドの指定と一致するか？	自分はすでにシステム・マスター装置になっているか？	取るべき行動
N	N	何もしない
N	Y	システム・マスターではなくなる
Y	N	新たにシステム・マスターになる
Y	Y	システム・マスターのままでいる

ASSIGN SYSTEM MASTER メッセージは，全員呼び出しデバイス ID(=7F)を通じて送信しなくてはならない。グループ・デバイス ID は，システム・マスターに指定してはならない。

49 **ASSIGN SYSTEM MASTER**
<count=01> **バイト数**
<device_ID> **(マスターに指定する)装置の ID 番号**
7F = 以前に指定したマスターを解除し，マスター無しにする

4A GENERATOR COMMAND [----]

タイムコード・ジェネレーターのランニング状態をコントロールする。

GENERATOR SET UP 情報フィールドをも参照のこと。

4A **GENERATOR COMMAND**
<count=01> **バイト数**
nn **動作**
00 = ストップ (Stop)
01 = ラン (Run)。GENERATOR SET UP 情報フィールドで指定されている場合は，そこでの指示通りにフレーム・ロックのこと。
02 = コピー／ジャム (Copy/Jam)。ラン中は GENERATOR SET UP 情報フィールドでの指示通りに，入力されているタイムコードを GENERATOR TIME CODE レジスターへ送る。

4B MIDI TIME CODE COMMAND [----]

MIDI タイムコードの生成を制御する。

MIDI TIME CODE SET UP 情報フィールドをも参照のこと。

4B
<count=01> **バイト数**
nn **動作**
00 = オフ(Off)
02 =MIDI TIME CODE SET UP 情報フィールドで指定されているタイムコードに従う。

4C　MOVE　[1234]

発信元(ソース)情報フィールドの内容を，宛先(デスティネーション)情報フィールドに転送する。宛先情報フィールドとして有効なのは，(コード01から1Fまでの間の)5データ・バイト・グループ中で，Read/Write 可能なフィールドである。

発信元情報フィールドで有効なのは，(コード01から1Fまでの間の)5データ・バイト・グループのものすべてである。

通常サブフレームを含んでいない({st} 型の)発信元の場合は,サブフレームの値はすべてゼロであると想定すべきである。発信元が({ff} 型で)サブフレームを含んでいても，宛先が({st} 型で)含まない場合には，サブフレーム・データは失われる。

4C	**MOVE**
<count=02+ext>	**バイト数(コマンドとカウントは含まない)**
<destination>	**有効な宛先情報フィールド名**
<source>	**有効な発信元情報フィールド名**

＜備考＞

1. タイムコード中に埋め込まれているフラグ類は(「3 標準仕様」参照)，次の例外を除いてすべて，発信元のフィールドから宛先フィールドに転送される。
   (a) {st} 型のフィールドから {ff} 型のフィールドに MOVE する場合は，サブフレームを 00 とすると共に，ビット i を 0 にリセットすること。
   (b) {ff} 型のフィールドから {st} 型のフィールドに MOVE する場合は，i=1，e=0，v=0 とし，d は決まりに従って 0 なり 1 に，n は 1 にそれぞれセットする。
2. MOVE コマンドは，動いているタイムコードの流れから即座に値を取り出すのに使える。例えば現在の SELECTED TIME CODE の値をロケート・ポイント・レジスター GP0 に移したり，あるいは現在の ACTUAL OFFSET を REQUESTED OFFSET に MOVE することで確保することができる。

＜日本語版注＞

3. ここに現われる <source> 及び <destination> は，上に示した意味で使われており，コマンド及びレスポンスのメッセージの構文定義で使われているものとは異なる。混同なきよう注意されたい。

4D　ADD　[-234]

2つのソース情報フィールドの内容を加え，結果をデスティネーション情報フィールドに置く。

[Destination]=[Source #1]+[Source #2]

結果は 00:00:00:00.00 から +/-23:59:59:nn.99 までの間の有効なタイムコード値である。ここに，nn は計算に用いられたフレーム・レートによる(負の結果が許されるかどうかは，デスティネーション情報フィールドの性質で決まる)。ソースのいずれかがドロップであるかノンドロップであるかにかかわらず，結果は常にノンドロップ・フレームの量として表される。

有効なデスティネーション情報フィールドは，(コード01から1Fまでの間の)5データ・バイト・グループ内の Read/Write 可能なフィールドである。

有効なソース情報フィールドは，(コード01から1Fまでの間の)5データ・バイト・グループのすべてである。

デスティネーション・フィールドがソース・フィールドの一方と同じであることはかまわない。この場合，計算の完了しないうちにソースのデータを壊してしまうことのないように注意が必要である。

通常サブフレームを含んでいない({st} 型の)ソースについては,すべてサブフレームの値はゼロであると想定すべきである。

いずれかのソースが({ff} 型で)サブフレームを含んでいても，デスティネーションが({st} 型で)サブフレームを含まない場合には，サブフレームのデータは失われる。

4D	**ADD**
<count=03+ext>	**バイト数(コマンドとカウントは含まない)**
<destination>	**有効なデスティネーション情報フィールド名**
<source #1>	**有効なソース情報フィールド名**
<source #2>	**有効なソース情報フィールド名**

＜備考＞

1. いずれのソース・フィールドについても，そのフレーム・レートやドロップ・フレームのステータスは，そのフィールド中に埋め込まれているタイムコード・ステータス・ビットで完全に確立している。TIME STANDARD 情報フィールドの内

容がこの計算に関わり合うことはない。

2. デスティネーション・フィールドが {ff} 型の場合(「3 標準仕様」参照), 埋め込まれるタイムコード・ステータス・フラグは, 次のようにセットされる。

tt(フレーム・タイプ)	:ソース #1 に同じ。ただしソース#1がドロップ・フレームの場合には, デスティネーションはノンドロップに変えられる。
c(カラー・フレーム)	:0
k(ブランク) :0	
g(符号)	:計算結果で決まる。もしデスティネーションが符号付きの負数データを受け入れない場合は, 負の結果にはまず"4時"を加えることで正の値に変えること。
(最終バイトID)	:0

3. デスティネーション・フィールドが {st} 型の場合(「3 標準仕様」参照), 埋め込まれるタイムコード・ステータス・フラグは, 次のようにセットされる:

tt,c,k,g	{ff} 型の場合に同じ
i(最終バイトID)	:1
e(推測コード)	:0
v(不正コード)	:0
d(ビデオ・フィールド1)	:決まりに従って 0 なり 1 に
n(タイムコード無し)	:1

4. 多くの装置において, 30 フレームのドロップとノンドロップとが混じっていても計算ができるものと思われる。しかしながら, 他のフレーム・レートでマッチしない場合には, 正しい結果の出せるものは極めて少ない。
5. ドロップ・フレーム・コードを含む計算は, 24 時の境界を「またがる」場に予測できない結果を生じることがある。
6. 「MOVE」の備考3参照。

4E SUBTRACT [-234]

ひとつのソース情報フィールドの内容をもう一方の内容から引き, 結果をデスティネーション情報フィールドに入れる。

[Destination]=[Source #1]-[Source #2]

ADD コマンドの条件はすべて, SUBTRACT コマンドにも当てはまる。

4E	**SUBTRACT**
<count=03+ext>	**バイト数(コマンドとカウントは含まない)**
<destination>	**有効なデスティネーション情報フィールド名**
<source #1>	**有効なソース情報フィールド名**
<source #2>	**有効なソース情報フィールド名**

4F DROP FRAME ADJUST [--34]

指定の情報フィールドの内容を, ドロップ・フレーム・フォーマットに変換する(「3 標準仕様」節中の「ドロップ・フレームの扱い」の項参照)。

もしもその内容がこの時点で 30 フレームのノンドロップ・フォーマットで表されていない場合には, 何もしない。

ADD あるいは SUBTRACT 演算の後で, ドロップ・フレームの結果を得るために用いることができる。

有効な情報フィールドは, (コード 01 から 1F までの間の)5データ・バイト・グループ中の, Read/Write 可能なフィールドである。

4F	**DROP FRAME ADJUST**
<count=01+ext>	**バイト数(コマンドとカウントは含まない)**

<name>　　有効な情報フィールド名

＜備考＞

指定情報フィールドのフレーム･レートやドロップ･フレームのステータスは，そのフィールド中に含まれているタイムコード・ステータス・ビットで完全に確立している。TIME STANDARD 情報フィールドの内容がこの計算に関わり合うことはない。

50　PROCEDURE　[--34]

プロセジャー（手続き）とは，コマンド群のつながり（ストリング）であって，コントローラーがこれを定義し，制御される装置内に保持される。いったん定義された後では,単に PROCEDURE[EXECUTE]コマンドを送るだけで実行されることになる。

フォーマット1：PROCEDURE[ASSEMBLE]

後での実行に備えてコマンド・ストリングを組み立てる。

プロセジャーは，PROCEDURE[DELETE]あるいは MMC RESET コマンドを受け取るまでは保持される。

すでに定義済みのプロセジャーを再定義することは，それまでの定義を先ずは削除することを意味するものとする。

（制御される装置は）プロセジャー中に組み込まれるコマンド群に付いては，“主要(Major)エラー”と“インプリメンテーション･エラー”がないかどうか，このプロセジャーのアセンブル時に予備チェックをしなければならない。もしもこうして埋め込まれるコマンド群の中にこれらのエラーの含まれる場合には：　*1

(i)　このプロセジャーはなかったものとする。

(ii)　COMMAND ERROR 情報フィールド中にそのエラー種別が記録される。

(iii)　（同じく COMMAND ERROR 情報フィールド中の）”PROCEDURE[ASSEMBLE]”エラーフラグがセットされる。

どんなにプロセジャー用記憶領域が利用可能であっても，それを受け入れるとオーバー・フローするようなプロセジャーが定義されようとした場合は，COMMAND ERROR 情報フィールド中の“プロセジャー・バッファ・オーバーフロー”エラーを発生する。

50　　PROCEDURE[ASSEMBLE]
<count=variable>　　バイト数
00　　“ASSEMBLE”のサブコマンド
<procedure>　　00 から 7E の範囲内のプロセジャー名（7F はリザーブされている）
<command #1>　　任意の MMC コマンドのリスト，ただし以下は除く
<command #2>　　(i)　別の PROCEDURE[ASSEMBLE]　*2　あるいは
<command #3>　　(ii)　現に定義しつつあるプロセジャー名を指定した
:　　PROCEDURE[EXECUTE]　*3

＜備考＞

* 1.　「主要(Major)エラー」と「インプリメンテーション・エラー」については，COMMAND ERROR 情報フィールドの解説部に説明がある。その備考2をも参照のこと。
* 2.　“ネストになった PROCEDURE[ASSEMBLE]”のエラーが引き起こされる。
* 3.　“リカーシブになった PROCEDURE[EXECUTE]”のエラーが引き起こされる。

フォーマット2：PROCEDURE[DELETE]

以前に定義されたコマンド・シーケンスを削除する。このコマンドの“全プロセジャーの削除”指定の場合を除いて，定義されていないプロセジャーを削除しようとした場合には，COMMAND ERROR 情報フィールド中の“未定義プロセジャー”エラーを発生する。

50	**PROCEDURE[DELETE]**
<count=02>	**バイト数**
01	**“DELETE”のサブコマンド**
<procedure>	**00 から 7E の範囲内のプロセジャー名**
	7F は“全プロセジャーの削除”を意味する

フォーマット3：PROCEDURE[SET]

この後の READ コマンドで, PROCEDURE RESPONSE 情報フィールドを指定された際に送るべきプロセジャーの名前（番号）をセットする。このコマンドの “全プロセジャーのセット” 指定の場合を除いて, 定義されていないプロセジャーの指定された場合には, COMMAND ERROR 情報フィールド中の “未定義プロセジャー” エラーを発生する。

50	**PROCEDURE[SET]**
<count=02>	**バイト数**
02	**“SET”のサブコマンド**
<procedure>	**00 から 7E の範囲内のプロセジャー名**
	7F は“全プロセジャーのセット”を意味する

フォーマット4：PROCEDURE[EXECUTE]

指定されたプロセジャーをただちに実行する。定義されていないプロセジャーの実行指示の場合には, COMMAND ERROR 情報フィールド中の” 未定義プロセジャー “エラーを発生する。

50	**PROCEDURE[EXECUTE]**
<count=02>	**バイト数**
03	**“EXECUTE”のサブコマンド**
<procedure>	**00 から 7E の範囲内のプロセジャー名**
	7F はリザーブされている

＜備考＞

1. MMC RESET コマンドは常に, すべてのプロセジャーを削除する。
2. [ASSEMBLE],[DELETE],[SET], それに[EXECUTE]以外のサブコマンドを指定した PROCEDURE コマンドに対しては, “規定のサブコマンドでない” の COMMAND ERROR を発生する。
3. EVENT[DEFINE]コマンド説明部の備考8に, PROCEDURE[ASSEMBLE]と PROCEDURE[EXECUTE]コマンドの例が示されている。

51　EVENT　[--34]

フォーマット1：EVENT[DEFINE]

制御される装置が，指定のタイムコード上の指定のトリガー時刻に，MIDI マシン・コントロールの任意のコマンドひとつを，実行することができるようにする。

すでに定義されているイベントを再度定義しようとした場合は，それまでの定義を先ずは削除することを意味するものとする。

どんなにイベント用記憶領域が利用可能であっても，それを受け入れるとオーバー・フローするようなイベントが定義されようとした場合は，COMMAND ERROR 情報フィールド中の“イベント・バッファ・オーバーフロー”エラーを発生する。

同様に，もしも指定された“トリガーソース”情報フィールドが何らかの理由で使えない場合には，“EVENT のトリガーソースが使えない，またはサポートされていない”のエラーを発生させ，当該イベントはなかったものとする。

（制御される装置は）イベント中に含まれるコマンドに付いては，“主要（Major）エラー”と“インプリメンテーション・エラー”がないかどうか，あらかじめイベントの定義時にチェックしなければならない。もしもこうして埋め込まれるコマンドがこれらのエラーを含む場合は：*9

(i)　このイベントはなかったものとする。

(ii)　COMMAND ERROR 情報フィールド中にそのエラー種別が記録される。

(iii)　(同じく COMMAND ERROR 情報フィールド中の）“EVENT[DEFINE]”エラー・フラグがセットされる。

51　**EVENT[DEFINE]**

<count=variable>　**バイト数**

00　**“DEFINE”のサブコマンド**

<event>　**00 から 7E の範囲内のイベント名（7F はリザーブされている）**

<flags>　**イベント・コントロール・フラグ：0 k 0 a 00 dd**

dd = 方向モード

00 = 前方向に進行中に限りトリガー

01 = 逆方向に進行中に限りトリガー

10 = どちら方向の進行中であってもトリガー

a = 全スピード・フラグ

0 = 固定速，あるいは可変速プレイのスピードで進行中に指定トリガー時刻と一致した場合に限りトリガー

1 = いかなるスピードであれ進行中に，指定トリガー時刻と一致したかあるいは通過したことの認知されしだいただちにトリガー

k = 非削除フラグ

0 = トリガーされるとただちにイベント定義を削除する（ES バス・モード）

1 = トリガーの後にもイベント・キューにイベント定義を残す *8

<trigger source>　**イベントがトリガーされる際に基準とするタイムコードを指定する情報フィールド名 *4**

00 = 拡張用にリザーブされている

01 = SELECTED TIME CODE *5

02 = SELECTED MASTER CODE

06 = GENERATOR TIME CODE

07 = MIDI TIME CODE INPUT

<name>　**トリガー時刻を保持する情報フィールド名　*3**

00 = 拡張用にリザーブされている

08 = GP0/LOCATE POINT

09 = GP1

0A = GP2

0B = GP3

0C = GP4
0D = GP5
0E = GP6
0F = GP7

<command..> **以下の場合を除く任意の単独コマンドと必要なデータ。*6**
(i) 別の EVENT[DEFINE]*10
or (ii)PROCEDURE[ASSEMBLE]*11

＜備考＞

1. イベント・コマンドのためには，GP0 から GP7 の汎用レジスターのうち，少なくとも1個はサポートされている必要がある。
2. トリガー時刻用レジスターに対して，この後どのように変化があっても，イベントについてはいっさい影響はない。言い換えれば，トリガー時刻はイベントが定義される時にこの汎用レジスターから読み取られ，内部の特に指定はしないイベント・トリガー時刻用エリアに移される。したがって EVENT RESPONSE においては，常にこの内部領域から時刻情報を送り返すことになる。

*3. サブフレーム精度でのイベント・トリガーをサポートするかどうかは，装置製造者の裁量範囲内である。

*4. 典型的なトリガーソースとしては SELECTED TIME CODE あるいは SELECTED MASTER CODE であろう。ひとつのトリガー・ソース（おそらくは SELECTED TIME CODE）のみをサポートしている装置においては，なんらかの制限事項が出て来よう。装置によっては複数のトリガー・ソースをサポートしているものがあろうが，サブフレームでのトリガーは，このうちのひとつのわずかなものに対してのみインプリメントされるであろう。

*5. タイムコードがなく，SELECTED TIME CODE が常にタコ・パルスによって更新されている場合には，その数の進み方は断続的であろうから，理想的なトリガー・ソースになっているとは言い難い。この問題はイベント定義の際に，“全スピード”でのトリガーと指定することで回避できるであろう（EVENT コマンドの解説を参照）。

*6 一度に複数のコマンドを実行させるには，コマンド PROCEDURE[EXECUTE]を用いること。

*7 前もって早目にトリガーする必要のある MIDI マシン・コントロールのコマンドは，制御される装置の側で検出し，そのトリガーのタイミングはそれなりに適宜早められるべきことが重要である（例えば RECORD STROBE は，記録開始の立ち上がり時の遅れを見込んで開始する必要がある）。この動作は，コントローラーにはトランスペアレントで，気にしなくて良いようになっていなければならない。

*8. 「非削除」モードを用いたイベントの例：

次のような簡単な「ルーピング」操作を考えて見る。テープ装置が「A」地点からプレイを開始し「B」地点にまで進めた後，この時点で「A」地点までロケートで戻り，再び始めから繰り返す。

```
F0 7F <device_ID> <mcc>
    <WRITE> <count=0C>
        <GP0> <5-byte loop start time“A”>
        <GP1> <5-byte loop end time“B”>
    <PROCEDURE> <count=06> <[ASSEMBLE]> <procedure_name>
        <LOCATE> <count=01> <GP0>
        <DEFERRED PLAY>
    <EVENT> <count=09> <[DEFINE]> <event_name>
        <flags=40> <SELECTED TIME CODE> <GP1>
        <PROCEDURE> <count=02> <[EXECUTE]> <procedure_name>
    <PROCEDURE> <count=02> <[EXECUTE]> <procedure_name>
F7
```

*9. 「主要（Major）エラー」と「インプリメンテーション・エラー」については COMMAND ERROR 情報フィールドの解説部に説明がある。その備考2をも参照のこと。

*10. “ネストになった EVENT[DEFINE]”のエラーが引き起こされる。

*11. “EVENT[DEFINE]中に PROCEDURE[ASSEMBLE]”のエラーが引き起こされる。

フォーマット2：EVENT[DELETE]

以前に定義されているイベントを削除する。

このコマンドの“全イベントの削除”指定の場合を除いて，定義されていないイベントを削除しようとした場合には，COMMAND ERROR 情報フィールド中の“未定義イベント”エラーを発生する。

51	**EVENT[DELETE]**
<count=02>	**バイト数**
01	**“DELETE”のサブコマンド**
<event>	**00 から 7E の範囲内のイベント名**
	7F は“全イベントの削除”を意味する

フォーマット3：EVENT[SET]

この後の READ コマンドで EVENT RESPONSE 情報フィールドを指定された際に送るべきイベントの名前番号をセットする。

このコマンドの“全イベントのセット”指定の場合を除いて，定義されていないイベントの指定された場合には，COMMAND ERROR 情報フィールド中の“未定義イベント”エラーを発生する。

51	**EVENT[SET]**
<count=02>	**バイト数**
02	**“SET”のサブコマンド**
<event>	**00 から 7E の範囲内のイベント名**
	7F は“全イベントのセット”を意味する

フォーマット4：EVENT[TEST]

指定されたイベントに含まれているコマンドを，あたかもトリガーがかかったかのように，ただちに実行する。イベントを実行後削除してはならない。

定義されていないイベントのテスト指示の場合には，COMMAND ERROR 情報フィールド中の“未定義イベント”エラーを発生する。

51	**EVENT[TEST]**
<count=02>	**バイト数**
03	**“TEST”のサブコマンド**
<event>	**00 から 7E の範囲内のイベント名**
	7F はリザーブされている

＜備考＞

1. MMC RESET コマンドは常に，すべてのイベントを削除する。
2. [DEFINE]，[DELETE]，[SET]，それに[TEST]以外のサブコマンドを指定した EVENT コマンドに対しては，“規定のサブコマンドでない”の COMMAND ERROR を発生する。

52　GROUP　[-234]

フォーマット1：GROUP[ASSIGN]

制御される装置は，受信したデバイス ID リスト中に自分の <device_ID> 番号があった場合には，指定のグループに割り付けられたことになる。

制御される装置は，ひとたびグループに割り付けられると以後は，本来の自分のデバイス ID で指定されたものに加えて，そのグループのデバイス ID で指示されたすべてのコマンドに対しても，これを受け付けるようになる。

グループへの割り付けは，MMC RESET か，適切な GROUP[DIS-ASSIGN]コマンドを受け取るまでは保持される。

グループへの割り付けは，各 GROUP[ASSIGN]メッセージごとに累積的に行なえる。例えば，すでに存在しているグループに新たな装置を付け加えるには，コントローラーは単に，新規の装置名だけをリストに持つ GROUP[ASSIGN]を送信するだけでよい。

装置に用意されている数以上のグループに割付けられた際には，COMMAND ERROR 情報フィールド中の”グループ・バッファ・オーバーフロー“エラーを発生する。 *3

52	**GROUP[ASSIGN]**
<count=variable>	**バイト数（コマンドとカウントは含まない）**
00	**“ASSIGN”のサブコマンド**
<group>	**グループ番号（7Fを除いては，使われていない任意のデバイス ID を，グループ番号として指定してよい）**
<device_ID>	**以後，このグループ番号に応答し始める装置のリスト**
<device_ID>	
:	
:	

フォーマット2：GROUP[DIS-ASSIGN]

制御される装置は，受信したデバイス ID リスト中に自分の <device_ID> 番号があった場合には，指定のグループから自分を取り除く。

52	**GROUP[DIS-ASSIGN]**
<count=variable>	**バイト数**
01	**“DIS-ASSIGN”のサブコマンド**
<group>	**グループ番号**
	7F = すべてのグループからの指定解除
<device_ID>	**上記のグループ番号から取り除かれる装置のリスト**
<device_ID>	**リスト中のどこかに7Fのある場合は，すべての装置を指定解除する *4**
:	
:	

＜備考＞

1. MMC RESET コマンドは常に，すべてのグループ割付けを解除する。
2. GROUP［ASSIGN］及び［DIS-ASSIGN］メッセージは通常，全員呼び出しデバイス ID（<device_ID>=7F）を通じて送信される。

* 3. 制御される装置は，同時には少なくとも16個のグループにまでは割り付けられるようになっていることが望ましい。

* 4. ひとつのグループをすっかり削除する際には，“装置のリスト” には標準的にはただひとつの番号（7F）だけが置かれる。例えば，すべての装置をすべてのグループから除くには，コントローラーは以下のように送る。

F0 7F 7F <mcc> <GROUP> <count=03> 01 7F 7F F7

5. ［ASSIGN］ 及び［DIS-ASSIGN］ 以外のサブコマンドを指定した GROUP コマンドに対しては，“規定のサブコマンドでない”の COMMAND ERROR を発生する。

53　COMMAND SEGMENT　[-234]

MMC システム・エクスクルーシブの定める最大データ・フィールド長(48 バイト)よりも長いコマンド(あるいはコマンド列)を，いくつかのセグメントに分割して，複数個のシステム･エクスクルーシブの形で部分部分ごとに送信することができるようにする。

制御される装置は，このようにして受信したコマンドは，あたかも一個のシステム・エクスクルーシブで全部一度に届けられたかのようにして実行するものとする。

COMMAND SEGMENT コマンドは，常にそのシステム・エクスクルーシブの中の先頭のコマンドでなければならず，またこのシステム・エクスクルーシブ中には，このコマンド・セグメントの本体部分に含まれているもの以外には，他のコマンドはいっさいあってはならない。

セグメントの切れ目は，コマンドの境界に合っている必要はない。制御される装置は，COMMAND SEGMENT システム･エクスクルーシブの終端部で出て来るであろうこの未完結コマンドを検出し，次のセグメントが届きしだい，正しくコマンド処理が続行できなくてはならない。

次に示す条件のいずれかに相当した場合には，制御される装置は COMMAND ERROR 情報フィールド中の“主要(Major)エラー”のひとつとして定義されている“セグメンテーション・エラー”を発生させる。

(a) COMMAND SEGMENT が，システム・エクスクルーシブ中の最初のコマンドでない。
or (b) バイト・カウントが，システム・エクスクルーシブ中に残っているバイト数と一致しない。
or (c) 先頭セグメントを受け取っていないのに，後続セグメントが受信された。
or (d) セグメントが順番を乱して受信された。

セグメンテーション・エラーが発生した場合は，セグメント復元は打ち切りにするものとする。

制御される装置が後続セグメントを期待して待っている時に，WAIT や RESUME メッセージ以外の，セグメント化されていない(普通の)システム・エクスクルーシブが受信された場合には，セグメント復元は打ち切りにして，こちらを通常通りに処理するものとする。

「2 全体構造－セグメンテーション」の項をも参照のこと。

53	**COMMAND SEGMENT**
<count=variable>	**バイト数(コマンド・ストリング・セグメント長 + 1)**
si	**セグメント ID：　0 f ssssss**
	f:　1 = 先頭セグメント
	0 = 後続セグメント
	ssssss = セグメント番号(降順，最後が 000000)
<..commands..>	**コマンド・ストリング・セグメント**

54　DEFERRED VARIABLE PLAY (MCS コマンド)　[-234]

VARIABLE PLAY コマンドと同等だが，装置がこの時もしも LOCATE(MCP)の実行中であった場合には，ロケート動作が完了するまでは Variable PLAY に強引に入ることはないという点が異なる。

他の MCS や MCP コマンドを受信すると，この DEFERRED(保留付き)の Variable PLAY 状態は取り消される。

ロケート動作の実行中にこのコマンドを受け取った場合は，MOTION CONTROL TALLY 情報フィールド中の“MCP

成功レベル”部は，ロケートしている間はずっと，「バリアブル・プレイの指示は保留されており，実際にはロケート中」なる旨を示していなければならない。

ロケートが完了すると：

(i)　自動的に VARIABLE PLAY コマンド（MCS）が出され，指定された方向とスピードの可変速プレイ・モードに滑らかに入る（もしも要求されたスピードの値が，制御される装置の可能な範囲を越える場合には，「できる範囲での最も近いスピード」で動作するものとして）。

(ii)　MOTION CONTROL TALLY 中の“MCSコマンド”バイトは“VARIABLE PLAY”に変わる。

(iii)同じく“MCP コマンド”バイトは“MCP は現在アクティブでない”に変えられ，それまでの LOCATE や Deferred Variable Play 標識は共にクリアーされる。

装置が LOCATE コマンドを実行中でなかったり，サポートしていなかった場合には，このコマンドではただちに可変速プレイ（Variable Playback ）モードに入るべきである。

54	**DEFERRED VARIABLE PLAY**
<count=03>	**バイト数**
sh sm sl	**標準スピード仕様**

＜備考＞

レコーディング［リハーサル］中のトラックは，DEFERRED VARIABLE PLAY コマンドを受け取っても，自動的にレコード［リハーサル］を止めることはない。その動作が望まれる場合は，<RECORD EXIT> <DEFERRED VARIABLE PLAY> と送る。

55　RECORD STROBE VARIABLE　[----]

制御される装置を RECORD MODE 情報フィールドの設定内容に応じて，レコード・モードなりリハーサル・モードに入れたり，あるいはこれを止めたりさせる。

RECORD STROBE VARIABLE コマンドは2種類の条件下でのみ効果を有する。

条件1：　制御される装置がすでにプレイ中

もし制御される装置がすでにプレイ中（すなわち，MOTION CONTROL TALLY 情報フィールド中の“直前に実行された MCS”が PLAY か Variable PLAY ）であれば，この RECORD STROBE VARIABLE コマンドは，この時点でレコードレディ状態にセットされているトラックすべてをレコード［またはリハーサル］状態にし，現在はレコーディング中［リハーサル中］であるが，もはやレコード・レディにはセットされていないトラックに対しては，レコード［リハーサル］動作を止めさせる。 *2 *9 *10

条件2：　制御される装置が停止中

RECORD STROBE VARIABLE コマンドを受け取った時に，もし制御される装置が明確な STOP あるいは PAUSE コマンドの結果として，完全に停止中であれば（すなわち，[i]MOTION CONTROL TALLY 情報フィールド中の“直前に実行された MCS”が STOP か PAUSE であって，[ii] “MCS 成功レベル”が“完全に停止している”を示しており，かつ[iii] “直前に実行された MCP”が“MCP は現在アクティブでない”になっている場合には）

(i) 自動的に VARIABLE PLAY（MCS）コマンドが出され，指定された方向とスピードの可変速プレイ・モードに滑らかに入り，もしも要求されたスピードの値が，制御される装置の可能な範囲を越える場合には，「できる範囲での最も近いスピード」で動作するものとして）*4 *5

(ii) 装置の可変速立ち上がりフェーズ中の適当な時点で，この時にレコード・レディ状態にセットされているトラック

すべてに対し，レコード[リハーサル]動作を開始させる。 *2　*6

55	**RECORD STROBE VARIABLE**
<count=03>	**バイト数**
sh sm sl	**標準スピード仕様**

＜備考＞

1. RECORD STROBE VARIABLE コマンドのレコーディング[リハーサル動作]特性には，RECORD STROBE コマンドとの違いはない。ふたつのコマンドの間の唯一の違いは，制御される装置が完全に停止していると自動的に出されるプレイ・モードのコマンドの種類にある。RECORD STROBE VARIABLE はしたがって，止まっている状態からの開始の際にバリアブル・スピードでのレコーディングを確保しなければならない時に用いられる。

*2. TRACK RECORD READY 情報フィールドを用いて，各トラックはレコード・レディ状態に入れたり外したり切り替えられる。

3. 制御される装置を MMC に準拠して新たに設計する際には，少なくともひとつのトラックがレコード・レディ状態となっているのでない限り，RECORD STROBE VARIABLE コマンドが来ても，レコード[リハーサル]状態にはしないようにすることが推奨される。

   しかしながら現行の非 MMC 装置の間では，(例えば RECORD MODE 情報フィールドに適当な値をセットする等により)レコード[リハーサル]がイネーブルになっていると，レコード[リハーサル]のコマンドを受け取った際に，たとえその時いずれのトラックもレコード・レディとなっていなくても，レコード[リハーサル]動作を開始させるといったことは，ごく普通に見られる。このような動作は，MMC の下でこれからも許されるものとする。ただし，この結果得られるステータス上は，RECORD STATUS 情報フィールドを見ると“アクティブなトラックはない”のビットが立っていることで，正しく示されることを条件とする。

*4. 条件2の下で VARIABLE PLAY(MCS)コマンドを自動的に出すのは，STOP または PAUSE 状態がはっきりしている場合に限られる。これ以外の場合には，RECORD STROBE VARIABLE コマンドは，プレイ・モードやプレイ・スピードに関していかなる関りも持たない。

＜日本語版注＞

したがって条件1の下で，すでに PLAY あるいは VARIABLE PLAY のプレイ・モードにある場合にも，スピードを改めてセットする機能は持っておらず，このコマンドのデータ部で用意された標準スピード仕様の3バイトは，まったく意味を持たずに無視される。

*5. 同じく条件2の下で VARIABLE PLAY(MCS)コマンドを自動的に出すことそのものは，レコード・レディのトラックがあるかどうかや，RECORD MODE がどうなっているかには関りなく行われる。

*6. RECORD PAUSE 状態があることによって，RECORD STROBE VARIABLE の動作内容に影響することはない(PAUSE 及び RECORD PAUSE コマンドを参照のこと)。

7. RECORD MODE 情報フィールドをサポートしており，それが“VTR: Insert”や，“VTR:Assemble”にセットされた装置において，上に概説した条件12の下でレコード[リハーサル]状態に移る際には，クリーンで正しい遷移過程を取ることが期待されている。(条件2の)STOP や PAUSE モードから立ち上がる際には，レコード[リハーサル]を始める前に，まずは装置の立ち上がりフェーズの完了するまで待つことが，通常は必要となるであろう。

8. 制御される装置が，すでにプレイ状態になっているのでもなければ完全なストップ状態でもない時に，RECORD STROBE VARIABLE コマンドを受け取った場合には，これを無視することとする。

*9. 条件1「制御される装置がすでにプレイ中」の下で RECORD STROBE VARIABLE が受け付けられるのに，PLAY や VARIABLE PLAY コマンドは必ずしも“成功した”状態にまでなっている必要はない。しかしながら，RECORD STROBE VARIABLE を受信した時に，もしもそのプレイ動作が未だ確立していなかった場合には，装置はその立ち上がりフェーズ中の適当な時点まで，レコード[リハーサル]に入るのを遅らせる必要があろう。

*10. さらにまた条件1の下では，MCP の実行中だからといって，レコード動作の妨げられることはないことにも注意されたい。

7C　WAIT　[-234]

制御される装置に対して，コントローラーの受信バッファがいっぱいになりつつあること，（あるいはコントローラーが他の理由で対応できない状態であること），したがってマシン・コントロール・レスポンスの送信は，コントローラーから RESUME を受け取るまで中断すべきであることを伝える。この時点で，送信中のレスポンスについてはその種類を問わず，通常通りシステム・エクスクルーシブ終了メッセージ EOX（F7）までは続けることが許される。引き続くレスポンスの送信は，コントローラーから RESUME を受信した後に再開してよい。

しかしながら WAIT と RESUME のレスポンスは，WAIT コマンドの後であっても拒絶されることはない。同様に WAIT コマンドの送信自体が，WAIT レスポンスを受信したからといって，拒否されることもない。

制御される装置は，以下の要請を満足しなければならない。

(i) この WAIT メッセージの EOX(F7)の到着から 10 m秒以内には，この WAIT 指示を認識できること。及び，

(ii) 続いて，この次に可能な MMC システム・エクスクルーシブ境界ですべての送信を停止すること（したがって，実際に停止状態となるまでには，最大 53 バイトまでの MMC での最長システム・エクスクルーシブは，送信せざるを得ないことが許される）。

WAIT コマンドは，常にひとつのシステム・エクスクルーシブ中の唯一のコマンドであり，“全員呼び出し”のアドレスで送信される。すなわち：F0 7F 7F <mcc> <WAIT> F7

```
7C          WAIT
```

＜備考＞

1. WAIT コマンドの正しい動作のためには，コントローラー内の MIDI 受信バッファには，ある最低限のサイズが必要となる。付録E「受信バッファ・サイズの決定」を参照。
2. コントローラーは，もしもその受信バッファにそれでもまだデータが来るようであったら，再度 WAIT コマンドを送信しても良い。

＜日本語版注＞

もしも複数のコントローラーがつながれている場合には，あるコントローラーにとっては，他のコントローラーの発した WAIT コマンドによってレスポンスが期待できなくなる場合や，他のコントローラーの発した RESUME コマンドによって，レスポンスが来はじめてしまうかも知れない場合に備える必要がある。コマンドが正しく受け取られなかった場合をも含めて，必要に応じては，任意回数の WAIT コマンドあるいは RESUME コマンドを繰り返すことが許されるものとする。

制御される装置にあっては，こうした重複する WAIT/RESUME コマンドに備える必要があるが，コマンドの発信源を特定できないこともあり，回数は問題とせずに，常に最後に受信した WAIT/RESUME コマンドに対応するものとする（2度の WAIT コマンドを受信しても，RESUME コマンドを2回待つことはしない）。

7F　RESUME　[-234]

コントローラーが，制御される装置からのマシン・コントロール・レスポンスを受信できる態勢にあることを知らせる。（電源オン時の）デフォルトの状態は，「受信準備完了」である。

RESUME コマンドは，主として制御される装置が WAIT コマンドの後で，再び送信を始められるようにするのに用いられる。

RESUME コマンドの送信は，WAIT レスポンスの受信後であっても拒否されることはない。

RESUME コマンドは，常にひとつのシステム・エクスクルーシブ中の唯一のコマンドであり，“全員呼び出し”のアドレスで送信される。すなわち：F0 7F 7F <mcc> <RESUME> F7

```
7F          RESUME
```

＜日本語版注＞　WAIT コマンド解説部の備考2，及び＜日本語版注＞参照。

6　レスポンスと情報フィールド詳説

制御される装置から制御する装置(コントローラー)へのメッセージ。

00　Reserved for extensions　[----]

拡張用にリザーブされている。

01　SELECTED TIME CODE [read/write] [1234]

制御される装置の現在位置を参照するのに, 通常使用するタイムコードの値を保持している(セルフ・コード, あるいはスレーブ・タイムコードと呼ばれる場合もある)。

情報フィールド SELECTED TIME CODE SOURCE は, このタイムコードの出所を決める。それは Longitudinal Time Code(LTC)か, Vertical Interval Time Code(VITC)か, あるいはたいていのテープ装置に見られる“テープ・カウンター”のうちから選ばれる。

01	**SELECTED TIME CODE**
hr mn sc fr st	**ステータス付き標準時刻仕様 ({st} タイプ)**

＜備考＞

1. タイムコードの選択肢に関しては, SELECTED TIME CODE SOURCE 情報フィールドの解説部に, より詳しい説明がある。
2. SELECTED TIME CODE のステータス・バイトは, 高速巻取モードの際にタイムコードの現在値が, 例えばタコメーターやコントロール・トラックからのパルスによって更新されたものであるかどうかを示す。
3. もしもタイムコードがまったく使えない場合には, SELECTED TIME CODE は, 通常タコメーターやコントロール・トラックからのパルスを使って更新を行う, たいていのテープ・マシンに付いているテープ・カウンターと同等になる。タイムコード・システムでの互換性のために, このテープ・カウンターは時分秒フレームの単位でカウントしなくてはならない。このカウンター用のタイムコードのモードは, 通常もしサポートされていれば TIME STANDARD 情報フィールドか, あるいは装置の側でセットできるようになっているデフォルトで決まる。ただし, コントローラーがこの SELECTED TIME CODE に値を書き込んだ場合には, タイムコード・モードは WRITE データ中の tt(フレーム・タイプ)フィールドで定まることになる。SELECTED TIME CODE には負符号の付いた値は認められていない。
4. 上に述べたタコ・オンリー・モードの動作を具合良く機能させるために, この SELECTED TIME CODE は“書き込み可能”と規定されている。この場合には, SELECTED TIME CODE の“カウンター”に新たな値をセットすることは, 認められている操作手続きと言える。しかしながら, テープからのタイムコードが使える場合には, このフィールドに新しい値を書き込むと, 予期できない結果を引き起こす可能性がある。
5. SELECTED TIME CODE から本物のタイムコードが得られるようになっている(すなわち, タイムコード・ステータス・ビット n = 0)のでない限り, 同期操作を行おうとするようなことはしないものと想定されている。
6. タイムコード中に埋め込まれているすべてのステータス・ビットの, SELECTED TIME CODE への正確な適用の仕方に関しては, 付録Bの「タイムコード・ステータスの運用細則」を参照。これらのビットの定義に関しては「3　標準仕様」を参照のこと。

02　SELECTED MASTER CODE [read only]　[---4]

すべての同期動作を基準として行なわれるタイムコードの時刻値を保持している(CHASE コマンド参照)。このタイムコードが, 制御される装置にどのようにして渡されるかは規定しない。

02 **SELECTED MASTER CODE**

hr mn sc fr st **ステータス付き標準時刻仕様（{st} タイプ）**

＜備考＞

1. MIDIマシン･コントロール仕様の将来のバージョンにおいては，このマスター･コードをいくつかの特定タイムコード･ソースの中から選択する手段を設けることが考えられる。
2. タイムコード中に埋め込まれているすべてのステータス・ビットの，SELECTED MASTER CODE への正確な適用の仕方に関しては，付録Bの「タイムコード・ステータスの運用細則」を参照。これらのビットの定義に関しては「3　標準仕様」を参照のこと。

03 REQUESTED OFFSET [read/write] [---4]

CHASE コマンドでの利用のために,SELECTED TIME CODE と SELECTED MASTER CODE の間の時間差(タイム・オフセット)として希望する値を保持する。その定義は：

REQUESTED OFFSET = SELECTED TIME CODE - SELECTED MASTER CODE

（例：制御される装置をマスター装置より1分だけ先行させたい場合，REQUESTED OFFSET は 00:01:00:00.00 となる。）

このオフセットは，マスターとスレーブの位置の間での望ましいフレーム差を表し，常にノンドロップ・フレームの数として表現される。

REQUESTED OFFSET は，プラス側あるいはマイナス側のいずれの範囲でも表すことができる。MMC装置は，例えば +23:00:00:00.00 のオフセットは，（いくつかあるうちのひとつの）-01:00:00:00.00 と同等であると見なすものとする。

3 **REQUESTED OFFSET**

hr mn sc fr ff **サブフレーム付き標準時刻仕様（{ff} タイプ）**

＜備考＞

タイムコード中に埋め込まれているすべてのステータス・ビットの，REQUESTED OFFSET への正確な適用の仕方に関しては，付録Bの「タイムコード・ステータスの運用細則」を参照。これらのビットの定義に関しては「3　標準仕様」を参照のこと。

04 ACTUAL OFFSET [read only] [---4]

同期に用いるためにこのフィールドは，SELECTED MASTER CODE と SELECTED TIME CODE の現在値同士間での，実際の時間差を保持している。ここに：

ACTUAL OFFSET = SELECTED TIME CODE - SELECTED MASTER CODE

（例：制御される装置がマスター装置より1分だけ進んでいる場合，ACTUAL OFFSET は 00:01:00:00.00 と言える。）

このオフセットは，スレーブとマスターの位置の差をフレームで表しており，常にノンドロップ･フレームの数として表現される。

-01:00:00:00.00 と +23:00:00:00.00 のような値の組み合わせに見られる“タイムコードの同等性”を基にすることで，ACTUAL OFFSET は +/-12:00:00:00.00 の範囲で表さなければならない。

04 **ACTUAL OFFSET**

hr mn sc fr ff **サブフレーム付き標準時刻仕様（{ff} タイプ）**

＜備考＞

タイムコード中に埋め込まれているすべてのステータス・ビットの，ACTUAL OFFSET への正確な適用の仕方に関しては，付録Bの「タイムコード・ステータスの運用細則」を参照。これらのビットの定義に関しては「3　標準仕様」を参照のこと。

05 LOCK DEVIATION [read only] [---4]

同期に用いるためにこのフィールドは，制御される装置の SELECTED TIME CODE と SELECTED MASTER

CODE の位置同士での時間差を, REQUESTED OFFSET で補正した後の値を保持している。
すなわち:

LOCK DEVIATION = ACTUAL OFFSET - REQUESTED OFFSET

あるいは:

LOCK DEVIATION = SELECTED TIME CODE - SELECTED MASTER CODE - REQUESTED OFFSET

LOCK DEVIATION は, 常にノンドロップ・フレームの数であり, +/- 12:00:00:00.00 の範囲で表現しなければならないものとする。

05	**LOCK DEVIATION**
hr mn sc fr ff	**サブフレーム付き標準時刻仕様（{ff} タイプ）**

＜備考＞

タイムコード中に埋め込まれているすべてのステータス・ビットの, LOCK DEVIATION への正確な適用の仕方に関しては, 付録Bの「タイムコード・ステータスの運用細則」を参照。これらのビットの定義に関しては「3　標準仕様」を参照のこと。

06　GENERATOR TIME CODE [read/write]　[----]

タイムコード・ジェネレーターから生成されている現在のタイムコードの値を保持している。

06	**GENERATOR TIME CODE**
hr mn sc fr st	**ステータス付き標準時刻仕様（{st} タイプ）**

＜備考＞

タイムコード中に埋め込まれているすべてのステータス・ビットの, GENERATOR TIME CODE への正確な適用の仕方に関しては, 付録Bの「タイムコード・ステータスフ運用細則」を参照。これらのビットの定義に関しては「3　標準仕様」を参照のこと。

07　MIDI TIME CODE INPUT [read only]　[----]

入力されているMIDIタイムコードの最新の値を保持している。

07	**MIDI TIME CODE INPUT**
hr mn sc fr st	**ステータス付き標準時刻仕様（{st} タイプ）**

＜備考＞

タイムコード中に埋め込まれているすべてのステータス・ビットの, MIDI TIME CODE INPUT への正確な適用の仕方に関しては, 付録Bの「タイムコード・ステータスの運用細則」を参照。これらのビットの定義に関しては「3　標準仕様」を参照のこと。

08　GP0／LOCATE POINT　[read/write]　[1234]

タイムコード，及び計算用の汎用レジスター0。

08	**GP0/LOCATE POINT**
hr mn sc fr ff	**サブフレーム付き標準時刻仕様（{ff} タイプ）**

＜備考＞

1. LOCATE コマンドのフォーマット1(LOCATE[I/F])では,そのターゲットとなるロケーションの時刻を,汎用レジスターにセットしておかなくてはならないと決めている。同様に EVENT コマンドでは，そのトリガー時刻を汎用レジスターから取り出す。したがって，この LOCATE または EVENT コマンドを用いるのであれば，少なくともひとつは，汎用レジスターがサポートされていなければならない。
2. 汎用レジスターは，動いているタイムコードを「オン・ザ・フライ」で捕えるのにも使われる(例えば SELECTED TIME CODE を GPn に MOVE する)。これによりコントローラーは，実際のタイムコードの値を絶えず読み返す必要がなくなり，オープン・ループ・モードでの操作が容易になる。
3. 汎用レジスターにおいては，符号付きタイムコードは認められるものとする。
4. タイムコード中に埋め込まれているすべてのステータス・ビットの，汎用レジスターへの正確な適用の仕方に関しては，付録Bの「タイムコード・ステータスの運用細則」を参照。これらのビットの定義に関しては「3　標準仕様」を参照のこと。

09	**GP1**	**[read/write] [-234]**
0A	**GP2**	**[read/write] [-234]**
0B	**GP3**	**[read/write] [-234]**
0C	**GP4**	**[read/write] [- - - -]**
0D	**GP5**	**[read/write] [- - - -]**
0E	**GP6**	**[read/write] [- - - -]**
0F	**GP7**	**[read/write] [- - - -]**

タイムコード，及び計算用の汎用レジスター1から7(汎用レジスター0についての注参照)

<name>	**GP1 ～ GP7**
hr mn sc fr ff	**サブフレーム付き標準時刻仕様（{ff} タイプ）**

21	**Short SELECTED TIME CODE**	**[read only]**	**[-234]**
22	**Short SELECTED MASTER CODE**	**[read only]**	**[- - -4]**
23	**Short REQUESTED OFFSET**	**[read only]**	**[- -- 4]**
24	**Short ACTUAL OFFSET**	**[read only]**	**[- -- 4]**
25	**Short LOCK DEVIATION**	**[read only]**	**[- - -4]**
26	**Short GENERATOR TIME CODE**	**[read only]**	**[- - - -]**
27	**Short MIDI TIME CODE INPUT**	**[read only]**	**[- - - -]**
28	**Short GP0／LOCATE POINT**	**[read only]**	**[- - - -]**
29	**Short GP1**	**[read only]**	**[-234]**
2A	**Short GP2**	**[read only]**	**[-234]**
2B	**Short GP3**	**[read only]**	**[-234]**
2C	**Short GP4**	**[read only]**	**[- - - -]**
2D	**Short GP5**	**[read only]**	**[- - - -]**
2E	**Short GP6**	**[read only]**	**[- - - -]**
2F	**Short GP7**	**[read only]**	**[- - - -]**

「標準短形式タイムコード」の定義については，「3　標準仕様」を参照。

いずれについてもデータの内容については，対応する5バイト・タイムコード情報フィールドを参照。

<name>	**短形式タイムコード情報フィールド名**
fr {st | ff}	**標準短形式タイムコード仕様**

40　SIGNATURE　[read only]　[-234]

制御される装置がサポートする(a)すべてのコマンド機能,及び(b)すべてのレスポンス／情報フィールド,の2個のビット･マップ配列。

どんな場合でも，該当する機能が部分的にであれサポートされている場合には，そのビットは1にセットされる。

制御される装置の製造者は，そのSIGNATURE情報フィールドを備考 *5に記されたフォーマットに従って，文書の形で公開しなければならない。

40	**SIGNATURE**
<count=variable>	**後ろに続くすべてのデータのバイト数　*1**
vi	**装置の採用したMMCのバージョン番号中，整数部分の2進表現（現バージョンの場合 vi=01）**
vf	**装置の採用したMMCのバージョン番号中，小数部分 00 ～ 99 の2進表現（00 ～ 63H）（現バージョンの場合 vf=00）**
va	**将来の拡張用に予約されている（00 でなければならない）**
vb	**将来の拡張用に予約されている（00 でなければならない）**
<count_1>	**コマンド用ビット・マップ配列のバイト数**
c00	**コマンド・ビット・マップ 00: Commands 00 ～ 06: 0 gfedcba** a = Command 00 b = Command 01 c = Command 02 d = Command 03 e = Command 04 f = Command 05 g = Command 06
c01	**コマンド・ビット・マップ 01: Commands 07 ～ 0D**
c02	**コマンド・ビット・マップ 02: Commands 0E ～ 14**
c03	**コマンド・ビット・マップ 03: Commands 15 ～ 1B**
c04	**コマンド・ビット・マップ 04: Commands 1C ～ 1F: 0000 dcba** a = Command 1C b = Command 1D c = Command 1E d = Command 1F
c05	**コマンド・ビット・マップ 05: Commands 20 ～ 26**
c06	**コマンド・ビット・マップ 06: Commands 27 ～ 2D**
c07	**コマンド・ビット・マップ 07: Commands 2E ～ 34**
c08	**コマンド・ビット・マップ 08: Commands 35 ～ 3B**
c09	**コマンド・ビット・マップ 09: Commands 3C ～ 3F**
c10	**コマンド・ビット・マップ 10: Commands 40 ～ 46**
c11	**コマンド・ビット・マップ 11: Commands 47 ～ 4D**
c12	**コマンド・ビット・マップ 12: Commands 4E ～ 54**
c13	**コマンド・ビット・マップ 13: Commands 55 ～ 5B**
c14	**コマンド・ビット・マップ 14: Commands 5C ～ 5F**
c15	**コマンド・ビット・マップ 15: Commands 60 ～ 66**
c16	**コマンド・ビット・マップ 16: Commands 67 ～ 6D**
c17	**コマンド・ビット・マップ 17: Commands 6E ～ 74**

c18	**コマンド・ビット・マップ 18: Commands 75 ～ 7B**
c19	**コマンド・ビット・マップ 19: Commands 7C ～ 7F**
c20 ～ c39	**コマンド・ビット・マップ 20 ～ 39:** **拡張コマンド　00 01 ～ 00 7F**
c40 ～ c59	**コマンド・ビット・マップ 40 ～ 59:** **拡張コマンド　00 00 01 ～ 00 00 7F**
<count_2>	**レスポンス／情報フィールド用配列のバイト数**
r00 ～ r19	**レスポンス／情報フィールド・ビット・マップ 00 ～ 19:** **レスポンスと情報フィールド　00 ～ 7F**
r20 ～ r39	**レスポンス／情報フィールド・ビット・マップ 20 ～ 39:** **拡張レスポンスと情報フィールド　00 01 ～ 00 7F**
r40 ～ r59	**レスポンス／情報フィールド・ビット・マップ 40 ～ 59:** **拡張レスポンスと情報フィールド　00 00 01 ～ 00 00 7F**

＜備考＞

*1. <count> の最大値は両拡張セットがすべてサポートされている場合で 7E となる。

2. それぞれの配列について，必要なだけのバイト数を送ること。

3. 送信された中に含まれていないコマンドとレスポンス／情報フィールドは，サポートされていないものとみなされる。

4. この SIGNATURE に加えて，すべての装置は MIDI インクワイアリー・メッセージをサポートすべきである。

*5. 文書で公開の際には，SIGNATURE は下記のフォーマットで表わすものとする。

vi　vf　va　vb
<count_1>
c00　c01　c02　c03　c04　c05　c06　c07　c08　c09
c10　c11　c12　c13　c14　c15　c16　c17　c18　c19
c20　c21　c22　etc.
<count_2>
r00　r01　r02　r03　r04　r05　r06　r07　r08　r09
r10　r11　r12　r13　r14　r15　r16　r17　r18　r19
r20　r21　r22　etc.

（付録C「SIGNATURE 表」も参照のこと。）

6. すべての制御される装置は，Command 00 については「サポートあり」として表示し，拡張コマンドをたとえひとつもサポートしていなくとも，構文上は正しく（読み飛ばし処理で）扱えるようになっているであろう。これとは違って Response 00 については，現バージョンにおいては常に「サポートなし」として表示される。将来のバージョンにおいては，Response 00 のビットはすべての拡張レスポンスビットの論理OR，すなわちひとつでも拡張レスポンスがサポートされると“1”とされる。

41　UPDATE　RATE　[read/write]　[-234]

UPDATE の繰り返し送信サイクルの最小時間間隔を設定する。

UPDATE コマンドの解説を参照のこと。

41	**UPDATE RATE**
<count=01>	**バイト数**
<interval>	**7ビット長のフレーム数で表された, UPDATE の繰返し送信の最小時間間隔** **無指定時の間隔は1フレーム（interval=01）**

＜日本語版注＞

最大値はしたがって, 127 フレーム（＝およそ 4.2 秒 /30fps;　5.1 秒 /25fps）。

42　RESPONSE　ERROR　[アクセス不可]　[-234]

制御される装置は, いずれの情報フィールドを指定した READ や UPDATE のリクエストに対しても, 何らかのレスポンスを返さなければならない。しかしながら, もしもこの要求された情報フィールドが：

(a)制御される装置のサポートしていない場合

or (b)MMCで定義されていない場合

or (c)MMCで[アクセス不可]と定義されている場合

期待された情報フィールドのレスポンスの代わりに, RESPONSE ERROR メッセージを返すものとする。

n個の異なる情報フィールドのリストを持った READ コマンドは, n個の異なるリクエストであるとみなして, そのいずれにも返答をするものとする。UPDATE コマンドでリクエストされた場合についても同様とする。

望まれれば, こうしたサポートされていない情報フィールド名いくつかを, ひとつの RESPONSE ERROR メッセージにまとめてもかまわない。

UPDATE コマンドでは通常, リクエストされた情報フィールドそれぞれが繰り返し送られるのであるが, サポートされていない情報フィールドのリクエストに対しては, RESPONSE ERROR は1回だけ送るものとする。

42	**RESPONSE ERROR**
<count=variable>	**バイト数（コマンドとカウントは含まない）**
<name>	**情報フィールド名（のリスト）**
:	

＜備考＞

1. 制御装置にとっては, このメッセージのあることによって, 通常の状況であれば, どんなデータの要求に対しても何らかのレスポンスへ戻ることが保証されていると言える。
2. 次の例は,3つの情報フィールドの READ に対して,そのうち2つは装置がサポートしておらず（“BAD1”と“BAD2”）, 3番目はサポートされている（“GOOD”）場合, 考えられるレスポンス列を示している：

コマンド　：F0 7F <device_ID> <mcc> <READ> <count=03> <BAD1> <BAD2> <GOOD> F7
レスポンス：F0 7F <device_ID> <mcr> <RESPONSE ERROR> <count=01> <BAD1> F7
F0 7F <device_ID> <mcr> <RESPONSE ERROR> <count=01> <BAD2> F7
F0 7F <device_ID> <mcr> <GOOD> <..data..> F7

43　COMMAND ERROR [read only]　[-234]

制御される装置はこの情報フィールドの内容を次の場合に送信する。

(a)コントローラーからの READ コマンドで指定して要求された場合

or (b)イネーブルにされたエラーが発生した場合に自動的に

エラーは,COMMAND ERROR LEVEL 情報フィールドをセットすることでイネーブルにされる。新たに検出されたエラーのエラーコードが, もしも COMMAND ERROR LEVEL の値よりも小さいか, あるいはそれと等しい場合には, そのエラーはイネーブルとなり“エラー発生”として報告され, それよりも大きい場合は, ディスエーブルとしてそのまま無視される。

43 **COMMAND ERROR**

<count=04+ext+count_1> **バイト数**

<flags> **エラー・フラグ・ビット:0 gfedcba**

a= エラー停止フラグ

0 = エラー停止とはなっていない(COMMAND ERROR RESET, MMC RESET や電源オンの直後の状態)

1 = エラー停止状態:イネーブルにされたエラーの発生によってセットされる。エラーを生じたコマンド以後に受け取ったすべてのコマンドは無視されている。COMMAND ERROR RESET を受け取るまでは, 以後いっさいのコマンドは処理されない。

b=PROCEDURE[ASSEMBLE]エラー・フラグ

0 = エラー無し

1 = プロセジャーのアッセンブリー中に埋め込まれている, コマンド群の予備チェック段階でエラーが見つかった。

c=EVENT[DEFINE]エラー・フラグ

0 = エラー無し

1 = イベント定義中に埋め込まれている, コマンドの予備チェック段階でエラーが見つかった。

d=0

e= リクエスト無し COMMAND ERROR レスポンス・フラグ

0 = READ のリクエストに対するレスポンスの場合。

1 = イネーブルにされたエラーの発生による, リクエスト無しの自動送信エラー・メッセージであり, この時結果的ににエラー停止フラグがセットされた場合。

f=COMMAND ERROR 送信済みフラグ

0 = 最新のエラーがここに記録された以降に, 未だその旨 COMMAND ERROR フィールドは伝えられていない。[初めての報告あり,] (個々のエラーの発生のつど, 0にリセットされる)。

1 = 最新のエラーを示す COMMAND ERROR フィールドは, すでに(1度以上) 送られている。(リクエストの有無に関わらず, COMMAND ERROR が送信され終えしだい, そのつど1にセットされる)。

g=0

<level> **COMMAND ERROR LEVEL 情報フィールドの現在の設定値**

<error> **エラー・コード**

00 = 拡張のため予約されている

01 ～ 7E = 後述のコマンド・エラー・コード・リストを参照

7F = パワー, オンあるいは MMC RESET 以後にエラーは起こっていない。

<count_1>	**以後に続く <offset> と <command string> のバイト数 「受信バッファ・オーバーフロー」や「コマンド・システム・エクスクルーシブ長規定違反」エラー, あるいはエラー・コードが 7F の場合は, <count_1> を 00 にセットし, <offset> と <command string> は省略する。**
<offset>	**エラーを生じたバイトの, <command string> の先頭からのオフセット値(先頭バイトを offset=00 とする)。 エラーの性質上, バイト位置が何とも言えないか決められない場合は 7F にセットすること。**
<command string>	**最も最近にエラーを生じたコマンド (レスポンスのシステム・エクスクルーシブとするにはコマンドが長過ぎるために後ろを打ち切ったり, あるいはエラーの性質上長さがはっきりしないなどの場合を除いて, これはそっくりあるがままの形で含まれていなければならない。) <command name> または <command name> <count> <command data>**

コマンド・エラー・コード・リスト

エラー・コードは, MMC のコマンドやレスポンスとまったく同じ方式で分類される。コード 00 は拡張用に予約されている。ある特定のエラー・クラスで示された特質は, これに対応する拡張クラスにも受け継がれる。例えば, エラー・コード 00 20 から 00 3F は, コード 20 から 3F までと同じく“即時オペレーション・エラー”に分類される。

制御される装置内部での“イネーブルになっている”エラーに対する対応は, そのエラーの種類に関わらず, どんなものに対しても同じである。

- COMMAND ERROR 情報フィールドの内容を更新
- エラー停止フラグをセット
- 自動的に(リクエスト無しに)COMMAND ERROR フィールドを送信
- エラーを生じたもの以降に受信したすべてのコマンドは読み捨てる
- (COMMAND ERROR RESET か, あるいは MMC RESET のいずれかによって)エラー停止フラグがリセットされるまでは, 以後のコマンドはすべて読み捨て, 処理はいっさい行なわないこと。

“ディスエーブルになっている”エラーに対する対応は, そのエラー種別によって決まり, 以下にそのケースごとに述べられている。

主要エラー(MAJOR ERRORS)

01= 受信バッファ・オーバーフロー

02= コマンド・システム・エクスクルーシブ長規定違反(EOX やステータス・バイトが正しいメッセージ境界に来ていない)

03= コマンドの <count> エラー(システム・エクスクルーシブの長さとの間で整合していない)

04= WRITE 時の情報フィールドの <count> エラー(システム・エクスクルーシブの長さとの間で整合していない)

05= 不正なグループ名(7F)

06= 不正なプロセジャー名(7F)

07= 不正なイベント名(7F)

08= 2次レベルを越える不正な拡張名(すなわち, “名前”を期待している所へ 00 00 00 を受け取った時)

09= セグメンテーション・エラー(COMMAND SEGMENT コマンドの解説参照)

主要エラーが「ディスエーブル」になっている場合の対応

- COMMAND ERROR 情報フィールドの内容を更新(ただし,「エラー停止」フラグはセットせず)。
- この時のシステム・エクスクルーシブの以後の構文解析はすべて打ち切る。
- エラーを含むコマンドは一切実行しない。
- できる限り早く通常の動作を再開する。

即時オペレーション・エラー（IMMEDIATE OPERATIONAL ERRORS）

PROCEDURE や EVENT に埋め込まれているコマンドは，これらのプロセジャーやイベントが実際に実行されるまでは，ここに分類されるエラーを生じることはない。*2

20= アップデート・リスト・オーバーフロー

21= グループ・バッファ・オーバーフロー

22= 未定義プロセジャー

23= プロセジャー・バッファ・オーバーフロー

24= 未定義イベント

25= イベント・バッファ・オーバーフロー

26= タイムコードが来ていない（ブランク・タイムコード）

即時オペレーション・エラーが「ディスエーブル」になっている場合の対応

- COMMAND ERROR 情報フィールドの内容を更新（ただし，エラー停止フラグはセットせず）。
- この時のシステム・エクスクルーシブの構文解析は，次のメッセージ境界部分から再開。
- このエラーを含むコマンドはいっさい実行しない。

インプリメンテーション・エラー（IMPLEMENTATION ERRORS）

40= サポートしていないコマンド

41= 規定のサブコマンドでない

42= 規定のコマンド・データでない

43= コマンド・データ部分にサポートしていない情報フィールド名

44= プロセジャー内の READ または UPDATE リクエスト中にサポートしていない情報フィールド名

45= EVENT のトリガーに，使えない，またはサポートしていないソースを指定

46= ネストになった PROCEDURE[ASSEMBLE]

47= リカーシブになった PROCEDURE[EXECUTE]

48= ネストになった EVENT[DEFINE]

49= EVENT[DEFINE]中に PROCEDURE[ASSEMBLE]がある

60= サポートしていない情報フィールドに WRITE しようとした

61=（定義によって，あるいはインプリメンテーション上）<read only> の情報フィールドに WRITE しようとした

62= 情報フィールドに WRITE すべきデータが規定通りでない

63= WRITE すべき情報フィールド・データ中に，サポートしていない情報フィールド名

インプリメンテーション・エラーが「ディスエーブル」になっている場合の対応

- COMMAND ERROR 情報フィールドの内容を更新（ただし，エラー停止フラグはセットせず）。
- この時のシステム・エクスクルーシブの構文解析は，次のメッセージ境界部分から再開。
- このエラーを含むコマンドはいっさい実行しない。
- もしもPROCEDURE や EVENT の定義中に埋め込まれているコマンドの予備チェック時にエラーの検出された場合は，この定義部分は読み捨て，PROCEDURE[ASSEMBLE]や EVENT[DEFINE]コマンド自体の後に続く次のメッセージ境界部分から解析を再開する。

＜備考＞

1. 電源オンや MMC RESET の後では，COMMAND ERROR 情報フィールドは次のような状態になっているものとする。

<count=04> <flags=00> <level=00> <error=7F> <count_1=00>

*2. PROCEDURE や EVENT に伴うエラー処理の詳細を明確にするために，PROCEDURE 中に EVENT の定義が含まれている次のような例を考えてみる。

F0 7F <device_ID> <mcc>
<PROCEDURE> <count=0A> <[ASSEMBLE]> <procedure_name>
<EVENT> <count=06> <[DEFINE]> <event_name>
<flags=00> <SELECTED TIME CODE> <GP6>
<RECORD STROBE>
F7

(a) もしもこのプロセジャー定義が，用意されているプロセジャー用メモリーより長い場合には，“プロセジャー・バッファ・オーバーフロー”のエラーが発生する。

(b) これに対して，もしもこの中にあるイベントの定義部分が，用意されているイベント用メモリーのスペースを超えてしまうとしても，“イベント・バッファ・オーバーフロー”のエラーは，このプロセジャー定義時には発生しない。ただし，やがてこのプロセジャーが実行されると，その時点で起こり得る。

(c) もしも <provedure_name> が 7F となっていると，“不正なプロセジャー名”エラーが起こることになる。

(d) もしも <event_name> が 7F となっていると，“不正なイベント名”エラーが起こり，かつ，プロセジャー中に埋め込まれている EVENT コマンドの予備チェック時にエラーが起こったのであるから，PROCEDURE[ASSEMBLE]エラー・フラグがセットされる。

(e) もしもレジスター <GP6> をこの装置がサポートしていなかった場合には，“コマンド・データ部分にサポートしていない情報フィールド名”のエラーが発生する。加えて，PROCEDURE[ASSEMBLE]エラーと EVENT[DEFINE]エラーの両フラグも共にセットされることとなる。

44 COMMAND ERROR LEVEL [read/write] [-234]

コマンド・エラーは，COMMAND ERROR LEVEL 情報フィールドをセットすることにより，“イネーブル”にされる。新たに検出されたエラーのエラー・コードが，もしも COMMAND ERROR LEVEL の値よりも小さいか，あるいはそれと等しい場合には，そのエラーはイネーブルとなり，それよりも大きい場合はディスエーブルとされる。

パワー・オンや MMC RESET 後の無指定時の状態は，「すべてのエラーはディスエーブル」になっているものとする。

44	**COMMAND ERROR LEVEL**
<count=01>	**バイト数**
vv	**レベル**
	00= すべてのエラーはディスエーブルに（デフォルト）
	01 ～ 7E: エラーの選択的イネーブル（COMMAND ERROR 情報フィールドの説明を参照）
	7F= すべてのエラーはイネーブルに

＜備考＞

1. 拡張セットのエラー・コードとの比較は，その（非ゼロの）最終バイトだけを使って行われる。
2. オープン・ループ構成で運用の際には，エラーはディスエーブルにしておくことが勧められる。
3. COMMAND ERROR LEVEL は典型的な形では，エラーをその分類区分に従ってイネーブルにするように用いるものである。例えば，1F のレベルはすべての“主要エラー”をイネーブルにし，2F は“主要エラー”と“オペレーション・エラー”をイネーブルにする，等々となる。
4. COMMAND ERROR 情報フィールド及び COMMAND ERROR RERSET コマンドの解説も併せて参照のこと。

45　TIME STANDARD [read/write]　[-234]

制御される装置が使用するタイムコードの公称フレーム・タイプを保持している。デフォルトの値は規定しない。

45	**TIME STANDARD**
<count=01>	**バイト数**
<type>	**フレーム・タイプ： 0 tt 00000**
	tt= フレーム・タイプ・コード
	00 = 24 フレーム
	01 = 25 フレーム
	10 = 30 ドロップ・フレーム
	11 = 30(ノンドロップ)フレーム

＜備考＞

1. それぞれの MMC タイムコード情報フィールドについては，ここでの公称設定内容は，実際のケースでは置き換えられることがある。例えば，SELECTED TIME CODE はタイムコード・リーダーの受信しているフレーム・レートを用いるであろうし，GENERATOR TIME CODE は，新たにタイムコード値がロードされると，TIME STANDARD とは違うフレーム・レートにセットされる場合もある，等。

   各タイムコード情報フィールドに埋め込まれた tt ビット(フレームタイプ)の使用方法に関しては,付録Bの「タイムコード・ステータスの運用細則」を参照。
2. TIME STANDARD を「きれいに」変更するためには，次のようなコマンド・シーケンスを用いることが望ましい。
   <MMC RESET> <TIME STANDARD> <count=01> <type>

46　SELECTED TIME CODE SOURCE [read/write]　[----]

SELECTED TIME CODE 情報フィールドに表示するタイムコードのソースを選択する。タイムコードにアクセスする手段を持たない装置においては，デフォルト設定は“テープ・カウンター”を選んでいること。その他の装置においては，Longitudinal Time Code(LTC)がデフォルトとなっているべきである。

46	**SELECTED TIME CODE SOURCE**
<count=01>	**バイト数**
ss	**ソース ID**
	00=Longitudinal Time Code（LTC）*1 *2
	01=Vertical Interval Time Code（VITC)*2 *3
	02=“テープ・カウンター”*2
	04= 自動 VITC/LTC 切り替え *2 *3 *4
	7F= 装置側で決まる通り(write のみ)

＜備考＞

* 1. パイロット・トーンやバイフェイズ・リーダーといった装置側でのインプリメンテーションは，これを元に Longitudinal Time Code(LTC)の形に合成して表示のこと。
* 2. LTC, VITC, 自動 VITC/LTC 切り替え等では,例えばテープの早巻き中など,選択されたタイムコードがない間は,タコメーターやコントロール・トラック・パルスなどで更新されることがある。
* 3. VITC または自動 VITC/LTC 切り替えのいずれかが選択されている場合で，VITC が使えない時には，指定なくとも LTC をこれに当てる。
* 4. 自動 VITC/LTC 切り替えの特性は，装置の側で決まる。

47　SELECTED TIME CODE USERBITS [read only] [----]

SELECTED TIME CODE 情報から抜き出されたユーザー・ビット群のうち, 最も最近のものを保持している。

情報フィールド SELECTED TIME CODE SOURCE は, このユーザー・ビットのソースを定めている。

47	**SELECTED TIME CODE USERBITS**
<count=09>	**バイト数（コマンドとカウントは含まない）**
u1 ... u9	**標準ユーザー・ビット仕様**

48　MOTION CONTROL TALLY [read only] [-234]

(a) 制御される装置の, 現在の“モーション・コントロール・ステート”を報告し, そのステートが正規に確立したかしたかどうかを示す。

(b) 制御される装置の, 現在の“モーション・コントロール・プロセス”を報告し, そのプロセスを完遂することに成功したかどうかを示す。

モーション・コントロール・ステート及びプロセスに付いては「3　標準仕様」に述べられている。

48	**MOTION CONTROL TALLY**
<count=03+ext>	**バイト数（コマンドとカウントは含まない）**
ms	**最も最近に実行された MCS コマンド**
	00 = 拡張のためリザーブされている
	01 = STOP
	02 = PLAY
	04 = FAST FORWARD
	05 = REWIND
	09 = PAUSE
	0A = EJECT
	45 = VARIABLE PLAY
	46 = SEARCH
	47 = SHUTTLE
	48 = STEP
mp	**最も最近に実行された MCP コマンド**
	00 = 拡張のためリザーブされている
	0B = CHASE
	44 = LOCATE
	7F = MCP は現在アクティブでない *1
ss	**ステータス及び成功レベル： 0 bbb 0 aaa**
	aaa = MCS 成功レベル（下記参照）
	bbb = MCP 成功レベル（下記参照）

各 MCS コマンドに関する有効な「MCS 成功レベル」

STOP	**000 = 状態が変りつつある**
	001 = 完全に停止している
	010 = 失敗した（Failure）
	011 = 推測される動作状態 *2
FAST FORWARD, REWIND,（リゾルブではない）PLAY	
	000 = 状態が変りつつある
	001 = 要求された動作状態になった

010 = 失敗した（Failure）
011 = 推測される動作状態 *2

Resolved PLAY
000 = 状態が変りつつある
001 = プレイ中であり，リゾルブになっている（サーボ・ロック）
010 = 失敗した（Failure）
101 = プレイ中であるが，リゾルブにはなっていない

PAUSE
000 = 状態が変りつつある
001 = 完全に停止している
010 = 失敗した（Failure）

EJECT
000 = 状態が変りつつある
001 = メディアはイジェクトされている／アンロードされている
010 = 失敗した（Failure）

VARIABLE PLAY, SEARCH, SHUTTLE
000 = 状態が変りつつある
001 = 要求された動作状態になった
010 = 失敗した（Failure）

STEP
000 = 状態が変りつつある（Transition in progress）
001 = STEP が完了し，待機位置にある
010 = 失敗した（Failure）
100 = STEP が進行中（STEP in progress）

各 MCP コマンドに関する有効な「MCP 成功レベル」

LOCATE
000 = 将にロケートを実行中
001 = ロケートは完了し，トランスポートは指定のロケート・ポイントにて停止中
010 = 失敗した（Failure）
100 = Deferred Play を保留にして，ロケートを実行中 *3
110 = Deferred Variable Play を保留にして，ロケートを実行中 *4

CHASE
000 = プレイ・モードにて同期させようと将に努力中
001 = 同期状態確立に成功している（プレイ・モードのみ）
010 = 失敗した（Failure）
100 = プレイ以外の（たいていは高速巻き等の）モードにてマスターを追いかけて
いる最中
110 = パーク状態にある *5

＜備考＞

* 1. （LOCATE が DEFERRED PLAY や DEFERRED VARIABLE PLAY では打ち切られないケースを除いて）MCS コマンドの受信により現在の MCP が打ち切られた際には，MCP コマンド・タリー・バイトは，この非動作状態に戻らなくてはならない。
電源オン時や MMC RESET の後もこの状態となる。
この“MCP は現在アクティブでない”となっている間は，“MCP 成功レベル”は常に，000 にリセットするものとする。
* 2. コントローラーと実際のトランスポートとの間にシンクロナイザーや他のインターフェースが介在している場合には，このインターフェースの外で（例えばそのトランスポート自体をオペレーターが操作して）コマンドが出されると，インター

フェースはこれを直接にモニターはできないかも知れないが，（例えばタイムコードの動き等から）正しく推測することができる場合があろう。こうして推定された動作状態は，これを MCS コマンド・タリーに示すものとする。

＜日本語版注＞

制御される装置のタリー情報を直接モニターできる場合にはこの“推測される動作状態”と遠慮する必要はない。例えば，上流からの MMC コマンド PLAY を指示した直後であっても，オペレーターによる操作か何らかの理由で STOP のタリーを確認した場合には，ms はあくまで STOP であり，その成功レベルもまた“001”でかまわない。MCS あるいは MCP のコマンドは，必ずしも MMC コマンドに限らないことに留意されたい（「モーション・コントロール・ステート及びプロセス」の項，及び EJECT コマンドの項参照）。ただし，状態変化に伴う過渡的なタリーは報告されるべきではない。例えば PLAY 中に STOP 指示のあった場合に，内部のブレーキ制御上の要請から一時的に REWIND 制御状態を経過するとしても，ms は REWIND を示してはならない。あくまで PLAY がしばらく続いた後，やがて STOP が，厳密な場合には成功レベル“000”を経由した上で，示されるべきである。

* 3. DEFERRED PLAY コマンドの解説を参照。

* 4. DEFERRED VARIABLE PLAY コマンドの解説を参照。

* 5. パーク状態とは，MCP コマンド CHASE の実行中にのみ意味を持ち，（チェイス中の）スレーブ装置が停止しており，（今は同じく停止中の）マスター装置に何時でも同期できる状態にあることを示している。このステータスは，マスター装置がある地点までの LOCATE を実行した時，スレーブがこれに「追いついた」かどうかをコントローラーが確かめなくてはならない場合に役に立つ。この場合には“ストップ中”との条件だけでは不十分であって，スレーブは自分をマスターの位置に合わせようとしている最中に，このストップ状態を（過渡的に）通過するかも知れないのである。

6. この情報フィールド中の“最も最近に実行された”のバイト部分に，「主要な」あるいは「インプリメンテーションの」コマンド・エラーを引き起こす MCS 及び MCP コマンドが出て来ることは，決してあってはならない（これらの用語の定義については COMMAND ERROR 情報フィールドを参照）。

これとは違って「オペレーショナルな」コマンド・エラーにぶつかるMCS，あるいは MCP コマンドは，MOTION CONTROL TALLY に顔を出すことはあり得る。ただし，“成功レベル”は“失敗した”にセットされている。

7 最新の MCS コマンドが EJECT である場合には，どんな MCS あるいは MCP コマンドもそれを実行しようとすると“失敗した”の成功レベルに終わるはずである。正常な操作が再開できるためには，普通はその前にオペレーターが介入して，イジェクトされたメディアを処置する必要がある。

49 VELOCITY TALLY [read only] [----]

トランスポートの実際の速度を常に示しており，この時実行しているモーション・コントロール・ステートやプロセスとは関係を持たない。

49	**VELOCITY TALLY**
<count=03>	**バイト数**
sh sm sl	**標準スピード仕様**

4A STOP MODE [read/write] [----]

制御される装置が，STOP コマンドの実行結果としてストップしている最中に，記録されている内容をモニターできるように努めるべきか否かを指定する。

4A	**STOP MODE**
<count=01>	**バイト数**
cc	**モード・コード**
	00 = モニター機能はディスエーブル
	01 = モニター機能はイネーブル *1
	7F = 装置側で定まる通り（デフォルト／ write のみ）

＜備考＞

* 1. モニター機能をイネーブルとするとSTOPコマンドは実効的にはPAUSEコマンドとなる。ただし,Record Pause モードのサポートされていないことと，レコーディング中のトラックが常にレコード状態でなくなる点が異なる。

2. 停止中にはモニターのできない装置においては，STOP MODE をサポートする必要はない。

3. この STOP MODE により，コントローラーはそれぞれの装置に対して，あらかじめオペレーターの希望に合うようにこれをセットアップしておくことにより，STOP コマンドを全装置に対して汎用的に適用することができるようになる。
4. STOP MODE を変更しても，すでに確立しているストップ状態に影響することはない。
5. STOP MODE は，コントローラーから明示的に出されたすべての STOP コマンドに対して適用される。装置の制御パネルからのストップ指令に対しては適用されない。モーション・コントロール・プロセス（MCP）の実行中に自動的に出される STOP 指令に対しても，適用されることはない。ただしその動作の結果，装置をストップ状態にして終わる MCP では，すべてその時点で STOP MODE に設定されている内容を尊重しなければならない。
6. STOP MODE は，VTR が停止中に絵の再生状態とするために用いられる。これはまた，カセット型の VTR が STOP コマンドのたびにテープをアンロード（“unthread-ing”）してしまうことのないようにするのにも使われよう。

4B　FAST MODE [read/write]　[----]

制御される装置が，コントローラーからこの後に出される FAST FORWARD や REWIND コマンドの実行中に，記録されている内容をモニターできるように努めるべきか否かを定める。

4B　**FAST MODE**
<count=01>　**バイト数**
cc　**モード・コード**
00 = モニターは伴わずに，最高の速度で移動する
01 = 記録されている内容が認識できる程度には充分な質でモニターのできる範囲内で，最高の速度
7F = 装置側で定まる通り（デフォルト／ write のみ）

＜備考＞

1. FAST MODE は，高速動作の実行中にこの2種のモニター機能のいずれもが，実際に可能な装置のみがサポートすれば良い。
2. FAST FORWARD や REWIND の動作を，現に実行している最中にこのフィールドを書き換えても，影響を与えることはない。
3. FAST MODE は，コントローラーから明示的に出された FAST FORWARD や REWIND コマンドに対してのみ適用される。装置の制御パネルからの指令に対しても，あるいはまたモーション・コントロール・プロセス（MCP）の実行中に，自動的に出される FAST FORWARD や REWINDP 指令に対しても，適用されることはない。
4. FAST MODE は，VTR がテープの巻取り中に，絵を再生状態とするために用いられる。これはまた，カセット型の VTR が高速動作を開始しようとする時に，テープをアンロード（“unthreading”）してしまうことのないようにするのにも使われよう。
5. 装置が映像と音声の両方の出力を有する場合には，FAST MODE の指定に従わなければならないのは映像のモニター機能だけである。音声の同時モニター機能は，装置製造者の裁量範囲とする。
6. FAST MODE は，音声出力を外部でのアッテネーターなしでも快適に聴けるだけのレベルに調節できるようになっているのでない限り，ATR のテープ・リフター機構の制御の目的には用いるべきでない（出力レベルとは関係なしに，ダイナミックにリフターをコントロールする機能は，MIDI マシン・コントロール中の別の所に用意されている）。
早巻きスピードでのモニター機能を制御できない ATR においては，FAST MODE はサポートすべきでない。

4C　RECORD MODE [read/write]　[-234]

これ以降の RECORD STROBE あるいは RECORD STROBE VARIABLE コマンドでの動作モードを選択する。いくつかのトラックに対して，現にレコーディング（あるいはリハーサル）の行われている最中にこのフィールドを書き換えても，当該トラックに影響することはない。

4C	**RECORD MODE**
<count=01>	**バイト数**
dd	**モード**
	00 = ディスエーブル
	01 = ATR: Record;　VTR: Insert　*1
	02 = ATR: Record;　VTR: Assemble
	04 = リハーサル(Rehearse)
	05 = ATR: Record;　VTR: Crash/Full Record
	7F = 装置側での選択通り（デフォルト／ write のみ）

ATR:　Record　TRACK RECORD READY 情報フィールドで指定されたすべてのトラックに対してレコード操作を行う。

Rehearse　古い記録内容を実際に消したり，新たに記録したりはしないが，他のすべてのモニター機能は，“01 = ATR:Record;VTR: Insert”モードでのレコーディング操作を行なった場合を真似た，同等の結果が得られる。*2

VTR:　Insert　コントロール・トラック情報があらかじめ記録済みのテープに対して，レコーディングを行う場合を想定する。レコーディング中にコントロール・トラックは保存される。

記録のパンチ・イン，パンチ・アウトのいずれの箇所においても，前に記録されていた素材と新たな素材とが，クリーンで正確なタイミングで切り替えることができるようになっている。

レコーディングを行うチャネルは，TRACK RECORD READY 情報フィールドで決定される。

VTR:　Assemble　コントロール・トラックとすべてのプログラム・チャネルは上書きされる。*3*4

前に記録されていた素材と新たな素材とは，記録のパンチ･インの箇所においてのみ，「そこに以前の素材が確かに記録されていること」との前提の上で，クリーンで正確なタイミングで切り替えられるようになっている。

VTR:　Crash/Full Record　コントロール・トラックとすべてのプログラム・チャネルは上書きされる。*3

前に記録されていた素材と新たな素材との間でのクリーンな切り替えを図ったり，コントロール・トラックのタイミングとマッチさせたりといったことには，いっさい意を払わないやり方を指す。

＜備考＞

*1.　たいていのレコーディングは“01 = ATR：Record：VTR: Insert”のレコード・モードで行われる。

*2.　多くの装置においては，RECORD STROBE コマンドを受け取ってから実際にリハーサルの開始するまでの遅れ時間は，RECORD STROBE コマンドを受け取ってから実際にレコーディングが開始するまでの時間との間で，わずかながら違いがある。

*3.　TRACK RECORD READY 情報フィールドは，サポートされていればであるが，“VTR: Assemble”と”VTR: Crash/Full Record”のいずれのモードが選ばれるかで影響を受けることはない。しかしながら，これらのモードは TRACK RECORD READY 情報フィールドでのすべての設定よりも優先し，すべてのトラックとコントロール・トラックをレコーディング状態にする。

*4.　現時点での MMC は，「トラック選択が可能」な“VTR: Assemble”モードをサポートはしていない。

5.　VTR 用のモード:“Insert”，“Assemble”，“Crash/Full Record”等は，同様のコントロール・トラック手法を採用している VTR 以外の装置に対しても，適用ができるであろう。

＜日本語版注＞

6.　RECORD MODE の装置側でのデフォルト状態について規定はない。最も“安全”であるという理由で,“Disabled”にすることは妥当のように思われる。しかし，RECORD STROBE コマンドを送るにあたって，RECORD MODE の設定を行わないコントローラーがあって，問題を起こしている。

よって,RECORD MODE に関しては,「Record 可状態を装置側のデフォルトとする」または,「装置側で,ユーザーが RECORD MODE（あるいはそのデフォルト）を変えられる」という仕様にすることを推奨せざるを得ない。

4D　RECORD STATUS [read only]　[-234]

制御される装置内で現に実行中のレコードあるいはリハーサルの動作内容を示す。

4D	**RECORD STATUS**
<count=01>	**バイト数**
ss	**ステータス：0 d c b aaaa**
	aaaa ＝ 現在のレコード／リハーサル動作（16 進桁）
	0 ＝ レコーディング／リハーサル中ではない
	1 ＝ ATR: レコーディング　VTR: Insert Recording
	2 ＝　VTR: Assemble Recording　*1
	4 ＝ リハーサル中
	5 ＝　VTR: Crash/Full Recording　*1
	6 ＝ レコード・ポーズ（Record-Pause）中
	b ＝ ローカル・レコード・インヒビット・フラグ（1 ＝ インヒビットされている）*2
	c ＝ ローカル・リハーサル・インヒビット・フラグ（1 ＝ インヒビットされている）*2
	d ＝(レコードやリハーサルで）アクティブなトラックはない *3
	TRACK RECORD STATUS の全ビットのNOR。
	aaaa がゼロでない時に限り有効。

＜備考＞

* 1.　RECORD STATUS が“VTR:　Assemble Recording”か“VTR:　Crash/Full Recording”を示している間，TRACK RECORD STATUS 情報フィールドは「使えるトラックはすべてレコーディング中」であることを示すことになる。
* 2.　ローカル・インヒビット・フラグは，オペレーターによる操作，カセットやディスクでの記録（許可／禁止）用タブの設定などによりセットされる。
* 3.　「アクティブなトラックはない」とは，1にセットされていると，「aaaa に示されるレコード［リハーサル］のステータス内容に関わりなく，実際にはレコーディング中［リハーサル中］のトラックがひとつもない」ことを表している。
　この状態が発生するかどうかは装置によって異なる。コントローラーはこのビットを無視することにしても良いし，「レコーディング中［リハーサル中］ではない」状態と見なしても良い。

4E　TRACK RECORD STATUS [read only]　[--3-]

現在レコーディング中［またはリハーサル中］のトラックのビット・マップを内容とする。レコーディング中かリハーサル中かは，RECORD STATUS 情報フィールドでわかる。

どんな場合でも，そのトラックがレコーディング中［リハーサル中］であれば，該当のビットが1にセットされる。使用しないビットは，すべて0でなければならない。制御される装置はこのレスポンスを，ちょうど必要とする分だけのバイトを送れば良い。

レスポンス送信中に含まれていないトラックは，レコーディング中［リハーサル中］ではないと見なされるものとする。レコーディング中［リハーサル中］のトラックがひとつもない場合には，<count> バイトを 00 としても良い。

4E	**TRACK RECORD STATUS**
<count=variable>	**（続くビット・マップの）バイト数**
r0 r1 r2	**標準トラック・ビット・マップ**

4F　TRACK RECORD READY [read/write][--3-]

各々のトラックは，そのトラック・ビット・マップ中の該当するビットが1にセットされると“レコード・レディ”状態となる。

RECORD MODE 情報フィールドでレコーディング，あるいはリハーサルがイネーブルにセットされている場合，次の RECORD STROBE あるいは RECORD STROBE VARIABLE コマンドが来ると，“レコード・レディ”になっているが現

在はレコーディング中[リハーサル中]ではないトラックは，レコード[リハーサル]状態になり，現在はレコーディング中[リハーサル中]であるが「レコード・レディ」になってはいないトラックは，レコード[リハーサル]状態から抜ける。

この情報フィールドの内容を変えることだけで，各トラックがレコード[リハーサル]状態になったり，これから抜けたりすることはない。

レスポンスとして読まれる時には，制御される装置は，必要とするだけの数のバイトを送信すれば良い。レスポンスの送信中に含まれていないトラックは，TRACK RECORD READY 状態にはないものと見なされる。レディ状態のトラックがひとつもない場合には，<count> バイトを 00 とすることができる。

WRITE コマンドで書き込まれる場合には，送信された中に含まれていないトラックは，レディでない状態にセットされる。全トラックをディスエーブルとする際には，<count> バイトを 00 とすることができる。

4F	**TRACK RECORD READY**
<count=variable>	**続くビット・マップのバイト数**
r0 r1 r2 ..	**標準トラック・ビット・マップ**

＜備考＞

1. レコーディングやリハーサルを行う前には，RECORD MODE 情報フィールドで，レコードまたはリハーサルをもイネーブルにしておかなくてはならない。
2. TRACK RECORD READY は，RECORD MODE 情報フィールドで“VTR: Assemble”や”VTR: Crash/Full Record”モードが選ばれても影響を受けることはない。しかしながらこれらのモードは，TRACK RECORD READY でのいかなる設定よりも優先し，すべてのトラックとコントロール・トラックをレコーディング状態にする。
3. TRACK RECORD READY 情報フィールド中の個々のビットを変えるにはMASKED WRITE コマンドが使える。

50 GLOBAL MONITOR [read/write] [--3-]

すべてのトラックについて，プレイバック・モニターかインプット・モニターかのモードを選択する。

2種類の再生モードの定義は：

(a) 大多数の装置にとっては，シンクロナス(同期)モードが「正常な」再生モードと見なされるであろう。この用語は，再生信号が今現在新たに記録している最中のどの信号とも同期しているという事実から来ている。

(b) リプロ(再生)モードとは，記録用とは別に再生専用に調整されたヘッドを独立に備えている ATR 装置に広く見られるモードである。再生ヘッドが記録ヘッドと物理的に離れていることから，再生信号は必然的に記録中の信号とは同期がずれている。

GLOBAL MONITOR の設定は，TRACK SYNC MONITOR や TRACK INPUT MONITOR，及び TRACK MUTE 両情報フィールドの設定によっては，トラックごとのベースでは無視されてしまうことがあり得る。

50	**GLOBAL MONITOR**
<count=01>	**バイト数**
dd	**モード**
	00 = プレイバック(シンクロ)モード(デフォルト)
	01 = インプット／フルEEモード
	02 = プレイバック(リプロ)モード *2
	7F = 装置側で設定の通り(WRITE のみ)

＜備考＞

1. 多くの ATR においては，リプロ・モードが本来の再生モードであると見なすであろうが，装置が MIDI マシン・コントロールの下で使われる場合には，シンクロナス・モードの方をデフォルト(無指定時設定)とすべきである。こうすることですべての制御される装置に対して，一様な扱いができることになる。

* 2. もしもリプロ・モードをサポートしていない場合には，シンクロナス・モードの再生を用いること。

51 RECORD MONITOR [read/write] [----]

記録動作中に，各トラックの入力がそれぞれの出力部でモニターされる際の条件を選択する。

シンクロナス・モードでの再生が選ばれたトラックについてのみ適用される。

この設定は装置の制御パネルで行うか，あるいは GLOBAL MONITOR または TRACK SYNC MONITOR 情報フィールドがサポートされていれば，これに書き込むことでセットされる。

タイムコード用として指定されたトラックについては，この RECORD MONITOR の設定で影響されないこととする。

51 **RECORD MONITOR**

<count=01> **バイト数**

dd **モード**

00 = 記録時のみ（入力をモニター）

01 = 記録時，及び非再生時

02 = 記録時，及びレコード・レディ時

7F = 装置側で設定の通り（デフォルト／ WRITE のみ）

記録時のみ： シンクロナス再生モードにセットされたトラックは，すべて記録中のみ入力をモニターする。これらのトラックは記録動作を終えると，元のシンクロナス再生モードに戻る。

記録時，及び非再生時： シンクロナス再生モードにセットされたトラックは，すべて記録中には入力をモニターする。記録動作を終えると，これらのトラックは元のシンクロナス再生モードに戻る。これに加えてレコード・レディのトラックは，すべてプレイ・モード中でなければ入力のモニター状態となる。

記録時，及びレコード・レディ時： シンクロナス再生モードにセットされたトラックと，レコード・レディあるいはレコーディング中のトラックは，すべて入力をモニターする。

＜備考＞

1. RECORD MONITOR は専ら，オーディオ・トラックについてのみ適用される。
2. 繰り返しになるが，3種の RECORD MONITOR の設定をテーブルの形にすると：

	Record Ready and PLAY	Record Ready and Non-play	Recording
00 記録時のみ	Sync	Sync	Input
01 記録時，及び非再生時	Sync	Input	Input
02 記録時，及びレコード・レディ時	Input	Input	nput

3. 実際のモニター機能は，TRACK INPUT MONITOR 及び／あるいは TRACK MUTE 情報フィールドが優先することで，この設定通りとならない場合がある。

52 TRACK SYNC MONITOR [read/write] [----]

それぞれの出力部にシンクロナス・モードの再生信号を出すべきトラックを個別的に選択する（シンクロナス・モードの再生については GLOBAL MONITOR 情報フィールドの項参照）。

TRACK SYNC MONITOR は，常にその指定トラックについては GLOBAL MONITOR の設定よりも優先するが，TRACK INPUT MONITOR 及び TRACK MUTE 情報フィールドは，これよりもさらに優先する。任意のあるトラックについて，この優先順位をテーブルにすると（x は，1,0 どちらであっても関係ないことを示す）。

TRACK SYNC MONITOR bit	TRACK INPUT MONITOR bit	TRACK MUTE bit	結果のモニター機能
0	0	0	GLOBAL MONITOR の定義通り
1	0	0	シンクロナス・モードの再生
x	1	0	インプット・モニター
x	x	1	ミュート

レスポンスとして読まれる場合には，制御される装置は TRACK SYNC MONITOR の必要なだけのバイト数を送れば良い。送信されたレスポンス中にないトラックは，個別的にはシンクロナス再生モードに選択されてはいないものと見なされる。個別的に選択されたトラックがひとつもない場合には，<count=00> としても良い。

WRITE コマンドで書かれる場合には，送られた中に含まれていないトラックの個別的 TRACK SYNC MONITOR ビットは，0にリセットされるものとする。全トラックをリセットする場合には，<count=00> としても良い。

52	**TRACK SYNC MONITOR**
<count=variable>	**続くビット・マップのバイト数**
r0 r1 r2..	**標準トラック・ビット・マップ**

＜備考＞

1. TRACK SYNC MONITOR は，GLOBAL MONITOR 情報フィールドでのリプロ再生モードを補完するものであって，これによりリプロ再生のトラックとシンク再生のトラックとが混じった組み合わせが可能となる。こうした類の機能は，元々 ATR のオーディオ・トラックに関連して用意されたものである。
2. レコーディング動作中のシンクロ・モニター・トラックにはさらに，RECORD MONITOR 情報フィールドが影響して来る。
3. TRACK SYNC MONITOR 情報フィールド中の個々のビットを変更するのには，MASKED WRITE コマンドを用いることができる。
4. READ や UPDATE に対しては，TRACK SYNC MONITOR の設定内容が戻されるのみであって，GLOBAL MONITOR，TRACK SYNC MONITOR 及び TRACK INPUT MONITOR 及び TRACK MUTE の組み合わせで決まってくる，最終的なトラックの構成を反映していない場合がある。

53 TRACK INPUT MONITOR [read/write] [----]

それぞれの出力部に入力のモニター信号を出すべきトラックを個別的に選択する。

TRACK INPUT MONITOR は常に，TRACK SYNC MONITOR と GLOBAL MONITOR 両情報フィールドのいずれの設定内容にも優先するが，TRACK MUTE 情報フィールドは，これよりもさらに優先する。任意のあるトラックについて，この優先順位をテーブルにすると（x は，1,0 どちらであっても関係ないことを示す）。

TRACK SYNC MONITOR bit	TRACK INPUT MONITOR bit	TRACK MUTE bit	結果のモニター機能
0	0	0	GLOBAL MONITOR の定義通り
1	0	0	シンクロナス・モードの再生
x	1	0	インプット・モニター
x	x	1	ミュート

レスポンスとして読まれる場合には，制御される装置は TRACK INPUT MONITOR の必要なだけのバイト数を送れば良い。送信されたレスポンス中にないトラックは，個別的にはインプット・モニター・モードに選択されてはいないものと見なされる。個別的に選択されたトラックがひとつもない場合には，<count=00> としても良い。

WRITE コマンドで書かれる場合には送られた中に含まれていないトラックの個別的 TRACK INPUT MONITOR ビットは，0にリセットされるものとする。全トラックをリセットする場合には，<count=00> としても良い。

53	**TRACK INPUT MONITOR**
<count=variable>	**続くビット・マップのバイト数**
r0 r1 r2 .	**標準トラック・ビット・マップ**

＜備考＞

1. TRACK INPUT MONITOR 情報フィールド中の個々のビットを変更するのには, MASKED WRITE コマンドが用いられることもある。
2. READ や UPDATE に対しては, TRACK INPUT MONITOR の設定内容が戻されるのみであって, GLOBAL MONITOR, TRACK SYNC MONITOR, TRACK INPUT MONITOR, 及び TRACK MUTE の組み合わせで決まってくる最終的なトラック・モニターの構成を反映していない場合がある。

54 STEP LENGTH [read/write] [----]

STEP コマンドで用いる距離の単位を規定する。

STEP LENGTH 自体は, タイムコードの1フレームの 1/100 の単位で表す。

54 **STEP LENGTH**
<count=01> **バイト数**
nn **1STEP 単位中の 1/100 フレーム(= サブフレーム)の数**
デフォルト値= 32 H(50 decimal = 1/2 フレーム)

55 PLAY SPEED REFERENCE [read/write] [-23-]

制御される装置が標準プレイ・モード時に, そのスピードを内部的に制御するのか, それとも外部のソースとなるものから直接プレイ・スピードを制御させるのかを定める。内部的な CHASE の実行中, あるいは"自由リゾルブ"のプレイ・モード中には, プレイ・スピードは同期に引き込むための内部的な制御の下にあるので, このフィールドの内容は意味を持たない。

55 **PLAY SPEED REFERENCE**
<count=01> **バイト数**
rr **基準**
00 = 内部
01 = 外部
7F = 装置側で選択された通り(デフォルト／ write のみ)

56 FIXED SPEED [read/write] [----]

制御される装置が2速以上をサポートしている場合に, 固定プレイ・スピードの公称値を選択する。これ以外のすべてのスピード測定においては, この固定スピードに対して相対的に計算するものとする。

56 **FIXED SPEED**
<count=01> **バイト数**
pp **スピード**
:
3F = その次に遅いスピード
40 = 中間, あるいは「標準の」スピード
41 = その次に速いスピード
:
7F = 装置側で選択された通り(デフォルト／ write のみ)

＜備考＞

1. もしも WRITE コマンドが範囲外のスピード値を指定した場合は，可能なスピード中で最も近い値を有効にする。
2. スピード値は連続的に割り付けられていること。
3. 例：
   (a) 3速の ATR があって，その低速，中速，高速をそれぞれ 3F,40,41 に割り振ったものとする。WRITE の際には，00 から 3F までの範囲の値は，すべて実際には低速を設定し，41 から 7E までの範囲の値は高速を設定する。
   (b) VHS のビデオ・デッキでは，40 を通常のスピードに割り当て，3F,3E は代わりに選べる長時間プレイ用の低スピードを指定することになろう(それぞれ標準：SP，2倍録画時間モード：LP，3倍モード：EP)。

57 LIFTER DEFEAT [read/write] [----]

リール・トゥ・リール型の制御される装置の，テープ・リフター機構を効かなくさせて，テープがヘッドに接するようにさせる。

57 **LIFTER DEFEAT**
<count=01> **バイト数**
cc **制御**
00 = リフターが働く(No deteat)(デフォルト時)
01 = リフターが効かず，テープがヘッドに接したまま
7F = 装置側で決まる通り(WRITE のみ)

＜備考＞

多くの ATR ではリフターを効かなくすると，オーディオ・モニター・レベルが過大となることがある。

58 CONTROL DISABLE [read/write] [---4]

ディスエーブルにされると，制御される装置は“トランスポート制御(Ctrl)と同期(Sync)”の両メッセージ・タイプに関わるすべてのコマンドと，このタイプの情報フィールドを指定したすべての WRITE を無視する。それが直接コントローラーから出されたものであるか，プロセジャーやイベントの実行の結果出たものであるかを問わない(メッセージ・タイプの定義については「インデックス・リスト」を参照)。その他のメッセージ・タイプは，アクティブのままになっている。

コントローラーは,これによって装置自体のいかなる直接制御もできなくなるが,すべての情報フィールドをモニター(READ)し続けることはできる。

58 **CONTROL DISABLE**
<count=01> **バイト数**
cc **制御**
00 = イネーブル(デフォルト)
01 = ディスエーブル
7F = 装置側で選択された通り(write のみ)

＜備考＞

1. この情報フィールドは，代表的にはコントローラーとターゲット装置との間に挿入される，シンクロナイザーやその他のインターフェースで使われる。この効果としては,シンクロナイザーがその制御出力部をディスエーブルにすることで,ターゲット装置を外部の制御からは完全に自由にすることにある。
2. CONTROL DISABLE 情報フィールド自体は，ディスエーブルにしないこと。

59　RESOLVED PLAY MODE [read/write][---4]

制御される装置が PLAY コマンドを受け取った際に，フォワード方向オペレーションの公称規定スピードに達する方法を選択する。

59	**RESOLVED PLAY MODE**
<count=01>	**バイト数**
dd	**モード**
	00 = リゾルブでない。(“Normal”)
	01 = フリー・リゾルブ・モード(“Free Resolve Mode”)
	7F = 装置側で選択された通り(デフォルト／ write のみ)
“Normal”:	PLAY SPEED REFERENCE で決められた方式で PLAY を確立する。タイムコードやその他のフレーム・レファレンスとの関係は一切含んでいない。
“Free Resolve Mode”:	SELECTED TIME CODE のフレーム・エッジを装置側で定めたフレーム・リファレンスに，そのデータ内容とは関わりなく一致させるという方式で PLAY を達成し，その後もデータとは関係なく融合状態(リゾルブ)を維持する。 *1

＜備考＞

*1. MIDI マシン・コントロール仕様の将来のバージョンにおいては，このフレーム・リファレンスを，例えばビデオ・リファレンス信号や，あるいは単純に入力 SELECTED MASTER CODE の同期ワードから，選択できるようにすることが考えられる。

2. この情報フィールドは，もともと自己リゾルブ式の(self-resolving，例えばほとんどの VTR のような)装置ではサポートする必要はない。

5A　CHASE MODE [read/write]　[---4]

制御される装置が CHASE コマンドを受け取った際に，その SELECTED TIME CODE と SELECTED MASTER CODE との間で同期を確立し，これを維持する方式を選択する。

5A	**CHASE MODE**
<count=01>	**バイト数**
dd	**モード**
	00 = 絶対標準モード(Absolute Standard Mode)
	01 = 絶対リゾルブ・モード(Absolute Resolve Mode)
	7F = 装置側で選択された通り(デフォルト／ write のみ)
Absolute Standard Mode:	SELECTED MASTER CODE のデータに関連を持って同期を確立し，その後もデータに依存しつつ同期を維持する。
Absolute Resolve Mode:	SELECTED MASTER CODE のデータに関連を持って同期を確立し,その後はデータとは独立に同期を維持する。

＜備考＞

リゾルブ，すなわち「データに依存しない」ロックとは，単に SELECTED TIME CODE のフレーム・エッジを適当なマスターのフレーム・リファレンスに同期させるやり方を指す。リゾルブ中は，タイムコードの実際の数値は無視される(MIDI マシン・コントロール仕様の将来のバージョンにおいては，このフレーム・リファレンスを，例えばビデオ・リファレンス信号や，単純に入力 SELECTED MASTER CODE の同期ワードから，選択できるようにすることが考えられる)。

5B　GENERATOR COMMAND TALLY　[read only] [----]

タイムコード・ジェネレーターの動作状態を報告する。

5B	**GENERATOR COMMAND TALLY**
<count=02>	**バイト数**
gg	**最後に受け取ったジェネレーター・コマンド**
	00 = ストップ(Stop)
	01 = ラン(Run)
	02 = コピー／ジャム(Copy /Jam)
ss	**ステータス(真なら 1)と成功レベル :　0 0 cb 0 aaa**
	aaa = 成功レベル
	000 = 状態が変わりつつある最中
	001 = うまくいっている(Successful)
	010 = うまくいっていない(Failure)
	b =(Copy や Jam の間に)元のタイムコード・データが来なくなった
	c = 外部からのフレーム同期基準信号がなくなった

5C　GENERATOR SET UP　[read/write] [----]

タイムコード・ジェネレーターの動作モードを制御する。

5C	**GENERATOR SET UP**
<count=03+ext>	**バイト数**
<reference>	**ジェネレーターのフレーム同期基準 :　0 yyy 0 nnn**
	nnn = Run モード時のフレーム同期基準
	000 = 内蔵の水晶“Standard” モード *1
	001 = 装置側で定まる, 外部からのフレーム同期基準
	010 = 内蔵の水晶“Drop A” モード *1
	011 = 内蔵の水晶“Drop B” モード *1
	111 = 装置側で定まる通り(write のみ)
	yyy = Copy/Jam モード時のフレーム同期基準
	000 = 元のタイムコードのフレーム・エッジ
	001 = 装置側で定まる, 外部からのフレーム基準
	111 = 装置側で定まる通り(write のみ)
<source>	**Copy/Jam 用タイムコード・ソース**
	00 = 拡張用にリザーブされている
	01 = SELECTED TIME CODE(デフォルト)
	02 = SELECTED MASTER CODE
	7F = 装置側で定まる通り(write のみ)
<copy/jam>	**Copy/Jam モード**
	00 = もしもコピーされる GENERATOR TIME CODE にとってのタイムコード・ソースが止まったり無くなった場合には, GENERATOR TIME CODE も止まること。
	01 = もしもコピーされる GENERATOR TIME CODE にとってのタイムコード・ソースが止まったり無くなった場合でも, GENERATOR TIME CODE の出力数値列はいささかも乱れることなしにランし続けること(別名, TimeCode Jam モードとも呼ばれる)。

＜備考＞

*1.　内蔵水晶の発振レート表

	GENERATOR TIME CODE のフレーム・タイプ(tt)			
nnn	24	25	30DF	30
000(Standard)	24	25	29.97	30
010(Drop A)	24	25	29.97	29.97
011(Drop B)	23.976	24.975	29.97	29.97

2.　MMC の将来のバージョンでは, GENERATOR のカラー・フレーミングをサポートすることが考えられる。

5D　GENERATOR USERBITS [read/write]　[----]

タイムコード・ジェネレーターの生成する現在のユーザー・ビットの内容を保持する。

5D　**GENERATOR USERBITS**
<count = 09>　**バイト数**
u1 thru u9　**標準ユーザー・ビット仕様**

5E　MIDI TIME CODE COMMAND TALLY [read only]　[----]

MIDI タイムコード・ジェネレーターの動作状態を報告する。

5E　**MIDI TIME CODE COMMAND TALLY**
<count=02>　**バイト数**
mm　**最後に受け取った MIDI TIME CODE COMMAND**
00 = MTC オフ
02 = タイムコードに従う(MTC オン)
ss　**ステータス : 0 0000 aaa**
aaa = 成功レベル
000 = 状態が変わりつつある最中
001 = うまくいっている(Successful)
010 = うまくいっていない(Failure)

5F　MIDI TIME CODE SET UP　[read/write] [----]

MIDI タイムコード・ジェネレーターの動作モードを制御する。

5F　**MIDI TIME CODE SET UP**

<count=02+ext>　**バイト数（コマンドとカウントは含まない）**

<flags>　**MTC フラグ : 0 gfedcba**

a =「ストップ中送信」フラグ

0 = タイムコード・ソースの止まったことが検出されると, MTC の送信をやめる

1 = ソースが止まっても, ビットbで指定されるデータ・タイプで送信を続ける（イネーブル）

b = ビットaがイネーブルの場合の「ストップ」・データ・タイプ

0 = フレーム値は増やさずに, クォーター・フレーム・メッセージを送信

1 = 特に規定はしないが, 一定のインターバルでフル・メッセージを送信

c =「高速動作中送信」フラグ

0 = タイムコード・ソースが公称フレーム・レートの少なくとも2倍以上で変わっていることが検出されると, MTC の送信をやめる

1 = タイムコード・ソースが公称フレーム・レートの少なくとも2倍以上で変わっていることが検出されると, ビットdで指定されるデータ・タイプで送信を続ける（イネーブル）

d= ビットcがイネーブルの場合の「高速時」・データ・タイプ

0 = クオーター・フレーム・メッセージを送信 *1

1 = 特に規定はしないが一定のインターバルで, フル・メッセージを送信

e =「ユーザー・ビット送信」フラグ

0 =MTC 中のユーザー・ビットの送信をやめる

1 = ユーザー・ビットの内容が変わった時は常に, あるいは特に規定はしない一定間隔で, ユーザー・ビット・メッセージを送信

f =MMC レスポンス側ケーブル・ミュート・フラグ

0 = 動作中には, MTC は MMC のレスポンス・ケーブル上に出力される

1 = MTC は, MMC のレスポンス・ケーブル上には出力しない。ただし, 特に規定はしない他の MIDI OUT ポート上に出すことはあり得る。

g = 0

<source>　**タイムコードのソース**

00 = 拡張用にリザーブされている

01 = SELECTED TIME CODE

02 = SELECTED MASTER CODE

06 = GENERATOR TIME CODE

07 = MIDI TIME CODE INPUT（ソフト THRU モード）

7F = 装置側で定まる通り（write のみ）

＜備考＞

*1. 高速のタイムコード・ソースに間違いなくついて行けるために, さらにまた MTC を正しく受信できるようにするために, ここでのクォーター・フレーム・メッセージ は：

(a) 公称フレーム・レートで進むこと。さらに,

(b) 送信は「バースト群」で行なわれること。ここに個々のバーストは, 通常通り1フレームずつ増えるタイムコード数フレーム分よりなる。

2. 送信される MIDI タイムコードのフレーム・タイプ・フラグ（tt）は, タイムコード・ソースのそれと同じにされる。

60　PROCEDURE　RESPONSE　[read only]　[--34]

コントローラーが，アセンブルされた任意のあるいはすべてのプロセジャーを読み返すことができるようにする。読み返すべきプロセジャーの名前は，始めに PROCEDURE[SET]コマンドによって確定しておかなければならない。

もしも PROCEDURE[SET]コマンドが“全プロセジャーのセット”を指定した場合は，制御される装置内にその時点でアセンブルされているプロセジャーごとに，それぞれ別々の PROCEDURE RESPONSE が送信される。PROCEDURE コマンドの項をも参照のこと。

60	**PROCEDURE RESPONSE**
<count=variable>	**バイト数（コマンドとカウントは含まない）**
<procedure>	**00 から 7E の範囲内のプロセジャー名** 7F = 不正規のプロセジャー・セット （すなわち，PROCEDURE[SET]コマンドが1度も出されていないか，あるいは，未定義のプロセジャーが指定されたか，または現在ひとつもプロセジャーが定義されていない場合）以下のデータは不要である（count=01）。
<command #1. .>	
<command #2. .>	
<command #3. .>	
:	
:	

61　EVENT　RESPONSE　[read only]　　[--34

コントローラーが，任意のあるいはすべての定義済みイベントを読み返すことができるようにする。読み返すべきイベントの名前は，始めに EVENT[SET]コマンドによって確定しておかなければならない。

もしも EVENT[SET]コマンドが“全イベントのセット”を指定した場合は，制御される装置内にその時点で定義されているイベントごとにそれぞれ別々の EVENT RESPONSE が送信される。

EVENT コマンドの項をも参照のこと。

61	**EVENT RESPONSE**
<count=variable>	**続くバイトのバイト数**
<event>	**00 から 7E の範囲内のイベント名** 7F = 不正規のイベント・セット （すなわち EVENT[SET]コマンドが1度も出されていないか，あるいは，未定義のイベントが指定されたか，または現在ひとつもイベントが定義されていない場合）以下のデータは不要である（count=01）。
<flags>	**イベント制御フラグ（ビットの定義に関しては EVENT[DEFINE]コマンドを参照）**
<trigger source>	**イベントがトリガーされる際に基準とするタイムコードを指定する情報フィールド名（EVENT[DEFINE]コマンドを参照）**
hr mn sc fr ff	**イベント時刻（{ff} タイプ）**
<command..>	**ひとつのコマンドと，これに伴うデータ**

62　TRACK MUTE [read/write]　[--3-]

出力信号にミュートを掛けるべき個々のトラックを選択する。

TRACK MUTE は他のモニター選択のいずれよりも優先される。

レスポンスとして読まれる場合は，制御される装置は TRACK MUTE の必要なだけのバイト数を送れば良い。送信されたレスポンス中にないトラックは，個別的にはミュートは選択されていないものと見なされる。個別的にミュートを選択されたトラックがひとつもない場合には，<count=00> としても良い。

WRITE コマンドで書かれる場合には送られた中に含まれていないトラックの個別的 TRACK MUTE ビットは0にリセットされる（ミュートが外される）。全トラックをミュート解除にする場合には，<count=00> としても良い。

62	**TRACK MUTE**
<count=variable>	**続くビット・マップのバイト数**
r0 r1 r2	**標準トラック・ビット・マップ *2**

＜備考＞

* 1.　TRACK MUTE 情報フィールド中の個々のビットを変更するのには，MASKED WRITE コマンドが用いられることもある。

* 2.　TRACK MUTE は，元来オーディオ・トラックを対象にしたものである。ビデオのビットは，通常ゼロにしておく。

63　VITC INSERT ENABLE [read/write]　[----]

制御される装置が，そのビデオ信号入力部から受信中のビデオ信号に VITC（Vertical Interval Time Code）を，挿入合成するかどうかを選択する。もしもこの入力ビデオ信号がこの後記録されると，これに伴いこの VITC 情報も一緒に記録されることとなる。

VITC はこの装置内のタイムコード・ジェネレーターから取り出される。したがって，GENERATOR COMMAND 及び GENERATOR SET UP 情報フィールドによっても制御される。*1

63	**VITC INSERT ENABLE**
<count=03>	**バイト数**
cc	**制御コード** 00 = ディスエーブル 01 = イネーブル 7F = 装置側で定まる通り（デフォルト／ write のみ）
h1	**VITC 挿入開始水平ライン番号** 0AH ～ 12H（NTSC） 06H ～ 14H（PAL） 7F = 装置側で定まる通り（デフォルト／ write のみ）
h2	**VITC 挿入第2水平ライン（非隣接）番号** ここに，h2 >(h1+1） 0CH ～ 14H（NTSC） 08H ～ 16H（PAL） 7F = 装置側で定まる通り（デフォルト／ write のみ）

＜備考＞

* 1.　VITC がイネーブルにされると，タイムコード・ジェネレーター部は必然的にビデオ信号に基準を求めることになるので，GENERATOR SET UP 情報フィールド中の <reference> データは，内部的にこれに合わせられることになる。

64　RESPONSE　SEGMENT　［アクセス不可］　［-234］

MMC システム・エクスクルーシブの定める最大データ・フィールド長（48バイト）よりも長いレスポンス（あるいはレスポンス列）を，いくつかのセグメントに分割して，複数個のシステム・エクスクルーシブの形で部分部分ごとに送信することができるようにする。

コントローラーは，このようにして受信したレスポンスは，あたかも一個のシステム・エクスクルーシブで，全部一度に届けられたかのようにして扱うものとする。

RESPONSE SEGMENT は，常にそのシステム・エクスクルーシブの中の先頭のレスポンスでなければならず，またこのシステム・エクスクルーシブ中には，このレスポンス・セグメントの本体部分に含まれているもの以外には，他のレスポンスは一切あってはならない。

セグメントの切れ目は，レスポンスの境界に合っている必要はない。コントローラーは，RESPONSE SEGMENT システム・エクスクルーシブの終端部で出てくるであろうこの未完結レスポンスを検出し，次のセグメントが届きしだい，正しくレスポンス処理が続行できなくてはならない。

コントローラーが，ある制御される装置からの後続セグメントを期待して待っている時に，WAIT や RESUME メッセージ以外のセグメント化されていない（普通の）システム・エクスクルーシブが，同じ制御される装置から送られた場合には，セグメント復元は打ち切りにして，こちらを通常通りに処理するものとする。　*1

説明と例については，「2 全体構造－セグメンテーション」の項を参照。

64	**RESPONSE SEGMENT**
<count=variable>	**バイト数（レスポンス・ストリング・セグメント長 + 1）**
si	**セグメント ID：　0 f ssssss**
	f:　1 = 先頭セグメント
	0 = 後続セグメント
	ssssss = セグメント番号（降順，最後が 000000）
<..responses..>	**レスポンス・ストリング・セグメント**

＜備考＞

*1.　コントローラーは，これにつながれた複数の制御される装置それぞれについて，別々のセグメント復元処理ができるようになっていなければならない（ある制御される装置からのセグメント処理の進行中に，他の装置からの通常のシステム・エクスクルーシブが来ても，これに影響されることのないこと）。

65　FAILURE　［アクセス不可］　［-234］

制御される装置で生じたどうにもならない不具合，すなわち装置側でのオペレーターによる直接介入を必要とするような状態の発生を知らせる。

65	**FAILURE**
<count>	**バイト数（データのない場合は <count=0>）**
<data ..>	**コントーラー側でオプショナルな表示に使える ASCII データ（表示のサイズに合わせて後部切り捨ての可能性あり）**

7C　WAIT ［アクセス不可］　［-234］

WAIT レスポンスはコントローラーに対して，制御される装置の受信バッファがいっぱいになりつつあること（あるいは装置が他の理由で対応できない状態であること），したがってマシン・コントロール・コマンドの送信は制御される装置からRESUME を受け取るまでの間，中断すべきことを知らせる。この時点で送信中であったコマンドについては，その種類を問わず，通常の EOX（F7）までは続けることが許されるようになっていなければならない。引き続くコマンドの送信は，制御される装置からの RESUME を受信した後に再開してよい。

しかしながら WAIT と RESUME のコマンドは，WAIT レスポンスの後であっても拒絶されることはない。同様にWAIT レスポンスの送信自体が，WAIT コマンドを受信したからといって拒否されることもない。

コントローラーは以下の要請を満足しなければならない。

(i)　この WAIT メッセージの EOX(F7)の到着から10m秒以内には，この WAIT 指示を認識できること；　及び，

(ii)　続いて，この次に可能な MMC システム・エクスクルーシブ境界ですべての送信を停止すること（したがって，実際に停止状態となるまでには，最大 53 バイトまでの MMC での最長システム・エクスクルーシブは，送信せざるを得ないことが許される）。

WAIT レスポンスは常に，ひとつのシステム・エクスクルーシブ中の唯一のレスポンスとして送信される。

すなわち：F0 7F <device_ID> <mcr> <WAIT> F7

```
7C          WAIT
```

＜備考＞

1. WAIT レスポンスの正しい動作のためには，制御される装置内の MIDI 受信バッファには，ある最低限のサイズが必要となる。付録E「受信バッファ・サイズの決定」を参照。
2. 制御される装置は，もしもその受信バッファにそれでも未だデータが来るようであったら，再度 WAIT レスポンスを送信しても良い。

7F　RESUME ［アクセス不可］　［-234］

コントローラーに対して，制御される装置がマシン・コントロール・コマンドを受信できる態勢になったことを知らせる。（電源オン時の）デフォルトの状態は，「受信準備完了」である。

RESUME レスポンスは，主としてコントローラーが WAIT レスポンスの後で，再び送信を始められるようにするのに用いられる。

RESUME レスポンスの送信は，WAIT コマンドの受信後であっても拒否されることはない。

RESUME レスポンスは，ひとつのシステム・エクスクルーシブ中の唯一のレスポンスとして送信される。

すなわち：F0 7F <device_ID> <mcr> <RESUME> F7

```
7F          RESUME
```

付録A　応用例

例1

以下のようなコマンドと情報フィールドをサポートするだけの，極めて基本的なテープ・トランスポートが作られたものとする。

コマンド　　:<STOP>,<DEFERRED PLAY>,<FAST FORWARD>,<REWIND>,
<RECORD STROBE>,<RECORD EXIT>,<MMC RESET>,<WRITE>,<LOCATE>,<MOVE>;

情報フィールド :<SELECTED TIME CODE>,<GPO/LOCATE POINT>.

製造者はこの装置の SIGNATURE を次のように公表する:

```
01 00 00 00
0C
7B 41 00 00 00 00 00 00 00 00
11 20
02
02 02
```

通信は，“オープン・ループ”モードのみで行われる。このトランスポートは，コマンドをその MIDI IN ポートで受けるが，MIDI OUT には応答を出力しない。この装置のデバイス ID は DIP スイッチで設定でき，これが 01H に設定されている。

以下のコマンド・シーケンスは，典型的な“オープン・ループ”スタイルでの MMC 操作である。

プレイ:

```
F0 7F 01 <mcc> <DEFERRED PLAY> F7
```

ストップ:

```
F0 7F 01 <mcc> <STOP> F7
```

「カウンター」をゼロにクリアーする（30ND フレーム）:

```
F0 7F 01 <mcc> <WRITE> <count=06> <SELECTED TIME CODE> 60 00 00 20 00 F7
```

ファースト・フォワード:

```
F0 7F 01 <mcc> <FAST FORWARD> F7
```

ストップ:

```
F0 7F 01 <mcc> <STOP> F7
```

ロケート・ポイントの設定:

```
F0 7F 01 <mcc> <MOVE> <count=02> <GPO/LOCATE POINT> <SELECTED TIME CODE> F7
```

プレイ:

```
F0 7F 01 <mcc> <DEFERRED PLAY> F7
```

パンチ・イン（Punch into record）:

```
F0 7F 01 <mcc> <RECORD STROBE> F7
```

パンチ・アウト（Punch out of record）:

```
F0 7F 01 <mcc> <RECORD EXIT> F7
```

ロケート・ポイントに戻って再度プレイ:

```
F0 7F 01 <mcc> <LOCATE> <count=02> <[I/F]> <GPO/LOCATE POINT> <DEFERRED PLAY> F7
```

リターン・トゥ・ゼロ（Return to Zero）:

```
F0 7F 01 <mcc> <LOCATE> <count=06> <[TARGET]> 60 00 00 00 00 F7
```

例2

次の2つの例では，MIDI マシン・コントロール（MMC）と MIDI タイムコード（MTC）とが結び付けられている。コンピューター・ベースのシーケンサーで1台の装置を制御する場合には，このうちのいずれかの方式が典型的に用いられることになるであろう。

例2A： 外部に Timecode to MTC 変換器を備えたオープン・ループの MMC

この接続方式の変わっている点は，制御される装置の正確なテープ上のタイムコード位置を，シーケンサーが（MTC を通して）つかんでいるにもかかわらず，装置自身はわかっていないことにある。たいていの場合，装置はその内部の <SELECTED TIME CODE> レジスターの内容を更新するのに，タコメーターからのパルスに頼ることになるのである。

MMC の通信は，もう一度言うが“オープン・ループ”モードのみになっている。装置は例1のものと同等であって，コマンドはその MIDI IN ポートで受け取るが MIDI OUT にそのレスポンスを送り出すことはしない。そのデバイス ID は01H にセットされている。シーケンサーからこの装置へのコマンドは頁の左側に，変換器からの MIDI タイムコードは右側に示すこととする。シーケンサーの操作には，マウスとキーボードを用いるものとしよう。

シーケンサーは常にそのセッションの最初には MMC Reset を送る。

```
F0 7F 01 <mcc> <MMC RESET> F7
```

オペレーターはシーケンサー画面上の“Remote Machine PLAY”ボタンをクリックする。このシーケンサーでは，このボタンはレコード状態からのパンチ・アウトにも用いる習慣に従っているので，送信されるメッセージは：

```
F0 7F 01 <mcc> <RECORD EXIT> <DEFERRED PLAY> F7
```

シーケンサーは MIDI タイムコードの解読を開始する。わずかな立ち上がり時間の後に，フレーム 01:02:03:04(30フレーム／秒，ノン・ドロップ）を受信する。

```
F1 04 . . F1 10 . . F1 23 . . F1 30 . .
F1 42 . . F1 50 . . F1 61 . . F1 76 . .
F1 06 . .
```

シーケンサーは，2フレーム時間にまたがる次の MTC 1ワード中のフレーム桁を受けしだい，ただちにこの MTC 現在時刻を逆に，制御される装置のタイムコード・レジスターにセットする。

```
F0 7F 01 <mcc> <WRITE> <count=06>
   <SELECTED TIME CODE> 61 02 03 26 00
F7
```

オペレーターは“Remote Machine RECORD”をクリックする。

シーケンサーは，内部にはこの時の MTC 時刻（01:02:13:20）をセーブの上，次のコマンドを送る。

```
F0 7F 01 <mcc> <RECORD STROBE> F7
```

オペレーターは，レコードからのパンチ・アウトのために“Remote Machine PLAY”をクリックする。

```
F0 7F 01 <mcc> <RECORD EXIT> <DEFERRED PLAY> F7
```

オペレーターは“Remote Machine STOP”をクリックする。

```
F0 7F 01 <mcc> <STOP> F7
```

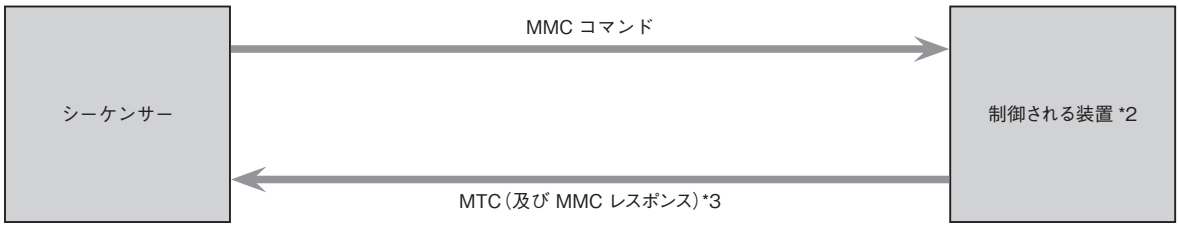
オペレーターはシーケンサーを操作して，このリモート・マシン上に録音された素材のレビューを行う。シーケンサーは装置を，前にストアしておいたパンチ・イン・ポイントの5秒手前 01:02:08:20 にまでロケートさせる。

```
F0 7F 01 <mcc>
  <LOCATE> <count=06> <[TARGET]> 61 02 08 14 00
  <DEFERRED PLAY>
F7
```

＜備考＞

1. MIDI タイムコードと，装置内でタコメーター・ベースで更新して作り出している値との間では，普通はある程度のドリフトがあるであろう。このドリフトの程度は，装置の機械的な条件に依存するばかりでなく，タイムコードが記録される際に，すでにわずかなスピードのズレ（off-speed）があった可能性のあることも効いて来るであろう。
   このドリフトの影響を最小にするためには，シーケンサーはある一定間隔ごとに，受信した最新の MTC の値を，装置の <SELECTED TIME CODE> レジスターに逆にセットしなおすことも考えられる。この際には，正しい現在時刻値だけを用いるよう，充分な注意が必要となる。
2. 以上の注意を払った上でもなお，LOCATE コマンドの実行に伴う，あの程度の位置づけエラーは避け難いであろう。
3. 装置がタコ・パルス・オンリーで動作している場合には，EVENT コマンドを用いてパンチ・イン／パンチ・アウトを行うことは，かなり問題を起こすであろう。

例2B：　内蔵の Timecode to MTC 変換器を備えたクローズド・ループの MMC



こちらの例では，例2Aに比べてかなり改善されている。ここでは装置自身がタイムコードから MTC への変換を行い，したがって正確なタイムコードの位置情報を知ることができる。

コントローラーから装置へのコマンドは頁の左側に，装置からのレスポンスと MIDI タイムコードは右側に示すものとする。制御される装置は，最小セットのガイドライン・レベル＃3と，加えて MIDI タイムコードのコマンドと情報フィールドをサポートしている。そのデバイス ID は <ident> で表す。

シーケンサーは常に，そのセッションの最初には MMC Reset を送る。

```
F0 7F <ident> <mcc> <MMC RESET> F7
```

次に装置の能力をチェックする。

```
F0 7F <ident> <mcc> <READ> <count=01> <SIGNATURE> F7
```

装置からの応答は：

```
F0 7F <ident> <mcr>
     <SIGNATURE> <count=2E> 01 00 00 00
     <count_1=14>
     7F 61 00 00 00 00 00 00 00 00
     7F 70 7F 00 00 00 00 00 00 09
     <count_2=14>
     02 1E 00 00 00 02 1E 00 00 00
     3F 62 07 01 0C 37 00 00 00 09
F7
```

シーケンサーはコマンド・エラー・レベルを，「主要エラー」と「即時オペレーション・エラー」のみにセットし，MIDI タイムコード発生器／変換器のセットアップを行う。

```
F0 7F <ident> <mcc>
   <WRITE> <count=03>
           <COMMAND ERROR LEVEL> <count=01> 3F
   <WRITE> <count=04>
           <MIDI TIME CODE SET UP> <count=02>
                   00 <SELECTED TIME CODE>
F7
```

オペレーターはプレイを要求する。シーケンサーはまた MIDI タイムコードをオンにすることを要求する。

```
F0 7F <ident> <mcc>
   <RECORD EXIT> <PLAY>
   <MIDI TIME CODE COMMAND> <count=01> 02
F7
```

シーケンサーの MIDI IN には MIDI タイムコード 03:02:20:28(ドロップ・フレーム)がやって来る。

```
F1 0C . . F1 11 . . F1 24 . . F1 31 . .
F1 42 . . F1 50 . . F1 63 . . F1 74 . .
```

シーケンサーは，装置が内部に持っているタイムコードの現在値のコピーが，受信した MTC と実際に同一かどうかをチェックする。

```
F0 7F <ident> <mcc>
   <READ> <count=01> <SELECTED TIME CODE>
F7
```

装置が MTC のフレーム 03:02:21:00 - 01 の中間で応答を返す。

```
F1 00 . . F1 10 . . F1 25 . . F1 31 . .
F1 42 . .
     F0 7F <ident> <mcr>
             <SELECTED TIME CODE> 43 02 15 21 00
     F7
      . . F1 50 . . F1 63 . . F1 74 . .
```

オペレーターは，オン・ザ・フライ方式でパンチ・イン・ポイント 03:02:27:08 をマークし，シーケンサーはこれを内部に記憶する。

しばらくの後，オペレーターはパンチ・アウト・ポイント 03:02:41:15 をマークし，シーケンサーはこちらも記憶する。

オペレーターは続いてシーケンサーに，トランスポートを巻き戻し，マークしたパンチ・イン／アウト・ポイント間で，レコード・モードに入れるよう指示を出す。レコーディングはトラックの1と2にのみ行うものとする。

シーケンサーはロケート動作を開始し，MIDI タイムコードはオフにし，ロケートの終わるまでの間，MOTION CONTROL TALLY をモニターする。

```
F0 7F <ident> <mcc>
   <MIDI TIME CODE COMMAND> <count=01> 00
   <LOCATE> <count=06> <[TARGET]> 43 02 16 08 00
   <UPDATE> <count=02> <[BEGIN]> <MOTION CONTROL TALLY>
F7
```

装置はただちに応答し，リバース方向（リワインド）にロケートの始まったことを知らせる。

```
F0 7F <ident> <mcr>
     <MOTION CONTROL TALLY> <count=03>
     <REWIND> <LOCATE> 01
F7
```

ロケートの途中では，装置はいくつかの異なった状態を辿る。

```
F0 7F <ident> <mcr>
     <MOTION CONTROL TALLY> <count=03>
            <FAST FORWARD> <LOCATE> 00
F7
  :
F0 7F <ident> <mcr>
     <MOTION CONTROL TALLY> <count=03>
            <FAST FORWARD> <LOCATE> 01
F7
  :
F0 7F <ident> <mcr>
     <MOTION CONTROL TALLY> <count=03>
            <STOP> <LOCATE> 00
F7
  :
F0 7F <ident> <mcr>
     <MOTION CONTROL TALLY> <count=03>
            <STOP> <LOCATE> 11
F7
```

この「ロケート完了」のタリーを知った上でシーケンサーは，MOTION CONTROL TALLY のモニターを止め，装置が本当に正しい位置にロケートされたかどうかをチェックする。

```
F0 7F <ident> <mcc>
   <UPDATE> <count=02> <[END]> 7F
   <READ> <count=01> <SELECTED TIME CODE>
F7
```

装置からのレスポンスは，3フレーム分のロケート・エラーのあることを示しているが，シーケンサーはこれを満足すべきものとする。

```
F0 7F <ident> <mcr>
     <SELECTED TIME CODE> 43 02 16 25 00
F7
```

シーケンサーは，ここで以後のレコード動作のためのすべてのセットアップを行い，その画面上の表示を更新するように，RECORD STATUS 情報フィールドをモニター対象に選ぶ。

```
F0 7F <ident> <mcc>
   <WRITE> <count=0F>
           <TRACK RECORD READY> <count=01> 60
           <GP1> 43 02 1B 08 00
           <GP2> 43 02 29 0F 00
   <EVENT> <count=06> <[DEFINE]> <event#=01>
           <flags=00> <SELECTED TIME CODE> <GP1>
           <RECORD STROBE>
   <EVENT> <count=06> <[DEFINE]> <event#=02>
           <flags=00> <SELECTED TIME CODE> <GP2>
           <RECORD EXIT>
   <UPDATE> <count=02> <[BEGIN]> <RECORD STATUS>
F7
```

装置はただちに，（“レコード中でない”の）RECORD STATUS を戻す。

```
F0 7F <ident> <mcr> <RECORD STATUS> <count=01> 00 F7
```

最後に PLAY コマンドが出され，MIDI タイムコードはオンにする。

```
F0 7F <ident> <mcc>
   <PLAY>
   <MIDI TIME CODE COMMAND> <count=01> 02
F7
```

パンチ・イン・ポイント（03:02:27:08）に達すると，装置は“レコード中である”のステータスを，MTC のメッセージ中に挿入する。

```
F1 08 . .
     F0 7F <ident> <mcr>
            <RECORD STATUS> <count=01> 01
     F7
       . . F1 10 . . F1 2B . . F1 31 . .
F1 42 . . F1 50 . . F1 63 . . F1 74 . .
```

同様にパンチ・アウト・ポイント（03:02:41:15）では，“レコード中でない”が戻される。 *1

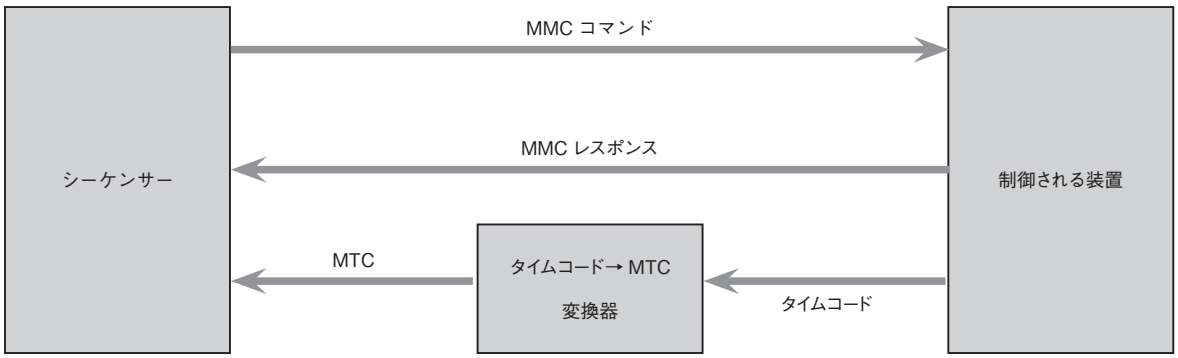
```
F1 0E . . F1 10 . . F1 29 . . F1 31 . .
F1 42 . .
     F0 7F <ident> <mcr>
            <RECORD STATUS> <count=01> 00
     F7
      . . F1 50 . . F1 63 . . F1 74 . .
```

レコーディングを終えるとトランスポートはストップさせ，MIDI タイムコードはオフにし，RECORD STATUS のモニターも止める。すべてのトラックは「レコードせず」の状態に戻す。

```
F0 7F <ident> <mcc>
   <STOP>
   <MIDI TIME CODE COMMAND> <count=01> 00
   <UPDATE> <count=02> <[END]> 7F
   <WRITE> <count=02>
           <TRACK RECORD READY> <count=00>
F7
```

＜備考＞

* 1. こうした MTC のタイミングがクリティカルな場面では，MIDI のレスポンスラインのトラフィック量を少なく抑えることは，本質的にコントローラー（シーケンサ）の責任ではあるが，インテリジェントな制御される装置には，MMC のレスポンスが MTC のメッセージにジッターを生じさせることのないよう，あらゆる努力を払うことが期待されている。
* 2. この構成を取っておれば，各種装置による多くのシステムが操作可能となる。このようなシステムを単独の仮想マシンとして扱えるように変えて（見せて）くれる，インテリジェントなトランスレーターがあれば具合が良いであろう。
* 3. シーケンサーにもうひとつ別の MIDI IN ポートが備わっていれば，外付けの Timecode to MTC 変換器を用いて，この例と同等の操作を行うことができる。ただし，MTC の生成を制御することはできなくなり，また装置にはそれでもやはり内部にタイムコード・リーダを備えている必要はなくならない。



例3

次の例は,これまで2つの例よりもかなり複雑な状況を示す。制御される装置は,2台の同等なシンクロナイザー(または,シンクロナイザー内蔵のテープ・トランスポート)であり, コントローラーは, タイムコードや他のトランスポート関連のステータスを表示する機能と共に, 基本的な自動化された“編集”シーケンスを起動させることができる。

通信はクローズド・ループで行われる。コントローラーの MIDI OUT は, どちらか一方の MIDI THRU を使って両方のデバイスの MIDI IN に供給される。2台の装置からの MIDI OUT はコントローラーの別々の MIDI IN に入力される。

装置のうち一方は同期動作のマスターになっていて, デバイス ID は 01H にセットされている。もう一方はスレーブで, デバイス ID は 02H になっている。こういったマスターを設定することや, “マスター・タイムコード”をスレーブ・シンクロナイザーに加えること等に関する操作は, すべて装置自身の側で行われるものとし, この場面においては MIDI システムは関知していない。

それぞれの装置は, 次のコマンドをサポートしている。

```
<STOP> <PLAY> <DEFERRED PLAY> <FAST FORWARD> <REWIND> <RECORD STROBE>
<RECORD EXIT> <CHASE> <COMMAND ERROR RESET> <MMC RESET> <WRITE> <READ>
<UPDATE> <LOCATE> <VARIABLE PLAY> <MOVE> <ADD> <SUBTRACT>
<DROP FRAME ADJUST> <PROCEDURE> <EVENT> <GROUP> <COMMAND SEGMENT>
<DEFERRED VARIABLE PLAY> <WAIT> <RESUME>
```

それぞれの装置は, 次のレスポンス／情報フィールドをサポートしている。

```
<SELECTED TIME CODE> <SELECTED MASTER CODE> <REQUESTED OFFSET>
<ACTUAL OFFSET> <LOCK DEVIATION> <GPO/LOCATE POINT> <GP1> <GP2> <GP3>
<Short time codes> <SIGNATURE> <UPDATE RATE> <RESPONSE ERROR>
<COMMAND ERROR> <COMMAND ERROR LEVEL> <TIME STANDARD> <MOTION CONTROL TALLY>
<RECORD MODE> <RECORD STATUS> <CONTROL DISABLE> <RESOLVED PLAY MODE>
<CHASE MODE> <PROCEDURE RESPONSE> <EVENT RESPONSE> <RESPONSE SEGMENT>
<FAILURE> <WAIT> <RESUME>
```

それぞれの装置について, 公開される SIGNATURE は次のようになる。

```
01 00 00 00
14
7F 71 00 00 00 00 00 00 00 00
3D 60 7F 00 00 00 00 00 00 09
14
3E 1E 00 00 00 3E 1E 00 00 00
3F 62 00 38 00 33 00 00 00 09
```

コントローラーから装置に送られるコマンドはページの左端に示し,装置からの応答は右にずらせて示す。マスターからのレスポンスか, スレーブからのものかは, それぞれの装置のデバイス ID(システム・エクスクルーシブの3バイト目)によって区別できる。

ここに示したコントロールの方法は, 主に MIDI マシン・コントロールの能力と柔軟性を示すために構成してみたものなので, この作業を行うための唯一の方法を示しているとか, まして最良の方法を表しているというわけでは決してない。

コントローラーは, まずこれまでの設定をすべてクリアーし, マスターとスレーブから成るグループを設定する。

```
F0 7F <all-call=7F> <mcc>
   <MMC RESET>
   <GROUP> <count=04> <ASSIGN> <group=7C> 01 02
F7
```

コントローラーは, 両方の装置に 30 フレームのタイム・スタンダードを設定し, またすべてのコマンド・エラーをイネーブルにする。

```
F0 7F <group=7C> <mcc>
   <WRITE> <count=06>
          <TIME STANDARD> <count=01> 03
```

```
            <COMMAND ERROR LEVEL> <count=01> <all=7F>
F7
```

オペレーターはマスターのプレイを開始し，続いてスレーブもプレイにする(まだチェイス動作ではないことに留意のこと)。

```
F0 7F <master=01> <mcc> <PLAY> F7
F0 7F <slave=02> <mcc> <PLAY> F7
```

コントローラーは両方の装置に対して，タイムコードとモーション・タリーを継続的に知らせることを要求する。

```
F0 7F <group=7C> <mcc>
   <UPDATE> <count=03> <[BEGIN]> <SELECTED TIME CODE> <MOTION CONTROL TALLY>
F7
```

マスターとスレーブは共に，5バイトのタイムコードとタリーを返す。
マスターは 00:22:05:12(00:16:05:0C Hex)の位置をPLAY中。
スレーブは 10:01:58:28(0A:01:3A:1C Hex)の位置をPLAY中。

```
F0 7F <master=01> <mcr>
     <SELECTED TIME CODE> 60 16 05 2C 00
     <MOTION CONTROL TALLY> <count=03> <PLAY> 7F 01
F7

F0 7F <slave=02> <mcr>
     <SELECTED TIME CODE> 6A 01 3A 3C 00
     <MOTION CONTROL TALLY> <count=03> <PLAY> 7F 01
F7
```

.

マスターからの次のタイムコード(00:22:05:13)は，フレームの桁だけが変化しているので“短”形式で送られる。モーション・コントロール・タリーにはまったく変化がない。

```
F0 7F <master=01> <mcr>
     <Short SELECTED TIME CODE> 2D 00
F7
```

次のスレーブ・コード(10:01:58:29)

```
F0 7F <slave=02> <mcr>
     <Short SELECTED TIME CODE> 3D 00
F7
```

.

.

さらにマスター(00:22:05:14)とスレーブ(10:01:59:00)の時刻。スレーブは秒の桁が変化したので，5バイトの全タイムコードが送られることに注意。

```
F0 7F <master=01> <mcr>
     <Short SELECTED TIME CODE> 2E 00
F7
```

```
F0 7F <slave=02> <mcr>
     <SELECTED TIME CODE> 6A 01 3B 20 00
F7
```

コントローラーのバッファがフル状態に近づくと，WAIT 要求を送信する。

```
F0 7F <all-call=7F> <mcc> <WAIT> F7
.
.     （すべての送信が停止する）
.
```

バッファがクリアされ（受信準備が整うと）

```
F0 7F <all-call=7F> <mcc> <RESUME> F7
```

さらにマスター（00:22:05:16）とスレーブ（10:01:59:02）からのタイムコード。

```
F0 7F <master=01> <mcr>
     <Short SELECTED TIME CODE> 30 00
F7
F0 7F <slave=02> <mcr>
     <Short SELECTED TIME CODE> 22 00
F7
```

オペレーターがマスターを止める。

```
F0 7F <master=01> <mcc> <STOP> F7
```

マスターのタリーは「ストップ状態」を表すように変化する。

```
F0 7F <master=01> <mcr>
     <MOTION CONTROL TALLY> <count=03> <STOP> 7F 01
F7
 .
 .
```

さらにスレーブ・タイムコード（10:01:59:03）。

```
F0 7F <slave=02> <mcr>
     <Short SELECTED TIME CODE> 23 00
F7
 .
 .
```

スレーブは（10:01:59:04）に。

```
F0 7F <slave=02> <mcr>
     <Short SELECTED TIME CODE> 24 00
F7
```

オペレーターがスレーブを止める。

```
F0 7F <slave=02> <mcc> <STOP> F7
```

スレーブ・タリーは「停止状態」を表す。

```
F0 7F <slave=02> <mcr>
     <MOTION CONTROL TALLY> <count=03> <STOP> 7F 01
F7
```

オペレーターは，現在のマスターとスレーブとの位置の差を，そのままスレーブの同期オフセットとするように要求する。

```
F0 7F <slave=02> <mcc>
   <MOVE> <count=02> <REQUESTED OFFSET> <ACTUAL OFFSET>
F7
```

コントローラーは，表示のためにオフセット値を読む。

```
F0 7F <slave=02> <mcc>
   <READ> <count=01> <REQUESTED OFFSET>
F7
```

スレーブがオフセット値 09:39:53:18.00(09:27:35:12.00 Hex)を返す。

```
F0 7F <slave=02> <mcr>
     <REQUESTED OFFSET> 69 27 35 12 00
F7
```

オペレーターが“スレーブ・チェイス”ボタンを押す。

```
F0 7F <slave=02> <mcc> <CHASE> F7
```

スレーブのタリーはチェイス・モードとなり，”stopped”及び”parked”のステータスを示すようになる。

```
F0 7F <slave=02> <mcr>
     <MOTION CONTROL TALLY> <count=03> <STOP> <CHASE> 61
F7
```

オペレーターは，マスターをプレイにする(スレーブは追従する)。

```
F0 7F <master=01> <mcc> <PLAY> F7
```

マスター(00:22:05:24)とスレーブ゛(10:01:59:09)は新しいタリーとタイムコードを返す。スレーブ゛は同期動作を始める。

```
F0 7F <master=01> <mcr>
     <Short SELECTED TIME CODE> 38 00
     <MOTION CONTROL TALLY> <count=03> <PLAY> 7F 01
F7

F0 7F <slave=02> <mcr>
     <Short SELECTED TIME CODE> 29 00
       <MOTION CONTROL TALLY> <count=03> <PLAY> <CHASE> 01
F7
```

オペレーターは同期動作中のスレーブの lock deviation(誤差)を監視することにする。

```
F0 7F <slave=02> <mcc>
   <UPDATE> <count=02> <[BEGIN]> <LOCK DEVIATION>
F7
```

スレーブがこれに従う(誤差は +5.02 フレーム)。

```
F0 7F <slave=02> <mcr>
     <LOCK DEVIATION> 60 00 00 05 02
F7
  .
  .
  .
```

次のマスター・タイム(00:22:05:25)。

```
F0 7F <master=01> <mcr>
     <Short SELECTED TIME CODE> 39 00
F7
```

スレーブのタイム(10:01:59:10)と誤差(-1.17 フレーム), まだ同期はしていない。

```
F0 7F <slave=02> <mcr>
     <Short SELECTED TIME CODE> 2A 00
     <Short LOCK DEVIATION> 41 17
F7
 .
 .
```

スレーブのタイム(10:02:01:00)と誤差とタリー(同期が成立した)。

```
F0 7F <slave=02> <mcr>
     <SELECTED TIME CODE> 6A 02 01 20 00
     <Short LOCK DEVIATION> 00 00
     <MOTION CONTROL TALLY> <count=03> <PLAY> <CHASE> 11
F7
```

マスターがプレイ中であり, スレーブは同期しているので, オペレーターはここでレコードの“イン・ポイント”と“アウト・ポイント”をマークすることにより, ちょっとした自動“編集”操作をやってみたいと考える。実際の記録動作はスレーブ・マシン上で行われる。

これ以降は, 主要な UPDATE レスポンスだけを示すことにする。

オペレーターはイン・ポイントをマークする。コントローラーは現在時刻を確定するために, グループ・デバイス ID を使ってマスターとスレーブ両装置の GP1 汎用レジスタに現在時刻を読み込ませる。

```
F0 7F <group=7C> <mcc>
   <MOVE> <count=02> <GP1> <SELECTED TIME CODE>
F7
.
.
```

ある時間の後, オペレーターはアウト・ポイントをマークする。すなわち, コントローラーは GP2 レジスタに現在時刻を読み込ませる。

```
F0 7F <group=7C> <mcc>
   <MOVE> <count=02> <GP2> <SELECTED TIME CODE>
F7
.
.
```

オペレーターはここで,「自動編集」の実行を指示する。

コントローラーは, GP1 の時刻でパンチ・イン, GP2 の時刻でパンチ・アウトするように, スレーブ内のイベントを設定する。どちらのイベントもトリガーをかけ終えた後は削除するものとし, さらにまた, トリガーがかかるのは, 共にフォワード方向で, かつプレイのスピードでの動作中のタイムコードの場合に限るようにする。

コントローラーは，さらにスレーブのレコード・モードをセットし，レコード・ステータスの自動アップデートを要求する。

```
F0 7F <slave=02> <mcc>
   <EVENT> <count=06> <[DEFINE]> <event=#01>
           <flags=00> <SELECTED TIME CODE> <GP1>
           <RECORD STROBE>
   <EVENT> <count=06> <[DEFINE]> <event=#02>
           <flags=00> <SELECTED TIME CODE> <GP2>
           <RECORD EXIT>
   <WRITE> <count=03> <RECORD MODE> <count=01> 01
   <UPDATE> <count=02> <[BEGIN]> <RECORD STATUS>
F7
```

スレーブは，ただちに現在の（非記録状態であるとの）RECORD STATUS を返す。

```
F0 7F <slave=02> <mcr>
     <RECORD STATUS> <count=01> 00
F7
```

次にコントローラーは，マスターを次のように設定する。

(a) GP1 が示すパンチ・イン・ポイントの5秒手前（プリロール・ポイント）にロケートする（GP0 を計算に使用する）。

(b) マスターが GP2 にあるパンチ・アウト・ポイントを2秒過ぎた点に達した時に，自動ストップさせるイベントをトリガーする（トリガー点は GP3 に設定される）。

```
F0 7F <master=01> <mcc>
   <WRITE> <count=06> <GPO/LOCATE POINT> 60 00 05 00 00
   <SUBTRACT> <count=03> <GPO/LOCATE POINT> <GP1> <GPO/LOCATE POINT>
   <LOCATE> <count=02> <[I/F]> <GPO/LOCATE POINT>
   <WRITE> <count=06> <GP3> 60 00 02 00 00
     <ADD> <count=03> <GP3> <GP3> <GP2>
   <EVENT> <count=06> <[DEFINE]> <event=#01>
           <flags=10> <SELECTED TIME CODE> <GP3>
           <STOP>
F7
```

タイムコードとスレーブの lock deviation に加えて，以下のようなマスターとスレーブの motion tally が返される。これらは，マスターがロケートを行っており，これにスレーブがチェイスで追従するようすを表している。

```
F0 7F <master=01> <mcr>
     <MOTION CONTROL TALLY> <count=03> <REWIND> <LOCATE> 01
F7
F0 7F <slave=02> <mcr>
     <MOTION CONTROL TALLY> <count=03> <REWIND> <CHASE> 41
F7
 .
```

やがてマスターはロケートを完了するが，スレーブはまだ動作中である。

```
F0 7F <master=01> <mcr>
     <MOTION CONTROL TALLY> <count=03> <STOP> <LOCATE> 11
F7

F0 7F <slave=02> <mcr>
     <MOTION CONTROL TALLY> <count=03> <REWIND> <CHASE> 41
F7
 .
 .
```

ついにスレーブがマスター位置に到達し，パーク状態となる。

```
F0 7F <slave=02> <mcr>
     <MOTION CONTROL TALLY> <count=03> <STOP> <CHASE> 61
F7
```

両装置がロケートとパークに成功したことを知って，コントローラーはマスターにプレイを指示し，これによって，あらかじめロードしておいた一連のイベントが自動的に実行される。

```
F0 7F <master=01> <mcc> <PLAY> F7
```

マスターはプレイになり，スレーブは同期しようとする。

```
F0 7F <master=01> <mcr>
     <MOTION CONTROL TALLY> <count=03> <PLAY> 7F 01
F7

F0 7F <slave=02> <mcr>
     <MOTION CONTROL TALLY> <count=03> <PLAY> <CHASE> 01
F7
.
.
```

やがてスレーブは同期する。

```
F0 7F <slave=02> <mcr>
     <MOTION CONTROL TALLY> <count=03> <PLAY> <CHASE> 11
F7
 .
 .
```

スレーブは予定されていた点で記録状態（パンチ・イン）に入る（そしてイベント #01 を削除する）。

```
F0 7F <slave=02> <mcr>
     <RECORD STATUS> <count=01> 01
F7
 .
 .
 .
```

ある時間の後，スレーブはパンチ・アウトする（そしてイベント #02 を削除する）。

```
F0 7F <slave=02> <mcr>
     <RECORD STATUS> <count=01> 00
F7
.
.
```

パンチ・アウトの2秒後，マスターは停止し，その結果スレーブも止まってパークする。

```
F0 7F <master=01> <mcr>
     <MOTION CONTROL TALLY> <count=03> <STOP> 7F 01
F7

F0 7F <slave=02> <mcr>
     <MOTION CONTROL TALLY> <count=03> <STOP> <CHASE> 61
F7
```

セッションの終わりにあたり，コントローラーはすべての UPDATE レスポンスを終了させ，両方の装置をディスエーブルにする。

```
F0 7F <group=7C> <mcc>
   <UPDATE> <count=02> <[END]> <all=7F>
   <WRITE> <count=03> <CONTROL DISABLE> <count=01> 01
F7
```

最後に，オペレーターはマスター装置の（実際には用意されていない）タイムコード・ジェネレーターを操作してみようとする。

```
F0 7F <master=01> <mcc>
   <READ> <count=01> <GENERATOR TIME CODE>
   <GENERATOR SET UP> <count=03> 00 <SELECTED TIME CODE> 01
   <GENERATOR COMMAND> <count=01> 01
F7
```

マスターは最初に，サポートしていない GENERATOR TIME CODE への READ に対してレスポンスを返す。

```
F0 7F <master=01> <mcr>
     <RESPONSE ERROR> <count=01> <GENERATOR TIME CODE>
F7
```

次に，GENERATOR SET UP コマンドが同じくサポートされていないことを知らせる。コントローラーがすべてのエラーをイネーブルにしているので，マスターは“エラー停止”モードに入り，すべてのコマンド処理を止める。最後のコマンド，GENERATOR COMMAND は無視されることになる。

```
F0 7F <master=01> <mcr>
     <COMMAND ERROR> <count=0A>
     <flags=11> <level=7F> <error=40>
     <count_1=06> <offset=00>
     <GENERATOR SET UP> <count=03> 00 <SELECTED TIME CODE> 01
F7
```

コントローラーは，COMMAND ERROR メッセージを認識し，マスターがコマンドの処理を再開できるようにする。

```
F0 7F <master=01> <mcc> <COMMAND ERROR RESET> F7
```

付録B　タイムコード・ステータスの運用細則

タイムコードのステータス・ビットについては,「3 標準仕様」で定義されている。ここでは, それぞれの MMC タイムコード情報フィールドでの正しい適用基準について述べる。

01　SELECTED TIME CODE　[read/write]

タイムコード中のテータス・ビット	パワー・オンや MMC RESET の直後の値	通常動作中の値	WRITE データ中に含まれるタイムコード・ステータス・ビットの解釈
tt （フレーム・タイプ）	TIME STANDARD あるいは他の内部デフォルトによる。	n=1 の場合（タイムコードを未だ読んでいない）: tt=TIME STANDARD（またはデフォルト値） または WRITE や［タイムコード計算］関係のコマンドでロードされた値 n=0 の場合： tt= 読んだタイムコードからの値	n=1 の（タイムコードを未だ読んでいない）場合にのみ, tt に WRITE し, そうでない場合は WRITE データ中の tt は無視すること。
c （カラー・フレーム・フラグ）	0	n=1 の場合：c=0 n=0 の場合：c= 読んだタイムコードからの値	WRITE データ中の c は無視すること。
k （ブランク）	1	k=0 とするのはタイムコードが以下の場合に限る： (a)（テープから）読み込まれた後 (b) タコ・パルスにより進められた後 (c) WRITE で書き込まれた後 (d)［タイムコード計算］関係のコマンドでロードされた後 そうでない場合は k=1	WRITE の後では常に k=0 にセットし, WRITE データ中の k の値は無視のこと。
g（符号）	0	0	WRITE データ中の g の値は無視
i（最終バイト標識）	1	1	WRITE データ中の i の値は無視
e （推測コード）	0	タイムコードの最後の変更が, タコやコントロール・トラック・パルスによる更新の結果であった場合に限り e=1。	WRITE の後では常に e=0 にセットし, WRITE データ中の e の値は無視のこと。
v （不正コード）	0	定義に従ってセット	WRITE の後では常に v=0 にセットし, WRITE データ中の v の値は無視のこと。
d （ビデオ・フィールド1標識）	0 （インプリメントされていれば 0/1）	インプリメントされていれば, 定義に従ってセット／リセット	WRITE データ中の d の値は無視
n （無効タイムコード・フラグ）	1	タイムコードが一度でもテープから読み取られた後に限り n=0	WRITE の後では常に n=1 にセット（すなわち「タイムコード未読」に「リセット」）し, WRITE データ中の v の値は無視のこと。

02 SELECTED MASTER CODE [read only]

WRITE 操作のあり得ないことを除いて, SELECTED TIME CODE に同じ。

03 REQUESTED OFFSET [read/write]

タイムコード中のステータス・ビット	パワー・オンや MMC RESET の直後の値	通常動作中の値	WRITE データ中に含まれるタイムコード・ステータス・ビットの解釈
tt (フレーム・タイプ)	右欄を参照	SELECTED TIME CODE がドロップ・フレーム(tt=10)の場合にノンドロップ・フレーム(tt=11)に留まることを除いては, SELECTED TIME CODE 中の tt に従う。	WRITE データ中の tt の値は無視のこと。 (MMCの将来のバージョンでは, ここに多少の変更の可能性がある。)
c(カラー・フレーム・フラグ)	0	0	WRITE データ中の c の値は無視
k (ブランク)	1	k=0 とするのはタイムコードが以下の場合に限る: (a) WRITE で書き込まれた後 (b) [タイムコード計算] 関係のコマンドでロードされた後 そうでない場合は k=1	WRITE の後では常に k=0 にセットし, WRITE データ中の k はの値無視のこと。
g(符号)	0	定義に従って 0/1	g ビットは WRITE 可
i (最終バイト標識)	0	0	WRITE データ中の i が 0 ならば, 最終データ・バイトはサブフレームとしてロードし, 1 ならば, 最終データ・バイトは無視してサブフレーム = 00 にセット。

04 ACTUAL OFFSET [read only]

05 LOCK DEVIATION [read only]

タイムコード中のステータス・ビット	パワー・オンや MMC RESET の直後の値	通常動作中の値	WRITE データ中に含まれるタイムコード・ステータス・ビットの解釈
tt （フレーム・タイプ）	右欄を参照	SELECTED TIME CODE がドロップ・フレーム（tt=10）の場合にノンドロップ・フレーム（tt=11）に留まることを除いては, SELECTED TIME CODE 中の tt に従う。	
c（カラー・フレーム・フラグ）	0	0	
k（ブランク）	0	0	
g（符号）	0	定義に従って 0/1	
i（最終バイト標識）	0	0	

06 GENERATOR TIME CODE [read/write]

タイムコード中のステータス・ビット	パワー・オンや MMC RESET の直後の値	通常動作中の値	WRITE データ中に含まれるタイムコード・ステータス・ビットの解釈
tt （フレーム・タイプ）	TIME STANDARD あるいは他の内部デフォルト設定による。	tt=TIME STANDARD（またはデフォルト設定値）または，WRITE やタイムコード計算関係のコマンドや，他のタイムコード・ソースからのコピー／ジャムによりロードされた値。	tt ビットは WRITE 可 この結果，タイムコード・ジェネレーターのカウント・モードが定まる。
c（カラー・フレーム・フラグ）	0	0	WRITE データ中の c の値は無視
k（ブランク）	0	0	WRITE データ中の k の値は無視
g（符号）	0	0	WRITE データ中の g の値は無視
i（最終バイト標識）	1	1	WRITE データ中の i の値は無視
e（推測コード）	0	0	WRITE データ中の e の値は無視
v（不正コード）	0	0	WRITE データ中の v の値は無視
d （ビデオ・フィールド1標識）	0 （インプリメントされていれば 0/1）	インプリメントされていれば，定義に従ってセット／リセット。	WRITE データ中の d の値は無視
n（無効タイムコード・フラグ）	0	0	WRITE データ中の n の値は無視

07 MIDI TIME CODE INPUT [read only]

タイムコード中のステータス・ビット	パワー・オンや MMC RESET の直後の値	通常動作中の値	WRITE データ中に含まれるタイムコード・ステータス・ビットの解釈
tt （フレーム・タイプ）	TIME STANDARD あるいは他の内部デフォルト設定による。	n=1 の場合（タイムコードを未だ読んでいない） tt=TIME STANDARD（またはデフォルト設定値） n=0 の場合： tt= MTCから読んだ値	
c（カラー・フレーム・フラグ）	0	0	
k （ブランク）	1	装置の MIDI IN 部から MTC を受信した後に限り k=0 そうでなければ k=1	
g（符号）	0	0	
i(最終バイト標識）	1	1	
e（推測コード）	0	0	
v（不正コード）	0	定義に従ってセットされる	
d（ビデオ・フィールド1標識）	0	0	
n （無効タイムコード・フラグ）	1	装置の MIDI IN 部から MTC を受信した後に限り n=0, そうでなければ n=1（すなわち, n=k）	

08　　GP0　　[read/write]

:　　:

0F　　GP7　　[read/write]

タイムコード中のステータス・ビット	パワー・オンや MMC RESET の直後の値	通常動作中の値	WRITE データ中に含まれるタイムコード・ステータス・ビットの解釈
tt （フレーム・タイプ）	TIME STAN-DARD あるいは他の内部デフォルト設定による。	tt=WRITE や [タイムコード計算] 関係のコマンドによりロードされた値。	tt ビットは WRITE 可
c （カラー・フレーム・フラグ）	0	c=WRITE や [タイムコード計算] 関係のコマンドによりロードされた値	c ビットは WRITE 可
k （ブランク）	1	k=0 とするのはタイムコードが以下の場合に限る： (a) WRITE で書き込まれた後 (b) [タイムコード計算] 関係のコマンドでロードされた後 そうでない場合は k=1	WRITE の後では常に k=0 にセットし，WRITE データ中の k の値は無視のこと。
g （符号）	0	g=WRITE や [タイムコード計算] 関係のコマンドによりロードされた値	g ビットは WRITE 可
i （最終バイト標識）	0	0	WRITE データ中の i が 0 ならば，最終データバイトはサブフレームとしてロードし，1 ならば，最終データ・バイトは無視してサブフレーム =00 にセット。

付録C　SIGNATURE 表

コマンドのビット・マップ配列

Byte	Bit7	Bit6 (40H)	Bit5 (20H)	Bit4 (10H)	Bit3 (08H)	Bit2 (04H)	Bit1 (02H)	Bit0 (01H)
c0	- 0	(06) RECORD STROBE	(05) REWIND	(04) FAST FORWARD	(03) DEFERRED PLAY	(02) PLAY	(01) STOP	(00) reserved
c1	- 0	(0D) MMC RESET	(0C) COMMAND ERROR RESET	(0B) CHASE	(0A) EJECT	(09) PAUSE PAUSE	(08) RECORD EXIT	(07) RECORD
c2	- 0	(14)	(13)	(12)	(11)	(10)	(0F)	(0E)
c3	- 0	(1B)	(1A)	(19)	(18)	(17)	(16)	(15)
c4	- 0	- 0	- 0	- 0	(1F)	(1E)	(1D)	(1C)
c5	- 0	(26)	(25)	(24)	(23)	(22)	(21)	(20)
c6	- 0	(2D)	(2C)	(2B)	(2A)	(29)	(28)	(27)
c7	- 0	(34)	(33)	(32)	(31)	(30)	(2F)	(2E)
c8	- 0	(3B)	(3A)	(39)	(38)	(37)	(36)	(35)
c9	- 0	- 0	- 0	- 0	(3F)	(3E)	(3D)	(3C)
c10	- 0	(46) SEARCH	(45) VARIABLE PLAY	(44) LOCATE	(43) UPDATE	(42) READ	(41) MASKED WRITE	(40) WRITE
c11	- 0	(4D) ADD	(4C) MOVE	(4B) MTC COMMAND	(4A) GENERATOR COMMAND	(49) ASSIGN SYS. MAS	(48) STEP	(47) SHUTTLE
c12	- 0	(54) DEFERRED VARI. PLAY	(53) COMMAND SEGMENT	(52) GROUP	(51) EVENT	(50) PROCEDURE	(4F) DROP FR. ADJUST	(4E) SUBTRACT
c13	- 0	(5B)	(5A)	(59)	(58)	(57)	(56)	(55) REC STROBE VARIABLE
c14	- 0	- 0	- 0	- 0	(5F)	(5E)	(5D)	(5C)
c15	- 0	(66)	(65)	(64)	(63)	(62)	(61)	(60)
c16	- 0	(6D)	(6C)	(6B)	(6A)	(69)	(68)	(67)
c17	- 0	(74)	(73)	(72)	(71)	(70)	(6F)	(6E)
c18	- 0	(7B)	(7A)	(79)	(78)	(77)	(76)	(75)
c19	- 0	- 0	- 0	- 0	(7F) RESUME	(7E)	(7D)	(7C) WAIT

レスポンス／情報フィールドのビット・マップ配列

Byte	Bit7	Bit6 (40H)	Bit5 (20H)	Bit4 (10H)	Bit3 (08H)	Bit2 (04H)	Bit1 (02H)	Bit0 (01H)
r0	- 0	(06) GENERATOR TIME CODE	(05) LOCK DEVIATION	(04) ACTUA OFFSET	(03) REQUESTED OFFSET	(02) SELECTE MASTER CODE	(01) SELECTED TIME CODE	(00) reserved
r1	- 0	(0D) GP5	(0C) GP4	(0B) GP3	(0A) GP2	(09) GP1	(08) GP0 /LOCATE POINT	(07) MTC INPUT
r2	- 0	(14)	(13)	(12)	(11)	(10)	(0F) GP7	(0E) GP6
r3	- 0	(1B)	(1A)	(19)	(18)	(17)	(16)	(15)
r4	-	- 0	- 0	- 0	(1F) 0	(1E)	(1D)	(1C)
r5	- 0	(26) Short GENTOR TIME CODE	(25) Sh't LOCK DEVIATION	(24) Sh't ACTUAL OFFSET	(23) Sh't REQ'D OFFSET	(22) Short SEL'D MASTER CODE	(21) Short SEL'D TIME CODE	(20) reserved
r6	- 0	(2D) Short GP5	(2C) Short GP4	(2B) Short GP3	(2A) Short GP2	(29) Short GP1	(28) Short GP0 LOCATE POINT	(27) Short MTC INPUT
r7	- 0	(34)	(33)	(32)	(31)	(30)	(2F) Short GP7	(2E) Short GP6
r8	- 0	(3B)	(3A)	(39)	(38)	(37)	(36)	(35)
r9	- 0	- 0	- 0	- 0	(3F)	(3E)	(3D)	(3C)
r10	- 0	(46) SEL'D TC SOURCE	(45) TIME STANDARD	(44) CMD ERROR LEVEL	(43) COMMAND ERROR	(42) RESPONSE ERROR	(41) UPDATE RATE	(40) SIGNATURE
r11	- 0	(4D) RECORD STATUS	(4C) RECORD MODE	(4B) FAST MODE	(4A) STOP MODE	(49) VELOCITY TALLY	(48) MOTION CTL TALLY	(47) SEL'D TC USERBITS
r12	- 0	(54) STEP LENGTH	(53) TRACK INP MONITOR	(52) TRACK SYNC MONITOR	(51) RECORD MONITOR	(50) GLOBAL MONITOR	(4F) TRACK REC READY	(4E) TRACK REC STATUS

Byte	Bit7	Bit6 (40H)	Bit5 (20H)	Bit4 (10H)	Bit3 (08H)	Bit2 (04H)	Bit1 (02H)	Bit0 (01H)
r13	- 0	(5B) GENERATOR CMD TALLY	(5A) CHASE MODE	(59) RESOLVED PLAY MODE	(58) CONTROL DISABL	(57) LIFTER DEFEAT	(56) FIXED SPEED	(55) PLAY SPEED REFERENCE
r14	- 0	- 0	- 0	- 0	(5F) MTC SETUP	(5E) MTC CMD TALLY	(5D) GENERATOR USERBITS	(5C) GENERATOR SETUP
r15	- 0	(66)	(65) FAILURE	(64) RESPONSE SEGMENT	(63) VITC INSERT ENABLE	(62) TRACK MUTE	(61) EVENT RESPONSE	(60) PROCEDURE RESPONSE
r16	- 0	(6D)	(6C)	(6B)	(6A)	(69)	(68)	(67)
r17	- 0	(74)	(73)	(72)	(71)	(70)	(6F)	(6E)
r18	- 0	(7B)	(7A)	(79)	(78)	(77)	(76)	(75)
r19	- 0	- 0	- 0	- 0	(7F) RESUME	(7E)	(7D)	(7C) WAIT

付録D　MIDI マシン・コントロールと MTC キューイング

装置によっては MIDI マシン・コントロールと MTC キューイングの両方をインプリメントすることが考えられる。そうした場合に，MMC のイベントと MTC のキューイング・イベント・リストとを一緒にできると好都合であろう。

他の規格を援用せずとも，それだけでも必要な機能は備えたものにすることと，ES バス方式を取っている装置にも適用できるようにすることに努めた結果，MIDI マシン･コントロールにおいて MTC のキューイングは，イベントのトリガー手段としては採用するに至らなかった。

本節においては，MIDI マシン・コントロールと MTC キューイングとが違っている部分と共に，重なっている部分を明らかにすることに努めた。

MIDI マシン・コントロールと MTC キューイングでのイベント規則の比較

MIDI マシン・コントロール	MTC キューイング
MMC のイベントは他の MMC コマンドだけをトリガーする。	MTC のイベントは，イベント・シーケンス，キュー，トラックのパンチ・イン及びパンチ・アウトをトリガーする。
イベントは EVENT コマンドで定義する。	イベントは MTC Set-Up メッセージで定義する。
各イベントはそれぞれ唯一のものであり，ひとつの7ビット長の名前で区別される。	同一のイベント番号を異なる時刻にトリガーすることができ，したがって，イベント・リストの各項目はトリガー時刻とイベント番号の組み合わせで区別される。
各々のイベント定義では，そのイベントがトリガーされる際に基準とするタイムコードのソースを指定する。	タイムコードのソースは指定しないが，MTC が前提となっている。
各イベントには付加的なフラグが含まれており，プレイ以外のスピードでもトリガーしたり，前方，後方，あるいは両方向への動作時にトリガーすることが指定できる。	動作上のバリエーションや制限事項は指定できない。
ES バスとの互換性から，各イベントはトリガー後に削除されるようにもできる。	トリガー発生後にイベントが削除されることはない。
イベントの総体的なイネーブル／ディスエーブル機能は持っていない。	イベント・リスト自体には影響を与えることなしに，イベントのイネーブル／ディスエーブル・コマンドによって，すべてのイベントをオン／オフすることができる。
「不正なイベント名」「未定義イベント」「イベント・バッファ・オーバーフロー」などのエラーはトラップされ，またイベント・トリガー時刻に実行されるコマンドについては完全な事前のチェックが行われる。	エラー・トラップの機能はない。

MTC キューイング・メッセージの復習，及び MMC との関係

01　Punch In points
02　Punch Out points
03　Delete Punch In point
04　Delete Punch Out point

これらの MTC メッセージは，完全な置き換えにはなっていないが機能的には MIDI マシン・コントロールのコマンドにも重複するものがある。

次の例は，MTC の Punch In メッセージを MMC のコマンドに翻訳してみたものである。

```
MTC Cueing: F0 7E <chan> 04 01 hr mn sc fr ff sl sm F7    (sl sm はトラック番号)

MMC:        F0 7F <device_ID> <mcc>
                        <PROCEDURE> <count=09> <[ASSEMBLE]> <procedure_name>
                                <MASKED WRITE> <count=04>
                                <TRACK RECORD READY> <byte #> <mask> <data=7F>
                        <RECORD STROBE>
                        <WRITE> <count=06> <GPO> hr mn sc fr ff
                        <EVENT> <count=09> <[DEFINE]> <event_name>
                                <flags=40> <MIDI TIME CODE INPUT> <GPO>
                                <PROCEDURE> <count=02> <[EXECUTE]> <procedure_name>
                F7
```

05　Event Start points
06　Event Stop points
07　Event Start points with additional info.
08　Event Stop points with additional info.
09　Delete Event Start point
0A　Delete Event Stop point
0E　Event Name in additional info.

MTC のイベント・スタート／ストップ・メッセージは，「イベント群の大きなシーケンスや，連続する一連のイベントの開始または停止を意味する」（MIDI 1.0 規格）。

MIDI マシン・コントロールでは，このスタイルのイベントに真に対応するものはない。

0B　Cue points
0C　Cue points with additional info.
0D　Delete Cue point

MTC のキュー・ポイントは，「個々のイベントの発生」を言う。例えば，サウンド・エフェクトの“hit”ポイントやエディットの基準点をマークする，等（MIDI 1.0 規格）。

MMC の EVENT とスタイルでは近いが，これら MTC のキュー・ポイントは，特定マシンのコマンドといった低レベルの実行を目的としたものではない。“追加情報”エリアが用意されているが，こうしたコマンドを定義するのにはどう見ても厄介な方法になっている。

そこで我々としては，MTC のキュー・ポイントは，MMC にちょうど対応するものが無な操作内容を表すものと見ることにしよう。

00: 00 00　Time Code Offset

MTC キューイングのタイムコード・オフセット・メッセージは，文字通りこれに対応する MMC コマンドに置き換えられる。

```
MTC Cueing: F0 7E <chan> 04 00 hr mn sc fr ff 00 00 F7

MMC:        F0 7F <device_ID> <mcc>
                  <WRITE> <count=06> <REQUESTED OFFSET> hr mn sc fr ff F7
```

00: 01 00　Enable Event List

00: 02 00　Disable Event List

MIDI マシン・コントロールでは現在，そのイベントに対してはこうしたコマンドをサポートしていない。これらのメッセージを受信した場合は，MTC メッセージで定義されたイベント群に対してのみ適用するものとし，MMC でのイベントに対してはいささかも影響のないようにすべきである。

00: 03 00　Clear Event List

MIDI マシン・コントロールでは，イベントの削除に関しては独自の方法を有している。このメッセージを受信した場合も上と同じく，MTC メッセージで定義されたイベント群に対してのみ適用するものとし，MMC でのイベントに対してはいささかも影響のないようにすべきである。

00: 04 00　System Stop

MTC キューイングのシステム・ストップ・メッセージは，次のような MMC のイベント・コマンドに置き換えられる。

```
MTC Cueing: F0 7E <chan> 04 00 hr mn sc fr ff 04 00 F7

MMC:        F0 7F <device_ID> <mcc>
                        <WRITE> <count=06> <GPO> hr mn sc fr ff
                        <EVENT> <count=06> <[DEFINE]> <event_name>
                        <flags=50> <SELECTED TIME CODE> <GPO>
                        <STOP>
                  F7
```

00: 05 00　Event List Request

MIDI マシン・コントロールのイベントは，EVENT RESPONSE 情報フィールドを使って読み返すことができる。Event List Request は MTC メッセージで定義されたイベントだけの送信を行うべきである。MMC のイベントを含むべきではない。

付録E　受信バッファ・サイズの決定

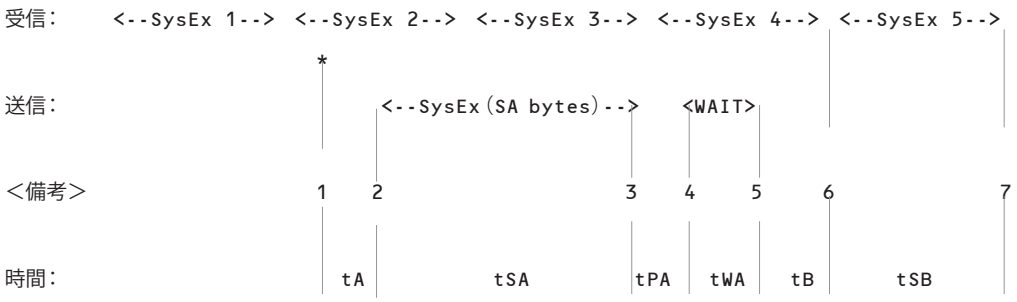
WAIT ハンドシェイクが期待通りに動作するためには，MMC 装置内の MIDI 受信バッファにはそれなりの最小限のサイズが必要となる。MIDI のシステム・エクスクルーシブが一度その送信を開始すると，途中でこれに割り込むわけには行かないことになっていることにより，WAIT メッセージを急いで送ろうにも待たざるを得ない場合があるばかりでなく，WAIT の要請に応えて送信を止めようとする際にも，ある遅れが生じ得る。MIDI データを帯域いっぱいのスピードで受信中との条件の下でも，WAIT を送って送信を止めて貰おうと判断した瞬間から，実際に受信データの流れが止まるまでの間にも，バッファ・オーバーフローが起こらないようにするためには，受信装置は十分なバッファ・スペースを備えていなければならない。

事態をより複雑にするのが外付けの MIDI マージ装置を使う場合であって，WAIT 遅れをかなり伸ばしてしまうか，あるいは少なくとも，遅れの程度を予測し難いものにしてしまう。

ここではまず外部マージ装置を使わない場合について,必要となる受信バッファサイズを吟味してみよう。すべての計算は最悪のケース」のシナリオとすることにする。

簡単な“クローズド・ループ”での WAIT 動作

次の図は MMC 装置Aでの受信と送信の様子を示したものである。ここでは他のもうひとつの装置Bが MIDI の帯域いっぱいで送信して来ており，対してこの装置Aはのんびりと送信すれば良いという状況を仮定しよう。我々の目的とするのは，基本的なポイント・トゥ・ポイントのクローズド・ループで接続されている条件で，装置Aにとっての最悪ケースに備えた最小限の受信バッファ・サイズを決定することにある。



1. 受信データが，装置A内の受信バッファ内で，あらかじめ決まっているスレッシュホールド HA バイト点を超えて溜まり始める。

2. ある遅れ tA の後に装置Aは,受信データがこのスレッシュホールドを超えたことを検知する。最悪のケースを考えるこの例では，装置Aの送信部では長さ SA バイトのシステム・エクスクルーシブの送信を将に開始したところである。MIDI のシステム・エクスクルーシブは，一度送信が始まると途中に割り込むことができないので，WAIT メッセージを送ることができるようになるには，この送信の完了までの時間 tSA を待つ必要がある。

3. システム・エクスクルーシブの送信を終えて，装置Aが WAIT メッセージを用意する間，短い遅れ tPA があるであろう。

4. 装置Aが WAIT メッセージを送る。これ自体は6バイト長であり，この送信には時間 tWA が取られることになる。

5. WAIT メッセージの送信が完了する。

6. 次には装置Bにおいて，Aからの WAIT メッセージが来たことを検知するまでに，遅れ tB が生じるであろう。前と同様に最悪のケースという条件の下では，装置Bはまったく新たなシステム・エクスクルーシブの送信を将に開始したところであり，EOX の後までは送信を止めることができない。

7. かくしてさらなるシステム・エクスクルーシブ長の遅れ tSB の後，装置Bはようやく送信を止める。

Aの受信バッファがスレッシュホールドを超えてから，Bのシステム・エクスクルーシブ送信が終わるまでの最大時間 tmax は，次式で与えられる。

tmax ＝ tAmax ＋ tSAmax ＋ tPAmax ＋ tWAmax ＋ tBmax ＋ tSB

tAmax = 受信データがバッファ・スレッシュホールドを超えたことを，装置Aが認識するまでの最悪のケースでの時間

tSAmax = 装置Aが MMC の最長システム・エクスクルーシブ（53 バイト）を送信するのに要する最大時間。装置Aは必ずしも MIDI の帯域いっぱいのスピードで送っているとは限らないので,この時間は(1MIDI バイト時間を 0.320ms として）53MIDI バイト時間か，それよりも長いであろう。

tPAmax = 装置Aがそのシステム・エクスクルーシブ送信を完了後，WAIT メッセージを用意するのに要する最大時間（ほとんどゼロであろうが）。

tWAmax =装置Aが WAIT を送信するのに要する最大時間（これもまた，6 × 0.320 ms か，それよりは長い）。

tBmax = 10ms。我々はここで,「MMC 装置は WAIT メッセージの着信を 10msec 以内に検知しなければならない」と言う，かなり緩やかな要請事項を決めることにする。

tSB = 装置Bが各 MIDI バイトに 0.320ms を使う帯域いっぱいのスピードで送信しており，MMC の最長システム・エクスクルーシブ（48 データ）が 53 バイトなので，16.96ms。

装置Aは,帯域いっぱいの MIDI データを tmax の間受信しているので,これから装置Aの受信バッファには,「予め決められたスレッシュホールド」の下にある HA バイトに加えて，少なくとも tmax /0.320 バイトの使えるスペースが必要であるということになる。HA の実際の値は，ここで決める必要はないが，ほとんどの場合は単独の MMC システム・エクスクルーシブであることから，それを受信するだけで WAIT メッセージを発生させるようなことがないだけの，十分な大きさを持っているべきであると言える。

最後の要素として XA が出て来る。これは時間 tmax の間に装置A内の処理ルーチンが，受信バッファから次の処理に向けて引き出すことのできるバイト数を表す。

以上の結果，装置の受信バッファの最小バイト数 z は次式で与えられる。

$$z = HA + \frac{tAmax + tSAmax + tPAmax + tWAmax + 26.96}{0.320} - XA$$

例えば，もし仮に，

HA ＝ 53

tAmax ＝ 10 ms

tSAmax ＝ 53 ×（.320 +.100）＝ 22.26（送信 MIDI バイト間の最大遅れは 100 μ s とする。）

tPAmax ＝ 0

tWAmax　＝ 6 ×(.320 +.100)＝ 2.52

XA ＝ 0(この値を信頼のできる予測をすることが困難なため, とりあえず無視する。)

とすれば,

$$z = 53 + \frac{10 + 22.26 + 0 + 2.52 + 26.96}{0.320} - 0 = 245.9$$

言い換えると, 上記の条件の下では, 基本的なポイント・トゥ・ポイントの MMC の通信は, 256 バイトの受信バッファがあれば極めて具合良く動作するであろうと言える。

外部MIDIマージャーの場合

MMC システムにおいてマージャーの典型的な使用法は, 複数の制御される装置からのレスポンスを集めて, これをコントローラーの MIDI IN に返してやることであろう。こうした接続はコントローラーにも制御させる装置にも大きな影響を与える。

コントローラーにおいては:

1. このタイプのマージ接続が問題なく動作するには, 明らかに, つながれている制御される装置群に対してコントローラーは, そのひとつの MIDI IN ポートで受けられる最大帯域を越えるようなデータのリクエストを出してはならない。
2. 接続されている制御される装置群に対してコントローラーが WAIT コマンドを発した時点で, マージャーはすでに, やがてはコントローラーに転送すべき相当量のバイト数を抱え込んでいるかもしれない。これに加えて, 最悪条件を考えると, それぞれの制御される装置は新たなシステム・エクスクルーシブの送信を始めたばかりで, EOX の送信を終えるまでは WAIT に対して手の打ちようがないことも考えられる。したがって, “バッファ・オーバーフロー”状態が決して起こらないだけの最小バッファ・サイズを決定するのは困難である。しかしながら, 5台位までの装置と結んでこれをひとつの MIDI IN ポートでサポートするコントローラーの場合, 1024 バイト程度の受信バッファ容量が適当なところかと思われる。

制御される装置においては:

制御される装置の出した WAIT メッセージは, 同様にして, 他の装置から出されたシステム・エクスクルーシブがすでにマージャー部でコントローラーへの転送待ちになっているために, ここで立ち往生となるかもしれない。

制御される装置においては, 5台位までの制御される装置からのデータを外部マージするシステムでも, 512 バイトにまで受信バッファを増やしておけば, うまく働いてくれそうに思われる。

付録F　コマンド / 情報フィールド　アルファベット順　索引

著作・発行●社団法人 音楽電子事業協会
〒101-0061　東京都千代田区三崎町 2-16-9　イトービル4F

MIDI 1.0 規格書

MIDI Standard & Recommended Practices(日本語版 98.1)

1998 年 12 月 15 日　第1版発行
2016 年 7 月 27 日　PDF 版発行

監修●
赤井電機 株式会社
カシオ計算機 株式会社
株式会社 河合楽器製作所
株式会社 コルグ
株式会社 第一興商
ティアック 株式会社
日本ビクター 株式会社
フォステクス 株式会社
松下電工 株式会社
松下電器産業 株式会社
ヤマハ 株式会社
ローランド 株式会社

翻訳●小町 明, 渡部 柾彦
編集●株式会社 リットーミュージック
（担当：中島 安貴彦）
DTP コーディネート●波多江 潤子
PDF 編集●上杉　尚史

本書記事の無断転載, 複製は固くお断りします。

ISBN4-8456-0348-9 C3055