リアルタイムOS上でのQoS制御手法の検討

処理モード方式を用いたQoS制御とリアルタイムOSによるQoS制御支援機能

金光　恭治†　　本田　晋也†　　高田　広章†

† 豊橋技術科学大学情報工学系　〒441-8580 愛知県豊橋市天伯町雲雀ヶ丘1-1
E-mail: †{kyouji,honda,hiro}@ertl.ics.tut.ac.jp

あらまし　近年，コンシューマ向け組込みシステムは多機能化が進み，マルチメディア処理を備えるものが多くなってきている．しかし，マルチメディア処理は付加価値的存在であり，システムが高負荷時には，十分な処理時間が与えられない可能性がある．本論文では，マルチメディア処理の品質に対してQoS制御を行い，この問題を解決する手法を提案する．具体的には，処理時間の異なる処理モードを搭載したマルチメディア処理アプリケーションを用意し，その処理モードを利用可能な処理時間に応じて選択する方法を取る．また，リアルタイムOSにより利用可能な処理時間を事前に決定し，アプリケーションに伝えるQoS制御支援機能についても提案する．また，提案した方式について，μITRON4.0仕様準拠のOS上への実装を行い，評価する．
キーワード　マルチメディア処理，QoS制御，複数の処理モード

Investigation of QoS control on real-time OS

Multiple processing mode and QoS control support function by real-time OS

Kyouji KANEMITSU†, Shinya HONDA†, and Hiroaki TAKADA†

† Department of Information and Computer Sciences, Toyohashi University of Technology　Tempaku-cho 1-1, Toyohashi-shi, 441-8580 Japan
E-mail: †{kyouji,honda,hiro}@ertl.ics.tut.ac.jp

Abstract　When the system is high traffic, enough processing time is not given to the multimedia processing. Because the multimedia processing is a value added value function in the embedded system. We propose the technique for solving this problem by using the QoS control for the quality of the multimedia processing. Multimedia application provides with processing mode. Those processing mode has different processing time. Multimedia application selects processing mode according to the time that can be used. Moreover, we propose QoS control support function by the real-time OS. This paper also presents implementation and evaluation of this technique.
Key words　Maltimedia Processing,　QoS Control,　Multiple Processing Mode

1. はじめに

近年，携帯電話に代表されるようなコンシューマ向けの組込みシステムは，市場での商品価値を高めるため，本来の機能の他に様々な機能を兼ね備えるようになってきている．特に音声や動画の再生といったマルチメディア処理機能はその最たるもので，ここ数年，付加機能として備えた機器が多く開発されている．

このような機器では，本来の機能を処理するタスクとマルチメディア処理を行うタスクが存在するが，本来の機能を処理するタスクはマルチメディア処理を行うタスクより優先して実行しなければならない．例えば，音楽再生機能を持つ携帯電話では，音楽再生よりも，通話はもちろん，それに伴う着信，また待機時における電波の監視などを優先して実行しなければならない．そのため，システムの高負荷時には，マルチメディア処理に十分な処理時間が与えられない可能性がある．

しかしながら，マルチメディア処理は，十分な処理時間が与えられない場合にも，音飛びや駒落ち等のユーザーに不快感を与える要因を発生させないことが重要である．すなわち，前述の音楽再生機能を持つ携帯電話で，音楽再生中に電波の監視処理により，その再生が途切れるといったことは避けたい．

この問題を解決する方法として，Quality of Service 制御（以

下，QoS 制御）と呼ばれる手法がある．この問題に対する QoS 制御は，マルチメディア処理は処理品質を落とすことにより処理時間を短縮することが可能であることに着目し，十分な処理時間が与えられないときは処理品質を落とすことにより再生を途切れさせない手法である．

QoS 制御を実現するためには，処理品質の異なる複数の処理モードを持つマルチメディア処理アプリケーションの実現と，そのアプリケーションが利用可能な処理時間の見積もりが必要となる．利用可能な処理時間の見積もりは，システム全体の状況により左右されるため，アプリケーションレベルではなく，システム全体を管理するリアルタイム OS により行った方が精度の高い見積もりが可能となると考えられる．

そこで，本研究では，組込みシステムにおけるマルチメディア処理の QoS 制御を支援するために，リアルタイム OS が持つべき機能について検討する．具体的には，マルチメディア処理として MP3 の再生処理を例として，まず処理品質の異なる複数の処理モードを持つ再生アプリケーションが構築可能であるか，またどの程度の処理時間の削減が可能か評価する．次にアプリケーションレベルで利用可能な処理時間の見積もりを行うことで QoS 制御を実現する．この結果を基に QoS 制御の支援のためにリアルタイム OS が持つべき機能について検討し，評価する．

MP3 を例としたのは，最近の携帯機器における音楽フォーマットとして幅広く用いられていることや，仕様や再生アプリケーションのソースなどがオープンであり，詳しい情報が比較的簡単に入手できる点などから評価対象として適当と考えたためである．

本論文では，2 章で本研究の前提と QoS 制御手法の実現について述べる．3 章では，処理モード選択方法を MP3 再生アプリケーション（以下，MP3 プレーヤ）に適応する妥当性の評価について述べる．4 章では，処理モード選択機能として，マルチメディア処理が利用可能な処理時間を見積もる方法について，その手法と評価について述べる．5 章ではリアルタイム OS による QoS 制御支援機能について述べる．最後 6 章では本論文のまとめと今後の展開について述べる．

2. 前提と目的

本章では，この研究を行なうにあたり，その前提と，組込みシステムに QoS 制御手法を用いる理由，およびその QoS 制御手法の実現にむけての目的を述べる．

2.1 前　　提

マルチメディア処理が付加機能である組込みシステムにおいては，本来の機能が最も高い優先度で実行されるため，マルチメディア処理には十分な処理時間が割り当てられない可能性がある．

マルチメディア処理に十分な処理時間が割り当てられない場合，通常は処理に十分なリソースをシステムに追加することにより解決することになる．例えば，プロセッサの高速化による処理時間の短縮，マルチプロセッサ化による処理の分担化などの方法がある．また，バッファを追加し，そこにプロセッサの空き時間に処理したデータを蓄えておき，高負荷時にはバッファの内容を出力する方法もある．しかし，組込みシステムにおいては，これらの方法は消費電力やコスト等の制約により，有効な手法ではない．また，バッファの追加はインタラクティブ的な利用時に遅延の原因となるため，常に利用できるとは限らない．また，組込みシステムは一度システムが構成されると，その構成が変えられることは少ない．そのため，リソースの追加自体が難しい場合も多い．しかし，その反面，構成が固定であればそのシステムの振る舞いはパソコン等の汎用システムに比べて比較的予測可能である．特にシステムにおける本来の機能は一般的にその処理が始まるとその動作は周期的であることが多い．例として，先の音楽再生機能付きの携帯電話の場合，マルチメディア処理より優先される電波監視機能は周期的に一定の処理を行なって監視している．よって，本来の機能としてのタスクは周期タスクであるとする．なお，本来の機能の処理時間がマルチメディア処理のデッドライン以上になる場合，マルチメディア処理はその処理を実行することができないため，そのようなシステムは想定していない．そのため，周期タスクの処理時間は，周期毎に変わる可能性があるが，その処理時間の差には大きな差はないものと考えられる．よって，周期毎の実行時間とその周期タスクの平均実行時間はほぼ等しくなるため，見積もる際には平均処理時間を利用できる．本研究では，タスクの実行時間は一定にして各評価を行っている．

2.2 QoS 制御手法

前述のマルチメディア処理に十分な処理時間が割り当てられない問題に対して，組込みシステムでは QoS 制御手法による解決が有効である．QoS 制御は前述の解決方法と異なり，リソースの追加がまず必要でないため，組込みシステムに適していると言える．

QoS 制御を実現するためには，マルチメディア処理を行うアプリケーションにあらかじめ処理時間の異なる処理モードを用意しておき，システムの負荷状況に応じてそれらを選択することで実現できる．

各処理モードは，その処理時間が長くなるほど高い処理品質となる．図 1 に利用可能な処理時間と処理品質の関係のグラフを示す．C の区間のように最高品質の処理を行なうのに十分な処理時間が与えられるときには，処理時間は長いが最高の処理品質である処理モードで実行する．一方，利用可能な処理時間が減少し，B の区間に入ると処理品質を徐々に落として，処理時間が短い処理モードで処理する．

しかしながら，提供する品質にも限度があり，ユーザーの許容範囲内の品質のものを提供する必要がある．そのため，A の区間程度の処理時間しか利用できない場合は，再生処理を行っても提供するに値する結果を得られないため，品質はゼロとなる．しかしながら，通常のマルチメディア処理では C の区間以外で処理することはできなかったものを，QoS 制御によって B の区間まで処理を可能にすることができるため，有効な手段と言える．

2.3 組込みシステムにおける QoS 制御手法の実現

組込みシステムにおいて QoS 制御手法を実現するためには，

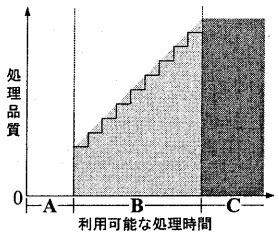

図1　処理品質と処理時間の関係

Fig. 1　Relation between processing quality and processing time

QoS 制御を行なうアプリケーションの実現と，利用可能な処理時間の見積もりを行なう必要がある．

QoS 制御を行なうアプリケーションの実現には，そのときに利用可能な処理時間に応じて，処理品質を変動させることが条件となる．そこで，前述の QoS 制御手法を用いることで，アプリケーションによる QoS 制御を可能にする．

利用可能な処理時間の見積もりは，処理モードを選択するには，利用可能な処理時間を把握しなくてはならないために必要となる．その方法の実現法としては，アプリケーション自身が処理時間を見積もる方法と，リアルタイム OS がその利用可能な処理時間を決め，アプリケーションに伝達する QoS 制御支援という方法が考えられる．しかしながら，現状のリアルタイム OS は，タスクを制御する立場にあっても，タスクの状況までは把握していない．そのため，事前にこの先起こる事象を予測することは非常に難しく，アプリケーションが利用可能な処理時間を正確に見積もることは出来ない．また，処理時間を前もって伝える機能も持っていない．

しかしながら，先にも述べたとおり，本来の機能などの優先される処理は周期タスクで構成されていることが多い．つまり，リアルタイム OS はそのタスクの起動周期と平均処理時間を把握さえすれば，このタスクが今からある一定時間内で使用する処理時間を見積もることも可能である．そこで本研究では周期タスクの起動周期と平均処理時間をリアルタイム OS に事前に把握させておくことで，マルチメディア処理のデッドラインまでの間に優先的に処理されるタスクの処理時間を見積もり，そのデッドラインからその処理時間を除いた残り時間をマルチメディア処理が利用可能な処理時間としてアプリケーションに伝える方法を検討する．

3.　複数の処理モードを持つ MP3 プレーヤ

本章では，処理時間の異なる処理モード持つ MP3 プレーヤの実現について述べる．

3.1　評価環境

評価に用いた MP3 プレーヤは，Xing Technology 製のデコードライブラリをベースに使用し，μ ITRON 上で動作する．

なお，測定環境として，プロセッサに SH-4（日立製 SH7750:200MHz）を用いた MS7750SE01（日立超 LSI システムズ）評価ボードを用いた．また，デコードに用いた MP3 ファイルのフォーマットは，MPEG1.0 LayerIII，サンプリング周波数：44.1KHz，ビットレート：128Kbit/s，音声タイプ：JointStereo，再生時間：61.99 秒である．

3.2　デコード処理の簡略化

MP3 はフレームと呼ばれる単位でデコード処理を行う．1 フレームには PCM データの 1152 サンプル分が圧縮して格納されている．つまり，どのフレームの再生時間も同一であり，処理の結果を得なければならないデッドラインが同じである．ただし，その圧縮率はフレームごとに異なるので，デコードに必要な処理時間はフレームごとに異なることになる．

MP3 の再生を途切れさせないためには，あるフレームの PCM データを出力中に，その次のフレームをデコードしておき，PCM データの出力終了後には，すぐ次のフレームの PCM データを出力できる状態にしておかなければならない．よって，1 フレームのデコード処理時間が 1 フレームの再生時間より長くなれば音が途切れることになる．

そのため，フレーム単位で処理モードを決定し，QoS 制御を行なうことが望ましい．QoS 制御を行うには，各フレームごとにデコード処理を部分的に簡略化することにより，処理時間の短縮を行うことになる．

3.3　簡略化項目とその評価

デコード処理において簡略化可能な箇所を調査した．ただし，簡略化もユーザーが聴きとれる許容範囲内で行なうものとする．その結果，簡略化可能な個所としては，次の 5 つの個所が挙げられる．

- 量子化ビット数の削減
  ダウンサンプリングを行なう．波形を少ないビット数で再現することで，出力データ量を減らして処理を早くする．最高 32 ビットで処理し，16 ビットで約半分，8 ビットで約 1/4 の処理時間になる．
- 出力周波数の低減
  出力する最大周波数を低減する．通常時は 24KHz で設定している．周波数が減ると波形の形が少ないデータ数で再現をされるため，処理時間が短縮される．理論上，いくらでも減らすことが可能であるが，可聴範囲としては 8KHz 以上推奨である．
- モノラル化
  モノラル化を行なう．データ量は半分となる．また，ステレオ処理を行なわないため，ステレオに関する処理が全く実行されないため，処理時間が短縮できる．
- 復元を行なうサブバンド数の削減
  1 つのサブバンドから復元するサブバンド数を指定する．サブバンドフィルタに圧縮された 1 つのサブバンドを入れると，32 個のサブバンドに復元される．サブバンドは周波数帯域を分割しているため，最低 1 つの復元を行わなければ，PCM データを生成することができない．
- エリアシング復元処理の省略
  MP3 はサブバンドフィルタ方式と MDCT 方式のハイブリッドフィルタ方式である．分割されたサブバンドの隣接部分に生じるエイリアシング歪みを含んだまま，

表1　簡略化処理の効果

Table 1　Effect of simplified processing

処理内容		所要時間 [ms]	削減率 [%]
簡略化しない		25192.21	-
周波数成分の量子化 bit 数の削減	16bit	17289.15	31.37
	8bit	9826.34	60.99
最大出力周波数の低減		20616.22	18.16
モノラル化		16703.97	33.69
エリアシング除去の省略		24819.77	1.48
サブバンド数の削減		24515.83	2.68
以上の全ての処理を省略		7702.68	69.42

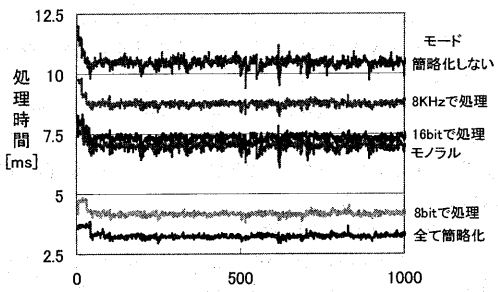

図2　各フレームの簡略化処理

Fig. 2　Simplified processing of each frame

MDCT は行われる．その結果にもエイリアシング歪みが含まれるため，この除去処理を行い，MDCT フィルタ方式のみの結果と同様の結果にする．そのため，復元時にこの処理を行わなければ，エイリアシング歪みが復元されず，サブバンド復元時に，隣接するエイリアシング歪みを打ち消す処理が実行されない．

これらの処理を簡略化することで十分に処理時間を減らすことができれば，これらの簡略化項目を組み合わせることで，複数のデコードモードを用意できることになる．

各部分および全ての処理の簡略化でどの程度処理時間が減少するか評価した結果を表 1 に示す．

以上の結果より全ての簡略化処理を併用化することで，処理時間を 69 %短縮することが可能であることを確認した．

次に 2 桁の削減率を示した簡略化項目について，各フレーム単位での効果を図 2 に示す．この結果から処理時間の短縮の効果は各フレームごとに出ているため，期待した効果が得られている．この結果より，QoS 制御を行うアプリケーションとして，この 6 種類の簡略化処理を各処理モードとして持つ MP3 プレーヤを作成した．

4.　アプリケーションレベルでの QoS 制御

本章では，処理モードを決定する方法として，アプリケーションレベルで利用可能な処理時間の見積もる方法について述べ，QoS 制御が行えることを確認する．

表2　タスク構成

Table 2　Task composition

	タスク 1	タスク 2
処理時間	3 msec	2 msec
起動位相	0 msec	0 msec
起動周期	12 msec	16 msec

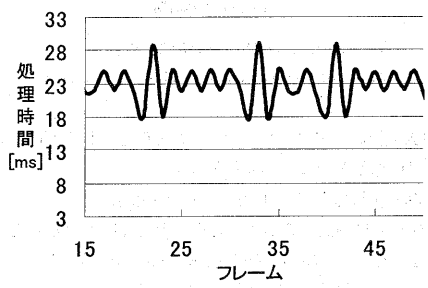

図3　デコード処理時間（通常再生）

Fig. 3　Decode processing time(Normal)

4.1　前フレームで利用した処理時間による見積もり

QoS 制御を行なうアプリケーションは前フレームのデコード処理において，その開始からデコード結果が得られるまでの処理時間およびその時の処理モードを把握しておく．それらの情報から前フレームに，どの程度他のタスクの処理時間が含まれているか計算する．なお，その計算を行なうにあたって，各処理モードの処理時間を知っておく必要がある．そのため，あらかじめ各処理モードの平均処理時間を求めておき，その値を使用して計算を行なう．そして，今から処理するフレームにも同程度の負荷がかかるものと仮定し，その負荷でも処理できるモードを選択する方法である．

この方法を前章で用いた環境に適応し測定を行なう．デコードに用いた MP3 ファイルは 44.1KHz であるため，以下の計算式より一フレームあたりの再生時間は約 26msec となる．

sec/sample : 1 ／ 44100 = 0.000022675

sec/frame : 0.000022675 × 1152 ＝ 0.02611 ≒ 26msec

つまり，26msec の再生時間の間に次のフレームのデコード値を出力できる状態にしておかなければ音が途切れることになる．そこで音の出力に関する処理などのデコード以外の部分のオーバーヘッドを考慮し，3msec ほどの余裕を設けた 23msec でデッドラインを設定した．なお，本来の機能などの優先度の高い処理を想定したその他のタスクは表 2 に示す 2 個の周期タスクがある．

まず，この 2 つの周期タスクが動作している環境において，通常の再生時におけるフレーム単位の処理時間を図 3 に示す．つまり，全てのフレームを簡略化しない，最高品質の処理モードで行なった結果である．なお，この処理時間は，デコードが開始されて結果が得られるまでに要した時間を示している．つ

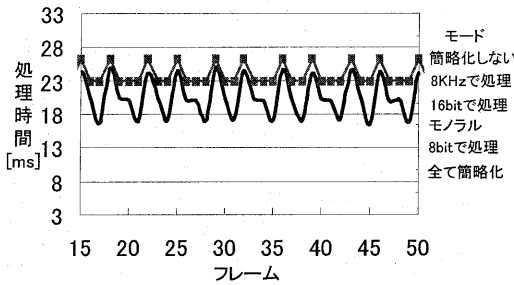

図4　アプリケーションによる QoS 制御（1）
Fig. 4　QoS control by application(1)

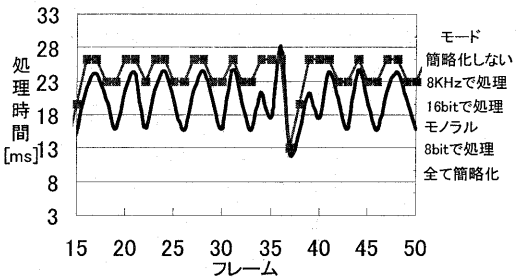

図5　アプリケーションによる QoS 制御（2）
Fig. 5　QoS control by application(2)

まり，デコードを開始してから，他のタスクに邪魔された場合，そのタスクの処理時間を含むことになる．

この結果を見れば，多くのフレームでデッドラインとして設定した 23msec を軽くオーバーしている．ただし，23msec は余裕を持って設定した値であるため，多少ならばオーバーしても再生が中断されることはない．しかし，いくつかのフレームでは 1 フレームの実再生時間である 26msec もオーバーしている．この状態において，処理モードを選択することにより，デッドラインをオーバーする個所が減らせれば，QoS 制御を行なえていると言える．そこで，図 4 にアプリケーションによる QoS 制御を行なった際の測定結果を示す．

正方形の点がある折れ線がそのフレームのデコードを行なう際に選択した処理モードであり，グラフの左にモードを示している．その並びは図 2 のグラフと同じで，上から処理時間が長いモードとなっている．太い線の方は図 3 同様，デコードが開始されて結果が得られるまでに要した時間を示している．

この結果より処理モードの設定後，QoS 制御により処理時間の短縮に成功しているのが確認できる．負荷が下がると，次のフレームは高い品質の処理モードで処理しようとし，負荷があがると処理時間の短い処理モードを選択して処理しているのが確認できる．また，図 3 に比べ，デッドラインをオーバーしているところは多少あるが，大幅に超えているところはない．

4.2 問 題 点

次にタスクの処理時間をそれぞれ 1msec 長く設定した場合の QoS 制御結果を図 5 に示す．35 から 40 フレームにかけて前のフレームと処理時間が大きく変動している．これは，次のフレームに同程度の負荷がかからない場合に，無駄な QoS 制御が行なわれている可能性があるためだと考えられる．

図 6 のような場合，1 フレーム目はタスク A と B に 2 回ずつ邪魔されている．そのため，デッドラインからこれらのタスクの処理時間を除いた時間が MP3 プレーヤの利用できる時間となる．この時間内で最高品質の処理が出来なかった場合，2 フレームの開始前に QoS 制御を行うわけだが，次も同じ負荷がかかると仮定したモードの設定をする．しかしながら，2 フレーム目ではタスク B には一度しか邪魔されていない．そのため，タスク B の処理時間 1 回分，無駄に品質を下げてしまうことになる．そのため，よりよい QoS 制御を行なうためには，予測ではなく，自身が本当の利用可能な処理時間を知る必要がある．しかし，これにはアプリケーションだけでは不可能であるため，リアルタイム OS による QoS 支援が必要になる．

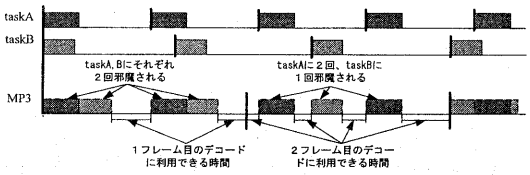

図6　適切な QoS 制御が行なわれない例
Fig. 6　Example of QoS control not accurate

5. RTOS による QoS 制御支援機能

本章では，処理モード選択の方法として，アプリケーションが利用可能な処理時間をリアルタイム OS からアプリケーションに伝える QoS 支援機能による方法をついて説明する．

5.1 タスク状態の把握による見積もり

アプリケーションが利用可能な処理時間を前もって知ることができれば，自ら予測する必要はない．しかし，その前もって知る処理時間はできるだけ正確でなければ，QoS 制御の精度は落ちることになる．そのため，全てのタスクがその上で動作しているリアルタイム OS によりその処理時間を決定する方法をとる．その具体的な方法は，リアルタイム OS にアプリケーションよりも優先度の高い他のタスクの状態を監視する機能を設ける．フレームのデコード開始時には，監視情報と事前に把握してある情報を元にその他のタスクが今から 1 フレーム分のデッドライン内に何回起動し，どの程度の処理時間を使うのかを見積もる．そして，その処理時間を考慮して，アプリケーションにどの程度利用可能な処理時間を与えられるか決定し，アプリケーションに伝達する．伝達してもらうことでアプリケーションは正確に処理モードが選択できることになる．

タスクの状況把握のため，リアルタイム OS には他のタスクを監視しておく機能を用意する．監視する項目は，現在のタス

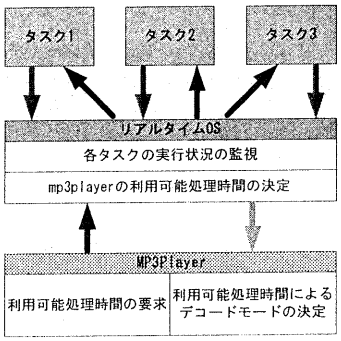

図7　リアルタイム OS によるタスク管理機能
Fig. 7　Task management function by real-time OS

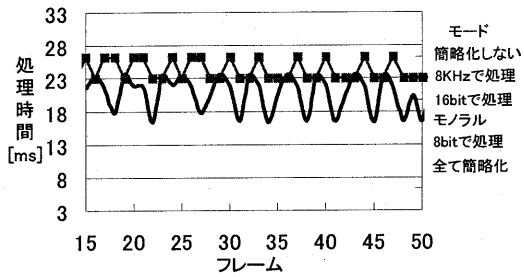

図8　QoS 制御支援機能を利用した QoS 制御
Fig. 8　QoS control using QoS control support function

クの状態とする．なお，タスクの起動周期・平均処理時間は事前に与えておく．これらの情報をもとに利用可能な処理時間を決定し，MP3 プレーヤからの問い合わせに応じて伝達することで実現できると考えられる．機能の概要を図 7 に示す．

利用可能な処理時間の決定には，まず各タスクが 1 フレーム分のデコードを行なう間に何回起動するかをその起動周期から計算する．その起動回数と平均処理時間の積をそのタスクが 1 フレーム分のデコード処理のデッドライン内での処理時間とする．ただし，起動するタスクが次のフレームに跨がる場合，またはデコード開始時にすでに起動中のタスクがある場合，このフレーム内での処理時間も計算し，出来るだけ正確な処理時間を見積もる．そして，各タスク毎に計算した結果を，1 フレームのデッドライン時間から引いたものがアプリケーションが 1 フレームのデコードで利用可能な処理時間となる．

5.2 評　価

この環境を実機上に構築する．ただし，今回はその効果を確かめるのが目的であるため，この機能をタスクで実現し，エミュレートする形で実現した．測定環境は，図 5 と同環境・同周期タスクで実行し，その効果を調べた．その結果を図 8 に示す．

この結果を見ると，図 5 までに比べ，23msec を大きく超えるフレームは存在していない．また，モード変更も簡略しないモードか，サンプリング周波数を 8KHz で処理するモードの 2 つのみで実行できている．そのため，図 5 に比べ，大きく変動しているところもない．これはこの先，最終フレームに至るまでその効果は確認できた．よって，QoS 制御支援を行なうことでより良い QoS 制御が行なえると言える．なお，この QoS 制御支援機能のオーバーヘッドは，10μsec 以下であるため，1 フレームの実再生時間の 26msec に対して，0.04 %程度である．また，MP3 プレーヤが各モードの切り替えに要する時間は 400msec 前後で実再生時間の約 1.5 %であり，この点も考慮してデッドラインを定めておけば，デコード処理に与える影響は無いに等しいと言える．

6. ま と め

本研究では，組込みシステム上でのマルチメディア処理に適した QoS 制御手法を検討した．その方式は，複数の処理モードを持つアプリケーションが，そのモードを利用可能な処理時間に応じて切り替えることで QoS 制御を実現するものである．

また，利用可能な処理時間を事前に見積もる方法として，リアルタイム OS にマルチメディア処理よりも優先度の高いタスクの状態を把握させることで，アプリケーションが利用可能な処理時間を決定する QoS 制御支援方法を提案した．

また，これらを実機上で評価した結果，フレーム単位の簡略化で最大約 69 %の効果を得ることができ，それに基づき設定した処理モードをリアルタイム OS の QoS 制御支援機能との組み合わせにより，無駄なモード変更がない QoS 制御が実現できた．この結果より，限られたリソース下でもマルチメディア処理を可能な限りつづけることができ，組込みシステムに適した方法であると言える．

今後の課題は，予期しない負荷発生時に備えて，1 フレーム分のバッファを用意した環境を実機上に構築し，その評価を行う．そして，その機能も踏まえたものを今のエミュレートするタスクから，実際にリアルタイム OS の機能として実装する．また，MP3 以外のマルチメディア処理についても，この方式が適応可能か調査も行なっていくことも考えている．

文　献

[1] D. Rosu, K. Schwan, S. Yalmanchili, and R. Jha, "On Adaptive Resource Allocation for Complex Real-Time Applications," IEEE Real-time System Symposium, pp.320-329, 1997
[2] 菅原智義, 高野陽介, "組み込みシステムに適した QoS 制御方式－ QoS テーブル方式の提案と実装," 情報処理学会論文誌, pp.1677-1687, 2000
[3] 浦田敏道, "詳細 MP3 マニュアル," エム研, 1999
[4] 高橋政雄, "VisualC++6.0 でつくる MP3Player プログラミング," エーアイ出版, 1999
[5] 社団法人トロン協会, "μITRON4.0 標準ガイドブック," パーソナルメディア, 2001
[6] 高橋政雄, "MP3 のデータ形式とその解析・MP3 プレーヤのプログラミング," CQ 出版社, Interface 2000 年 8 月号, pp.91-112, 2000
[7] 中村伸一, "MP3 の符号化技術の基礎," CQ 出版社, Interface 2000 年 8 月号, pp.79-87, 2000
[8] 日立製作所, "SH7750 ハードウェアマニュアル," 日立製作所半導体グループ電子統括営業本部, 2000