

RAM 容量削減のためのプリンタエンジン制御方法

Printer Engine Control Method to Reduce RAM Capacity

近藤 正生*
Kondo, Masao

日高 真聡*
Hitaka, Masatoshi

土井 隆広*
Doi, Takahiro

要旨

カラータンデムプリンタのエンジン制御において、レジスト（色重ねズレ）補正制御におけるパターン位置検出の演算を1パターン分のデータをサンプリングした時点から順次実行し、演算の完了したデータから順次外部解析装置に掃き出すことでRAM使用量を大幅に削減し、プリンタエンジン制御ではCPU内蔵RAMのみを使用するようにした。

Abstract

The RAM-size needed for color registration adjustment control in an engine controller of a tandem color printer is significantly reduced by sequential processing of data, to require use of only the internal RAM in CPU.

Data of a pattern of registration adjustment control are sequentially sampled, part by part, and the position of the sensed pattern of each part is computed after its sampling has finished. When computing of the data of a pattern part is completed, it is sent out sequentially to an external analysis device.

1 はじめに

カラープリンタ市場においては、年々価格競争が激化しており、新製品においては機能、性能を向上させて且つ、コスト削減をすることが要求されている。その実現のため、従来もモータ制御や高電圧電源制御などのフィードバック制御をソフトウェアで行うことで、ハード回路のコストダウンを図ってきている。今回は、エンジン制御基板上の拡張RAMを削減することで、部品費のコストダウン、更にRAM削除に伴う実装面積とCPU-RAM間のバス配線のための面積の削減で基板のコストダウンを実施し、プリンタエンジンとして製品への搭載の目的が立ったために報告する。

2 従来のプリンタエンジンにおけるRAM容量の分析

従来までのタンデム方式のプリンタエンジンでは、CPU内蔵のRAM（12～32Kbyte）、更に拡張RAM（128Kbyte）を使用する構成であった。コストダウンのために、この拡張RAMを削除するという今回の課題に対して、最初にRAM使用量の分析を行った。その結果、Fig. 1に示すように当初の予想通り画像安定化制御、レジスト補正制御で大量（約90Kbyte）のRAMを使用していることが確認できた。他の制御においても、RAMに余裕があるための設計、コーディングとなっていることも確認できたため、画像安定化制御、レジスト補正制御でのRAM使用量を減らすことができれば、外部の拡張RAMを削除しても製品仕様を満足できると判断した。そこで、上記2つの

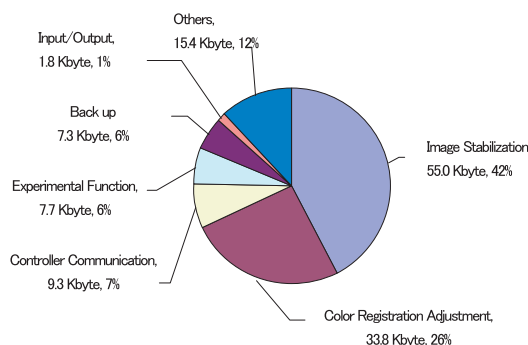


Fig.1 RAM share in a current method

* コニカミノルタビジネステクノロジーズ(株)
機器開発本部 機器第3開発センター 第33開発部

制御について、内部の処理を詳細に分析し、RAMの削減方法の検討を行った。

①画像安定化制御では、Fig. 2に示すようなパターンの濃度をセンサーにて検出して、最適な現像条件と露光条件を決定している。この制御においては、サンプリング周期は約1mm程度、現像条件を決定した後、その条件で現像したパターンを使って露光条件を決定する。これらの決定処理は順次計算となるため、本来必要となるRAMサイズは、各色毎の現像条件と露光条件の数十byteとなる。しかし、これらのデータは、カラープリンタの画質決定の基準となるため、画質に問題が発生した場合の解析には、サンプリングした濃度データ、演算過程のデータを全て必要とすることがあった。そのため、RAMに全データを保持していたことから、約55KbyteのRAMを必要としていた。

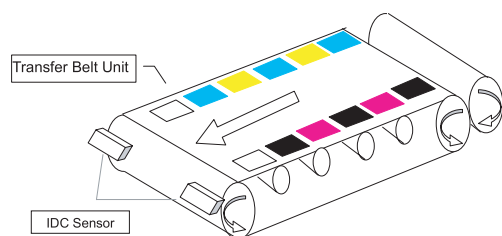


Fig.2 Sample pattern for image stabilization control

②レジスト補正制御では、Fig. 3のようなパターンを用いて、パターン間隔の測定を行い、その結果から画像書き出しタイミングの調整を行っている。つまり、各色毎の画像書き出し位置補正量を求めるのがレジスト補正制御である。測定用のパターンをパターン検出用センサーにてモニターするとFig. 4のような波形が得られる。これは、センサーのアーチャー径が数mmあることから、パターン通りの検出波形とはならない。センサーによる読み取り（サンプリング周期）を書き込みと同じ解像度（600dpi）で行っても、やはりFig. 4のような波形となって検出される。このセンサーのアーチャー径が600dpi相当もしくはそれ以下であれば、このセンサー検出波形からエッジを検出することで、パターンの正確な位置が把握でき、各色毎の画像書き出し位置補正量は簡単に求めることができる。本制御では、各色のパターンとそのパターン間の余白部も全て、600dpi周期でサンプリングするため、パターン長が長くなるとそれだけ、サンプリング数が増えることとなる。今回の検討は、各色のパターン幅が1mm、間隔が5mm、パターンの繰り返しが16回（16ユニット）として検討した。この場合、総パターン長が384mmとなる。これを600dpiでサンプリングするとサンプリング回数は9071回、更にセンサーは左右に2つあるため、計18142個のデータをサンプリングすることとなる。従来のカラータンデムプリンタにおいては、このサンプリングデータを全て一旦RAMに保持し、サンプリ

ングの完了で演算を行っていた。つまり、RAM容量はシステム（製品）として必要なパターン長とそのサンプリング周期に大きく影響されていた。ここでの演算は、以下のような処理を行い、各色の基準色からの書き出し位置補正量を算出している。

サンプリングしたデータに対して、

- ・ノイズ除去、スムージングなどのデータ処理
- ・各パターンの中心（もしくは重心）算出
- ・1ユニット毎に各色毎の基準色との距離の算出
- ・16ユニットの平均から補正量の算出

ここまでで、副走査方向の画像書き出し位置補正量を算出し、更に別のパターンを用いて、同様の演算処理にて主走査方向の画像書き出し位置補正量、倍率補正量を求めている。このような制御を実施していたことで、レジスト補正制御のデータのサンプリングのために約35KbyteのRAMを使用していた。

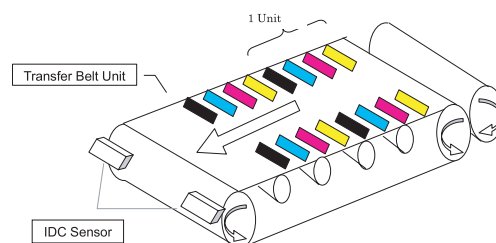


Fig.3 Sample pattern for color registration adjustment control

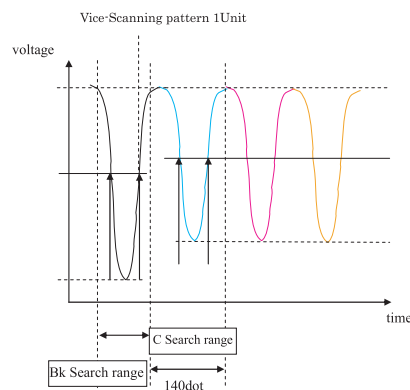


Fig.4 Sensor detected waveform for the pattern in Fig. 3

3 RAM 容量削減手法と課題

RAM容量を削減する方法としては、サンプリング数を減らす（600dpi → 300dpi、パターンのユニット数削減）か、サンプリングしたデータを全てRAMに保持するのではなく、リアルタイムで演算を行っていく方法でしか削減はできない。これは、画像安定化制御、レジスト補正制御の両方の制御に当てはまる。

サンプリング数を減らすとした場合には、従来同等の精度を維持できるのかという点で課題があり、課題をク

リアするために、他部署を含めた検証に非常に多くの工数を要することが考えられることから今回は選択を見送った。

残ったリアルタイムで演算を行う方法については、①演算の処理時間によって他の制御に影響を与えないのか、②従来可能であった評価時、問題発生時にサンプリングデータを外部解析装置で取り込んで解析するという開発ツールとしての機能をどうやって実現するのかという2つの課題があった。

1つ目の課題の判断基準はCPUの負荷がどの程度になるのかということである。プリンタエンジン制御は一定周期でFig. 1で抽出したような決められた処理を繰り返し、或いはシーケンシャルに実行しているが、この一定周期の中でCPUがどれだけ処理を行っているかという割合をCPU負荷としている。予定した処理がこの周期内で完了しないと言うことは、シーケンスが正しく動かない可能性があるため、エンジンの動作を保証できないということになる。従来の経験から、この負荷が80%程度であれば、突発的な要因が発生しても、現状のシステムであれば正しく制御していけると判断している。したがって、このリアルタイムでの演算も一度にどれだけデータをどれだけ時間で演算するかによって処理時間が異なるため、CPU負荷は変動する。しかしこの演算処理は、従来機にて行っているアルゴリズムであるため、その時間を見積もることができる。そこで、CPU負荷として問題とならないような演算の周期とデータ数を求めることで1つ目の課題を解決できると考えた。

2つ目の課題のとして挙げた開発ツールとしての機能維持については、直接センサー出力をモニターして、記憶させるような外部治具を新規に設計して対応する方法が提案された。しかし、この場合は、外部治具でサンプリングしたデータと、プリンタエンジン側のサンプリングタイミングとの同期が取れないこと、エンジンがどのように検出して、その結果を導き出したのかが判断できないことから従来機能を満足しないために採用できないと判断した。また、開発段階のみ拡張RAMを設けて、量産段階では拡張RAMを未実装にするという提案もあったが、これについては当初の目的である基板面積の削減に繋がらず、更には量産ライン等でのデータ取り等が一切行えないということから採用しないこととした。そこで、最終的には従来からの外部解析装置とプリンタエンジン間のシリアル通信の速度を上げて、リアルタイムで演算したサンプリングデータをそのまま外部解析装置へ送信することで従来機と同等の開発ツールとしての機能を満足させる方法を選択した。しかし、通信速度が上がるということは、ある周期内に処理するデータ数が増えることとなり、CPU負荷が上がることになる。したがって、2つ目の課題を解決する場合も、CPUの負荷を考慮する必要があった。

3. 1 リアルタイム演算処理

最初に、サンプリングしながらの演算は、演算が可能な単位のデータのサンプリングが完了した時点でを行い、演算の完了したデータエリア (RAM) は開放して、次のデータの一時保管用に使用することで、サンプリングに使用するRAMエリアを削減できる。

今回の検討は、A4タンデムプリンタのミドルクラスへの適用ということで、システム速度120mm/s、エンジン制御の基準周期を10msとして検討を行った。現状の解像度は、600dpiであるので、サンプリングも600dpiで行う。従って、サンプリング周期は、

$$25.4\text{mm}/600\text{dpi}/120\text{mm/s} \times 1000 = 0.3528\text{ms}$$

となるので、1周期でのサンプリングデータ数は、

$$10\text{ms}/0.3528\text{ms} = 28.3\text{個} \rightarrow 29\text{個}$$

したがって、今回のシステムでは1周期で29個分のRAMが必要となる。センサー出力は、1サンプル当たり8bitのA/D変換結果として取得するため、29byte分のRAMが必要である。システム速度が上がれば、この必要容量が増えることとなる。レジスト補正制御では、この1周期分のデータで演算が行える訳ではなく、Fig. 4のBk Search Range (探索範囲)、C Search Rangeに相当するデータをサンプリングした後にそのパターンを中心 (もしくは重心) を求める演算を行って、各パターンの位置を算出し、そこから各色毎の画像書き出し位置補正量を求めていく。この演算処理には、非常に多くの処理時間を必要とすることから、上記の1周期 (10ms) 内で1度に処理するのは困難である。したがって、これを複数の周期に跨って演算を行うことで、CPU負荷の集中を避けることとした。

今回のレジスト補正パターンからFig. 4で示す探索範囲内のサンプリング数は、140dotであるが、初期のズレ量を考慮して、その1.25倍の範囲を探索範囲として設計を行う。この場合の探索範囲内のドット数は、175dotである。175dotに対して、ノイズ除去、スムージングなどのデータ処理を行って、パターンを中心 (重心) まで算出する処理時間を見積もると、15ms程度である。従来機の実績において、プリント中、演算を含まないレジスト補正制御の処理は、1周期において3ms程度であることが、複数のCPUにおいて検証ができていた。したがって、4周期 (40ms) をかけて演算を完了すれば、他の処理に影響を与えないで演算できると判断した。1周期にサンプリングされるデータ29個を基準に必要なRAMサイズを求める。4周期では、 $29 \times 4 = 116\text{dot}$ がサンプリングされるが、175dot単位での演算となるため、RAMとしては、175dot以上で且つ116dotの倍数となる容量 (232byte) が必要となる。演算タイミングとの関係からこれを2セット用意することでサンプリングしたデータに対して確実に演算を行ってからRAMを開放するという制御を実現できる。したがって、センサー1つに対して、464byte (29

× 4 × 2 × 2 = 464) のRAMが必要となり、センサーが2つの今回のシステムでは、928byteのRAMが、レジスト補正制御のデータのサンプリングに必要と言うことになる。これにて、従来機と比較して30Kbyte以上のRAMの削減が可能となる。

ここで、RAMサイズの妥当性を確認する。上記1周期内でのサンプリングに必要なRAMサイズ29個を1つのバッファと定義してFig. 5を使って説明する。

- ・サンプリング周期
 $25.4\text{mm}/600\text{dpi}/120\text{mm/s} = 352.8\ \mu\text{s}$
- ・140dot分のサンプリング時間 t1:
 $352.8\ \mu\text{s} \times 140\text{dot} = 49.392\text{ms}$
- ・上記t1間に使用されたバッファ数
 $140\text{dot}/29 = 4.827\text{個}$
- ・175dot分のサンプリング&演算時間 t2:
 $352.8\ \mu\text{s} \times 175\text{dot} + 40\text{ms} = 101.74\text{ms}$
- ・上記t2間に使用されたRAM容量
 $101.74\text{ms}/352.8\ \mu\text{s} = 288.37\text{byte}$
- ・上記t2間に使用したバッファ数
 $288.37/29 = 9.944\text{個}$

サンプリング開始から最初の演算タイミングは、175dotのサンプリングが完了した時点である(CP1)。そこからは、演算を行いながらサンプリングを継続して実行している。その為、バッファは開放することができず、演算が完了した時点で初めてRAMの開放が可能となる(CP2)。以降は、140dot周期で演算を実行する(CP3, CP4)。各チェックポイント(CP1~CP4)でのバッファ残量は、

$$\begin{aligned} \text{CP1} &: 16 - 9.944 = 6.056\text{個} \\ \text{CP2} &: 6.056 + 4.827 = 10.883\text{個} = \text{CP4} \\ \text{CP3} &: 10.883 - 4.827 = 6.056\text{個} \end{aligned}$$

となり、常に6個以上のバッファが空いており、演算が完了してないバッファが新しいデータで上書きされないことが確認できたので、現状確保したバッファサイズで十分に今回の制御を実現できることが確認できた。

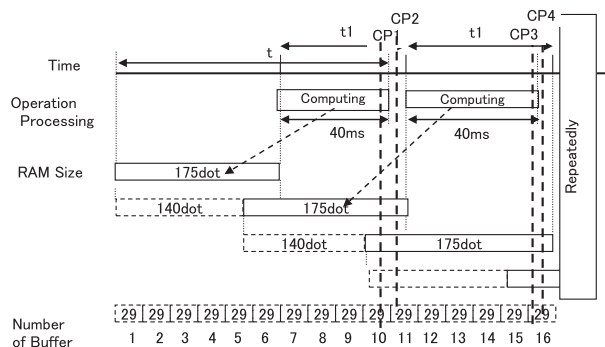


Fig.5 Verification of necessary RAM size

3. 2 リアルタイムデータ掃き出し

単にRAMを削減するだけでは、開発ツールの一部としての機能を損なうため、同等の機能を維持するためにサンプリングデータを演算が完了した所から順次解析装置へと掃き出すことが必要となる。プリンタエンジンと外部解析装置とは、シリアル通信にてデータの送受信を行っている。外部解析装置へのリアルタイムでのデータ掃き出しの課題は、処理時間である。ここで言う処理時間は、①単純にデータを送信する時間と②通信に必要なソフト的な処理に要する時間である。①は、RAM使用量に関係し、②はCPUの処理時間というところに関連する。

3. 2. 1 転送速度

最初に、通信速度であるが、サンプリングより早い速度でデータを外部に送ることができれば、演算処理の完了したデータから順次解析装置に送信することで、RAMを開放することができる。

1周期でのサンプリング数は、センサーが2つであるので、 $29 \times 2 = 58\text{dot}$ つまり、サンプリング速度は

$$58\text{dot}/10\text{ms} \rightarrow 5800\text{byte/s}$$

つまり 実質の通信速度がこれ以上であれば、通信のために余分なRAMを必要としないこととなる。今回、115200 bpsを選択することで、スタートビット、ストップビット、パリティビットを考慮しても

$$115200\text{bps}/11\text{bit} \rightarrow 10472.7\text{byte/s}$$

となるため、目標に対して十分に速い速度でサンプリングデータを解析装置に送信できる。

3. 2. 2 CPU処理能力

リアルタイムデータ掃き出しにおけるもう一つの課題は、通信速度があがることによるCPU負荷の増大であった。通信速度が上がることで、割り込み回数が増え、そのオーバーヘッド等が心配されたが、画像書き出し位置補正量の演算処理同様に現在使用が考えられるCPUでの処理時間を見積もったところ、通信割り込みの締める割合が1周期の約20%以下(今回の場合2ms以下)であった。

“3. 1 リアルタイム演算処理”で記載したように、従来の処理で3ms、上記からリアルタイムデータ掃き出しのための通信処理で2ms ということでは1周期の約50%を使っている。別途40ms間で実行する演算処理が15ms必要ということでは、4周期内の空き時間 20ms内で演算処理を実行することが可能であることから、CPUの能力としても今回のリアルタイム演算処理、リアルタイムデータ掃き出し処理を実現できることが確認できた。同時に、演算周期を40msとしたことの妥当性も確認できた。

4 RAM 削減の実現

画像安定化制御，レジスト補正以外の処理については，レジスト補正制御がないために元々外部拡張RAMを使用しない設計をしている4サイクルカラープリンタと同様の設計方法に改めた。

画像安定化制御においては，サンプリングデータ，演算過程データを順次外部解析装置に掃き出すこと（リアルタイムデータ掃き出し）で，約55Kbyte使用していたRAMを数十byteへと削減できた。レジスト補正においては，サンプリングしたデータをパターンごとに演算するようにし（リアルタイム演算処理），更には演算の完了したデータを画像安定化制御同様に順次外部解析装置側へ高速で掃き出すことで，従来約35Kbyte使用していたRAMを1Kbyte以下とすることが確認できた。これら，全ての対応を実施することで，Fig. 6に示すように外部拡張RAM無しでプリンタエンジン制御を実現できることが確認できた。

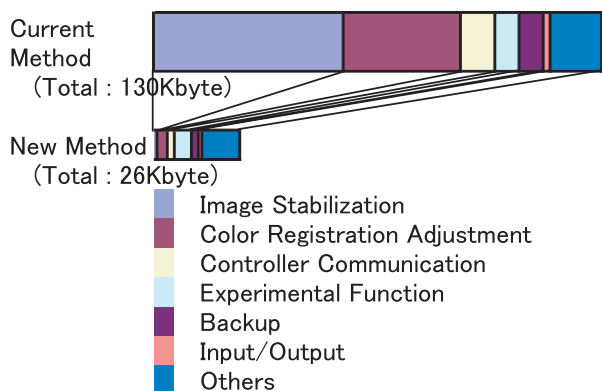


Fig.6 Improvement of RAM utilization

5 おわりに

今回は，コストダウンのための外部拡張RAMの削除という目標をA4カラータンデムプリンタのミドルクラスの製品で達成可能であることが検証できた（今回の検証は，システム速度120mm/sにて行ったが，160mm/sまでであれば今回の制御方法が適用できる見込みである）。次は，今回課題として挙げ，検討したものを更に条件を厳しくしてシミュレーションを行い，高速機分野（A3機（MFP）を含む）への適用を考えていきたい。