お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

Application Note

HD64413A Q2SD

アプリケーションノート

SuperH™ RISC engine 周辺LSI

Quick 2D Graphics Renderer with Synchronous DRAM Interface

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　1999.09

ご注意

1 本書に記載の製品及び技術のうち「外国為替及び外国貿易法」に基づき安全保障貿易管理関連貨物・技術に該当するものを輸出する場合，または国外に持ち出す場合は日本国政府の許可が必要です。
2 本書に記載された情報の使用に際して，弊社もしくは第三者の特許権，著作権，商標権，その他の知的所有権等の権利に対する保証または実施権の許諾を行うものではありません。また本書に記載された情報を使用した事により第三者の知的所有権等の権利に関わる問題が生じた場合，弊社はその責を負いませんので予めご了承ください。
3 製品及び製品仕様は予告無く変更する場合がありますので，最終的な設計，ご購入，ご使用に際しましては，事前に最新の製品規格または仕様書をお求めになりご確認ください。
4 弊社は品質・信頼性の向上に努めておりますが，宇宙，航空，原子力，燃焼制御，運輸，交通，各種安全装置， ライフサポート関連の医療機器等のように，特別な品質・信頼性が要求され，その故障や誤動作が直接人命を脅かしたり，人体に危害を及ぼす恐れのある用途にご使用をお考えのお客様は，事前に弊社営業担当迄ご相談をお願い致します。
5 設計に際しては，特に最大定格，動作電源電圧範囲，放熱特性，実装条件及びその他諸条件につきましては，弊社保証範囲内でご使用いただきますようお願い致します。
保証値を越えてご使用された場合の故障及び事故につきましては，弊社はその責を負いません。
また保証値内のご使用であっても半導体製品について通常予測される故障発生率，故障モードをご考慮の上，弊社製品の動作が原因でご使用機器が人身事故，火災事故，その他の拡大損害を生じないようにフェールセーフ等のシステム上の対策を講じて頂きますようお願い致します。
6 本製品は耐放射線設計をしておりません。
7 本書の一部または全部を弊社の文書による承認なしに転載または複製することを堅くお断り致します。
8 本書をはじめ弊社半導体についてのお問い合わせ，ご相談は弊社営業担当迄お願い致します。

はじめに

HD64413A（Q2SD:Quick 2D Graphics Renderer with Synchronous DRAM Interface）は、SuperH™*で行っていたソフトウェアによる描画の一部をハードウェアに置き換え、描画処理にて多種多様な表現を行います。

また、HD64413A（Q2SD）は SuperH のチップセットであり、Q シリーズ（Quick シリーズ）の第２弾の製品である HD64412（Q2i）の描画コマンド、背景画面表示機能、描画の中断・再開機能を継承し、さらにビデオ取り込み機能を追加した製品です。

本アプリケーションノートでは、ソフトウェア設計を行う際の情報として、SuperH とのインタフェースを行うときの要点、およびサンプルプログラムを掲載しています。

【注】* SuperH **は（株）日立製作所の商標です。**

目次

なお、転載のすべてのプログラム例は、下記の条件のもとで作成しました。

動作環境： SH7709 ソリューションエンジン（MS7709SE01）
Q2SD ドータボード（MS4413DB01） 株式会社 日立超 LSI システムズ

- SuperH の条件
  SH7709(HD6417709)を使用
  動作周波数 ：内部80MHz、外部20MHz
  メインメモリの容量 ：32MバイトDRAM(MH5164805TT6×4個)
- Q シリーズの条件
  Q2SD (HD644113A)を使用
  CLK0の周波数 ：66MHz
  CLK1の周波数 ：7.15909MHz(sample7.cのみ、14.31818を使用)
  表示サイズ ：640×240画素 (sample7.cのみ、640×480画素)
  スキャンモード ：ノンインタレースモード

（sample7.cのみ、インタレースシンク&ビデオモードを使用）

表現色　　：65536色(16bit/pixel)
UGMの容量　　：32MバイトSDRAM(μPD4564323G5×1個)
NTSCエンコーダ　　：CXA2075M

- サンプルプログラム作成環境

プログラム言語　　：C言語
コンパイラ　　：株式会社 日立製作所 SHシリーズC コンパイラ
SH SERIES C Compiler Ver. 5.0
リンケージエディタ　　：株式会社 日立製作所 SHシリーズリンケージエディタ
H SERIES LINKAGE EDITOR Ver. 5.3
オブジェクトコンバータ：株式会社 日立製作所 Hシリーズオブジェクトコンバータ
H SERIES SYSROF STYPE OBJECT CONVERTER Ver. 1.5B

1. システム構成例

HD64413A は、Super H のチップセットで、SDRAM と共にダイレクトに接続し、システムを構成することが可能です。

また、表示用のドットクロック CLK1 信号および、HD64413A の動作クロック（MCLK）は、お互いの周波数が MCLK≧2×CLK1 の関係が成立する範囲で、非同期のクロックを使用することができます。

表示サイズは、CLK1 に入力可能な最大クロック周波数で決定されます。例えば、HD64413A がノンインタレースモードで動作する時の表示サイズは、320×240、480×240 ドット程度になり、インタレースシンク&ビデオモードでは、640×480 ドット程度になります。

HD64413A を TV 同期モードにし、外部のデバイスから、HSYNC, VSYNC, ODDF および CLK1 を HD64413A に供給することで、外部の映像信号と表示合成を行うことが可能です。

また、ビデオ信号をデジタルエンコードしてビデオ取り込みを行うことでビデオ表示ができます。

HD64413A のシステム構成例を**図** 1.1 に示します。

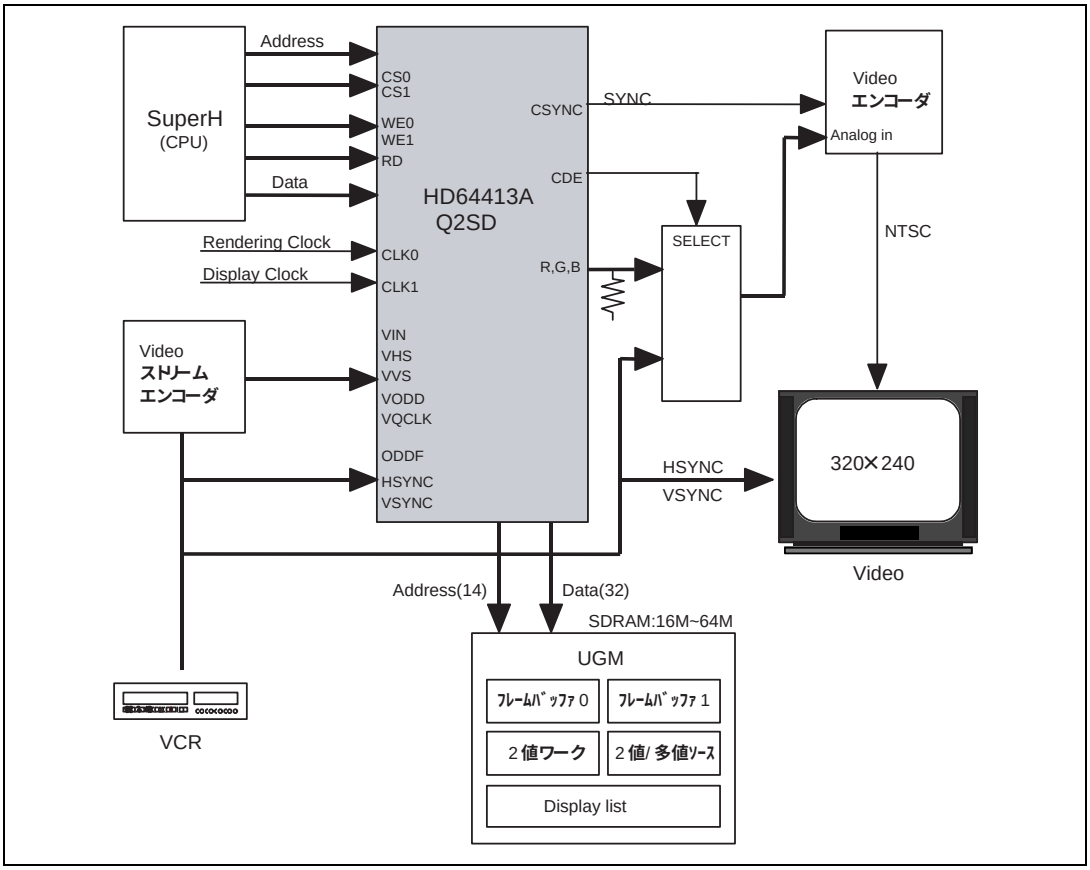


図 1.1　**システム構成例**

1. システム構成例

2. SuperH とのインタフェース

2.1 クロックの決定

HD64413A に供給するクロックには、CLK1 端子に入力するクロックと、CLK0 端子に入力するクロックがあります。CLK1 端子に入力されるクロックは、表示制御用のクロックとして使用し、CLK0 端子に入力されるクロックは、動作クロックとして使用します。

（1） CLK0端子に使用可能なクロックの種類として、以下の（a）～（b）に示すいずれかのクロックが使用可能です。

（a） SuperHのCKIO端子から出力されるクロックを使用する方法
CPUに3.3V動作のSuperH（SH-3、SH-4）を使用する場合は、CKIO端子から出力されるクロックをCLK0端子の入力クロックとして使用できます。
また、CKIO端子のファンアウトを増すためにも、CKIO端子の出力クロックをバッファ回路を経由させてからHD64413AのCLK0端子に入力させてください。

（b） CPUのCKIO端子から出力されるクロック以外のクロックを使用する方法
3.3Vレベルのクロックを CLK0端子の入力クロックとして使用できます。

（2） CLK1端子に入力するクロックは、下記の条件を満足するクロックを入力してください。

MCLK[Hz]≧2×CLK1 [Hz]（CLK1≦33.3MHz）
MCLK＝N×CLK0（Nは逓倍数、1、2、4のいずれか）

2.2 ソフトウェアウェイトの設定

Super H のソフトウェアウェイトサイクルは、SuperH の外部バス動作周波数（CKIO）と HD64413A の内部動作周波数（MCLK）の関係で決まります。

HD64413A が出力する WAIT 信号を SuperH が見つけられるように、SuperH と HD64413A の AC タイミングの両方を考慮し、ソフトウェアウェイトサイクル数を設定してください。

ここでは、SH-3 を使用し、CKIO＝20MHz、MCLK＝66MHz で使用する場合の例を挙げます。**図** 2.1 に示すように、SuperH のソフトウエアウェイトサイクル（Tw）を 2 サイクルにすることで、SuperH の WAIT 端子の規定である tWTS および tWTH の規定を守れるようになり、SuperH と HD64413A 間のハードウェアサイクル（Twx）を確定できるようになります。（tWAS1＝3tcyc0 + 15 ns（MAX））

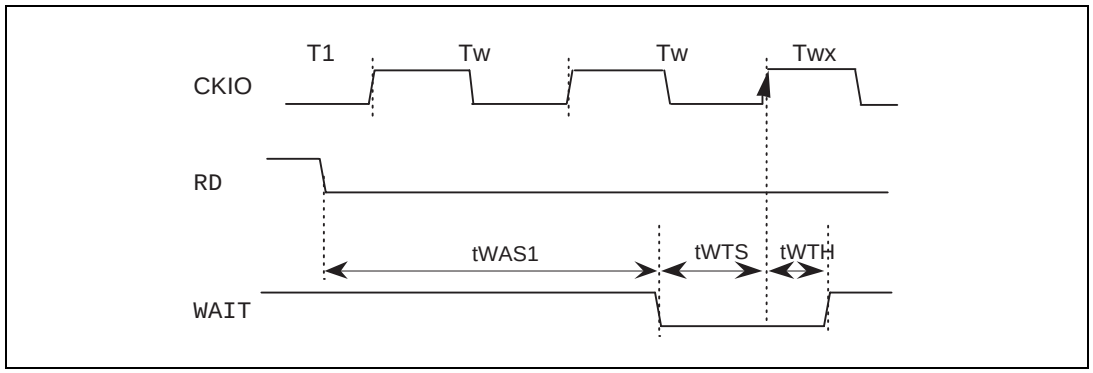


図 2.1 **インタフェースタイミング例**

2.3 接続時の注意事項

HD64413A に SuperH を接続する際には、下記のことに注意してください。

（1） SuperHのCS端子の初期値が入力ポートであり、この端子からHD64413AのCS0、CS1端子に接続する信号を生成する場合、ハードウェアリセット解除後の電圧レベルが不安定にならないようにSuperHのCS端子をプルアップしてください。

（2） SuperHに内蔵されているDMACを使用する場合、初期値の設定でDACK端子がアクティブハイの場合には、外部回路にてDACK端子の信号を反転したものを、HD64413AのDACK端子に接続してください。なお、このときのDACK端子は、アクティブハイのままで使用してください。

（3） SuperHにSH-4を使用する場合、HD64413AのWAIT端子から出力される信号を外部回路にて反転し、それをSH-4のRDY端子に入力させてください。

2.4 アドレスマップドレジスタの初期化手順

HD64413A のアドレスマップドレジスタに初期値を設定する際の標準的な設定手順について説明します。（1）～（4）の順番に設定を行ってください。

（1） システム制御レジスタにSRES＝0、DRES＝1、DEN＝0を設定し、表示同期動作を停止させます。また、この値を設定してから、表示同期動作を開始するまでの間に、SuperHおよびDMACにてのUGMをアクセスを行わせないようにしてください。
（2） レジスタアドレスH’002～H’025および02B間のレジスタに初期値設定します。
特に、02Bの各ビットの初期値によっては、それらビットに関連したレジスタにも、初期値を設定する必要があります。詳細は、**「**HD64413A Q2SD**ユーザーズマニュアル」**をご参照ください。
（3） GBM2～0の組み合わせで、８ビット／画素の表示面を表示さる場合やカーソル表示を行う場合には、カラーパレットレジスタに初期値を設定してください。
（4） システム制御レジスタにSRES＝0、DRES＝0を設定し、表示同期動作を開始させます。
この設定を行うことで、初めて、SuperHはUGMにアクセスを行うことが可能になります。
なお、HD64413Aが描画した図形を確認できるようにするために、通常、システム制御レジスタのDBMには、オート・レンダリングモードまたは、マニュアル・ディスプレイ・チェンジモードを指定します。

アドレスマップドレジスタの初期化を行う例をサンプルプログラム内の ginit 関数に示します。サンプルプログラムについては、**「**5. **サンプルプログラム集」**をご参照ください。

2.5 メモリ割り当て

2.5.1 HD64413A のメモリマッピング

HD64413A のアドレスマップドレジスタおよび UGM は、SuperH のメモリ空間のキャッシュスルー空間にマッピングします。**図** 2.2 に UGM として 64M ビットのシンクロナス DRAM を使用した時のメモリマップ例を示します。また、HD64413A の A22～A1 端子には、HD64413A の UGM のアドレスを直接入力させる必要があります。この例の場合は、A1 から A22 を、UGM のアドレスを直接示すためのアドレス信号として用いるようにします。例えば、SuperH にて UGM の H'000000 番地にアクセスを行う場合には、HD64413A の A22～A1 端子すべてを"０"にしてください。

図 2.2 では、SuperH が UGM をアクセスした際に、キャッシュ・スルー空間をアクセスするように H’A8000000 から UGM を配置しています。

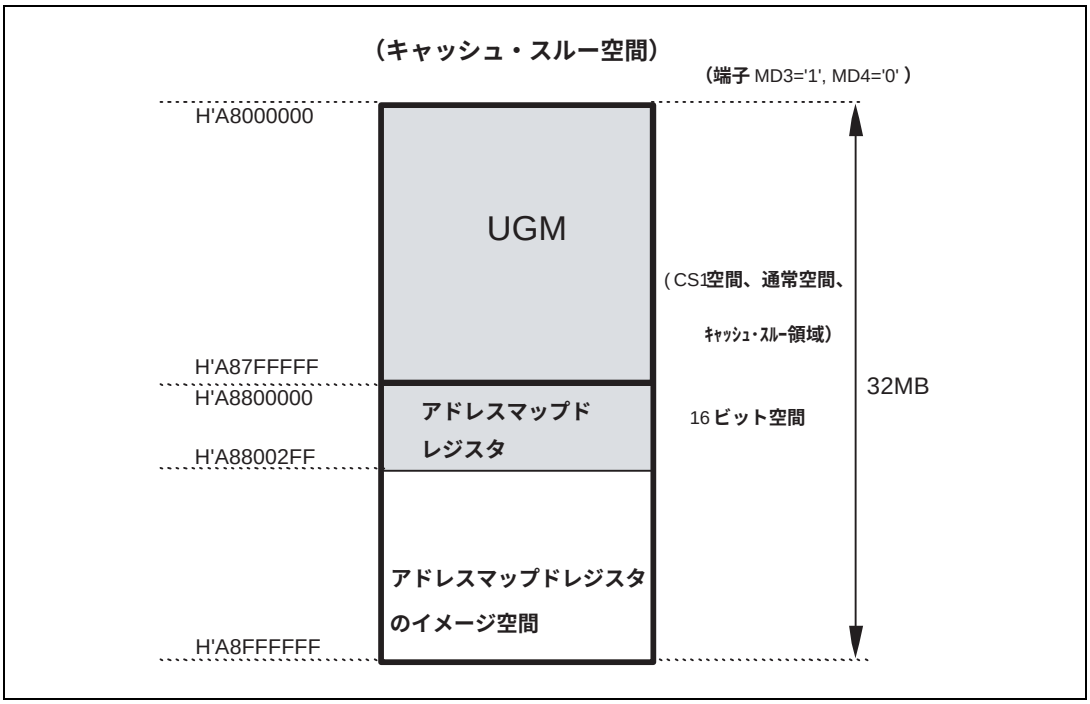


図 2.2 **メモリマッピング例（**SH7709 **を使用）**

2.5.2 UGM における領域の配置例

UGM 内の領域配置の例を**図** 2.3 に示します。なお、表示サイズは 640×240（最大 640×480）ドットとします。

(1) フレームバッファ領域（FB0、FB1）

ダブルバッファ制御時に、表示領域および描画領域（レンダリング座標）として使用するための領域です。

これらの領域の表示アドレス（DSA0、DSA1）には、Y 軸に接する 256 ドットおきの位置に相当する UGM アドレスを設定してください。

(2) ビデオ格納領域（V0、V1、V2）

ビデオ取り込み機能を使用する際に、取り込んだデータストリームを格納するための領域です。これらの領域は、VVS 端子に同期信号が入力される毎に V0、V1、V2 の順番に使用されます。ここでは取り込みサイズを 320×240 画素としています。

これらの領域の表示アドレス（VSAR0～2）には、UGM を 16 ビット/画素と見たときに、Y 軸に方向に 16 ドットおきで、かつ、X 軸方向に 32 ドットおきの位置に相当する UGM アドレスを設定してください。

なお、ビデオ取り込み機能やビデオウインドウを表示しない場合、本領域は使用しませんので不要となります。

(3) ワーク領域（BWAREA）

ワーク座標として使用するための領域です。ワーク座標の X 軸の最大画素は、レンダリング・モードレジスタの MWX ビットで指定した画素数になります。このため、ワーク座標として必要なメモリ容量は、レンダリング・モードレジスタの GBM ビットに関係なく、（MWX ビットで指定した画素数）×（Y 軸方向の表示画素数）/8[Bytes]になります。ワーク領域アドレス（WASH、WASL）には、Y 軸に接する 16 ドットおきの位置に相当する UGM アドレスを設定してください。

なお、多角形などの任意形状パターンでの描画を行わない場合、本領域は使用しませんので不要となります。

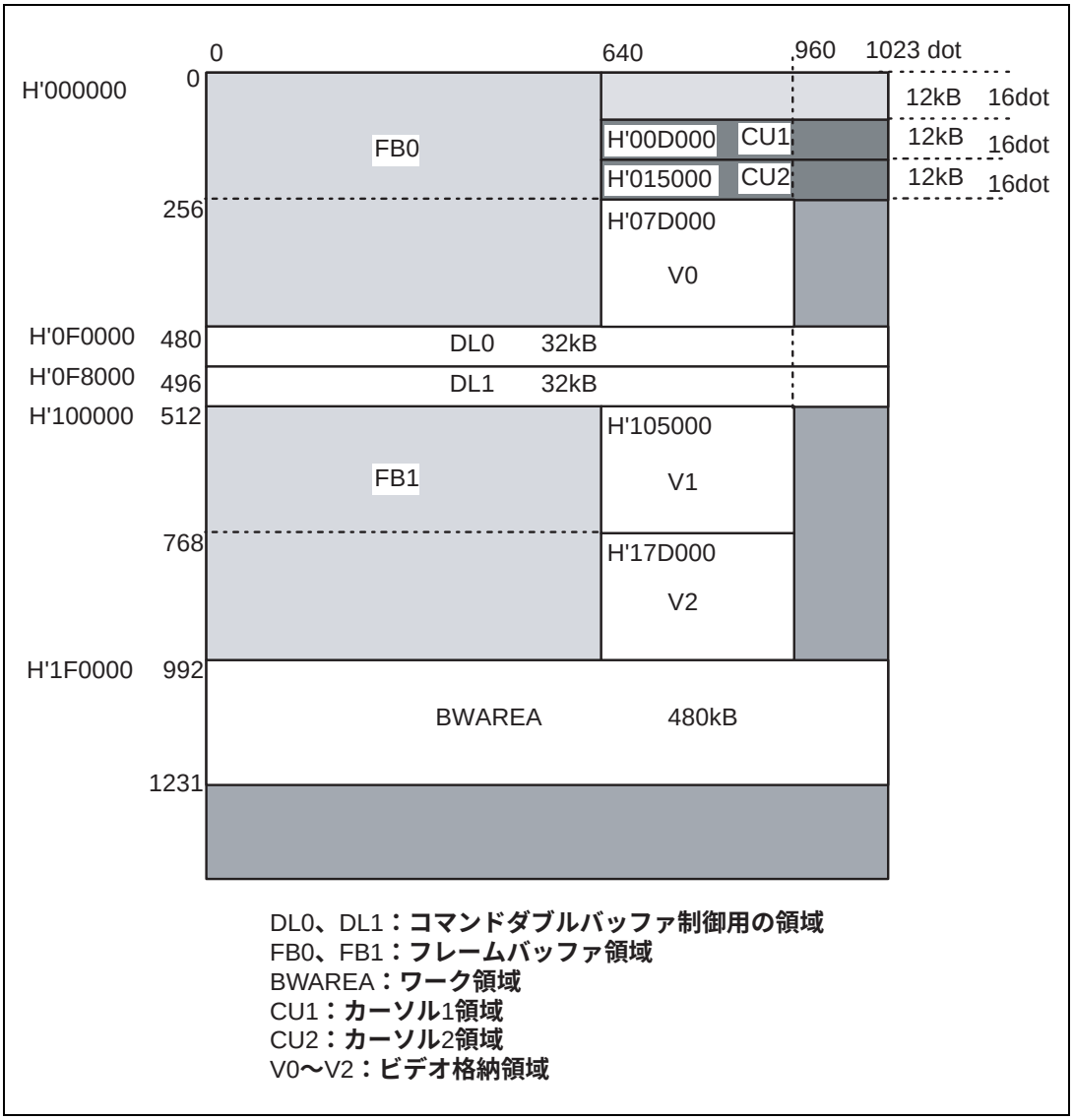
(4) ディスプレイリスト領域（DL0、DL1）

ディスプレイリストを格納するための領域です。DL0 と DL1 の片方の領域を HD64413A がディスプレイリストをフェッチするためのリード領域、もう片方を SuperH がディスプレイリストを置くためのライト領域として使用します。DL0 と DL1 はソフトウェア制御で交互に使用します。ディスプレイリスト開始アドレス（DLSAH、DLSAL）には、任意のワード（16 ビット）アドレスで指定可能です。

(5) カーソル１、２領域（CU1、CU2）

カーソルの形状パターンを格納するための領域です。HD64413A では２個のカーソルを表示させることが可能で、それぞれの形状を CU1、CU2 におのおの格納してください。また、カーソル自身は、8 ビット/画素で表示されますので、必ずカラーパレットにカーソルの表示色を設定してください。

なお、CU1、CU2 共に、使用されるメモリ容量は 2kB です。



DL0、DL1：**コマンドダブルバッファ制御用の領域**
FB0、FB1：**フレームバッファ領域**
BWAREA：**ワーク領域**
CU1：**カーソル**1**領域**
CU2：**カーソル**2**領域**
V0〜V2：**ビデオ格納領域**

図 2.3　UGM **メモリマップ**

2.5.3 UGM におけるアドレスの連続性

図 2.4 に示すように、SuperH から UGM を見たとき、UGM のアドレスは、メモリ１単位毎に連続したタイル状のアドレスとして見えます。このため、FB0、FB1 等の領域割り当てで使用しなかったメモリ１単位を複数個使用することで、この空間を、アドレスが連続したメモリ空間として使用できます。

HD64413A の場合、アドレスが連続したメモリ空間に配置可能なものは、２値ソース、多値ソース、カーソルパターンがあり、通常、これらをこの領域に配置させます。

例えば、FB0 の右脇には、X＝640、Y＝0 の位置から（1024-640）画素×16 ライン＝6114 画素、つまり、1 画素＝2B の関係から 12kB の容量を持つアドレスの連続したメモリ空間を獲得できます。ここに CU1 の割り当てなどを行います。

詳細は、**「**HD64413A Q2SD **ユーザーズマニュアル」**の**「**3.2.3 **メモリマップ」**の項目をご参照ください。

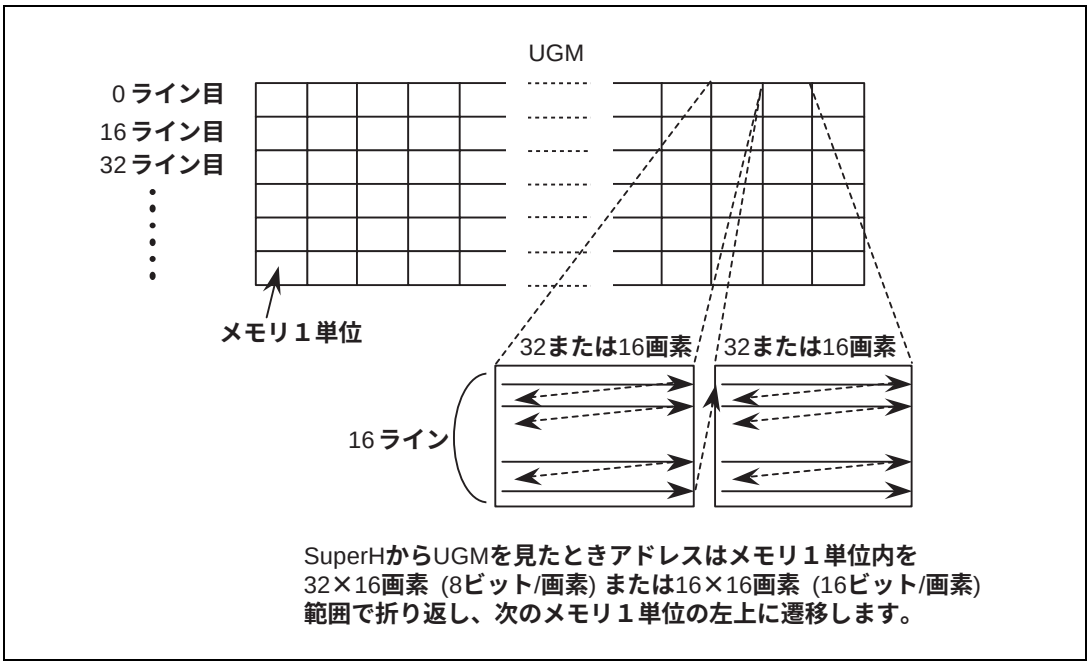


図 2.4 UGM **のアドレス遷移概要**

2.6 UGM へのデータ転送における注意事項

SuperH または DMA コントローラで、２値データやディスプレイリスト等のデータを UGM に転送するには、初めに HD64413A のアドレスマップドレジスタに初期設定を行い、表示同期動作を開始させることで、初めて SuperH と UGM 間のデータ伝送が可能になります。

表示同期動作を行っていない時に、SuperH または DMA コントローラが UGM にアクセスを行うと、データ転送が停止する場合がありますので、表示同期動作を行っていない時の UGM へのアクセスは行わないでください。

なお、UGM にアクセスできるバスマスターは、一つだけです。このため、HD64413A のシステム制御レジスタ内の DMA モードが通常モードのときは、SuperH のみが UGM にアクセス可能です。同様に、DMA モードが DMA 転送モードのときは、DMA コントローラのみが UGM にアクセス可能です。

データの転送は、HD64413A が描画処理を行っている最中であっても、転送可能です。SuperH に内蔵されている DMAC を使用する場合、DMA 転送の終了は、TE（トランスファエンドフラグビット）を確認した後に、必ず、HD64413A のステータスレジスタの DMF フラグをチェックするようにしてください。

3. 表示制御の行い方

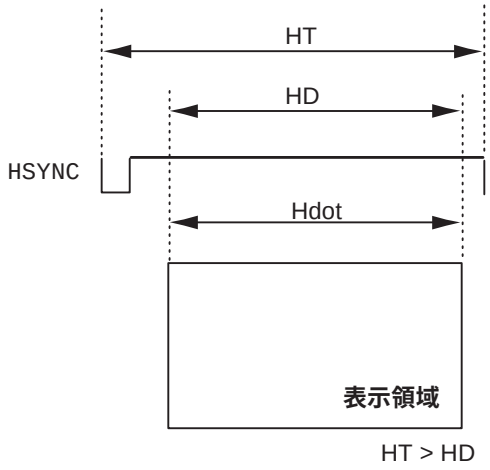
3.1 表示サイズの決め方

水平方向の表示画素数（Hdot）は、下記の式を満たす値である必要があります。例えば、CLK0＝33MHz、N＝1、HD＝44.7μs とすると、Hdot は、737 画素以下にしてください。

また、表示ドットクロック（CLK1）の周波数は　CLK1＝Hdot / HD（Hz）にしてください。

$$\frac{Hdot}{HD} \leqq \frac{CLK0 \times N}{2} = MCLK$$

ただし、
- **NはQ2SDの逓倍数**
- **CLKOはQ2SDのCLK0端子に入力するクロック**(Hz)



HT：HSYNCの周期（秒）
HD：HSYNC内の表示時間（秒）
Hdot：水平方向の表示画素数（画素）

図 3.1　**表示タイミング例**

3.2 表示画面の選択方法

HD64413A は、以下の（1）～（3）の３つの表示面があります。

（1） フレーム面 ：最前面に表示される面です。8または16ビット/画素で表示可能で、主に描画処理での動画を実現する際に使用します。

（2） 背景面 ：最背面に表示される面です。8または16ビット/画素で表示可能で、主に画素単位でのスクロールを実現する際に使用します。

（3） ビデオウインドウ ：フレーム面と背景面の間に表示される面です。ビデオ取り込み機能にて取り込まれたストリームデータを表示する際に使用します。

各表示面の選択は、フレーム面は FBD ビット、背景面は BG ビット、ビデオウインドウは VWE ビットで選択できます。

3.3 同期信号の設定方法

HD64413A では表示制御を行うために、アドレスマップドレジスタに同期信号の設定を行う必要があります。以下に、本アプリケーションノートで使用している同期信号のレジスタ設定例を示します。

(1) TV 同期モードがマスタモード、かつスキャンモードがノンインタレースの時

同期信号の設定例を示します。表示サイズは、320×240 ドットとします。

なお、CLK1＝（水平表示画素）/（xw の時間）（Hz）にしてください。

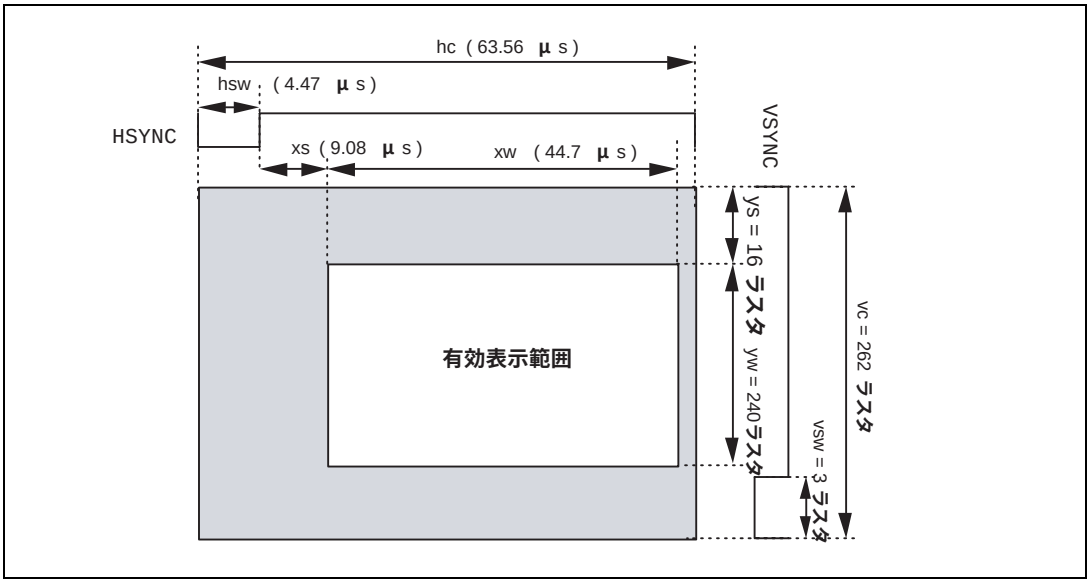


図 3.2 ノンインタレースモード時の表示タイミング例

表 3.1 **変数の設定例（（**TVM1、0）＝（0、0）、（SCM1、0）＝（0、0））

変数名	計算式	表示例での値
hsw	4.47μs×CLK1	32
xs	9.08μs×CLK1	65
xw	44.7μs×CLK1	320
hc	63.56μs×CLK1	455

CLK1＝7.159MHz

表 3.2 **レジスタ設定例（（**TVM1、0）＝（0、0）、（SCM1、0）＝（0、0））

レジスタ名称	計算式（マスターモード）	表示例での設定値
DSX	xw	320
DSY	yw	240
HDS	hsw＋xs－11	68
HDE	hsw＋xs－11＋xw	406
VDS	ys－2	14
VDE	ys－2－yw	254
HSW	hsw－1	31
HC	hc－1	454
VSP	vc－vsw－1	258
VC	vc－1	261

(2) TV 同期モードがマスタモード、かつスキャンモードがインタレース＆ビデオモードの時

同期信号の設定例を示します。表示サイズは、640×480 ドットとします。
また、CLK1＝（水平表示画素）/（xw の時間）（Hz）にしてください。
サンプルプログラムとして「5.2.12　sample7.c のソースファイル」をご参照ください。

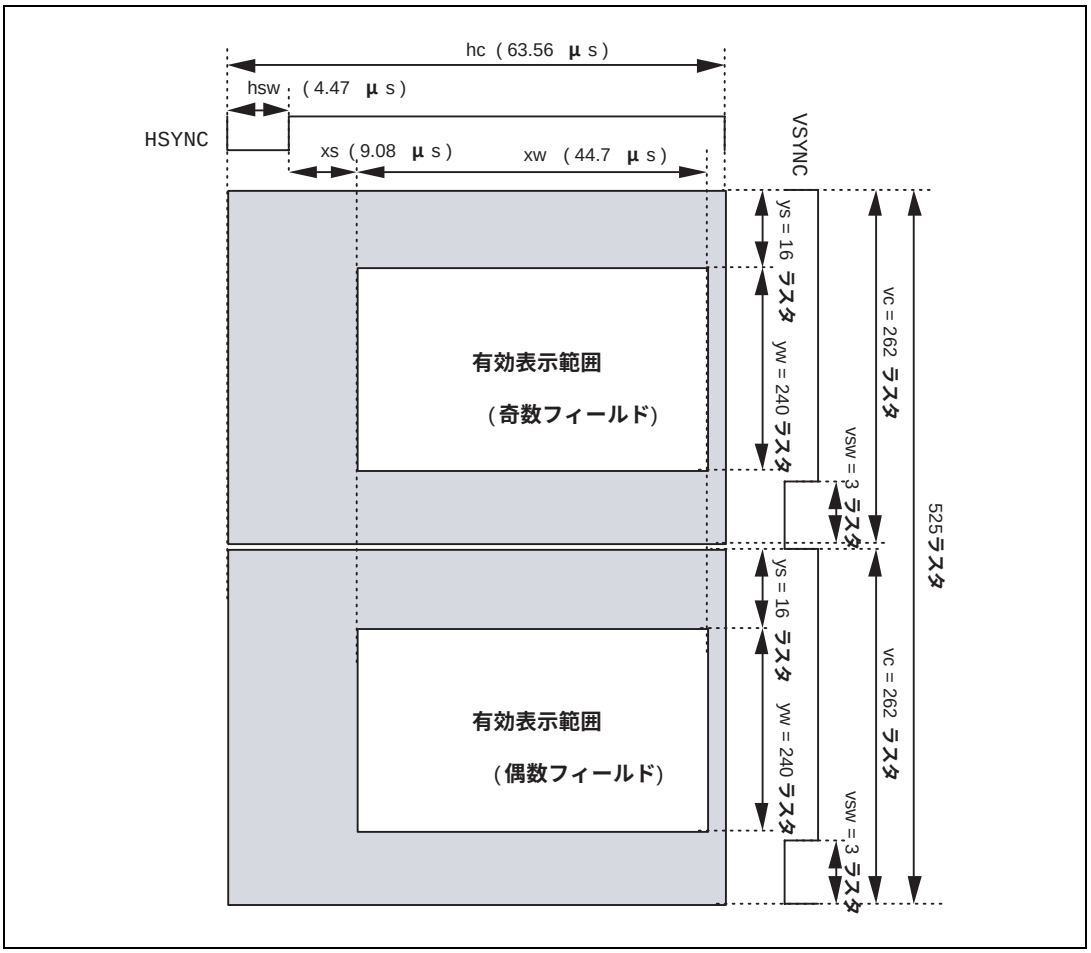


図 3.3　**インタレースシンク＆ビデオモード時の表示タイミング例**

表 3.3 **変数の設定例（（**TVM1、0**）＝（**0、0**）、（**SCM1、0**）＝（**1、1**））**

変数名	計算式	表示例での値
hsw	4.47μs×CLK1	64
xs*	9.08μs×CLK1	131
xw	44.7μs×CLK1	640
hc	63.56μs×CLK1	910

CLK1＝14.318MHz

【注】 * **ビデオエンコーダを使用する場合は、有効表示範囲がカラーバーストに重ならないように、**xs **を決めてください。**

表 3.4 **レジスタ設定例（（**TVM1、0**）＝（**0、0**）、（**SCM1、0**）＝（**1、1**））**

レジスタ名称	計算式（マスターモード）	表示例での設定値
DSX	xw	640
DSY	yw	240
HDS	hsw＋xs－11	184
HDE	hsw＋xs－11＋xw	824
VDS	ys－2	14
VDE	ys－2＋yw	254
HSW	hsw－1	63
HC	hc－1	909
VSP	vc－vsw－1	258
VC	vc－1	261

3.4 表示制御に関連するレジスタ値の設定および変更方法

3.4.1 カラーパレットの設定方法

HD64413A のカラーパレットは、２ワード連続アクセスで、カラーパレットのライトまたは、カラーパレットのリードを行う仕様になっております。このため、カラーパレットに値を設定するときは、R を含むレジスタに続けて、必ず、G および B を含むレジスタを設定してください。

また、同様に、カラーパレットから値を読みとるときも、R を含むレジスタに続けて、必ず、G および B を含むレジスタを読み出してください。

3.4.2 同期モードの移行手順

マスタモードから TV 同期モード等へ同期モードの変更は、同期方式切り換えモードを経由して行います。同期方式切り換えモードへの移行は、TVM1＝0、TVM0＝1 を設定することで行えます。

また、同期方式切り換えモードの時、HD64413A は、UGM にリフレッシュを行わなくなりますので、DRES＝1、DEN＝0 の設定を行って、HD64413A が UGM をリフレッシュするモードに移行させてから、同期方式切り換えモードに移行させます。以下に手順を示します。

なお、DRES＝1、DEN＝0 が有効になっている間、HD64413A は UGM のリフレッシュを行いますので、SuperH または DMAC による UGM アクセスを行わないでください。

同期方式切り換えモードへの移行手順

（1） BG＝0、VWE＝0、CE1＝0、CE2＝0を設定します。
（2） DRES＝1、DEN＝0を設定します。この設定でUGMに対し、リフレッシュのみ行います。
（3） TVM1＝0、TVM0＝1を設定します。HD64413Aは同期方式切り換えモードへ移行します。

同期方式切り換えモードからの復帰手順

（4） CLK1端子にクロックを入力してください。また、TV同期モード（TVM1＝1、TVM0＝0）に移行する場合は、EXHSYNC、EXVSYNV、ODDF端子にも信号を入力してください。
（5） 表示サイズを変更したい場合は、HD64413Aのアドレスマップドレジスタに値を設定してください。
（6） TVM1＝0、TVM0＝0または、TVM＝1、TVM0＝0の設定により、CLK1端子からの入力クロックが有効になります。さらに、必要に応じて、BG ＝ 1、VWE ＝ 1、CE1＝1、CE2＝1を設定してください。
（7） DRES＝0、DEN＝1を設定します。内部更新が行われると、HD64413Aは表示を開始します。

3.5 ビデオ取り込み機能の行い方

HD64413A は、ビデオ取り込みモードレジスタ（VIMR）の VIE が 1 の時に、UGM 上に用意したビデオ格納領域（V0、V1、V2）を順番に使用して、外部のビデオ・ストリーム・デコーダで生成されたビデオ・ストリーム・データを格納していきます。ビデオ格納領域の垂直方向のサイズ（VSIZEY）は、ODEN1、ODEN0 の設定値に依存します。以下に ODEN1、0 の説明および VSIZEY の計算式を示します。

- ODEN1＝0、ODEN0＝0：
  ビデオ格納領域の開始アドレスを指定するタイミングは、VVS単位になり、VVS単位にデータを取り込みます。
  VSIZEY＝（１回分のVVS信号内に存在する有効ライン数）×（ビデオ取り込み間引き率）
- ODEN1＝0、ODEN0＝1：
  ビデオ格納領域の開始アドレスを指定するタイミングは、２VVS単位になり、偶数および奇数フィールドの両方のデータを取り込みます。
  VSIZEY＝（２回分のVVS信号内に存在する有効ライン数）×（ビデオ取り込み間引き率）
- ODEN1＝1：
  ビデオ格納領域の開始アドレスを指定するタイミングは、２VVS単位になり、偶数または奇数フィールドのどちらか一方のデータを取り込みます。
  VSIZEY＝（１回分のVVS信号内に存在する有効ライン数）×（ビデオ取り込み間引き率）

また、ビデオ格納領域にビデオ・ストリーム・データを格納するときに、HD64413A にて、水平方向に 1、2、3、4 画素、垂直方向に 1、2、3、4、6 画素ごとにビデオ・ストリーム・データ省くことも可能です。

ビデオ取り込み機能を使用する場合には、MCLK を 64MHz 以上にし、UGM のデータバス幅が 32 ビットになるようにしてください。また、ビデオ取り込みを実行する際には、VQCLK 端子のクロックの立ち上がりエッヂにて、VID0～7 端子に入力されるビデオ・ストリーム・データを取り込み、ビデオ格納領域に転送します。本アプリケーションノートでは、ロックウェル社製のビデオ・ストリーム・デコーダ（Bt829）を使用し、ビデオ・ストリーム・データが存在するタイミングでのみ、VQCLK が発生するようにしています。サンプルプログラムとして、**「5.2.18　init_bt.c のソースファイル」**をご参照ください。

以下の（1）または（2）に、格納されたビデオ・ストリーム・データの使用方法を示します。

(1) ビデオウインドウの表示データとして使用する方法

VIE＝1 の場合、表示モードレジスタ２（DSMR2）の VWE ビットが 1 であると、ビデオ格納領域に格納された最新のビデオ・ストリーム・データをリアルタイムに表示します。サンプルプログラムとして、**「5.2.17　v_wind.c のソースファイル」**をご参照ください。

なお、ビデオウインドウを使用してビデオ・ストリーム・データを表示させる場合、下記の（a）または（b）を行ってください。

（a） VMIR中のRGBビットが１の時には、DSMR2中のVWRYに0を設定してください。

（b） VMIR中のRGBビットが0の時には、DSMR2中のVWRYに1を設定してください。

(2) 多値ソースとして利用する方法

VIE＝0 を設定してビデオ取り込みを停止した場合、VIMR の VID1、0 ビットに最新のビデオ・ストリーム・データが格納されたビデオ格納領域が示されます。VMIR 中の RGB ビットが１である状態でビデオ取り込み機能を実行したならば、VID1、0 が指し示しているビデオ格納領域を 16 ビット/画素の多値ソースして参照することが可能です。

なお、VID1、0 は、ビデオ取り込みを行っている最中（VIE＝1）のときは意味を持ちません。VID1,0 を参照する場合には、ビデオ取り込みを停止（VIE＝0）してから参照を行ってください。

3.6 カーソル表示の行い方

HD64413A では、UGM 上の配置された 32×32 画素の大きさのカーソルを 2 個、表示させることができます。1 個のカーソルはカーソルブリンク形状 A とカーソルブリンク形状 B の 2 つの形状をもち、それらは、BLNKA および BLNKB で設定されたタイミングで交互に表示されます。このため、1 個のカーソルあたり、2kB のアドレスが連続した領域が必要です。

UGM 上に 2kB のアドレスが連続した領域を割り当てるには、**図** 3.4 のように、水平方向にメモリ 1 単位を 4 個使用することでこの領域を割り当てることができます。（メモリ 1 単位については、「2.5.3 UGM **におけるアドレスの連続性」**をご参照ください）

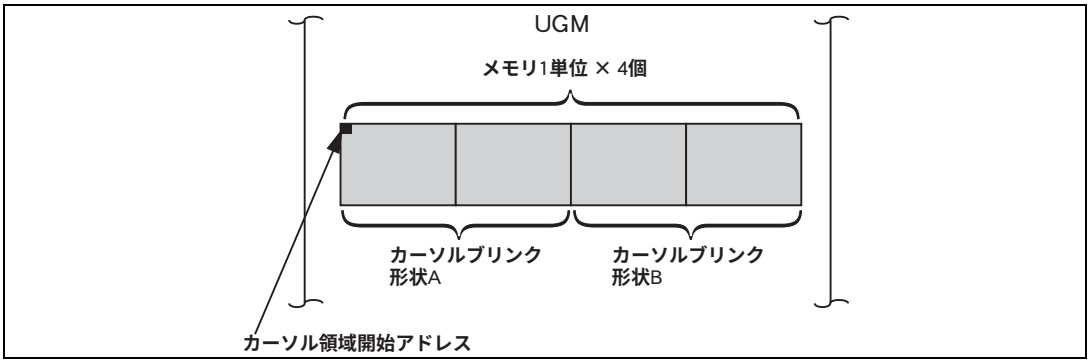


図 3.4 **カーソル割り当て**

HD64413Aがカーソルを表示する際には、カーソル領域開始アドレスレジスタで指定されたアドレスから順番にカーソル形状を読み取り、読み取ったデータにもとづいてカラーパレットを参照して、カラー化され、表示します。

4. 描画の行い方

4.1 描画の開始方法

HD64413A は、ディスプレイリストと呼ばれるコマンドの集まりをもとに、レンダリング座標およびワーク座標に描画を行います。以下に描画を行う際の手順を示します。

（1） SuperHで、LCOFS、SCLIPコマンドをディスプレイリストとして、UGMに配置します。
このディスプレイリストは、HD64413Aのローカルオフセットおよびシステムクリップ範囲の初期値を設定するためのものです。
（2） フレームチェンジタイミングと描画開始タイミングを同期させるために、（1）で配置したディスプレイリストに続けて、SuperHでVBKEMコマンドをUGMに配置します。
（3） HD64413Aに描画を行わせるために、（2）で配置したディスプレイリストに続けて、SuperHで、POLYGON4系コマンド等を使用したディスプレイリストをUGMに配置します。
（4） ディスプレイリストの終了を示すために、（3）で配置したディスプレイリストに続けて、TRAPコマンドを配置します。この時点で、ディスプレイリストの作成は終了です。
（5） レンダリング開始アドレスの設定を行った後、RSビットに１を設定してください。
このレジスタの設定で、HD64413Aに描画を行わせることができます。

POLYGON4 系コマンドを使用したサンプルプログラムとして、**「**5.2.6　sample.c **のソースファイル」**～**「**5.2.8　sample3.c **のソースファイル」**をご参照ください。なお、（1）の項目は、一番最初に生成するディスプレイリストの先頭で行えばよい項目であり、それ以降は、必要に応じて LCOFS、SCLIP コマンドを使用してください。

4.2 フレームチェンジの行い方

フレームチェンジを行うには、以下に示す（a）、（b）の２種類の方法が可能です。

（a） DBMに設定したダブルバッファ制御に従い、フレームチェンジを行う方法。
（b） 内部更新でフレームチェンジを行う方法
この方法は、DBMをマニュアルディスプレイチェンジモード固定にし、SuperHで表示開始アドレスDSA0およびDSA1を管理して、内部更新でフレームチェンジを行わせる方法です。

描画の中断・再開機能を使用する場合には、描画開始アドレスおよび表示開始アドレスを制御できる（b）の方法が有効です。この方法を行うには、初めに、ステータスレジスタ内の DBF ビットを調べ、DSA0、DSA1 のどちらが表示開始アドレスを決定するレジスタになっているのかを判定する必要があります。DBF＝0 の時、DSA0 が表示開始アドレスを決定するレジスタになります。同様に、DBF＝1 の時、DSA1 が表示開始アドレスを決定するレジスタになります。**表** 4.1 に DBF と DSA0, DSA1 の関係を示します。

表 4.1 DBF **と表示面（**FG**）の関係**

	DSA0	DSA1
DBF＝0	**表示面**	**描画面**
DBF＝1	**描画面**	**表示面**

DBF が 0 である場合を例に挙げ、DSA0 と DSA1 の管理手順を（1）～（4）に示します。

（1） 内部更新期間の終了を待ちます。内部更新の終了は、FRMビットをクリアし、その後、FRMビットが１になるのを確認することで行えます。
（2） WPRコマンドにて、DSA0に次の内部更新で表示をさせたい位置の表示開始アドレスを設定します。
DSA0に設定された表示開始アドレスは、すぐに有効な値としては反映されません。
この設定された値は、内部更新を経過することで初めて有効になります。
（3） WPRコマンドにて、RSAEに1を、RSARに描画開始アドレスを設定します。
（4） ディスプレイリストを転送し終えた後に、システム制御レジスタ内のRSビットに1を設定し、描画を開始させます。

以上の（1）から（4）を繰り返すことで、 内部更新により、DSA0 に設定した表示開始アドレスが有効になり、フレームチェンジを行えます。
内部更新によるフレームチェンジを使用した時の描画および表示タイミングを**図** 4.1 に示します。

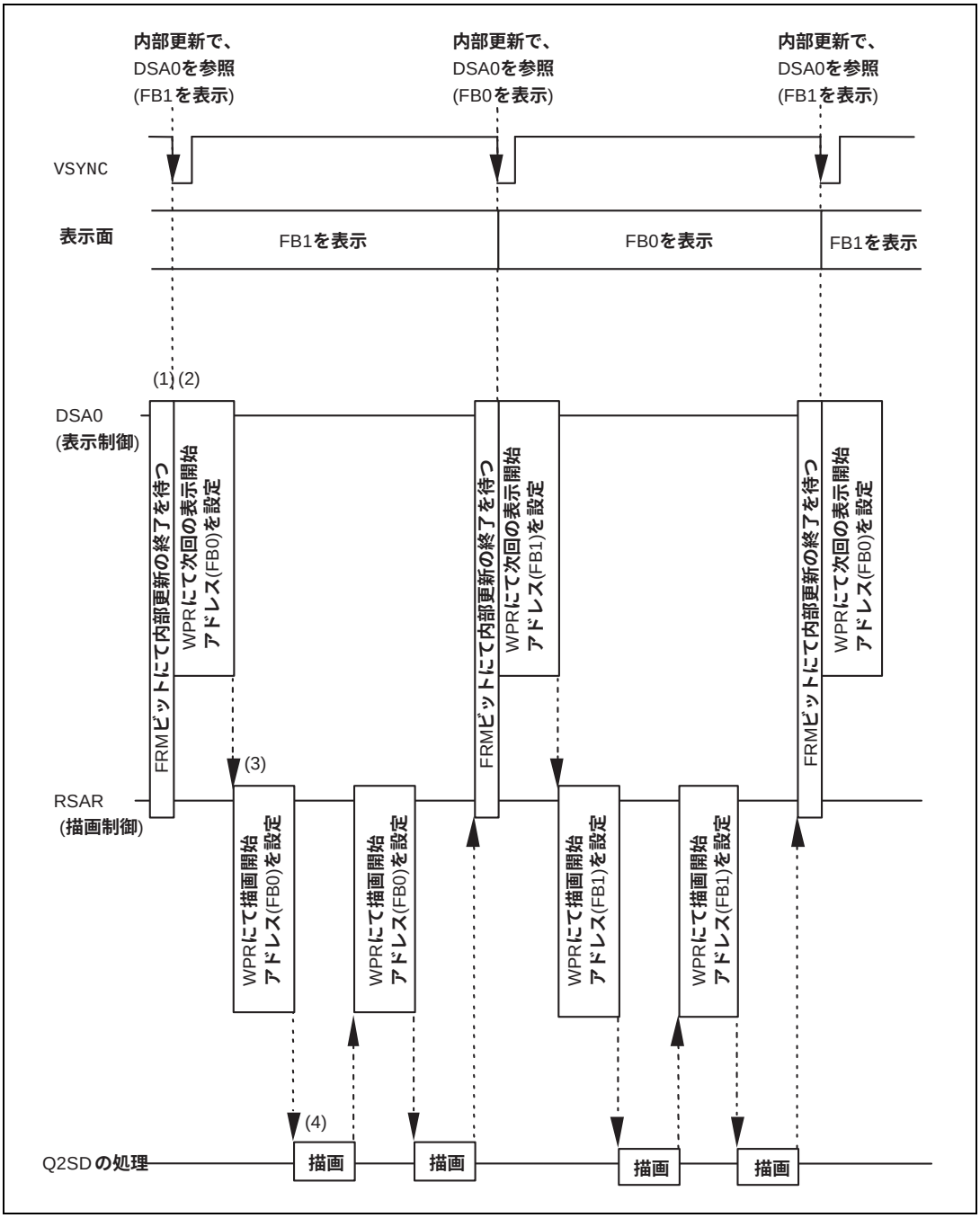


図 4.1　DBF＝0 **時の表示・描画制御タイミングチャート**

4.3 描画コマンドの使用例

4.3.1 多角形の描画

HD64413A で、多角形をレンダリング座標に描画させるには、レンダリング属性の一つであるワーク参照および、ワーク座標を使用することで行えます。
以下に HD64413A が行う描画手順を示します。

（1） ワーク領域をクリアするために、CLRWコマンドを実行します。
（2） FTRAPコマンドで描画を行わせたい多角形の形を、ワーク座標に描画します。
（3） レンダリング属性のWORKビットに１を設定したPOLYGON4Cコマンドを使用し、ワーク座標に描画済みの形状で、多角形を描画します。
（4） LINEコマンドにて、多角形の縁取り線を描画します。

実際は、上記の手順を行わせるためのディスプレイ・リストを SuperH で生成し、生成したディスプレイ・リストをもとに HD64413A が描画を行います。
サンプルプログラムとして、**「5.2.9 sample4.c のソースファイル」**をご参照ください。

4.3.2 任意の形の描画

任意の形のパターンをレンダリング座標に描画させる方法として、下記の二つの方法があります。

- 固定の任意形状パターンを部分参照して、その形状で描画を行う方法
  サンプルプログラムとして、**「5.2.14 sample9.cのソースファイル」**をご参照ください。
  このプログラムでは、ワーク座標に配置された２値の固定されたパターン（１文字が16×16画素サイズの文字を、水平方向に32個、垂直方向に7個配置したパターン）を使用して、この形状を部分的に参照して描画を行った例です。
  なお、ワーク座標に２値パターンを配置する前に、ワーク座標をゼロクリアする必要があります。本サンプルプログラムでは、SuperHとHD64413Aによるワーク座標への描画の競合を避けるために、ワーク座標のゼロクリアは、CLRWコマンドによるゼロクリアではなく、SuperHで直接、ゼロクリアを行うようにします。

- 使用したい任意形状パターンを変えながら描画を行う方法
  サンプルプログラムとして、**「5.2.11 sample6.cのソースファイル」**をご参照ください。
  このプログラムでは、16×16画素の大きさの２値パターンを使用して、描画を行った例です。この２値パターンは、コマンドごとに指定が可能ですので、ディスプレイリストを作成する際に、参照パターンを指定し直すことで、描画させたい形状を変えることができます。

4.3.3 円・楕円の描画

HD64413A で円・楕円の描画を行うには、SuperH で楕円の軌道を算出し、算出した結果をパラメータとして LINE コマンドにて実現します。サンプルプログラムとして、**「**5.2.16　elsp.c **のソースファイル」**をご参照ください。このプログラムでは、Bresenhamの円アルゴリズムを使用して楕円の軌道を算出します。円の描画は楕円の x と y の半径を同じドット数にすることによって実現できます。

ワーク領域に描画する必要がある場合には、_fill_ellipse（）関数中で使用している LINE コマンドを LINEW コマンドに変更してください。

_fill_ellipse（）関数の仕様

プロトタイプ：　**`void _fill_ellipse (short xc, short yc, short rx, short ry, short color);`**

引数：

`xc`	中心点の **x** 座標
`yc`	中心点の **y** 座標
`rx`	**x** 方向の軸長半径
`ry`	**y** 方向の軸長半径
`color`	色コード

戻り値：　なし

4.3.4 ソースデータを使用した描画

HD64413A でソースを参照する描画コマンドを使用する場合、一般的にアプリケーションソフト側でソースデータが UGM に格納されているか判定する必要があります。システムによっては、判定するのが困難であったり、処理が冗長だったりする場合があります。これを回避するための方法を説明します。例としてディスプレイリストにソースデータを含ませることで、描画コマンドとソースの参照位置を関係付ける方法があります。

ソースデータをディスプレイリストに埋め込むには、**図** 4.2 のように描画コマンドの直後に JUMP コマンドを配置し、ソースデータをスキップさせるようなディスプレイリストを配置することで行えます。

多値ソースをディスプレイリストに含ませる場合も、同様な方法で実現可能です。この場合には、レンダリング属性の LNi ビットに 1 を設定した POYLGON4A コマンドを使用してください。

また、POYLGON4B コマンドを使用した場合のサンプルプログラムとして**「**5.2.7　sample2.c **のソースファイル」**をご参照ください。このプログラムでは、**図** 4.2 のような 2 値ソースデータを含んだディスプレイリストを生成し、描画を行います。

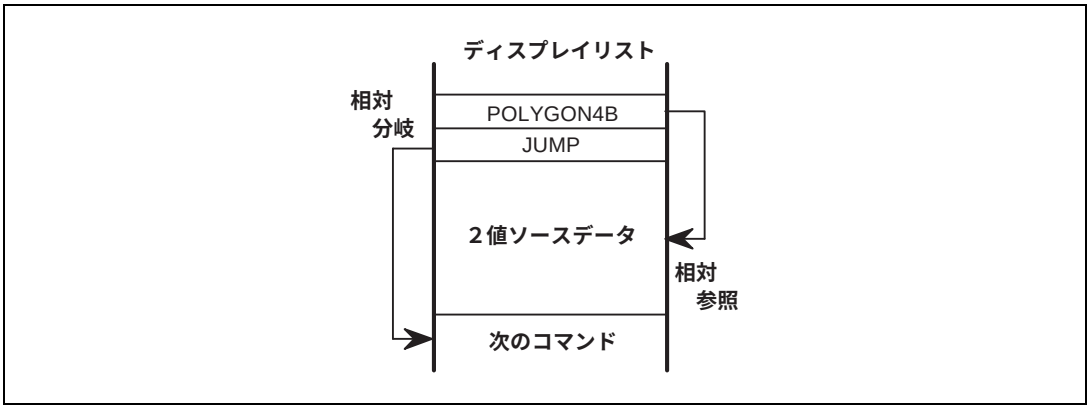


図 4.2　**参照・分岐例**

4.3.5　3 次元空間を表現させる方法

3 次元空間を表現したサンプルプログラムとして、**「**5.2.13　sample8.c **のソースファイル」**をご参照ください。

このプログラムでは、20 面の四角形で構成された立体図形を 2 個用意し、各々を回転させています。立体図形内の各四角形の各 4 頂点は、X、Y および Z の 3 次元の座標値を持ち、それらは、配列変数 coordindex で決定されます。

このプログラムで行っている処理手順を以下に示します。

（1）　set_polygon_sample（）関数で、３次元の座標値で表現される四角形を定義します。
この関数で定義された四角形の集まりが、一つの立体図形になります。

（2）　rotate_a_rectangle（）関数にて、四角形ごとに座標値の回転演算を行わせます。

（3）　z_sort関数にて、四角形を描画する順番を決定します。描画の順番は、奥から手前への順番とします。順番の決め方は、各四角形のZの値の平均値を求め、それをもとに順番を決定します。また、ソート方法はプログラムの記述を簡単にするために、バブルソート法を使用しています。このため、ソートされる要素数をnとすると、おおよそn×n回のソートが行われます。
一般的にヒープソート等の高速なソート方法を使用した場合、おおよそn×log n 回でソートを終了できます。

（4）　convert_d3_into_d2（）関数で、四角形ごとに３次元の座標値を、2次元の座標値に変換します。
このときの変換は、各々の頂点のZの値が、変数 _z_sizeで示されるZ軸の奥行き内のどの位置に存在するのかを、Zの値と_z_sizeの比で示し、それを各々の4頂点に乗算させることで変換を行っています。

（5）　（3）で得た順番および、（4）で得た2次元の座標値を使用して、POLYGON4Cコマンドをディスプレイリストとしてを生成します。

以上の手順が終了すると、UGM 上にディスプレイリストが生成されますので、HD64413A に描画を行わせた結果、3 次元空間を表現できます。

4.4 描画コマンドを使用する際の注意事項

4.4.1 ローカルオフセットとカレントポインタの関係に関する注意

ローカルオフセット、およびカレントポインタは、コマンドの実行される順番によって、それぞれの値が決まります。このため、ローカルオフセットとカレントポインタの関係を考慮しながら、コマンドを配置してください。以下にコマンドの配置の優先度を示します。優先度が小さいほど、先に配置すべき描画コマンドとなります。

1. lcofsコマンド ：ローカルオフセットの初期値を設定するコマンド
2. rlcofsコマンド ：現在のローカルオフセットに対して、相対値でローカルオフセットを移動するコマンド
3. moveコマンド ：現在のローカルオフセットを加算したカレントポインタが設定されるコマンド
4. rmoveコマンド ：現在のカレントポインタに対して相対値でカレントポインタを移動するコマンド

4.4.2 相対系コマンドを使用する際の注意

相対座標で座標のパラメータを管理するコマンドを相対系コマンドといいます。この相対系コマンドを使用する時は、前もって、move コマンド等でカレントポインタを生成する必要があります。また、相対系コマンド以外のコマンドは、カレントポインタを演算用のレジスタとして使用し、カレントポインタを破壊します。このため、相対系コマンドでかつ、描画を行うコマンドを使用する際には、相対系コマンド間に、それ以外のコマンドを挿入しないでください。

4.4.3 ソースデータを使用する際の注意

HD64413A が UGM に配置された 2 値・多値ソースを使用する場合、HD64413A 内部に存在するソースバッファにソースデータを取り込み、蓄積されたソースデータを使用して描画を行います。このソースバッファは 16 ワードの容量をもち、HD64413A は、UGM のアドレスが 32 バイトの境界を超えるごとに 32 バイトづつソースバッファにデータを格納します。このため、2 値・多値ソースを使用する際には、ソースバッファの更新を起こさせるように考慮しながら、HD64413A に描画を行わせる必要があります。また、レンダリング属性の STYL ビットによってソースバッファの更新の行われ方が決まります。（1）および（2）に説明します。

(1) レンダリング属性の STYL を 0 に設定した場合

STYL＝0 の場合、ソースの容量が 32 バイト以内の時に、ソースバッファの更新を起こさせるように考慮する必要があります。以下の方法が考えられます。

(a) コマンド毎に異なるソースアドレスを指定する。

例えば、POLYGON4B コマンドで 32 バイト以内の 2 値ソースを参照させたい場合、POLYGON4B コマンドのパラメータである SOURCE ADDRESSH および SOURCE ADDRESSL をコマンドごとに異なるアドレスを指定する方法があります。

(b) 透過指定を使用する。

32 バイトを超える 2 値ソースを用意し、透過指定を有効にした描画コマンドで、描画の際に、必要な部分の 2 値ソースのみを描画させる方法があります。

(2) レンダリング属性の STYL を 1 に設定した場合

STYL＝1 の場合、ソースの繰り返し参照を行います。このため、ソースの参照開始アドレスから数えて、32 バイト以内のアドレスでソースの参照が終了した場合に、ソースバッファの更新を起こさせるように考慮する必要があります。以下の方法が考えられます。

(a) コマンド毎に異なるソースアドレスを指定する。

例えば、POLYGON4B コマンドで 32 バイト以内の 2 値ソースを参照させたい場合、POLYGON4B コマンドのパラメータである SOURCE ADDRESSH および SOURCE ADDRESSL をコマンドごとに異なるアドレスを指定する方法があります。

4.5 描画処理を支援する機能

4.5.1 描画の中断・再開

描画の中断・再開は、HD64412（Q2i）以降からサポートされた描画機能を支援するための機能です。この機能は、背景面（BG）描画中にフレームバッファ（FB）に対して描画処理を行う場合や、強制的に描画処理を割り込ませる場合などに使用します。描画の中断・再開の使用方法を以下に説明します。

なお、本機能は、システム制御レジスタの DBM ビットに 10 を設定して、ダブルバッファ制御をマニュアルディスプレイチェンジに固定した状態でのみ使用可能です。

(1) 描画の中断

「描画の中断」は、現在行っている描画を中断させるための方法です。描画の中断は、システム制御レジスタ（SYSR）の RBRK ビットに１を設定することで行えます。SuperH で RBRK ビットに 1 を設定すると、HD64413A は、現在実行している描画コマンドの処理が終了した後の次のコマンドの先頭で、LSI 内部のレジスタに設定された値（カレントポインタ、ローカルオフセット、クリッピング範囲、GOSUB コマンドのリターンアドレス）をレンダリング制御レジスタ２に設定し、描画処理を中断します。

また、SuperH で描画の中断を判定するには、 RBRK ビットに１を設定後に、TRA ビットおよび BRK ビットを読み出してください。TRA ビットが１である場合は、RBRK による中断ではなく TRAP コマンド実行による描画の終了であるため、それ以降の描画の再開は行わないようにしてください。また 、BRK ビットが 1 である場合は、描画の中断が行われたことを意味します。このため、BRK が１になったことを確認することで、描画の中断を判定できます。

BRK ビットが 1 になったら、レンダリング制御レジスタ２、レンダリングモードレジスタの描画スタートアドレスイネーブル（RSAE）、描画スタートアドレスレジスタ（RSAR）、および、コマンドステータスレジスタ（CSTR）に設定されている値を SuperH のソフトウェア処理にてリードし、SuperH のメモリ上に待避してください。待避した値は、中断した描画を再開する時に使用します。

その後、中断中に描画させたいディスプレイリストを生成し、実行してください。

(2) 描画の再開

「描画の再開」は、「描画の中断」で中断させた描画を再開させるための方法です。描画の再開時の処理は、SuperH にて描画の再開を行うためのディスプレイリストを UGM に配置し、このディスプレイリストへの描画開始を行い（システム制御レジスタの RS ビットに 1 を設定）、RS ビットが 0 に戻ることを確認することで行えます。描画の再開を行うためのディスプレイリストの構成を以下の（1）から（8）に示します。

◆ 描画の再開時に使用するディスプレイリストのコマンドの並び順（（1）から（8）の順）

（1） WPRコマンド（描画の中断時に待避した描画スタートアドレスレジスタの値を、描画開始アドレスレジスタ（RSAR）に設定する）
（2） WPRコマンド（描画の中断時に待避した描画スタートアドレスイネーブルの値を、描画スタートアドレスイネーブル（RSAE）に設定する）
（3） WPRコマンド（描画の中断時に待避したGOSUBコマンドのリターンアドレスを、リターンアドレスレジスタ（RTNR）に設定する）
（4） UCLIPコマンド（描画の中断時に待避したUCLIPの値を復帰させる）
（5） SCLIPコマンド（描画の中断時に待避したSCLIPの値を復帰させる）
（6） LCOFSコマンド（描画の中断時に待避したローカルオフセットの値を復帰させる）
（7） MOVEコマンド（描画の中断時に待避したカレントポインタの値を復帰させる）

（8） JUMPコマンド（描画の中断時に待避したコマンドステータスレジスタの値を復帰させる）

描画の中断・再開を行うサンプルプログラムとして、**「5.2.15　tst_brk.c のソースファイル」**をご参照ください。

このプログラムでは、初めに 256 個の矩形を描画するためのディスプレイリストを UGM に配置し、HD64413A がこのディスプレイリストに基づいて描画処理を行っている最中に「描画の中断」を行って描画処理を中断させています。

描画処理を中断させた後の割り込みの描画処理として、POLYGON4C コマンドにて画面全体を塗りつぶしています。

さらに、「描画の再開」で、先ほど中断した矩形描画の続きを行うようにしています。

割り込み描画の処理を行う時も、描画の再開と同様に、RS ビットに 1 を設定した後で、RS ビットが 0 に戻ることを確認してください。

このプログラム上での描画の中断・再開は、五つの処理から成り立っています。以下の（1）から（5）に、それらの処理の概要を示します。

（1） 内部更新によるフレームチェンジ（表示開始位置の更新）
（2） 描画の中断
（3） 割り込み描画処理（POYLGON4Cコマンドによる塗りつぶし）
（4） 矩形描画の再開
（5） フレームチェンジを行うためのレジスタ設定

5. サンプルプログラム集

以下にサンプルプログラムのリストを示します。sample.c から sample3.c は POLYGON4 系コマンドの基本機能を使用したサンプルプログラムで、SuperH で計算した矩形の４頂点座標を元に、矩形の拡大、縮小および回転を行います。なお、本サンプルプログラムでは、日立超 LSI システムズ製 SolutionEngine（SH7709 搭載）および HD64413A（Q2SD）ドータボード上で作成されたものです。このため、SuperH のメインメモリおよび Q2SD のマッピングアドレスは、以下のアドレスを前提とします。

SuperH のメインメモリ	:H’C000000～H’C3FFFFF
Q2SD の UGM	:H’B4000000～
Q2SD のアドレスマップドレジスタ	:H’B4800000～

また、サンプルプログラムでは、おのおので、表示サイズが 640×240 画素になるように Q2SD のアドレスマップドレジスタにて表示サイズを設定しています。（ただし、sample7.c のみ、640×480dot）

（ヘッダー、ライブラリ）

Q2SD_REG.h : Q2SD のレジスタアドレスの定義。

Q2SD_mac.h : サンプルプログラムで使用した Q2SD のコマンドのマクロ関数のサンプル。

Q2SDL.inc : コマンドダブルバッファ制御のソースライブラリ。

MS7709.inc : SH7709 が Q2SD へアクセスするためのバスステートコントローラの設定。

（基本プログラム例）

sample.c : POLYGON4C を使用したサンプルプログラム。
単色の四角形を拡大、縮小、回転、移動をさせる。

sample2.c : POLYGON4B を使用したサンプルプログラム。
24×24 ドットサイズの文字を拡大、縮小、回転、移動させる。

sample3.c : POLYGON4A を使用したサンプルプログラム。
カラーパターン（多い値）を拡大、縮小、回転させる。

（応用プログラム例）

sample4.c : 多角形描画を行うサンプルプログラム
サンプルプログラムでは、6 角形を描画させる。
NET 指定をおこなっているので、NTSC エンコーダ回路の出力タイミングによっては、表示色が異なる場合があります。

sample5.c : WPR コマンドにて描画開始アドレスを変更しながら描画を行うサンプルプログラム

sample6.c : POLYGON4A を使用し、100 個の移動速度が異なるボールを移動させるサンプルプログラム。なお、MSAREA 内に 16×16 画素の 10 個のボール状の多値パターンが配置されていることを前提とします。

sample7.c : インタレースシンク＆ビデオモードで動作させた時のサンプルプログラム（表示サイズは 640×480 画素）

sample8.c : 2 個の立体図形を回転させるサンプルプログラム。1 個の立体図形は、20 面で構成される。

sample9.c : 2 値ワーク座標に配置した 2 値パターンを使用して、描画を行ったサンプルプログラム。

tst_brk.c : 描画処理を一時的に中断させた後、画面クリアを行い、さらに、先ほど中断させた描画処理の続きを行うサンプルプログラムです。
なお、制御方式の関係から、下記の関数は使用しておりません。

```
draw_start ()
draw_end (DBmode,&first,quick);
```

elps.c : 楕円描画を行うサンプルプログラム。

v_wind.c : ビデオ取り込み機能にて取り込んだビデオデータを表示するサンプルプログラムです。
ビデオ取り込みを行う際には、フィールド単位で取り込みながら、水平方向のみを半分に減らし、320×240 画素のビデオデータを UGM に格納しています。
なお事前に、INIT_BT.C から生成されえるオブジェクトプログラムを実行させ、1 フィールドあたりに 640×240 画素のビデオデータが発生するようにしてください。

Init_bt.c : Q2SD ドータボードに実装されているロックウエル社製の BT829 を初期化するためのサンプルプログラムです。
1 フレームで、640×480 画素のビデオデータを出力します。

各サンプルプログラムでの描画例を**図** 5.1～**図** 5.11 に示します。

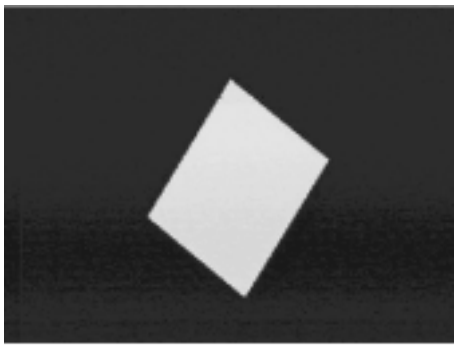


図 5.1　sample.c **の描画例（四角（単色）を描画した例）**

図 5.2　sample2.c **の描画例（文字（２値）を描画した例）**

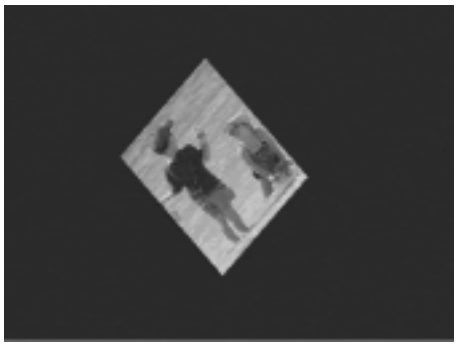


図 5.3　sample3.c **の描画例（自然画を描画した例）**

図 5.4　sample4.c **の描画例（多角形を描画した例）**

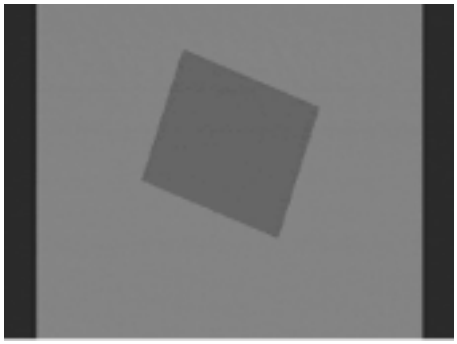


図 5.5　sample5.c **の描画例（描画開始位置を変更した例）**

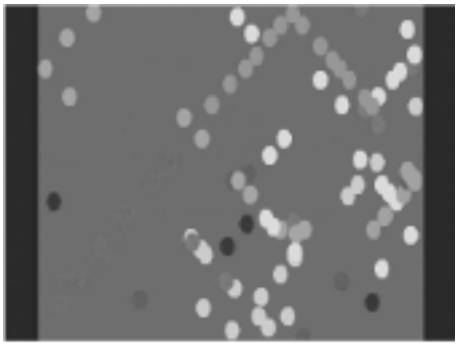


図 5.6　sample6.c **の描画例（任意パターンで描画した例）**

図 5.7　sample7.c **の描画例（表示サイズ** 640×480**）**

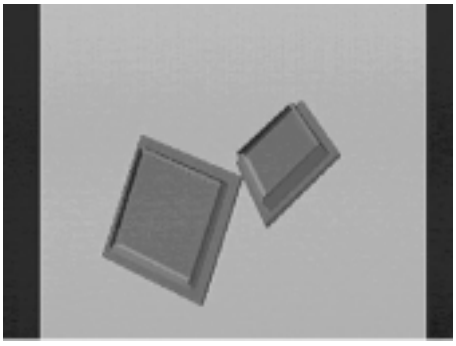


図 5.8　sample8.c **の描画例（**3 **次元空間を表現した例）**

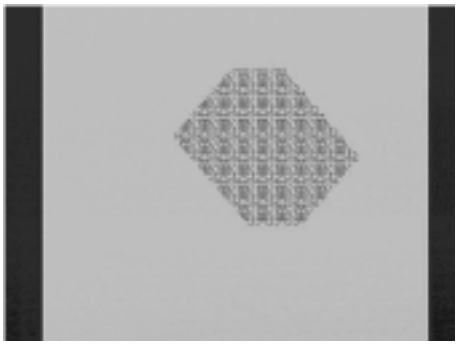


図 5.9　sample9.c **の描画例（任意パターンで描画した例）**

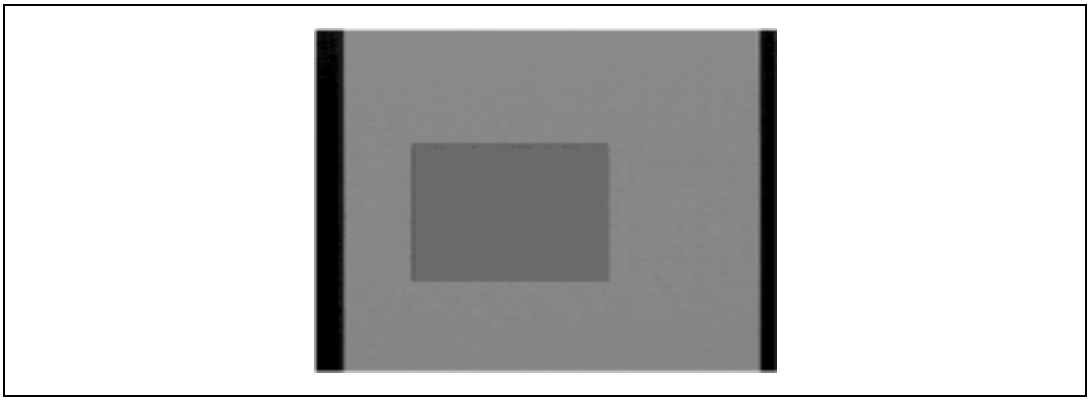


図 5.10　tst_brk.c **の描画例（描画の中断・再開を行った例）**

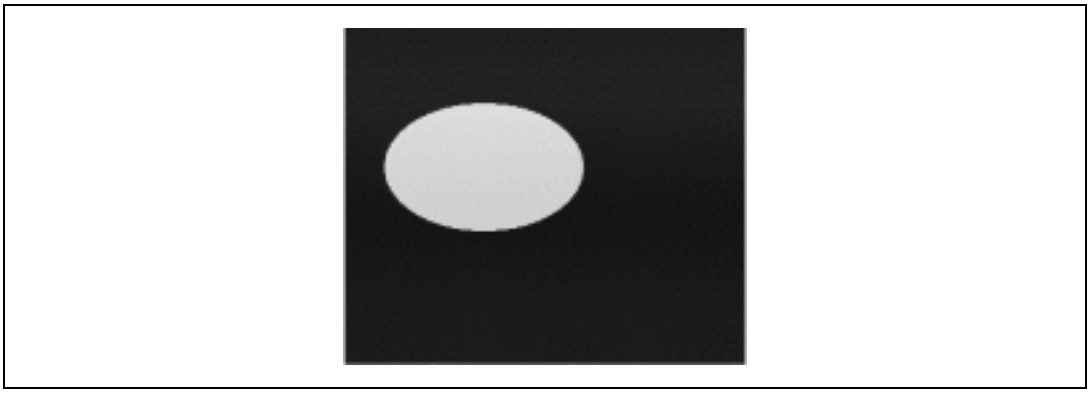


図 5.11　elps.c **の描画例（楕円を描画した例）**

5.1 サンプルプログラムの説明

5.1.1 基本関数の構成

本アプリケーションノートに記載しているサンプルプログラムは、（1）～（3）に示す関数を基本関数として使用しています。

(1) 初期化関数

アドレスマップドレジスタに初期値を設定するための関数です。

（a） ginit ()関数：本関数では、**「3.3 同期信号の設定方法」**で求めた値を使用して、初期化を行います。

(2) コマンド・ダブルバッファ制御関数

ディスプレイリストの配置および、描画の開始・終了を管理するための関数です。本関数は、DL0 および DL1 領域をソフトウェア制御にて交互に使用し、描画処理とディスプレイリストの配置を並行して行います。

（a） init_stat ()関数 ： コマンド・ダブルバッファ制御の初期化を行います。
（b） draw_stat ()関数 ： コマンド・ダブルバッファ制御を開始するための関数。
（c） draw_end ()関数 ： コマンド・ダブルバッファ制御を終了し、その後、描画を行う関数。
（d） change_com_buffer ()関数 ： draw_end ()関数で使用される関数。
本関数で描画の終了判定を行います。

また、コマンド・ダブルバッファ制御関数では、以下の変数を使用します。

（a） DrawBuffer ： DL0, DL1領域に対し、領域の選択を行う際に使用する変数。
（b） DISPLIST_ptr ： ディスプレイリストの配置場所を記憶するためのポインタ変数。
（c） flag ： change_com_buffer ()関数で使用される変数。

(3) 入出力関数（マクロ関数）

アドレスマップドレジスタにアクセスを行うための関数です。本関数はマクロ関数であり、volatile 宣言を使用して変数の最適化を禁止させることで、アドレスマップドレジスタのアクセスに適した関数を実現しています。（通常、外部デバイスにアクセスを行う場合、volatile 宣言をした変数を使用して行ってください。）

（a） outport ()関数 ： アドレスマップドレジスタに値を設定します。
（b） inport ()関数 ： アドレスマップドレジスタから値を得ます。

5.1.2 サンプルプログラムの記述規則

以下に、サンプルプログラムの記述規則を示します。記述は、（1）から（9）の順番で行います。また、draw_start 関数と draw_end 関数の間を、ディスプレイリストの生成単位とし、その単位ごとに、DL0 および DL1 領域を交互に使用しながら、ディスプレイリストの配置およびレンダリング（描画）を行います。

なお、DL0 および DL1 領域の開始アドレスは、下に示すプログラムのように Q2SD_REG.h ファイル内の DISPLIST0、DISPLIST1 として定義されます。

```
#define     DISPLIST0         0x190000L/* DL0 area start address */
#define     DISPLIST1         0x198000L/* DL1 area start address */
```

Q2SD_mac.h ファイルに Q2SD コマンドのマクロ関数とプロトタイプ記述していますのでご参考ください。

サンプルプログラムの記述規則を以下に示します。

```
void main(void)
{
          short array[8];
          short DBmode;
/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
          char  first,quick;
          DBmode = 1;
          first   = ON;
          quick  = OFF;

 /*-----------------------------------------------------------------------------------------------------
```

（1） init_BSC()関数にて、SH7709のバスステートコントローラに初期値を設定します。

```
 -----------------------------------------------------------------------------------------------------*/
          init_BSC (DBmode);          /* Initialize SH7709 bus state controller */

 /*-----------------------------------------------------------------------------------------------------
```

（2） ginit()関数にて、Q2SDのアドレスマップドレジスタに初期値を設定します。

```
 -----------------------------------------------------------------------------------------------------*/
          ginit(DBmode);              /* Initialize Q2 */
```

```
/*-----------------------------------------------------------------------------------------------------
```

(3) init_start()関数にて、コマンド・ダブルバッファ制御の初期化を行います。

```
-----------------------------------------------------------------------------------------------------*/
        init_start();               /* Initialize Transfer Procedure */

/*-----------------------------------------------------------------------------------------------------
```

(4) draw_start()関数にて、コマンド・ダブルバッファ制御を開始します。
さらに、vbkemコマンドをディスプレイリストとしてUGMに配置します。

```
-----------------------------------------------------------------------------------------------------*/
        draw_start();               /* Start Transfer Display List to UGM */

/*-----------------------------------------------------------------------------------------------------
```

(5) Q2SDのコマンドマクロ関数を記述します。
なお、sclip()マクロ関数およびlcofs()マクロ関数は、
必ず、一番最初に記述して下さい。

```
-----------------------------------------------------------------------------------------------------*/
        sclip(0x0000, 319,239);
        lcofs(0x0000,0,0);

/*-----------------------------------------------------------------------------------------------------
```

(6) draw_end()関数にて、Q2SDのコマンドマクロ関数で生成したディスプレイリストにもとずいて、Q2SDがレンダリング（描画）を行います。

```
-----------------------------------------------------------------------------------------------------*/
        draw_end(DBmode,&first,quick);


/*-----------------------------------------------------------------------------------------------------
```

(7) draw_start()関数にて、コマンド・ダブルバッファ制御を再開始します。
その後、Q2SDのコマンドマクロ関数を記述します。
なお、２回目以降のdraw_start()関数において、sclip()マクロ関数およびlcofs()マクロ関数は、必要に応じて記述して下さい。

```
-----------------------------------------------------------------------------------------------------*/
        draw_start();               /* Start Transfer Display List to UGM */
        ...........
        draw_end(DBmode,&first,quick);

/*-----------------------------------------------------------------------------------------------------
```

(8) サンプルプログラムは、returnを記述して、プログラムを終了を明示しています。

```
-----------------------------------------------------------------------------------------------------*/
        return;                     /* EXIT */
}
```

```
/*-------------------------------------------------------------------------------------------------
```

（9） 最後に、MS7709.incとQ2SDL.incファイルをインクルードしています。

```
-------------------------------------------------------------------------------------------------*/

#include "MS7709.inc"

#include "Q2SDl.inc"
```

5.1.3 プログラム作成上の注意点

下記の（1）～（3）に注意点を示します。

(1) VSYNC に同期させる

SuperH と HD64413A による描画システムでは、VSYNC に同期したプログラムを作成することがプログラム作成時の基本スタイルとなります。このため、下記の（a）または（b）を行ってください。本アプリケーションノートに記載されているサンプルプログラムでは、（a）の方法を採用しています。

（a） ディスプレイリストにVBKEMを使用してVSYNCと同期化させる。
（b） VBKまたはFRMを使用してIRL端子割り込みにてVSYNCと同期化させる。

(2) ノンキャッシュ領域を使用してアクセスを行う

SuperH は内部にキャッシュを内蔵しているものがあります。このため HD64413A へのアクセスはノンキャッシュ領域で行って下さい。

キャッシング領域で使用する場合には、キャッシュとのコヒーレンシを確保するため、ディスプレイリストを UGM へ配置後、描画開始するにはキャッシュの書き戻し／無効化を行ってください。

(3) 最適化を無効にした変数を使用する

HD64413A のように、SuperH にとって外部デバイスとなる LSI にアクセスを行う場合、最適化を無効にした変数を使用してアクセスを行ってください。また、最適化を無効にする宣言として volatile 宣言をご使用ください。

（記述例）

```
#define VU_SHORT (volatile unsigned short * const)
#define     outport(add,data) ( *VU_SHORT(add) ) = ( (unsigned short)(data) )
#define     inport(add)       ( *VU_SHORT(add) )
#define _Q2SYSR               0x000L + 0xA8800000L
```

上記のような定義を行った場合、コンパイルを行うことで下記の（a），(b）のような volatile 宣言を含む記述に置換されます。

（a） 値を設定する場合の記述

■ 置換前

```
/* _Q2SYSR register is set to 0x2080. */
 outport(_Q2SYSR, 0x2080);
```

■ 置換後

```
/* _Q2SYSR register is set to 0x2080. */
 ( *(volatile unsigned short * const)(0x000L + 0xA8800000L) ) = ( (unsigned
short)(0x2080) );
```

(b) 値を得るための記述

■ 置換前

```
/* Read value from _Q2SYSR register. */
 abc = inport(_Q2SYSR);
```

■ 置換後

```
/* Read value from _Q2SYSR register. */
 abc = ( *(volatile unsigned short * const)(0x000L + 0xA8800000L) );
```

5.2 サンプルプログラムのソースリスト

5.2.1 BuildB.bat(BuildL.bat)ビルドファイル

```
SHC /CPU=SH3 /OPT=1 /SPEED /ENDIAN=BIG /LOOP %1.c
if errorlevel 1 goto end
LNK %1 /OUTPUT=%1 /LIB=c:¥SHC¥V50¥LIB¥shc3npb.LIB /FORM=A /START=P(0C000000)
if errorlevel 1 goto end
CNVS %1 %1.txt
DEL %1.obj
DEL %1.abs
:end
```

5.2.2 MS7709.inc のソースファイル

```
/*
                SH7709 bus state controller for MS7709SE01/MS4413DB01
                        Copyright(c) Hitachi Ltd. 1999
*/

/*======================    DEFINE TYPES     ==========================*/
#define   BCR2      *(volatile unsigned short *)0xFFFFFF62  /* bus control register 1 */
#define   WCR1      *(volatile unsigned short *)0xFFFFFF64  /* wait state control register 1 */
#define   WCR2      *(volatile unsigned short *)0xFFFFFF66  /* wait state control register 2 */
void init_BSC(void)                           /* Initialize SH7709 bus state controller */
{
          BCR2 = (BCR2 & 0x3fc0) | 0x0020;    /* set 16bit bus width for CS2(area-2) */
          WCR1 = (WCR1 & 0x3fc3) | 0x0010;    /* set 1 idle cycle to CS2(area-2) */
          WCR2 = (WCR2 & 0xffe7) | 0x0010;    /* set 2 wait state cycle to CS2(area-2) */
}
```

5.2.3 Q2SDL.inc のソースファイル

```
/*
                                        Command double buffer control for Q2SD
                                        ((for 16bit/Pixel 65536 Color mode))
                                Copyright(c) Hitachi Ltd. 1999
*/
void  init_start(void);                                 /* Initilaze Display List Double Buffering */
short draw_start(void);                                 /* Initialize Display List Address pointer */
short draw_end(short DBmode,char *first,char quick);    /* Display List Execution */
void  change_com_buffer(void);                          /* Change Display List */

short    DrawBuffer;                                    /* Display List Double Buffering Destination Number */
short    flag;

/* ---------------------------------------------------------------------- */

void init_start(void)                                   /* Initialze transfer procedure */
{
        DrawBuffer = 0;
        flag = 0;
}

short draw_start(void)                                          /* Initialize Display List Address Pointer */
{
  long  CMD_StartAddress;
        if(DrawBuffer == 0)
                        /* Display List Start Address */
                CMD_StartAddress = (long)(DISPLIST1+UGMBASE); /* Execute Buffer0 -> Transfer Display List to
Buffer1 */
        else
                CMD_StartAddress = (long)(DISPLIST0+UGMBASE); /* Execute Buffer1 -> Transfer Display List to
Buffer0 */

        DISPLIST_ptr = (unsigned short *)CMD_StartAddress;

        vbkem(0x0000,0x0000,0x0000);                            /* Add 'VBKEM commnad'. */

        return( 0 );
}
```

```
short draw_end(short DBmode,char *first,char quick)            /* Execute Display List */
{
        trap(0);                    /* Add 'trap' command */

.       change_com_buffer();

        switch(DBmode) {

                case 2:                     /* Manual Change */
                        /* Renddering start */
                        outport(_Q2SYSR,0x2180);

                        if ( quick == OFF ) {
                                outport(_Q2SYSR,0x2280);    /* Frame change    */
                                /* Wait Display Change Bit '1' -> '0' */
                                while( ( inport(_Q2SYSR) & 0x0200 ) != 0 );
                        }

                        break;


                case 1:                     /* Auto Renderring */
                        /* Renddering start */
                        outport(_Q2SYSR,0x2140);
                        break;

                default:          /* Auto Change */
                        outport(_Q2SRCR,0x0800);            /* Clear VBK bit */
                        while( ( inport(_Q2SR) & 0x0800 ) == 0); /* Wait VBK */

                        /* Renddering start */
                        outport(_Q2SYSR,0x2100);
        }
        return( 0 );
}

void change_com_buffer(void)
{
  unsigned long CMD_StartAddress;
```

```
    unsigned short st;

      if (flag) {
                  while(1){
                              st=inport(_Q2SR);
                              if ((st & 0x0400)!=0) break;          /* Check TRA bit */
                              if ((st & 0x0200)!=0) {
                                       outport(_Q2SRCR,0x0200);     /* Clear CSF bit   */
                                       break; /* Check CSF bit */
                              }
                  }
            }
            else {
                  flag = 1;
            }

            outport(_Q2SRCR,0x0400);              /* Clear TRA bit   */

            if(DrawBuffer == 0) {
                  DrawBuffer = 1;
                  CMD_StartAddress = DISPLIST1;
            }
            else {
                  DrawBuffer = 0;
                  CMD_StartAddress = DISPLIST0;
            }
            outport( _Q2DLSAH,(CMD_StartAddress >> 16L) & 0xffff ); /* Change DLSAR */
            outport( _Q2DLSAL, CMD_StartAddress               );
    }
```

5.2.4 Q2SD_mac.h のソースファイル

```
/*

                                    Q2 command macro sample

                            Copyright(c) Hitachi Ltd. 1999

*/

/* ------------- Q2 Display List Functions ---------------- */
void polygon4a(short draw_mode, short txs,short tys,short tdx,short tdy, short poly[]);
void polygon4b(short draw_mode, short src_h,short src_l,short tdx,short tdy, short poly[], short color0,short
color1);
void polygon4c(short draw_mode, short poly[], short color1);
void line(short draw_mode,short color,short n, short poly[]);
void rline(short draw_mode,short color,short n, short poly[]);
void pline(short draw_mode,short color0,short color1,short src_h,short src_l,short tdx,short n, short poly[]);
void rpline(short draw_mode,short color0,short color1,short src_h,short src_l,short tdx,short n, short poly[]);
void linew(short draw_mode,short n, short poly[]);
void rlinew(short draw_mode,short n, short poly[]);
void clrw(short draw_mode,short xmin,short ymin,short xmax,short ymax);
void ftrap(short draw_mode,short n, short dxl, short poly[]);
void rftrap(short draw_mode,short n, short dxl, short poly[]);
void move(short draw_mode,short xc,short yc);
void rmove(short draw_mode,short xc,short yc);
void lcofs(short draw_mode,short xo,short yo);
void rlcofs(short draw_mode,short xo,short yo);
void sclip(short draw_mode,short xmax,short ymax);
void uclip(short draw_mode,short xmin,short ymin,short xmax,short ymax);
void jump(short draw_mode,short adr_h,short adr_l);
void gosub(short draw_mode,short adr_h,short adr_l);
void ret(short draw_mode);
void trap(short draw_mode);
void nop3(short draw_mode,short dummy1,short dummy2);
void wpr(short draw_mode,short rn,short data);
void vbkem(short draw_mode,short dummy1,short dummy2);

/* ----- Q2 function macro ------------------------------- */
#define polygon4a( draw_mode, txs,tys, tdx,tdy, array )¥
{¥
```

```
  short *_q2_array = array;¥
         *DISPLIST_ptr++ = 0x0000 | draw_mode;                /* POLYGON4A */¥
         *DISPLIST_ptr++ = txs;           *DISPLIST_ptr++ = tys;¥
         *DISPLIST_ptr++ = tdx;           *DISPLIST_ptr++ = tdy;¥
         *DISPLIST_ptr++ = *_q2_array++;  *DISPLIST_ptr++ = *_q2_array++;¥
         *DISPLIST_ptr++ = *_q2_array++;  *DISPLIST_ptr++ = *_q2_array++;¥
         *DISPLIST_ptr++ = *_q2_array++;  *DISPLIST_ptr++ = *_q2_array++;¥
         *DISPLIST_ptr++ = *_q2_array++;  *DISPLIST_ptr++ = *_q2_array++;¥
}

#define polygon4b( draw_mode, src_h,src_l, tdx,tdy, array, color0,color1 )¥
{¥
  short *_q2_array = array;¥
         *DISPLIST_ptr++ = 0x0800 | draw_mode;                /* POLYGON4B */¥
         *DISPLIST_ptr++ = src_h;         *DISPLIST_ptr++ = src_l;¥
         *DISPLIST_ptr++ = tdx;           *DISPLIST_ptr++ = tdy;¥
         *DISPLIST_ptr++ = *_q2_array++;  *DISPLIST_ptr++ = *_q2_array++;¥
         *DISPLIST_ptr++ = *_q2_array++;  *DISPLIST_ptr++ = *_q2_array++;¥
         *DISPLIST_ptr++ = *_q2_array++;  *DISPLIST_ptr++ = *_q2_array++;¥
         *DISPLIST_ptr++ = *_q2_array++;  *DISPLIST_ptr++ = *_q2_array++;¥
         *DISPLIST_ptr++ = color0;¥
         *DISPLIST_ptr++ = color1;¥
}

#define polygon4c( draw_mode, array, color1 )¥
{¥
  short *_q2_array = array;¥
         *DISPLIST_ptr++ = 0x1000 | draw_mode;                /* POLYGON4C */¥
         *DISPLIST_ptr++ = *_q2_array++;  *DISPLIST_ptr++ = *_q2_array++;¥
         *DISPLIST_ptr++ = *_q2_array++;  *DISPLIST_ptr++ = *_q2_array++;¥
         *DISPLIST_ptr++ = *_q2_array++;  *DISPLIST_ptr++ = *_q2_array++;¥
         *DISPLIST_ptr++ = *_q2_array++;  *DISPLIST_ptr++ = *_q2_array++;¥
         *DISPLIST_ptr++ = color1;¥
}

#define line( draw_mode, color, n, array )¥
{¥
  short *_q2_array = array;¥
  short i;¥
```

```
        *DISPLIST_ptr++ = 0x6000 | draw_mode;                /* Line */¥
        *DISPLIST_ptr++ = color;¥
        *DISPLIST_ptr++ = n;¥
        for(i = 0; i < (n)*2; i++) *DISPLIST_ptr++ = *_q2_array++;¥
}

#define pline( draw_mode, color0,color1, src_h,src_l, tdx, n, array )¥
{¥
  short *_q2_array = array;¥
  short i;¥
        *DISPLIST_ptr++ = 0x7101 | draw_mode;                /* PLINE */¥
        *DISPLIST_ptr++ = color0;¥
        *DISPLIST_ptr++ = color1;¥
        *DISPLIST_ptr++ = src_h;¥
        *DISPLIST_ptr++ = src_l;¥
        *DISPLIST_ptr++ = tdx;¥
        *DISPLIST_ptr++ = n;¥
        for(i = 0; i < (n)*2; i++) *DISPLIST_ptr++ = *_q2_array++;¥
}

#define clrw( draw_mode, xmin, ymin, xmax, ymax )¥
{¥
        *DISPLIST_ptr++ = 0xa000 | draw_mode;                /* CLeaR Work */¥
        *DISPLIST_ptr++ = xmin;¥
        *DISPLIST_ptr++ = ymin;¥
        *DISPLIST_ptr++ = xmax;¥
        *DISPLIST_ptr++ = ymax;¥
}

#define ftrap( draw_mode, n, dxl, array )¥
{¥
  short *_q2_array = array;¥
  short i;¥
        *DISPLIST_ptr++ = 0x4000 | draw_mode;                /* Filled TRAPezoid */¥
        *DISPLIST_ptr++ = n;¥
        *DISPLIST_ptr++ = dxl;¥
        for(i = 0; i < (n)*2; i++) *DISPLIST_ptr++ = *_q2_array++;¥
}
```

```
#define linew( draw_mode, n, array )¥
{¥
  short *_q2_array = array;¥
  short i;¥
        *DISPLIST_ptr++ = 0x5000 | draw_mode;                /* Line Work */¥
        *DISPLIST_ptr++ = n;¥
        for(i = 0; i < (n)*2; i++) *DISPLIST_ptr++ = *_q2_array++;¥
}

#define move( draw_mode, xc, yc )¥
{¥
        *DISPLIST_ptr++ = 0x8000 | draw_mode;                /* MOVE */¥
        *DISPLIST_ptr++ = xc;¥
        *DISPLIST_ptr++ = yc;¥
}

#define lcofs( draw_mode, xo,yo )¥
{¥
        *DISPLIST_ptr++ = 0x9000 | draw_mode;                /* LoCal OFfSet */¥
        *DISPLIST_ptr++ = xo;¥
        *DISPLIST_ptr++ = yo;¥
}

#define sclip( draw_mode, xmax,ymax )¥
{¥
        *DISPLIST_ptr++ = 0xb800 | draw_mode;                /* System CLIP */¥
        *DISPLIST_ptr++ = xmax;¥
        *DISPLIST_ptr++ = ymax;¥
}

#define uclip( draw_mode, xmin, ymin, xmax, ymax )¥
{¥
        *DISPLIST_ptr++ = 0xa800 | draw_mode;                /* User CLIP */¥
        *DISPLIST_ptr++ = xmin;¥
        *DISPLIST_ptr++ = ymin;¥
        *DISPLIST_ptr++ = xmax;¥
        *DISPLIST_ptr++ = ymax;¥
```

```
}

#define jump( draw_mode, adr_h, adr_l)¥
{¥
        *DISPLIST_ptr++ = 0xc000 | draw_mode;                /* Jump */¥
        *DISPLIST_ptr++ = adr_h;¥
        *DISPLIST_ptr++ = adr_l;¥
}

#define gosub( draw_mode, adr_h, adr_l )¥
{¥
        *DISPLIST_ptr++ = 0xc800 | draw_mode;                /* GO SUBroutine */¥
        *DISPLIST_ptr++ = adr_h;¥
        *DISPLIST_ptr++ = adr_l;¥
}

#define ret( draw_mode )¥
{¥
        *DISPLIST_ptr++ = 0xd800 | draw_mode;                /* RETurn from subroutine */¥
}

#define nop3( draw_mode, dummy1, dummy2 )¥
{¥
        *DISPLIST_ptr++ = 0xf000 | draw_mode;                /* NOP3 */¥
        *DISPLIST_ptr++ = dummy1;¥
        *DISPLIST_ptr++ = dummy2;¥
}

#define wpr( draw_mode, rn, data )¥
{¥
        *DISPLIST_ptr++ = 0xb000 | draw_mode;                /* Write PaRameter */¥
        *DISPLIST_ptr++ = rn;¥
        *DISPLIST_ptr++ = data;¥
}

#define vbkem( draw_mode, dummy1, dummy2 )¥
{¥
        *DISPLIST_ptr++ = 0xd000 | draw_mode;                /* Vertical BlanKing Edge Maker */¥
        *DISPLIST_ptr++ = dummy1;¥
```

```
        *DISPLIST_ptr++ = dummy2;¥
}

#define trap( draw_mode )¥
{¥
        *DISPLIST_ptr++ = 0xf800 | draw_mode;                /* TRAP */¥
}

#define rftrap( draw_mode, n, dxl, array )¥
{¥
  short *_q2_array = array;¥
  short i;¥
        *DISPLIST_ptr++ = 0x4800 | draw_mode;                /* Relative Filled TRAPezoid */¥
        *DISPLIST_ptr++ = n;¥
        *DISPLIST_ptr++ = dxl;¥
        for(i = 0; i < n; i++) *DISPLIST_ptr++ = *_q2_array++;¥
}

#define rlinew( draw_mode, n, array )¥
{¥
  short *_q2_array = array;¥
  short i;¥
        *DISPLIST_ptr++ = 0x5800 | draw_mode;                /* Rerlative Line Work */¥
        *DISPLIST_ptr++ = n;¥
        for(i = 0; i < n; i++) *DISPLIST_ptr++ = *_q2_array++;¥
}

#define rline( draw_mode, color, n, array )¥
{¥
  short *_q2_array = array;¥
  short i;¥
        *DISPLIST_ptr++ = 0x6800 | draw_mode;                /* Relative Line */¥
        *DISPLIST_ptr++ = color;¥
        *DISPLIST_ptr++ = n;¥
        for(i = 0; i < n; i++) *DISPLIST_ptr++ = *_q2_array++;¥
}

#define rpline( draw_mode, color0, color1, src_h,src_l, tdx, n, array )¥
{¥
```

```
  short *_q2_array = array;¥
  short i;¥
        *DISPLIST_ptr++ = 0x7901 | draw_mode;                /* Relative PLINE */¥
        *DISPLIST_ptr++ = color0;¥
        *DISPLIST_ptr++ = color1;¥
        *DISPLIST_ptr++ = src_h;¥
        *DISPLIST_ptr++ = src_l;¥
        *DISPLIST_ptr++ = tdx;¥
        *DISPLIST_ptr++ = n;¥
        for(i = 0; i < n; i++) *DISPLIST_ptr++ = *_q2_array++;¥
}

#define rmove( draw_mode, xc, yc )¥
{¥
        *DISPLIST_ptr++ = 0x8800 | draw_mode;                /* Relative MOVE */¥
        *DISPLIST_ptr++ = ((xc) << 8) | ((yc) & 0xff);¥
}

#define rlcofs( draw_mode, xo, yo )¥
{¥
        *DISPLIST_ptr++ = 0x9800 | draw_mode;                /* Relative LoCal OFfSet */¥
        *DISPLIST_ptr++ = ((xo) << 8) | ((yo) & 0xff);¥
}
```

5.2.5 Q2SD_REG.h のソースファイル

```
/*_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/

            Q2SD Display List / SHC sample program
              (Q2SD_REG.h)

                 Copyright(c) Hitachi Ltd. 1999

  _/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/*/

#define   Area2          1              /* Q2SD internal register/UGM address area 2 on MS7709SE01 */
#define   Area5          0              /* Q2SD internal register/UGM address area 5 on MS7709SE01 */

#define   AddressArea    Area5          /* set Area2 or Area5 */

#define OFF                             0
#define ON                              1


/*=======================    DEFINE TYPES     ==========================*/
/*----- Define new type -----*/
#define VU_SHORT (volatile unsigned short * const)


/*====================    DEFINE I/O FUNCTIONS    ======================*/
/*----- Define I/O function -----*/
void outport(unsigned long address, unsigned short data);
unsigned short inport(unsigned long address);
#define   outport(add,data) ( *VU_SHORT(add) ) = ( (unsigned short)(data) )
#define   inport(add)       ( *VU_SHORT(add) )


/*====================    DEFINE MAPPING ADDRESS    ======================*/
/*----- Define commnad double buffer start address -----*/
#define   DISPLIST0 0x0F0000L /* DL0 area start address */
#define   DISPLIST1 0x0F8000L /* DL1 area start address */

#if       AddressArea
#define BASE_ADDRESS       0xA8800000L                   /* Q2SD internal register base address */
                                                         /* Byte address (MS7709SE01/MS4413DB01) */
```

```
#define UGMBASE            0xA8000000L                 /* UGM base address */
                                                       /* Byte address (MS7709SE01/MS4413DB01) */
#else
#define BASE_ADDRESS       0xB4800000L                 /* Q2SD internal register base address */
                                                       /* Byte address (MS7709SE01/MS4413DB01) */

#define UGMBASE            0xB4000000L                 /* UGM base address */
                                                       /* Byte address (MS7709SE01/MS4413DB01) */
#endif

#define _Q2SYSR            0x000L + BASE_ADDRESS       /* No.000 */
#define _Q2SR              0x002L + BASE_ADDRESS       /* No.001 */
#define _Q2SRCR            0x004L + BASE_ADDRESS       /* No.002 */
#define _Q2IER             0x006L + BASE_ADDRESS       /* No.003 */
#define _Q2MEMR            0x008L + BASE_ADDRESS       /* No.004 */
#define _Q2DSMR            0x00AL + BASE_ADDRESS       /* No.005 */
#define _Q2REMR            0x00CL + BASE_ADDRESS       /* No.006 */
#define _Q2IEMR            0x00EL + BASE_ADDRESS       /* No.007 */

#define _Q2DSX             0x010L + BASE_ADDRESS       /* No.008 */
#define _Q2DSY             0x012L + BASE_ADDRESS       /* No.009 */
#define _Q2DSA0            0x014L + BASE_ADDRESS       /* No.00A */
#define _Q2DSA1            0x016L + BASE_ADDRESS       /* No.00B */
#define _Q2DLSAH           0x018L + BASE_ADDRESS       /* No.00C */
#define _Q2DLSAL           0x01AL + BASE_ADDRESS       /* No.00D */
#define _Q2SSAR            0x01CL + BASE_ADDRESS       /* No.00E */
#define _Q2WSAR            0x01EL + BASE_ADDRESS       /* No.00F */
#define _Q2DMASH           0x020L + BASE_ADDRESS       /* No.010 */
#define _Q2DMASL           0x022L + BASE_ADDRESS       /* No.011 */
#define _Q2DMAWL           0x024L + BASE_ADDRESS       /* No.012 */

#define _Q2HDS             0x026L + BASE_ADDRESS       /* No.013 */
#define _Q2HDE             0x028L + BASE_ADDRESS       /* No.014 */
#define _Q2VDS             0x02AL + BASE_ADDRESS       /* No.015 */
#define _Q2VDE             0x02CL + BASE_ADDRESS       /* No.016 */
#define _Q2HSW             0x02EL + BASE_ADDRESS       /* No.017 */
#define _Q2HC              0x030L + BASE_ADDRESS       /* No.018 */
#define _Q2VSP             0x032L + BASE_ADDRESS       /* No.019 */
```

```
#define _Q2VC              0x034L + BASE_ADDRESS          /* No.01A */
#define _Q2DOR             0x036L + BASE_ADDRESS          /* No.01B */
#define _Q2DOG_DOB         0x038L + BASE_ADDRESS          /* No.01C */
#define _Q2CDR             0x03AL + BASE_ADDRESS          /* No.01D */
#define _Q2CDG_CDB         0x03CL + BASE_ADDRESS          /* No.01E */
#define _Q2CSTH            0x03EL + BASE_ADDRESS          /* No.01F */
#define _Q2CSTL            0x040L + BASE_ADDRESS          /* No.020 */

#define _Q2ISAH            0x042L + BASE_ADDRESS          /* No.021 */
#define _Q2ISAL            0x044L + BASE_ADDRESS          /* No.022 */
#define _Q2IDSX            0x046L + BASE_ADDRESS          /* No.023 */
#define _Q2IDSY            0x048L + BASE_ADDRESS          /* No.024 */
#define _Q2IDE             0x04AL + BASE_ADDRESS          /* No.025 */

#define _Q2BGSX            0x04CL + BASE_ADDRESS          /* No.026 */
#define _Q2BGSY            0x04EL + BASE_ADDRESS          /* No.027 */
#define _Q2DMAWH           0x050L + BASE_ADDRESS          /* No.028 */

#define _Q2EQW             0x052L + BASE_ADDRESS          /* No.029 */
#define _Q2SPW             0x054L + BASE_ADDRESS          /* No.02A */
#define _Q2DSMR2           0x056L + BASE_ADDRESS          /* No.02B */
#define _Q2HVP             0x058L + BASE_ADDRESS          /* No.02C */
#define _Q2VVP             0x05AL + BASE_ADDRESS          /* No.02D */

#define _Q2VSAH0           0x062L + BASE_ADDRESS          /* No.031 */
#define _Q2VSAL0           0x064L + BASE_ADDRESS          /* No.032 */
#define _Q2VSAH1           0x066L + BASE_ADDRESS          /* No.033 */
#define _Q2VSAL1           0x068L + BASE_ADDRESS          /* No.034 */
#define _Q2VSAH2           0x06AL + BASE_ADDRESS          /* No.035 */
#define _Q2VSAL2           0x06CL + BASE_ADDRESS          /* No.036 */
#define _Q2VSIZEX          0x06EL + BASE_ADDRESS          /* No.037 */
#define _Q2VSIZEY          0x070L + BASE_ADDRESS          /* No.038 */
#define _Q2VIMR            0x072L + BASE_ADDRESS          /* No.039 */
#define _Q2HCSR1           0x074L + BASE_ADDRESS          /* No.03A */
#define _Q2VCSR1           0x076L + BASE_ADDRESS          /* No.03B */
#define _Q2HCS2            0x078L + BASE_ADDRESS          /* No.03C */
#define _Q2VCS2            0x07AL + BASE_ADDRESS          /* No.03D */
#define _Q2CSAR1           0x07CL + BASE_ADDRESS          /* No.03E */
#define _Q2CSAR2           0x07EL + BASE_ADDRESS          /* No.03F */
```

```
#define _Q2XC              0x080L + BASE_ADDRESS          /* No.040 */
#define _Q2YC              0x082L + BASE_ADDRESS          /* No.041 */
#define _Q2XO              0x084L + BASE_ADDRESS          /* No.042 */
#define _Q2YO              0x086L + BASE_ADDRESS          /* No.043 */
#define _Q2UXMIN           0x088L + BASE_ADDRESS          /* No.044 */
#define _Q2UYMIN           0x08AL + BASE_ADDRESS          /* No.045 */
#define _Q2UXMAX           0x08CL + BASE_ADDRESS          /* No.046 */
#define _Q2UYMAX           0x08EL + BASE_ADDRESS          /* No.047 */
#define _Q2SXMAX           0x090L + BASE_ADDRESS          /* No.048 */
#define _Q2SYMAX           0x092L + BASE_ADDRESS          /* No.049 */
#define _Q2RTNH            0x094L + BASE_ADDRESS          /* No.04A */
#define _Q2RTNL            0x096L + BASE_ADDRESS          /* No.04B */
#define _Q2RSAR            0x098L + BASE_ADDRESS          /* No.04C */
```

5.2.6 sample.c のソースファイル

```
/*
                                        Q2 Display List / SHC sample program (1)
                        (Zoom/Shrink/Rotate/Move 'Solid' Polygon)

                                        Copyright(c) Hitachi Ltd. 1999
*/


#include <machine.h>
#include <stdio.h>
#include <math.h>
#include "Q2SD_REG.h"
#include "Q2SD_mac.h"


unsigned short *DISPLIST_ptr;


#define   PI                                        3.14159

void  init_BSC(void);                               /* Initialize SH7709 bus state controller */
void  init_start(void);                             /* Initialize Display List Double Buffering (Only 1st) */
short draw_start(void);                             /* Initialize Display List Address pointer */
short draw_end(short DBmode,char *first,char quick);  /* Display List Execution */
void  change_com_buffer(void);
/* ------------------------------------------------------- */

void  clrscrn(void);
void  ginit(short DBmode);                          /* Graphics System Open  (Q2SD Initialize) */


void main(void)
{
          long   count;
          short  array[8];
          short  d, i, h, j, k, del;
          double rad;
          short  tx,ty;
          short  DBmode;
```

```
/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
          char  first,quick;

          DBmode = 1;
          first  = ON;
          quick  = OFF;

          init_BSC();                              /* Initialize SH7709 bus state controller */

          ginit(DBmode);                           /* Initialize Q2 */
          init_start();                            /* Initialize Transfer Procedure */
          draw_start();                            /* Start Transfer Display List to UGM */
          sclip(0x0000, 639,239);
          lcofs(0x0000,0,0);

          del = 4;

          for( count = 0 ; count < 2 ; count++ ){

/* Zoom ------------------------------------ */
                    array[0] =  150;    array[1] =  100;
                    array[2] =  165;    array[3] =  100;
                    array[4] =  165;    array[5] =  115;
                    array[6] =  150;    array[7] =  115;

                    for(i=0;i<=80;i++) {
                           clrscrn();

                           array[0] -=  del;     array[1] -=  del;
                           array[2] +=  del;     array[3] -=  del;
                           array[4] +=  del;     array[5] +=  del;
                           array[6] -=  del;     array[7] +=  del;
                           polygon4c(0x0000,array,0x001F);               /* POLYGON4C */

                           draw_end(DBmode,&first,quick);
                           draw_start();
```

```
                }

/* Shrink ----------------------------------- */
                for(i=0;i<=80;i++) {
                        clrscrn();

                        array[0] +=  del;     array[1] +=  del;
                        array[2] -=  del;     array[3] +=  del;
                        array[4] -=  del;     array[5] -=  del;
                        array[6] +=  del;     array[7] -=  del;
                        polygon4c(0x0000,array,0x001F);                /* POLYGON4C */

                        draw_end(DBmode,&first,quick);
                        draw_start();
                }

/* Rotate ----------------------------------- */
                for(d=0;d<=360;d+=5) {
                        clrscrn();

                        array[0] =  -60;     array[1] =  -70;
                        array[2] =   40;     array[3] =  -70;
                        array[4] =   40;     array[5] =   50;
                        array[6] =  -60;     array[7] =   50;

                        rad = PI * d / 180;
                        for(i=0;i<=6;i+=2) {
                                tx =(short)  (array[i] * cos(rad)-  array[i+1] * sin(rad));
                                ty =(short)  (array[i] * sin(rad)+  array[i+1] * cos(rad));
                                array[i]   =  tx;
                                array[i+1] =  ty;
                        }

                        lcofs(0x0000, 160, 115);
                        polygon4c(0x0000,array,0x001F);/* POLYGON4C */
                        lcofs(0x0000, 0, 0);
                        draw_end(DBmode,&first,quick);
                        draw_start();
                }
```

```
/* Move ------------------------------------ */
                    h=260;
                    j=110;
                    for(d=0;d<=360;d+=5) {
                            clrscrn();

                            array[0] =  -50;     array[1] =  -60;
                            array[2] =   50;     array[3] =  -60;
                            array[4] =   50;     array[5] =   60;
                            array[6] =  -50;     array[7] =   60;

                            rad = PI * d / 180;
                            for(i=0;i<=6;i+=2) {
                                    tx =(short)  (array[i] * cos(rad)-  array[i+1] * sin(rad));
                                    ty =(short)  (array[i] * sin(rad)+  array[i+1] * cos(rad));
                                    array[i]  =  tx;
                                    array[i+1] =  ty;
                            }

                            lcofs(0x0000, h, j);
                            polygon4c(0x0000,array,0x001F);                /* POLYGON4C */
                            lcofs(0x0000, 0, 0);
                            h-=5;
                            draw_end(DBmode,&first,quick);
                            draw_start();
                    }
            }
            return;   /* EXIT */
}


void clrscrn(void)
{
            short array[8];

/* ploygon4c (clear) ---------------------------------------------------- */
            array[0] =   0;     array[1] =   0;
            array[2] = 639;     array[3] =   0;
```

```
        array[4] =  639;    array[5] =  239;
        array[6] =   0;     array[7] =  239;
        polygon4c(0x0000, array, 0x0000);
}

void ginit(short DBmode)
{                   /* Q2SD */
  unsigned short hsw =  64;
  unsigned short xs  = 131;
  unsigned short xw  = 640;
  unsigned short hc  = 910;

  unsigned short vsw =  3;
  unsigned short ys  =  16;
  unsigned short yw  = 240;
  unsigned short vc  = 262;

        outport(_Q2SYSR, 0x4080);                       /* Initilaize Draw/Display */
        outport(_Q2SRCR, 0xfe80);                       /* Clear SR register      */

        /* Set InterFace Control Registers */
        outport(_Q2IER ,   0x0000);
        outport(_Q2MEMR,   0x0031);
        outport(_Q2DSMR,   0x0005);
        outport(_Q2DSMR2,  0x0000);
        outport(_Q2REMR,   0x0041);
        outport(_Q2IEMR,   0x0000);

        /* Set Memory Control Regisers    */
        outport(_Q2DSX,       xw);   /* Display size of x                   */
        outport(_Q2DSY,       yw);   /* Display size of y                   */
        outport(_Q2DSA0,   0x0000);  /* Frame buffer 0 area start address(H) */
        outport(_Q2DSA1,   0x0010);  /* Frame buffer 1 area start address(H) */
        outport(_Q2DLSAH,  0x0019);  /* Display list area start address(H)   */
        outport(_Q2DLSAL,  0x0000);  /* Display list area start address(L)   */
        outport(_Q2SSAR,   0x0008);  /* Color area sorce start address(H)    */
        outport(_Q2WSAR,   0x001F);  /* Work area start address(H)           */
        outport(_Q2DMASH,  0x0000);  /* DMA transfer start address(H)        */
        outport(_Q2DMASL,  0x0000);  /* DMA transfer start address(L)        */
```

```
        outport(_Q2DMAWL,  0x0000);  /* DAM transfer word                  */

        /* Display Contral Registers */
        outport(_Q2HDS,              hsw+xs-11    );
        outport(_Q2HDE,              hsw+xs-11+xw );
        outport(_Q2VDS,              ys-2         );
        outport(_Q2VDE,              ys-2+yw      );
        outport(_Q2HSW,              hsw-1        );
        outport(_Q2HC,               hc-1         );
        outport(_Q2VSP,              vc-vsw-1     );
        outport(_Q2VC,               vc-1         );
        outport(_Q2DOR,              0x0000);
        outport(_Q2DOG_DOB,          0x007C);
        outport(_Q2CDR,              0x00FC);
        outport(_Q2CDG_CDB,          0xFCFC);

        /* Input Data Control Registers */
        outport(_Q2ISAH,   0x0000);
        outport(_Q2ISAL,   0x0000);
        outport(_Q2IDSX,   0x0000);
        outport(_Q2IDSY,   0x0000);
        outport(_Q2IDE,    0x0000);

        switch(DBmode) {                          /* Enable Draw/Display */
                case 0:
                        outport(_Q2SYSR, 0x2000);  /* Idle,Auto Change mode,No DMA */
                        break;
                case 1:
                        outport(_Q2SYSR, 0x2040);  /* Idle,Auto Renderring mode,No DMA */
                        break;
                default:
                        outport(_Q2SYSR, 0x2080);  /* Idle,manual change moe,No DMA */
        }
}

#include "MS7709.inc"
#include "Q2SDl.inc"
```

5.2.7 sample2.c のソースファイル

```
/*
                                        Q2 Display List / SHC sample program (2)
                          (Zoom/Shrink/Rotate/Move 'Text 24×24')

                          Copyright(c) Hitachi Ltd. 1999
*/
#include <machine.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Q2SD_REG.h"
#include "Q2SD_mac.h"


unsigned short *DISPLIST_ptr;


#define   PI          3.14159


void  init_BSC(void);                                  /* Initialize SH7709 bus state controller */
void  init_start(void);                                /* Initialize Display List Double Buffering (Only 1st) */
short draw_start(void);                                /* Initialize Display List Address pointer */
short draw_end(short DBmode,char *first,char quick);  /* Display List Execution */
void  change_com_buffer(void);

/* ------------------------------------------------------ */

void  clrscrn(void);
void  ginit(short DBmode);                             /* Graphics System Open  (Q2 Initialize) */


void main()
{
          long  count;
          short array[8];
          short rel_adr_h, rel_adr_l;
          short d, i, h, j, k, del, lp;
          double rad;
```

```
          short tx,ty;
          short txs, tys;
          short DBmode;
/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
  char  first,quick;
  short forecolor = 0x0f0f; /* charactor  color */
  short backcolor = 0x0101; /* background color */


/* Define Font Data "漢" ( 24 dots × 24 lines ) ---------------------- */
  unsigned short font_pattern[]={
       /*----------------------------------------------*/
       /* Note : Q2 uses font pattern from LSB to MSB. */
       /*----------------------------------------------*/
       /* 'font_pattern' must be mapped at SuperH's big endian area. */

       /* (MSB  LSB)   (MSB  LSB)  (MSB  LSB)                 */
             0x1c00,      0x0e07,     0x430c,
             0x0c1c,      0xd8e3,     0xffff,
             0x0c00,      0x0003,     0x030c,
             0x0140,      0x4718,     0x3fff,
             0x636e,      0x2c18,     0x1863,
             0x6320,      0x3018,     0x1fff,
             0x6310,      0x1800,     0x1860,
             0xff98,      0x0c3f,     0x0060,
             0x600f,      0xec60,     0xffff,
             0xb00c,      0x0c01,     0x0338,
             0x1c0c,      0x0c0e,     0x3c0e,
             0x038c,      0xecf8,     0x6000
       };

  unsigned font_pattern_word_counter = sizeof(font_pattern)/sizeof(short);


          DBmode = 1;
```

```
        first  = ON;
        quick  = OFF;

        init_BSC();                                 /* Initialize SH7709 bus state controller */

        ginit(DBmode);                              /* Initialize Q2 */


/* ---------------------------------------------------------------------- */

        init_start();                               /* Initialize Transfer Procedure */
        draw_start();                               /* Start Transfer Display List to UGM */
        sclip(0x0000, 639,239);
        lcofs(0x0000,0,0);

        del = 4;

{
  int jump_address  = (font_pattern_word_counter + 3)*2;
        rel_adr_h = (short)( (jump_address >> 13L) & 0xffffL );
        rel_adr_l = (short)(  jump_address       & 0x1fffL );
}

        for( count = 0 ; count < 2 ; count++ ){

/* Zoom ------------------------------------ */
                array[0] =  150; array[1] =  100;
                array[2] =  165; array[3] =  100;
                array[4] =  165; array[5] =  115;
                array[6] =  150; array[7] =  115;

                for(i=0;i<=80;i++) {
                        clrscrn();

                        array[0] -=  del;     array[1] -=  del;
                        array[2] +=  del;     array[3] -=  del;
                        array[4] +=  del;     array[5] +=  del;
                        array[6] -=  del;     array[7] +=  del;

                        polygon4b(0x0040, 0,(15+3)*2, 24,24, array,
```

```
                                        backcolor,forecolor);      /* polygon4b */

                jump( 0x0040, rel_adr_h, rel_adr_l);           /* jump */

                /* Transfer Font Data "漢" ( 24 dots × 24 lines ) */
                for(lp=0;lp<font_pattern_word_counter;lp++) {
                        *DISPLIST_ptr++ = font_pattern[lp];
                }

                draw_end(DBmode,&first,quick);
                draw_start();
        }

/* Shrink ------------------------------------ */
        for(i=0;i<=80;i++) {
                clrscrn();

                array[0] +=  del;     array[1] +=  del;
                array[2] -=  del;     array[3] +=  del;
                array[4] -=  del;     array[5] -=  del;
                array[6] +=  del;     array[7] -=  del;

                polygon4b(0x0040, 0,(15+3)*2, 24,24, array,
                                        backcolor,forecolor);      /* polygon4b */

                jump( 0x0040, rel_adr_h, rel_adr_l);           /* jump */

                /* Transfer Font Data "漢" ( 24 dots × 24 lines ) */
                for(lp=0;lp<font_pattern_word_counter;lp++) {
                        *DISPLIST_ptr++ = font_pattern[lp];
                }

                draw_end(DBmode,&first,quick);
                draw_start();
        }

/* Rotate ------------------------------------ */
        for(d=0;d<=360;d+=5) {
                clrscrn();
```

```
                array[0] =  -60;     array[1] =  -70;
                array[2] =   40;     array[3] =  -70;
                array[4] =   40;     array[5] =   50;
                array[6] =  -60;     array[7] =   50;

                rad = PI * d / 180;
                for(i=0;i<=6;i+=2) {
                        tx =(short)  (array[i] * cos(rad)-  array[i+1] * sin(rad));
                        ty =(short)  (array[i] * sin(rad)+  array[i+1] * cos(rad));
                        array[i]  =  tx;
                        array[i+1] =  ty;
                }

                lcofs(0x0000, 160, 115);
                polygon4b(0x0040, 0,(15+3)*2, 24,24, array,
                                backcolor,forecolor);      /* polygon4b */

                jump( 0x0040, rel_adr_h, rel_adr_l);           /* jump */

                /* Transfer Font Data "漢" ( 24 dots × 24 lines ) */
                for(lp=0;lp<font_pattern_word_counter;lp++) {
                        *DISPLIST_ptr++ = font_pattern[lp];
                }

                lcofs(0x0000, 0, 0);

                draw_end(DBmode,&first,quick);
                draw_start();
        }

/* Move ------------------------------------- */
        h=260;
        j=110;
        for(d=0;d<=360;d+=5) {
                clrscrn();

                array[0] =  -50;     array[1] =  -60;
                array[2] =   50;     array[3] =  -60;
                array[4] =   50;     array[5] =   60;
                array[6] =  -50;     array[7] =   60;
```

```
                        rad = PI * d / 180;
                        for(i=0;i<=6;i+=2) {
                                tx =(short)  (array[i] * cos(rad)-  array[i+1] * sin(rad));
                                ty =(short)  (array[i] * sin(rad)+  array[i+1] * cos(rad));
                                array[i]   =  tx;
                                array[i+1] =  ty;
                        }

                        lcofs(0x0000, h, j);
                        polygon4b(0x0040, 0,(15+3)*2, 24,24, array,
                                        backcolor,forecolor);      /* polygon4b */

                        jump( 0x0040, rel_adr_h, rel_adr_l);           /* jump */

                        /* Transfer Font Data "漢" ( 24 dots × 24 lines ) */
                        for(lp=0;lp<font_pattern_word_counter;lp++) {
                                *DISPLIST_ptr++ = font_pattern[lp];
                        }

                        lcofs(0x0000, 0, 0);

                        h-=5;
                        draw_end(DBmode,&first,quick);
                        draw_start();
                }

        }
        return;   /* EXIT */
}

void clrscrn(void)
{
        short array[8];

/* ploygon4c (clear) ------------------------------------------------------ */
        array[0] =   0;     array[1] =   0;
        array[2] =  639;     array[3] =   0;
        array[4] =  639;     array[5] =  239;
        array[6] =   0;     array[7] =  239;
```

```
        polygon4c(0x0000, array, 0x0000);
}


void ginit(short DBmode)
{                                   /* Q2SD */
  unsigned short hsw =  64;
  unsigned short xs  = 131;
  unsigned short xw  = 640;
  unsigned short hc  = 910;

  unsigned short vsw =   3;
  unsigned short ys  =  16;
  unsigned short yw  = 240;
  unsigned short vc  = 262;

        outport(_Q2SYSR, 0x4080);                       /* Initilaize Draw/Display */
        outport(_Q2SRCR, 0xfe80);                       /* Clear SR register       */

        /* Set InterFace Control Registers */
        outport(_Q2IER ,   0x0000);
        outport(_Q2MEMR,   0x0031);
        outport(_Q2DSMR,   0x0005);
        outport(_Q2DSMR2,  0x0000);
        outport(_Q2REMR,   0x0041);
        outport(_Q2IEMR,   0x0000);

        /* Set Memory Control Regisers    */
        outport(_Q2DSX,                 xw);  /* Display size of x               */
        outport(_Q2DSY,                 yw);  /* Display size of y               */
        outport(_Q2DSA0,   0x0000);   /* Frame buffer 0 area start address(H) */
        outport(_Q2DSA1,   0x0010);   /* Frame buffer 1 area start address(H) */
        outport(_Q2DLSAH,  0x0019);   /* Display list area start address(H)   */
        outport(_Q2DLSAL,  0x0000);   /* Display list area start address(L)   */
        outport(_Q2SSAR,   0x0008);   /* Color area sorce start address(H)    */
        outport(_Q2WSAR,   0x001F);   /* Work area start address(H)           */
        outport(_Q2DMASH,  0x0000);   /* DMA transfer start address(H)        */
        outport(_Q2DMASL,  0x0000);   /* DMA transfer start address(L)        */
        outport(_Q2DMAWL,  0x0000);   /* DAM transfer word                    */
```

```
        /* Display Contral Registers */
        outport(_Q2HDS,              hsw+xs-11    );
        outport(_Q2HDE,              hsw+xs-11+xw );
        outport(_Q2VDS,              ys-2         );
        outport(_Q2VDE,              ys-2+yw      );
        outport(_Q2HSW,              hsw-1        );
        outport(_Q2HC,               hc-1         );
        outport(_Q2VSP,              vc-vsw-1     );
        outport(_Q2VC,               vc-1         );
        outport(_Q2DOR,              0x0000);
        outport(_Q2DOG_DOB,          0x007C);
        outport(_Q2CDR,              0x00FC);
        outport(_Q2CDG_CDB,          0xFCFC);

        /* Input Data Control Registers */
        outport(_Q2ISAH,   0x0000);
        outport(_Q2ISAL,   0x0000);
        outport(_Q2IDSX,   0x0000);
        outport(_Q2IDSY,   0x0000);
        outport(_Q2IDE,              0x0000);

        switch(DBmode) {                            /* Enable Draw/Display */
                case 0:
                        outport(_Q2SYSR, 0x2000);   /* Idle,Auto Change mode,No DMA */
                        break;
                case 1:
                        outport(_Q2SYSR, 0x2040);   /* Idle,Auto Renderring mode,No DMA */
                        break;
                default:
                        outport(_Q2SYSR, 0x2080);   /* Idle,manual change moe,No DMA */
        }
}

#include "MS7709.inc"
#include "Q2SDl.inc"
```

5.2.8 sample3.c のソースファイル

```
/*
                                        Q2 Display List / SHC sample program (1)
                        (Zoom/Shrink/Rotate/Move 'Solid' Polygon)

                        Copyright(c) Hitachi Ltd. 1999
*/


#include <machine.h>
#include <stdio.h>
#include <math.h>
#include "Q2SD_REG.h"
#include "Q2SD_mac.h"


unsigned short *DISPLIST_ptr;


#define  PI                             3.14159

#define _Q2_SSAR_REG_NO 0x00e

void  init_BSC(void);                                /* Initialize SH7709 bus state controller */
void  init_start(void);                              /* Initialize Display List Double Buffering (Only 1st) */
short draw_start(void);                              /* Initialize Display List Address pointer */
short draw_end(short DBmode,char *first,char quick); /* Display List Execution */
void  change_com_buffer(void);
/* ------------------------------------------------------ */

void  clrscrn(void);
void  ginit(short DBmode);                           /* Graphics System Open  (Q2SD Initialize) */


void main(void)
{
          long   count;
          short  array[8];
          short  d, i, h, j, k, del;
          double rad;
```

```
          short  tx,ty;
          short  DBmode;
/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
          char  first,quick;

          DBmode = 1;
          first  = ON;
          quick  = OFF;

          init_BSC();                              /* Initialize SH7709 bus state controller */

          ginit(DBmode);                           /* Initialize Q2 */
          init_start();                            /* Initialize Transfer Procedure */
          draw_start();                            /* Start Transfer Display List to UGM */
          sclip(0x0000, 639,239);
          lcofs(0x0000,0,0);

          wpr(0x0000, _Q2_SSAR_REG_NO, 0x10);      /* Sst the SSAR (0x100000) */

          del = 4;

          for( count = 0 ; count < 2 ; count++ ){

            /* POYLGON4A commnad reference to V1 area */
            short txs=640,tys=  0; /* Reference start point */
            short tdx=320,tdy=240; /* Reference size */

/* Zoom ------------------------------------ */
                    array[0] =  150;     array[1] =  100;
                    array[2] =  165;     array[3] =  100;
                    array[4] =  165;     array[5] =  115;
                    array[6] =  150;     array[7] =  115;

                    for(i=0;i<=80;i++) {
                           clrscrn();
```

```
                array[0] -= del;     array[1] -= del;
                array[2] += del;     array[3] -= del;
                array[4] += del;     array[5] += del;
                array[6] -= del;     array[7] += del;
                polygon4a(0x0000, txs,tys, tdx,tdy, array );   /* POLYGON4A */

                draw_end(DBmode,&first,quick);
                draw_start();
        }

/* Shrink ------------------------------------ */
        for(i=0;i<=80;i++) {
                clrscrn();

                array[0] += del;     array[1] += del;
                array[2] -= del;     array[3] += del;
                array[4] -= del;     array[5] -= del;
                array[6] += del;     array[7] -= del;
                polygon4a(0x0000, txs,tys, tdx,tdy, array );   /* POLYGON4A */

                draw_end(DBmode,&first,quick);
                draw_start();
        }

/* Rotate ------------------------------------- */
        for(d=0;d<=360;d+=5) {
                clrscrn();

                array[0] =  -60;     array[1] =  -70;
                array[2] =   40;     array[3] =  -70;
                array[4] =   40;     array[5] =   50;
                array[6] =  -60;     array[7] =   50;

                rad = PI * d / 180;
                for(i=0;i<=6;i+=2) {
                        tx =(short)  (array[i] * cos(rad)-  array[i+1] * sin(rad));
                        ty =(short)  (array[i] * sin(rad)+  array[i+1] * cos(rad));
                        array[i]   =  tx;
                        array[i+1] =  ty;
```

```
                        }

                        lcofs(0x0000, 160, 115);
                        polygon4a(0x0000, txs,tys, tdx,tdy, array ); /* POLYGON4A */
                        lcofs(0x0000, 0, 0);
                        draw_end(DBmode,&first,quick);
                        draw_start();
                }

/* Move ------------------------------------- */
                h=260;
                j=110;
                for(d=0;d<=360;d+=5) {
                        clrscrn();

                        array[0] =  -50;     array[1] =  -60;
                        array[2] =   50;     array[3] =  -60;
                        array[4] =   50;     array[5] =   60;
                        array[6] =  -50;     array[7] =   60;

                        rad = PI * d / 180;
                        for(i=0;i<=6;i+=2) {
                                tx =(short)  (array[i] * cos(rad)-  array[i+1] * sin(rad));
                                ty =(short)  (array[i] * sin(rad)+  array[i+1] * cos(rad));
                                array[i]  =  tx;
                                array[i+1] =  ty;
                        }

                        lcofs(0x0000, h, j);
                        polygon4a(0x0000, txs,tys, tdx,tdy, array );   /* POLYGON4A */
                        lcofs(0x0000, 0, 0);
                        h-=5;
                        draw_end(DBmode,&first,quick);
                        draw_start();
                }
        }
        return;                                                        /* EXIT */
}
```

```
void clrscrn(void)
{
          short array[8];

/* ploygon4c (clear) ---------------------------------------------------- */
          array[0] =   0;     array[1] =   0;
          array[2] =  639;     array[3] =   0;
          array[4] =  639;     array[5] =  239;
          array[6] =   0;     array[7] =  239;
          polygon4c(0x0000, array, 0x0000);
}

void ginit(short DBmode)
{               /* Q2SD */
  unsigned short hsw =  64;
  unsigned short xs  = 131;
  unsigned short xw  = 640;
  unsigned short hc  = 910;

  unsigned short vsw =  3;
  unsigned short ys  =  16;
  unsigned short yw  = 240;
  unsigned short vc  = 262;

          outport(_Q2SYSR, 0x4080);                       /* Initilaize Draw/Display */
          outport(_Q2SRCR, 0xfe80);                       /* Clear SR register      */

          /* Set InterFace Control Registers */
          outport(_Q2IER ,   0x0000);
          outport(_Q2MEMR,   0x0031);
          outport(_Q2DSMR,   0x0005);
          outport(_Q2DSMR2,  0x0000);
          outport(_Q2REMR,   0x0041);
          outport(_Q2IEMR,   0x0000);

          /* Set Memory Control Regisers    */
          outport(_Q2DSX,                 xw);  /* Display size of x                */
          outport(_Q2DSY,                 yw);  /* Display size of y                */
```

```
outport(_Q2DSA0,   0x0000);  /* Frame buffer 0 area start address(H) */
outport(_Q2DSA1,   0x0010);  /* Frame buffer 1 area start address(H) */
outport(_Q2DLSAH,  0x0019);  /* Display list area start address(H)   */
outport(_Q2DLSAL,  0x0000);  /* Display list area start address(L)   */
outport(_Q2SSAR,   0x0008);  /* Color area sorce start address(H)    */
outport(_Q2WSAR,   0x001F);  /* Work area start address(H)           */
outport(_Q2DMASH,  0x0000);  /* DMA transfer start address(H)        */
outport(_Q2DMASL,  0x0000);  /* DMA transfer start address(L)        */
outport(_Q2DMAWL,  0x0000);  /* DAM transfer word                    */

/* Display Contral Registers */
outport(_Q2HDS,              hsw+xs-11    );
outport(_Q2HDE,              hsw+xs-11+xw );
outport(_Q2VDS,              ys-2         );
outport(_Q2VDE,              ys-2+yw      );
outport(_Q2HSW,              hsw-1        );
outport(_Q2HC,               hc-1         );
outport(_Q2VSP,              vc-vsw-1     );
outport(_Q2VC,               vc-1         );
outport(_Q2DOR,              0x0000);
outport(_Q2DOG_DOB,          0x007C);
outport(_Q2CDR,              0x00FC);
outport(_Q2CDG_CDB,          0xFCFC);

/* Input Data Control Registers */
outport(_Q2ISAH,   0x0000);
outport(_Q2ISAL,   0x0000);
outport(_Q2IDSX,   0x0000);
outport(_Q2IDSY,   0x0000);
outport(_Q2IDE,    0x0000);

switch(DBmode) {                              /* Enable Draw/Display */
          case 0:
                 outport(_Q2SYSR, 0x2000);      /* Idle,Auto Change mode,No DMA */
                 break;
          case 1:
                 outport(_Q2SYSR, 0x2040);      /* Idle,Auto Renderring mode,No DMA */
                 break;
          default:
```

```
                        outport(_Q2SYSR, 0x2080);       /* Idle,manual change moe,No DMA */
        }
}

#include "MS7709.inc"
#include "Q2SDl.inc"
```

5.2.9 sample4.c のソースファイル

```
/*
                                        Q2 Display List / SHC sample program
                        (Draw hexagon with no pattern)

                        Copyright(c) Hitachi Ltd. 1999
*/
#include <machine.h>
#include <stdio.h>
#include "Q2SD_REG.h"
#include "Q2SD_mac.h"


unsigned short *DISPLIST_ptr;


#define   MAX_WORD  512

void  init_BSC(void);                                   /* Initialize SH7709 bus state controller */
void  init_start(void);                                 /* Initilaze Display List Double Buffering (Only 1st) */
short draw_start(void);                                 /* Initialize Display List Address pointer */
short draw_end(short DBmode,char *first,char quick);    /* Display List Execution */
void  change_com_buffer(void);
/* ------------------------------------------------------ */

void  ginit(short DBmode);                              /* Graphics System Open  (Q2 Initialize) */


void main(void)
{
          short array[MAX_WORD], poly[MAX_WORD];
          short DBmode;
          short xmin,xmax,ymin,ymax;
          short location_of_base_line;
          short fill_color = 0x0404;
          short fill_point = 6;
/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
```

```
 DBmode: 2 ... Manual Change
*/
          char  first,quick;

          DBmode = 1;
          first  = ON;
          quick  = OFF;

          init_BSC();                              /* Initialize SH7709 bus state controller */

          ginit(DBmode);                           /* Initialize Q2 */

          init_start();                            /* Initialize Transfer Procedure */
          draw_start();                            /* Start Transfer Display List to UGM */
          sclip(0x0000, 639,239);
          lcofs(0x0000,0,0);

/* ploygon4c ( clear screen ) ------------------------------------------- */
          array[0] =   0;     array[1] =   0;      /* Left  Up    */
          array[2] = 639;     array[3] =   0;      /* Right Up    */
          array[4] = 639;     array[5] = 239;      /* Left  Down  */
          array[6] =   0;     array[7] = 239;      /* Right Down  */
          polygon4c(0x0000, array, 0x0000);        /* polygon4c   */

          array[0] =   0;     array[1] =   0;      /* Left  Up    */
          array[2] = 159;     array[3] =   0;      /* Right Up    */
          array[4] = 159;     array[5] = 119;      /* Left  Down  */
          array[6] =   0;     array[7] = 119;      /* Right Down  */
          polygon4c(0x0000, array, 0xffff);        /* polygon4c   */

          array[0] = 160;     array[1] = 120;      /* Left  Up    */
          array[2] = 319;     array[3] = 120;      /* Right Up    */
          array[4] = 319;     array[5] = 239;      /* Left  Down  */
          array[6] = 160;     array[7] = 239;      /* Right Down  */
          polygon4c(0x0000, array, 0xffff);        /* polygon4c   */

/* Define hexagon ------------------------------------------------------- */
          array[ 0] =  40;     array[ 1] = 110;
          array[ 2] = 130;     array[ 3] =  70;
          array[ 4] = 190;     array[ 5] =  80;
```

```
          array[ 6] =  200;     array[ 7] =  150;
          array[ 8] =  140;     array[ 9] =  210;
          array[10] =   60;     array[11] =  160;
          array[12] = array[ 0];     array[13] = array[1];

/* Draw hexagon at work screen ---------------------------------------------- */
          xmin =  40; /* minimum of x */
          xmax = 200; /* maximum of x */
          ymin =  70; /* minimum of y */
          ymax = 210; /* maximum of y */

          location_of_base_line = xmin;

          /* Clear work screen for ftrap */
          clrw(0x0000, xmin, ymin, xmax, ymax);                          /* clrw */

          /* Draw hexagon at work screen */
          ftrap(0x0000, fill_point+1, location_of_base_line, array);     /* ftrap */

/* Draw hexagon with no pattern at rendering screen referencing
                                                                         to work screen -*/
          poly[ 0] =  xmin;     poly[ 1] =  ymin;
          poly[ 2] =  xmax;     poly[ 3] =  ymin;
          poly[ 4] =  xmax;     poly[ 5] =  ymax;
          poly[ 6] =  xmin;     poly[ 7] =  ymax;

          polygon4c(0x0021, poly, fill_color);                           /* polygon4c */

          line(0x0000, fill_color, fill_point+1, array);                 /* line */

          draw_end(DBmode,&first,quick);

          return;                                                        /* EXIT */
}


void ginit(short DBmode)
{              /* Q2SD */
  unsigned short hsw =  64;
  unsigned short xs  = 131;
```

```
unsigned short xw  = 640;
unsigned short hc  = 910;

unsigned short vsw =  3;
unsigned short ys  =  16;
unsigned short yw  = 240;
unsigned short vc  = 262;

        outport(_Q2SYSR, 0x4080);                                   /* Initilaize Draw/Display */
        outport(_Q2SRCR, 0xfe80);                                   /* Clear SR register      */

        /* Set InterFace Control Registers */
        outport(_Q2IER ,   0x0000);
        outport(_Q2MEMR,   0x0031);
        outport(_Q2DSMR,   0x0005);
        outport(_Q2DSMR2,  0x0000);
        outport(_Q2REMR,   0x0041);
        outport(_Q2IEMR,   0x0000);

        /* Set Memory Control Regisers    */
        outport(_Q2DSX,       xw);    /* Display size of x                   */
        outport(_Q2DSY,       yw);    /* Display size of y                   */
        outport(_Q2DSA0,   0x0000);   /* Frame buffer 0 area start address(H) */
        outport(_Q2DSA1,   0x0010);   /* Frame buffer 1 area start address(H) */
        outport(_Q2DLSAH,  0x0019);   /* Display list area start address(H)   */
        outport(_Q2DLSAL,  0x0000);   /* Display list area start address(L)   */
        outport(_Q2SSAR,   0x0008);   /* Color area sorce start address(H)    */
        outport(_Q2WSAR,   0x001F);   /* Work area start address(H)           */
        outport(_Q2DMASH,  0x0000);   /* DMA transfer start address(H)        */
        outport(_Q2DMASL,  0x0000);   /* DMA transfer start address(L)        */
        outport(_Q2DMAWL,  0x0000);   /* DAM transfer word                    */

        /* Display Contral Registers */
        outport(_Q2HDS,              hsw+xs-11   );
        outport(_Q2HDE,              hsw+xs-11+xw );
        outport(_Q2VDS,              ys-2        );
        outport(_Q2VDE,              ys-2+yw     );
        outport(_Q2HSW,              hsw-1       );
        outport(_Q2HC,               hc-1        );
```

```
        outport(_Q2VSP,              vc-vsw-1    );
        outport(_Q2VC,               vc-1        );
        outport(_Q2DOR,              0x0000);
        outport(_Q2DOG_DOB,          0x007C);
        outport(_Q2CDR,              0x00FC);
        outport(_Q2CDG_CDB,          0xFCFC);

        /* Input Data Control Registers */
        outport(_Q2ISAH,    0x0000);
        outport(_Q2ISAL,    0x0000);
        outport(_Q2IDSX,    0x0000);
        outport(_Q2IDSY,    0x0000);
        outport(_Q2IDE,     0x0000);

        switch(DBmode) {                            /* Enable Draw/Display */
                case 0:
                        outport(_Q2SYSR, 0x2000);   /* Idle,Auto Change mode,No DMA */
                        break;
                case 1:
                        outport(_Q2SYSR, 0x2040);   /* Idle,Auto Renderring mode,No DMA */
                        break;
                default:
                        outport(_Q2SYSR, 0x2080);   /* Idle,manual change moe,No DMA */
        }
}

#include "MS7709.inc"
#include "Q2SDl.inc"
```

5.2.10 sample5.c のソースファイル

```
/*
                                        Q2 Display List / SHC sample program (5)

                          ( Move 'Solid' Polygon )

                           Copyright(c) Hitachi Ltd. 1999
*/
#include <machine.h>
#include <stdio.h>
#include <math.h>
#include "Q2SD_REG.h"
#include "Q2SD_mac.h"


unsigned short *DISPLIST_ptr;


#define   PI    3.14159

#define _Q2_REMR_REG_NO 0x006
#define _Q2_RSAR_REG_NO 0x04C


void  init_BSC(void);                                /* Initialize SH7709 bus state controller */
void  init_start(void);                              /* Initialize Display List Double Buffering (Only 1st) */
short draw_start(void);                              /* Initialize Display List Address pointer */
short draw_end(short DBmode,char *first,char quick); /* Display List Execution */
void  change_com_buffer(void);
/* ------------------------------------------------------ */

void  ginit(short DBmode);                           /* Graphics System Open  (Q2 Initialize) */

void main(void)
{
          long   count;
          short array[8];
          short DBmode;
/*
 DBmode: 0 ... Auto Change
```

```
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
          char  first,quick;

          DBmode = 1;
          first  = ON;
          quick  = OFF;

          init_BSC();                              /* Initialize SH7709 bus state controller */

          ginit(DBmode);                           /* Initialize Q2 */


          init_start();                            /* Initialize Transfer Procedure */
          draw_start();                            /* Start Transfer Display List to UGM */
          sclip(0x0000, 639,239);
          lcofs(0x0000,0,0);

          for( count = 0 ; count < 2 ; count++ ){

          short i,d;

          /* MOVE */
                    short kj = 50;
                    double rad;

                    for(d=0;d<=360;d+=5) {

                    /* Job No.1 : clear screen */
                           array[0] =   0;     array[1] =   0;
                           array[2] =  639;    array[3] =   0;
                           array[4] =  639;    array[5] =  239;
                           array[6] =   0;     array[7] =  239;
                           polygon4c(0x0000, array, 0x0606);


                    /* Set drawing start address in the RSAR */
                           if ( (inport( _Q2SR ) & 0x0100) != 0 ) {
                                  wpr( 0x0000, _Q2_RSAR_REG_NO, inport( _Q2DSA0 ) );
```

```
        } else {
                wpr( 0x0000, _Q2_RSAR_REG_NO, inport( _Q2DSA1 ) );
        }

/* Enable the RSAR */
        wpr( 0x0000, _Q2_REMR_REG_NO, inport(_Q2REMR) | 0x8000 );


/* Job No.2 : draw two solid rectangles */
        array[0] =  -50;     array[1] =  -(60);
        array[2] =   50;     array[3] =  -(60);
        array[4] =   50;     array[5] =  -(-60);
        array[6] =  -50;     array[7] =  -(-60);

        rad = PI * d / 180;
        for(i=0;i<=6;i+=2) {
                short tx =(short)  (array[i] * cos(rad)-  array[i+1] * sin(rad));
                short ty =(short)  (array[i] * sin(rad)+  array[i+1] * cos(rad));
                array[i]  =  tx;
                array[i+1] =  ty;
        }

        lcofs(0x0000, kj,100);
        polygon4c(0x0000,array,0xF800);                /* POLYGON4C */

        lcofs(0x0000, 320-kj,100);
        polygon4c(0x0000,array,0xF800);                /* POLYGON4C */

        lcofs(0x0000, 0,0);


/* Disable the RSAR */
        wpr( 0x0000, _Q2_REMR_REG_NO, inport(_Q2REMR) & 0x7fff );


        kj+=5;

/* Drawing start */
        draw_end(DBmode,&first,quick);
        draw_start();
```

```
                } /* for */
        }
        return;                                                 /* EXIT */
}


void ginit(short DBmode)
{               /* Q2SD */
  unsigned short hsw =  64;
  unsigned short xs  = 131;
  unsigned short xw  = 640;
  unsigned short hc  = 910;

  unsigned short vsw =   3;
  unsigned short ys  =  16;
  unsigned short yw  = 240;
  unsigned short vc  = 262;

        outport(_Q2SYSR, 0x4080);         /* Initilaize Draw/Display  */
        outport(_Q2SRCR, 0xfe80);         /* Clear SR register        */

        /* Set InterFace Control Registers */
        outport(_Q2IER ,   0x0000);
        outport(_Q2MEMR,   0x0031);
        outport(_Q2DSMR,   0x0005);
        outport(_Q2DSMR2,  0x0000);
        outport(_Q2REMR,   0x0041);
        outport(_Q2IEMR,   0x0000);

        /* Set Memory Control Regisers    */
        outport(_Q2DSX,       xw);    /* Display size of x                */
        outport(_Q2DSY,       yw);    /* Display size of y                */
        outport(_Q2DSA0,   0x0000);   /* Frame buffer 0 area start address(H) */
        outport(_Q2DSA1,   0x0010);   /* Frame buffer 1 area start address(H) */
        outport(_Q2DLSAH,  0x0019);   /* Display list area start address(H)   */
        outport(_Q2DLSAL,  0x0000);   /* Display list area start address(L)   */
        outport(_Q2SSAR,   0x0008);   /* Color area sorce start address(H)    */
        outport(_Q2WSAR,   0x001F);   /* Work area start address(H)           */
```

```
        outport(_Q2DMASH,  0x0000);   /* DMA transfer start address(H)        */
        outport(_Q2DMASL,  0x0000);   /* DMA transfer start address(L)        */
        outport(_Q2DMAWL,  0x0000);   /* DAM transfer word                    */

        /* Display Contral Registers */
        outport(_Q2HDS,               hsw+xs-11    );
        outport(_Q2HDE,               hsw+xs-11+xw );
        outport(_Q2VDS,               ys-2         );
        outport(_Q2VDE,               ys-2+yw      );
        outport(_Q2HSW,               hsw-1        );
        outport(_Q2HC,                hc-1         );
        outport(_Q2VSP,               vc-vsw-1     );
        outport(_Q2VC,                vc-1         );
        outport(_Q2DOR,               0x0000);
        outport(_Q2DOG_DOB,           0x007C);
        outport(_Q2CDR,               0x00FC);
        outport(_Q2CDG_CDB,           0xFCFC);

        /* Input Data Control Registers */
        outport(_Q2ISAH,   0x0000);
        outport(_Q2ISAL,   0x0000);
        outport(_Q2IDSX,   0x0000);
        outport(_Q2IDSY,   0x0000);
        outport(_Q2IDE,    0x0000);

        switch(DBmode) {                          /* Enable Draw/Display */
                case 0:
                        outport(_Q2SYSR, 0x2000);      /* Idle,Auto Change mode,No DMA */
                        break;
                case 1:
                        outport(_Q2SYSR, 0x2040);      /* Idle,Auto Renderring mode,No DMA */
                        break;
                default:
                        outport(_Q2SYSR, 0x2080);      /* Idle,manual change moe,No DMA */
        }
}

#include "MS7709.inc"
#include "Q2SDl.inc"
```

5.2.11 sample6.c のソースファイル

```
/*
                                        Q2 Display List / SHC sample program (6)
                        ( Move 'Image')

                Must be load "bolls.bmp" to (txs,tys) = (0,832), before.
                SSAR = 0x80000

                        Copyright(c) Hitachi Ltd. 1999
*/
#include <machine.h>
#include <stdio.h>
#include <math.h>
#include "Q2SD_REG.h"
#include "Q2SD_mac.h"


unsigned short *DISPLIST_ptr;


static short speed[]={2,5,4,8,3,4,6,1,7,5,
              6,9,2,4,3,6,5,4,2,6,
              7,2,4,3,1,5,1,4,9,3,
              4,9,7,2,4,2,3,7,2,4,
              6,4,3,8,2,4,6,8,6,3,
              6,2,7,6,3,5,4,1,6,6,
              2,5,4,6,3,9,6,6,2,7,
              6,4,3,9,5,1,3,7,5,1,
              6,3,8,4,1,1,6,2,5,9,
              3,6,1,5,3,4,5,6,9,6
              };

static unsigned short color_table[]={
                                     (255/8)*2048 | (128/4)*32 | ( 64/8),
                                     (255/8)*2048 | (  0/4)*32 | (128/8),
                                     (128/8)*2048 | (128/4)*32 | (255/8),
                                     (128/8)*2048 | (255/4)*32 | (255/8),
                                     (  0/8)*2048 | (255/4)*32 | (128/8),
                                     (255/8)*2048 | (255/4)*32 | (128/8),
```

```
                                    (255/8)*2048 | (  0/4)*32 | (255/8),
                                    (  0/8)*2048 | (  0/4)*32 | (255/8),
                                    (255/8)*2048 | (  0/4)*32 | (  0/8),
                                    (128/8)*2048 | (128/4)*32 | (128/8)
};

#define OFF               0
#define ON                1

void  init_BSC(void);                                /* Initialize SH7709 bus state controller */
void  init_start(void);                              /* Initilaze Display List Double Buffering (Only 1st) */
short draw_start(void);                              /* Initialize Display List Address pointer */
short draw_end(short DBmode,char *first,char quick);  /* Display List Execution */
void  change_com_buffer(void);
/* ------------------------------------------------------ */

void  clrscrn(void);
void  ginit(short DBmode);                           /* Graphics System Open  (Q2 Initialize) */


void main(void)
{
  long  count;
  short block_count = 100;
  short lp;
  short dx[100], dy[100];
  short array[100][8];
  short DBmode;
  short ah,al;
  short size_x=16, size_y=16;
  char first,quick;

          DBmode = 1;
          first  = ON;
          quick  = OFF;

          init_BSC();                                /* Initialize SH7709 bus state controller */

          ginit(DBmode);                             /* Initialize Q2 */
```

```
/* Transfer Data "●" ( 16 dots × 16 lines ) */
{
  short i;
  unsigned short *address;
  unsigned long SrcAdr;
  unsigned short crcl_pattern[]={
          /* (MSB LSB) */
            0x0000,
            0x07e0,
            0x1ff8,
            0x3ffc,
            0x3ffc,
            0x7ffe,
            0x7ffe,
            0x7ffe,
            0x7ffe,
            0x7ffe,
            0x7ffe,
            0x3ffc,
            0x3ffc,
            0x1ff8,
            0x07e0,
            0x0000
       };

           SrcAdr = 0x006000L + UGMBASE;
           address = (unsigned short *)SrcAdr;
           for(i=0;i<16;i++) {
                   *address++ = crcl_pattern[i];
           }

           ah = (short)( ((SrcAdr-UGMBASE) >> 13L) & 0xffffL );
           al = (short)(  (SrcAdr-UGMBASE) & 0x1fffL );
}

for(lp=0; lp<block_count; lp++){
           dx[lp] = speed[lp];
           dy[lp] = speed[lp];
```

```
                array[lp][0] = 0  +lp*speed[lp]/2;   array[lp][1] = 0 +lp*speed[lp]/2;
                array[lp][2] = size_x-1+lp*speed[lp]/2; array[lp][3] = 0+lp*speed[lp]/2;
                array[lp][4] = size_x-1+lp*speed[lp]/2; array[lp][5] = size_y-1+lp*speed[lp]/2;
                array[lp][6] = 0  +lp*speed[lp]/2;   array[lp][7] = size_y-1+lp*speed[lp]/2;
        }

        init_start();             /* Initialize Transfer Procedure */
        draw_start();             /* Start Transfer Display List to UGM */
        sclip(0x0000, 639,239);
        lcofs(0x0000,0,0);

        for( count = 0 ; count < 1000 ; count++ ){
          short tmp_dx;

                clrscrn();
                for(lp=0; lp<block_count; lp++){
                        if ( array[lp][0] <   0 ) dx[lp] =  speed[lp];
                        if ( array[lp][2] > 319 ) dx[lp] = -speed[lp];
                        if ( array[lp][1] <   0 ) dy[lp] =  speed[lp];
                        if ( array[lp][7] > 239 ) dy[lp] = -speed[lp];

                        array[lp][0] += dx[lp];   array[lp][1] += dy[lp];
                        array[lp][2] += dx[lp];   array[lp][3] += dy[lp];
                        array[lp][4] += dx[lp];   array[lp][5] += dy[lp];
                        array[lp][6] += dx[lp];   array[lp][7] += dy[lp];

                        if(dx[lp] < 0)
                                tmp_dx = -dx[lp];
                        else
                                tmp_dx = dx[lp];
                        polygon4b(0x0200, ah, al, 16, 16, array[lp], 0x0000, color_table[tmp_dx-1]);
                }

                draw_end(DBmode,&first,quick);
                draw_start();

        }
}

static void clrscrn(void)
```

```
{
          short array[8];

/* ploygon4c (clear) ---------------------------------------------------- */
          array[0] =   0;     array[1] =   0;
          array[2] =  639;     array[3] =   0;
          array[4] =  639;     array[5] =  239;
          array[6] =   0;     array[7] =  239;
          polygon4c(0x0000, array, 0x0000);
}


void ginit(short DBmode)
{               /* Q2SD */
  unsigned short hsw =  64;
  unsigned short xs  = 131;
  unsigned short xw  = 640;
  unsigned short hc  = 910;

  unsigned short vsw =   3;
  unsigned short ys  =  16;
  unsigned short yw  = 240;
  unsigned short vc  = 262;

          outport(_Q2SYSR, 0x4080);                       /* Initilaize Draw/Display */
          outport(_Q2SRCR, 0xfe80);                       /* Clear SR register      */

          /* Set InterFace Control Registers */
          outport(_Q2IER ,   0x0000);
          outport(_Q2MEMR,   0x0031);
          outport(_Q2DSMR,   0x0005);
          outport(_Q2DSMR2,  0x0000);
          outport(_Q2REMR,   0x0041);
          outport(_Q2IEMR,   0x0000);

          /* Set Memory Control Regisers    */
          outport(_Q2DSX,       xw);   /* Display size of x                   */
          outport(_Q2DSY,       yw);   /* Display size of y                   */
          outport(_Q2DSA0,   0x0000);  /* Frame buffer 0 area start address(H) */
```

```
outport(_Q2DSA1,   0x0010);  /* Frame buffer 1 area start address(H) */
outport(_Q2DLSAH,  0x0019);  /* Display list area start address(H)   */
outport(_Q2DLSAL,  0x0000);  /* Display list area start address(L)   */
outport(_Q2SSAR,   0x0008);  /* Color area sorce start address(H)    */
outport(_Q2WSAR,   0x001F);  /* Work area start address(H)           */
outport(_Q2DMASH,  0x0000);  /* DMA transfer start address(H)        */
outport(_Q2DMASL,  0x0000);  /* DMA transfer start address(L)        */
outport(_Q2DMAWL,  0x0000);  /* DAM transfer word                    */

/* Display Contral Registers */
outport(_Q2HDS,              hsw+xs-11    );
outport(_Q2HDE,              hsw+xs-11+xw );
outport(_Q2VDS,              ys-2         );
outport(_Q2VDE,              ys-2+yw      );
outport(_Q2HSW,              hsw-1        );
outport(_Q2HC,               hc-1         );
outport(_Q2VSP,              vc-vsw-1     );
outport(_Q2VC,               vc-1         );
outport(_Q2DOR,              0x0000);
outport(_Q2DOG_DOB,          0x007C);
outport(_Q2CDR,              0x00FC);
outport(_Q2CDG_CDB,          0xFCFC);

/* Input Data Control Registers */
outport(_Q2ISAH,   0x0000);
outport(_Q2ISAL,   0x0000);
outport(_Q2IDSX,   0x0000);
outport(_Q2IDSY,   0x0000);
outport(_Q2IDE,    0x0000);

switch(DBmode) {                              /* Enable Draw/Display */
          case 0:
                 outport(_Q2SYSR, 0x2000);      /* Idle,Auto Change mode,No DMA */
                 break;
          case 1:
                 outport(_Q2SYSR, 0x2040);      /* Idle,Auto Renderring mode,No DMA */
                 break;
          default:
                 outport(_Q2SYSR, 0x2080);      /* Idle,manual change moe,No DMA */
```

```
        }
}

#include "MS7709.inc"
#include "Q2SDl.inc"
```

5.2.12 sample7.c のソースファイル

```
/*
                                    Q2 Display List / SHC sample program (7)
                        (Zoom/Shrink/Rotate/Move 'Text 24×24')

                                    Copyright(c) Hitachi Ltd. 1999

                        CLK1 = 14.32 MHz

                        Color depth  ; 16bit/pixel
                        Display size : 640 pixel X 480 line
                        Scan mode    : Interrace sync & video (TVM = 11)

                        Frame 0 start addtess          : 0x000000          (  0,   0)
                        Frame 1 start address          : 0x100000          (  0, 512)
                        Display list 0 start address   : 0x0F0000          (  0, 480)
                        Display list 1 start address   : 0x1F0000          (  0, 992)
                        Color sorce area start address : Undefine
                        Mono pattern area start address: 0x006000          (640,   0)
                        Work area start address        : Undefine
*/
#include <machine.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Q2SD_REG.h"
#include "Q2SD_mac.h"


#undef      DISPLIST0
#undef      DISPLIST1

#define     DISPLIST0 0x0F0000L        /* (0,480) */
#define     DISPLIST1 0x1F0000L        /* (0,992) */


unsigned short *DISPLIST_ptr;

#define   PI                           3.14159
```

```
void  init_BSC(void);                                   /* Initialize SH7709 bus state controller */
void  init_start(void);                                 /* Initilaze Display List Double Buffering (Only 1st) */
short draw_start(void);                                 /* Initialize Display List Address pointer */
short draw_end(short DBmode,char *first,char quick);    /* Display List Execution */
void  change_com_buffer(void);

/* ------------------------------------------------------ */

void  clrscrn(void);
void  ginit(short DBmode);                              /* Graphics System Open  (Q2 Initialize) */


void main()
{
          long  count;
          short array[8];
          short ah,al;
          short d, i, h, j, k, del;
          double rad;
          short tx,ty;
          short txs, tys;
          short DBmode;
/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
          char  first,quick;
          int       SrcAdr;         /* Source Address */
          short     forecolor;      /* charactor  color */
          short     backcolor;      /* background color */

          DBmode = 1;
          first  = ON;
          quick  = OFF;

          init_BSC();               /* Initialize SH7709 bus state controller */

          ginit(DBmode);            /* Initialize Q2 */
```

```
/* Transfer Font Data "漢" ( 24 dots × 24 lines ) ----------------------- */
{
  short i;
  unsigned short *address;
  unsigned short font_pattern[]={
       /*---------------------------------------------*/
       /* Note : Q2 uses font pattern from LSB to MSB. */
       /*---------------------------------------------*/
       /* 'font_pattern' must be mapped at SuperH's big endian area. */

       /* (MSB  LSB)   (MSB  LSB)  (MSB  LSB)                 */
             0x1c00,      0x0e07,     0x430c,
             0x0c1c,      0xd8e3,     0xffff,
             0x0c00,      0x0003,     0x030c,
             0x0140,      0x4718,     0x3fff,
             0x636e,      0x2c18,     0x1863,
             0x6320,      0x3018,     0x1fff,
             0x6310,      0x1800,     0x1860,
             0xff98,      0x0c3f,     0x0060,
             0x600f,      0xec60,     0xffff,
             0xb00c,      0x0c01,     0x0338,
             0x1c0c,      0x0c0e,     0x3c0e,
             0x038c,      0xecf8,     0x6000
       };

          SrcAdr = 0x6000L + UGMBASE; /* (640,0) */
          address = (unsigned short *)SrcAdr;
          forecolor = 0x0F0F;
          backcolor = 0x0101;
          for(i=0;i<36;i++) {
                    *address++ = font_pattern[i];
          }

}

/* ---------------------------------------------------------------------- */

          init_start();             /* Initialize Transfer Procedure */
```

```
        draw_start();             /* Start Transfer Display List to UGM */
        sclip(0x0000, 639,479);
        lcofs(0x0000,0,0);

        del = 4;
        ah = (short)( ((SrcAdr-UGMBASE) >> 13L) & 0xffffL );
        al = (short)(  (SrcAdr-UGMBASE) & 0x1fffL );

        for( count = 0 ; count < 2 ; count++ ){

/* Zoom ----------------------------------- */
        array[0] =  200;     array[1] =  200;
        array[2] =  215;     array[3] =  200;
        array[4] =  215;     array[5] =  315;
        array[6] =  200;     array[7] =  315;

                for(i=0;i<=45;i++) {
                        clrscrn();

                        array[0] -=  del;     array[1] -=  del;
                        array[2] +=  del;     array[3] -=  del;
                        array[4] +=  del;     array[5] +=  del;
                        array[6] -=  del;     array[7] +=  del;
                        polygon4b(0x0000, ah,al, 24,24, array,
                                                backcolor,forecolor);     /* polygon4b */

                        draw_end(DBmode,&first,quick);
                        draw_start();
                }

/* Shrink --------------------------------- */
                for(i=0;i<=45;i++) {
                        clrscrn();

                        array[0] +=  del;     array[1] +=  del;
                        array[2] -=  del;     array[3] +=  del;
                        array[4] -=  del;     array[5] -=  del;
                        array[6] +=  del;     array[7] -=  del;
                        polygon4b(0x0000, ah,al, 24,24, array,
                                                backcolor,forecolor);     /* polygon4b */
```

```
                draw_end(DBmode,&first,quick);
                draw_start();
        }

/* Rotate ------------------------------------ */
        for(d=0;d<=360;d+=8) {
                clrscrn();

                array[0] =  -60;     array[1] =  -70;
                array[2] =   40;     array[3] =  -70;
                array[4] =   40;     array[5] =   50;
                array[6] =  -60;     array[7] =   50;

                rad = PI * d / 180;
                for(i=0;i<=6;i+=2) {
                        tx =(short)  (array[i] * cos(rad)-  array[i+1] * sin(rad));
                        ty =(short)  (array[i] * sin(rad)+  array[i+1] * cos(rad));
                        array[i]  =  tx;
                        array[i+1] =  ty;
                }

                lcofs(0x0000, 160, 115);
                polygon4b(0x0000, ah,al, 24,24, array,
                                backcolor,forecolor);     /* polygon4b */
                lcofs(0x0000, 0, 0);
                draw_end(DBmode,&first,quick);
                draw_start();
        }

/* Move -------------------------------------- */
        h=600;
        j=110;
        for(d=0;d<=720;d+=8) {
                clrscrn();

                array[0] =  -50;     array[1] =  -60;
                array[2] =   50;     array[3] =  -60;
                array[4] =   50;     array[5] =   60;
```

```
                        array[6] =  -50;     array[7] =   60;

                        rad = PI * d / 180;
                        for(i=0;i<=6;i+=2) {
                                tx =(short)  (array[i] * cos(rad)-  array[i+1] * sin(rad));
                                ty =(short)  (array[i] * sin(rad)+  array[i+1] * cos(rad));
                                array[i]   =  tx;
                                array[i+1] =  ty;
                        }

                        lcofs(0x0000, h, j);
                        polygon4b(0x0000, ah,al, 24,24, array,
                                              backcolor,forecolor);     /* polygon4b */
                        lcofs(0x0000, 0, 0);
                        h-=10;
                        draw_end(DBmode,&first,quick);
                        draw_start();
                }

        }
        return;                                                         /* EXIT */
}

void clrscrn(void)
{
        short array[8];

/* ploygon4c (clear) ---------------------------------------------------- */
        array[0] =   0;     array[1] =   0;
        array[2] =  639;     array[3] =   0;
        array[4] =  639;     array[5] =  479;
        array[6] =   0;     array[7] =  479;
        polygon4c(0x0000, array, 0x0034);
}


void ginit(short DBmode)
{               /* Q2SD */
  unsigned short hsw =  42;
```

```
unsigned short xs  = 131;
unsigned short xw  = 640;
unsigned short hc  = 910;

unsigned short vsw =  3;
unsigned short ys  =  16;
unsigned short yw  = 240;
unsigned short vc  = 262;

        outport(_Q2SYSR, 0x4080);        /* Initilaize Draw/Display  */
        outport(_Q2SRCR, 0xfe00);        /* Clear SR register        */

        /* Set InterFace Control Registers */
        outport(_Q2IER ,   0x0000);
        outport(_Q2MEMR,   0x0031);
        outport(_Q2DSMR,   0x0035);
        outport(_Q2DSMR2,  0x0000);
        outport(_Q2REMR,   0x0041);
        outport(_Q2IEMR,   0x0000);

        /* Set Memory Control Regisers    */
        outport(_Q2DSX,       xw);   /* Display size of x                */
        outport(_Q2DSY,       yw);   /* Display size of y                */
        outport(_Q2DSA0,   0x0000);  /* Frame buffer 0 area start address(H) */
        outport(_Q2DSA1,   0x0010);  /* Frame buffer 1 area start address(H) */
        outport(_Q2DLSAH,  0x0019);  /* Display list area start address(H)   */
        outport(_Q2DLSAL,  0x0000);  /* Display list area start address(L)   */
        outport(_Q2SSAR,   0x0008);  /* Color area sorce start address(H)    */
        outport(_Q2WSAR,   0x001F);  /* Work area start address(H)           */
        outport(_Q2DMASH,  0x0000);  /* DMA transfer start address(H)        */
        outport(_Q2DMASL,  0x0000);  /* DMA transfer start address(L)        */
        outport(_Q2DMAWL,  0x0000);  /* DAM transfer word                    */

        /* Display Contral Registers */
        outport(_Q2HDS,             hsw+xs-11    );
        outport(_Q2HDE,             hsw+xs-11+xw );
        outport(_Q2VDS,             ys-2         );
        outport(_Q2VDE,             ys-2+yw      );
        outport(_Q2HSW,             hsw-1        );
```

```
        outport(_Q2HC,              hc-1        );
        outport(_Q2VSP,             vc-vsw-1    );
        outport(_Q2VC,              vc-1        );
        outport(_Q2DOR,             0x0000);
        outport(_Q2DOG_DOB,         0x007C);
        outport(_Q2CDR,             0x00FC);
        outport(_Q2CDG_CDB,         0xFCFC);

        /* Input Data Control Registers */
        outport(_Q2ISAH,   0x0000);
        outport(_Q2ISAL,   0x0000);
        outport(_Q2IDSX,   0x0000);
        outport(_Q2IDSY,   0x0000);
        outport(_Q2IDE,    0x0000);

        switch(DBmode) {                        /* Enable Draw/Display */
                case 0:
                        outport(_Q2SYSR, 0x2000);  /* Idle,Auto Change mode,No DMA,No Cache,7709 */
                        break;
                case 1:
                        outport(_Q2SYSR, 0x2040);  /* Idle,Auto Renderring mode,No DMA,No Cache,7709 */
                        break;
                default:
                        outport(_Q2SYSR, 0x2080);  /* Idle,manual change moe,No DMA,No Cache,7709 */
        }
}

#include "MS7709.inc"
#include "Q2SDl.inc"
```

5.2.13 sample8.c のソースファイル

```
/*_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/

            Q2 Display List / SHC sample program (8)
              ( Rotating 'a solid model' )
                                  ( A structure of solid model has 20 surfaces )

                Copyright(c) Hitachi Ltd. 1999

 _/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/*/

#include <machine.h>
#include <stdio.h>
#include <stdlib.h>
#include "Q2SD_REG.h"
#include "Q2SD_mac.h"


/* Coordinate3 */
          short point[][3] = {
                    0, 360, -2048,
                    0, 360, 256,
                    0, 448, 256,
                    0, 448, -2048,
                    2304, 360, -2048,
                    2304, 448, -2048,
                    2304, 360, 256,
                    2304, 448, 256,
                    224, 0, -32,
                    224, 0, -1792,
                    288, 0, -1792,
                    288, 0, -32,
                    224, 360, -1792,
                    288, 360, -1792,
                    224, 0, 32,
                    224, 360, 32,
                    288, 360, -32,
                    2016, 0, -32,
                    2016, 0, 32,
```

```
                2080, 0, 32,
                2080, 360, 32,
                2016, 360, -32,
                2016, 0, -1760,
                2080, 0, -1760,
                2080, 0, -1824,
                2080, 360, -1824,
                2016, 360, -1760,
                256, 0, -1760,
                256, 0, -1824,
                256, 360, -1824,
                256, 360, -1760
        };

/* IndexedFaceSet */
        short coordIndex[][5] = {
                        0, 1, 2, 3, -1,
                        4, 0, 3, 5, -1,
                        6, 4, 5, 7, -1,
                        1, 6, 7, 2, -1,
                        2, 7, 5, 3, -1,
                        1, 0, 4, 6, -1,
                        8, 9, 10, 11, -1,
                        10, 9, 12, 13, -1,
                        9, 14, 15, 12, -1,
                        11, 10, 13, 16, -1,
                        14, 8, 17, 18, -1,
                        14, 19, 20, 15, -1,
                        17, 11, 16, 21, -1,
                        18, 22, 23, 19, -1,
                        19, 24, 25, 20, -1,
                        22, 17, 21, 26, -1,
                        27, 28, 24, 23, -1,
                        24, 28, 29, 25, -1,
                        27, 22, 26, 30, -1,
                        28, 27, 30, 29, -1
        };
```

```
/* X,Y and Z are constance. */
#define X 0
#define Y 1
#define Z 2

#define _Z1 2
#define _Z2 5
#define _Z3 8
#define _Z4 11

/* N define count of surface. */
#define N 100

#define _DUMMY 0

short sin_table[91]={
   0, 142, 285, 428, 571, 713, 856, 998,1140,1281,
1422,1563,1703,1842,1981,2120,2258,2395,2531,2667,
2801,2935,3068,3200,3331,3462,3591,3719,3845,3971,
4096,4219,4341,4461,4580,4698,4815,4930,5043,5155,
5265,5374,5481,5586,5690,5792,5892,5991,6087,6182,
6275,6366,6455,6542,6627,6710,6791,6870,6947,7021,
7094,7164,7233,7299,7362,7424,7483,7540,7595,7647,
7697,7745,7791,7834,7874,7912,7948,7982,8012,8041,
8067,8091,8112,8130,8147,8160,8172,8180,8187,8190,8192
};


unsigned short *DISPLIST_ptr;



void  init_BSC(void);                                /* Initialize SH7709 bus state controller */
void  init_start(void);                              /* Initilaze Display List Double Buffering (Only 1st) */
short draw_start(void);                              /* Initialize Display List Address pointer */
short draw_end(short DBmode,char *first,char quick);   /* Display List Execution */
void  change_com_buffer(void);
/* ------------------------------------------------------- */

void clrscrn(void);
```

```
void ginit(short DBmode);                              /* Graphics System Open  (Q2 Initialize) */

/* This is function for rotation. */
void rotate_a_rectangle( short array[12], short angle, short center, short dx, short dy, short dz );

/* This is function for change from 3D to 2D. */
void convert_d3_into_d2( short array[12], short dx, short dy, short dz, short _z_size );

/* This is function for Z sorting. */
void z_sort( short array[N][12], short st[N], short n );

void set_polygon_sample(short array[N][12], short point[N][3],
                                        short coordIndex[N][5], short _n_point);


void main(void)
{
  long  count;
  char  first = ON;
  char  quick = OFF;
  short _z_size = 1024;

/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
  short DBmode = 1;

          init_BSC();                       /* Initialize SH7709 bus state controller */

          ginit(DBmode);                    /* Initialize Q2 */
          init_start();                     /* Initialize Transfer Procedure */
          draw_start();                     /* Start Transfer Display List to UGM */
          sclip(0x0000, 639,239);
          lcofs(0x0000,0,0);

          for( count = 0 ; count < 5 ; count++ ){

                    {
```

```
  short poly[N][12], st[N];
  short poly2[N][12], st2[N];
  short color[N];
  short zz = 300;
  short d;
  short n_point;
  short lp;

n_point = sizeof(coordIndex)/sizeof(short)/5;

for(lp=1;lp<=n_point;lp++)
        color[lp] = (((lp%16)<<8) & 0xff00) | (lp%16);

for(d=0;d<360*3;d+=5) {
        clrscrn();

        /* Object No.1 */
        set_polygon_sample(poly, point, coordIndex, n_point );

        /* Object No.2 */
        set_polygon_sample(poly2, point, coordIndex, n_point );

                for(lp=0;lp<n_point;lp++){
                        rotate_a_rectangle( poly[lp], d, X, _DUMMY, 20, -100 );
                        rotate_a_rectangle( poly[lp], d, Z, 0, 0, _DUMMY );

                        rotate_a_rectangle( poly2[lp], d+30, X, _DUMMY, 20, -100 );
                        rotate_a_rectangle( poly2[lp], d+30, Z, 0, 0, _DUMMY );
                }

                /* Z sorting */
                z_sort( poly, st, n_point );
                z_sort( poly2, st2, n_point );

                /* Convert 3D data into 2D data */
                for(lp=0;lp<n_point;lp++){
                        convert_d3_into_d2( poly[lp], 100, 100, zz, _z_size );
                        convert_d3_into_d2( poly2[lp], 100+125, 100-20, zz+200, _z_size );
                }
```

```
                                /* POLYGON4C */
                                for(lp=0;lp<n_point;lp++){
                                        polygon4c(0x0000, poly2[st2[lp]], color[st2[lp]] );
                                        polygon4c(0x0000, poly[st[lp]], color[st[lp]] );
                                }
                                draw_end(DBmode,&first,quick);
                                draw_start();
                        }
                }

        } /* for */
        return;                                                 /* EXIT */
}

void clrscrn(void)
{
  short array[8];

/* ploygon4c (clear) ------------------------------------------------- */
        array[0] =   0;     array[1] =   0;
        array[2] =  639;     array[3] =   0;
        array[4] =  639;     array[5] =  239;
        array[6] =   0;     array[7] =  239;
        polygon4c(0x0000, array, 0x0010);
}

void set_polygon_sample(short array[N][12], short point[N][3],
                                        short coordIndex[N][5], short _n_point)
{

#define _DIV 20

  short i;
  short loc = 0;

        for(i=0; i<_n_point; i++){

          /* X */
          array[i][loc++] = point[ coordIndex[i][0] ][0] / _DIV;
          /* Y */
```

```
            array[i][loc++] = point[ coordIndex[i][0] ][1] / _DIV;
            /* Z */
            array[i][loc++] = point[ coordIndex[i][0] ][2] / _DIV;

            /* X */
            array[i][loc++] = point[ coordIndex[i][1] ][0] / _DIV;
            /* Y */
            array[i][loc++] = point[ coordIndex[i][1] ][1] / _DIV;
            /* Z */
            array[i][loc++] = point[ coordIndex[i][1] ][2] / _DIV;

            /* X */
            array[i][loc++] = point[ coordIndex[i][2] ][0] / _DIV;
            /* Y */
            array[i][loc++] = point[ coordIndex[i][2] ][1] / _DIV;
            /* Z */
            array[i][loc++] = point[ coordIndex[i][2] ][2] / _DIV;

            /* X */
            array[i][loc++] = point[ coordIndex[i][3] ][0] / _DIV;
            /* Y */
            array[i][loc++] = point[ coordIndex[i][3] ][1] / _DIV;
            /* Z */
            array[i][loc++] = point[ coordIndex[i][3] ][2] / _DIV;

            loc = 0;

          }
}


void ginit(short DBmode)
{               /* Q2SD */
  unsigned short hsw =  64;
  unsigned short xs  = 131;
  unsigned short xw  = 640;
  unsigned short hc  = 910;

  unsigned short vsw =   3;
  unsigned short ys  =  16;
```

```
unsigned short yw  = 240;
unsigned short vc  = 262;

        outport(_Q2SYSR, 0x4080);         /* Initilaize Draw/Display */
        outport(_Q2SRCR, 0xfe80);         /* Clear SR register      */

        /* Set InterFace Control Registers */
        outport(_Q2IER ,   0x0000);
        outport(_Q2MEMR,   0x0031);
        outport(_Q2DSMR,   0x0005);
        outport(_Q2DSMR2,  0x0000);
        outport(_Q2REMR,   0x0041);
        outport(_Q2IEMR,   0x0000);

        /* Set Memory Control Regisers    */
        outport(_Q2DSX,                 xw);/* Display size of x                */
        outport(_Q2DSY,                 yw);/* Display size of y                */
        outport(_Q2DSA0,   0x0000);  /* Frame buffer 0 area start address(H) */
        outport(_Q2DSA1,   0x0010);  /* Frame buffer 1 area start address(H) */
        outport(_Q2DLSAH,  0x0019);  /* Display list area start address(H)   */
        outport(_Q2DLSAL,  0x0000);  /* Display list area start address(L)   */
        outport(_Q2SSAR,   0x0008);  /* Color area sorce start address(H)    */
        outport(_Q2WSAR,   0x001F);  /* Work area start address(H)           */
        outport(_Q2DMASH,  0x0000);  /* DMA transfer start address(H)        */
        outport(_Q2DMASL,  0x0000);  /* DMA transfer start address(L)        */
        outport(_Q2DMAWL,  0x0000);  /* DAM transfer word                    */

        /* Display Contral Registers */
        outport(_Q2HDS,              hsw+xs-11    );
        outport(_Q2HDE,              hsw+xs-11+xw );
        outport(_Q2VDS,              ys-2         );
        outport(_Q2VDE,              ys-2+yw      );
        outport(_Q2HSW,              hsw-1        );
        outport(_Q2HC,               hc-1         );
        outport(_Q2VSP,              vc-vsw-1     );
        outport(_Q2VC,               vc-1         );
        outport(_Q2DOR,              0x0000);
        outport(_Q2DOG_DOB,          0x007C);
        outport(_Q2CDR,              0x00FC);
```

```
        outport(_Q2CDG_CDB,          0xFCFC);

        /* Input Data Control Registers */
        outport(_Q2ISAH,   0x0000);
        outport(_Q2ISAL,   0x0000);
        outport(_Q2IDSX,   0x0000);
        outport(_Q2IDSY,   0x0000);
        outport(_Q2IDE,    0x0000);

        switch(DBmode) {                                /* Enable Draw/Display */
                case 0:
                        outport(_Q2SYSR, 0x2000);       /* Idle,Auto Change mode,No DMA */
                        break;
                case 1:
                        outport(_Q2SYSR, 0x2040);       /* Idle,Auto Renderring mode,No DMA */
                        break;
                default:
                        outport(_Q2SYSR, 0x2080);       /* Idle,manual change moe,No DMA */
        }
}


long sin_t(short d)
{
        d%=360;
        if(d>=  0 && d< 90) return  sin_table[   d];
        if(d>= 90 && d<180) return  sin_table[180-d];
        if(d>=180 && d<270) return -sin_table[d-180];
                            return -sin_table[360-d];
}

long cos_t(short d)
{
        d%=360;
        if(d>=  0 && d< 90) return  sin_table[90 -d];
        if(d>= 90 && d<180) return -sin_table[d -90];
        if(d>=180 && d<270) return -sin_table[270-d];
                            return  sin_table[d-270];
}
```

```
void rotate_a_rectangle( short array[12], short angle, short center, short dx, short dy, short dz )
{
  short i;
  short tx, ty, tz;
          switch( center ){
            case Z:
                    for(i=0;i<12;i+=3) {
                            tx =(short)  (( (array[i]-dx) * cos_t(angle)-  (array[i+1]-dy) * sin_t(angle) )>>13);
                            ty =(short)  (( (array[i]-dx) * sin_t(angle)+  (array[i+1]-dy) * cos_t(angle) )>>13);
                            array[i  ] =  tx+dx;
                            array[i+1] =  ty+dy;
                    }
                    break;
            case Y:
                    for(i=0;i<12;i+=3) {
                            tz =(short)  (( (array[i+2]-dz) * cos_t(angle)-  (array[i]-dx) * sin_t(angle) )>>13);
                            tx =(short)  (( (array[i+2]-dz) * sin_t(angle)+  (array[i]-dx) * cos_t(angle) )>>13);
                            array[i+2] =  tz+dz;
                            array[i  ] =  tx+dx;
                    }
                    break;
            case X:
                    for(i=0;i<12;i+=3) {
                            ty =(short)  (( (array[i+1]-dy) * cos_t(angle)-  (array[i+2]-dz) *
sin_t(angle) )>>13);
                            tz =(short)  (( (array[i+1]-dy) * sin_t(angle)+  (array[i+2]-dz) *
cos_t(angle) )>>13);
                            array[i+1] =  ty+dy;
                            array[i+2] =  tz+dz;
                    }
                    break;
          }
}

void convert_d3_into_d2( short array[12], short dx, short dy, short dz, short _z_size )
{
  short x0, y0;
```

```
  short x1, y1;
  short x2, y2;
  short x3, y3;

          x0 = (short)(array[ 0] * ((long)_z_size - array[ 2] - dz ) / _z_size);
          y0 = (short)(array[ 1] * ((long)_z_size - array[ 2] - dz ) / _z_size);
          x1 = (short)(array[ 3] * ((long)_z_size - array[ 5] - dz ) / _z_size);
          y1 = (short)(array[ 4] * ((long)_z_size - array[ 5] - dz ) / _z_size);
          x2 = (short)(array[ 6] * ((long)_z_size - array[ 8] - dz ) / _z_size);
          y2 = (short)(array[ 7] * ((long)_z_size - array[ 8] - dz ) / _z_size);
          x3 = (short)(array[ 9] * ((long)_z_size - array[11] - dz ) / _z_size);
          y3 = (short)(array[10] * ((long)_z_size - array[11] - dz ) / _z_size);

          array[0] = x0+dx; array[1] = y0+dy;
          array[2] = x1+dx; array[3] = y1+dy;
          array[4] = x2+dx; array[5] = y2+dy;
          array[6] = x3+dx; array[7] = y3+dy;
}

void z_sort( short array[N][12], short st[N], short n )
{
  short i,j,k;
  short max;
  short z[N];

          for(i=0; i<n; i++){
                    z[i] = 0;
                    for (k=_Z1;k<=_Z4;k+=3){
                              z[i] += array[i][k];
                    }
                    z[i] /= 4;
          }

          st[0] = 0;

          for(i=1; i<n; i++){
                    for(j=0; j<i && z[st[j]]>z[i];j++);
                    for(k=i; k>j;              k--) {
                              st[k]=st[k-1];
```

```
                                                    }
                    st[j]=i;
          }
}

#include "MS7709.inc"
#include "Q2SDl.inc"
```

5.2.14 sample9.c のソースファイル

```
/*
                                        Q2 Display List / SHC sample program (9)

                          ( Move 'Solid' Polygon with pattern which is in work area )

                                        Copyright(c) Hitachi Ltd. 1999
*/
#include <machine.h>
#include <stdio.h>
#include <math.h>
#include "Q2SD_REG.h"
#include "Q2SD_mac.h"


unsigned short *DISPLIST_ptr;


#define         PI                      3.14159


#define X_WIDTH 1024


#define _Q2_REMR_REG_NO 0x006
#define _Q2_RSAR_REG_NO 0x04C


void  init_BSC(void);                                   /* Initialize SH7709 bus state controller */
void  init_start(void);                                 /* Initialize Display List Double Buffering (Only 1st) */
short draw_start(void);                                 /* Initialize Display List Address pointer */
short draw_end(short DBmode,char *first,char quick);  /* Display List Execution */
void  change_com_buffer(void);
/* ------------------------------------------------------ */


void  ginit(short DBmode);                              /* Graphics System Open  (Q2 Initialize) */


void main(void)
{
          long  count;
```

```
          short array[8];
          short DBmode;
/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
          char  first,quick;

          DBmode = 1;
          first  = ON;
          quick  = OFF;

          init_BSC();                              /* Initialize SH7709 bus state controller */

          ginit(DBmode);                           /* Initialize Q2 */


/* Clear work area */
{
  unsigned short x,y;
  unsigned short *wk  = VU_SHORT (0x1F0000L | UGMBASE);
          for(y=1;y<=240;y++){
                    for(x=1;x<=X_WIDTH/16;x++){
                            *wk = 0x0000;
                            ++wk;
                    }
          }
}


/* Transfer Font Data "漢" at work area ---------------------------------- */
{
  unsigned short work_area_width = X_WIDTH / 16;
  unsigned long  SrcAdr;                           /* Source Address */
  short i,k,m;
  unsigned short *address;
  unsigned short font_pattern[16]={
                    0x8112, 0x4ffe, 0x2110, 0x0000,
```

```
                0x87fc, 0x4444, 0x1444, 0x17fc,
                0x2040, 0x27fc, 0xc040, 0x4ffe,
                0x40a0, 0x4110, 0x4608, 0x5806
        };

    SrcAdr = inport(_Q2WSAR), SrcAdr <<= 16, SrcAdr += UGMBASE;

    SrcAdr += work_area_width * 90;
    address = (unsigned short *)SrcAdr;

    for(m=1;m<8;m++){
            for(i=0;i<16;i++){
                    for(k=0;k<work_area_width;k++){
                            *address++  = font_pattern[i];
                    }
            }
    }
}

    init_start();                           /* Initialize Transfer Procedure */
    draw_start();                           /* Start Transfer Display List to UGM */
    sclip(0x0000, 639,239);
    lcofs(0x0000,0,0);

    for( count = 0 ; count < 5 ; count++ ){

    /* MOVE */
      short kj = 50;
      double rad;
      short d,i;

            for(d=0;d<=360;d+=5) {

            /* Job No.1 : clear screen */
                    array[0] =   0;     array[1] =   0;
                    array[2] =  639;    array[3] =   0;
                    array[4] =  639;    array[5] =  239;
                    array[6] =   0;     array[7] =  239;
                    polygon4c(0x0000, array, 0x0606);
```

```
/* Set drawing start address in the RSAR */
        if ( (inport( _Q2SR ) & 0x0100) != 0 ) {
                wpr( 0x0000, _Q2_RSAR_REG_NO, inport( _Q2DSA0 ) );
        } else {
                wpr( 0x0000, _Q2_RSAR_REG_NO, inport( _Q2DSA1 ) );
        }

/* Enable the RSAR */
        wpr( 0x0000, _Q2_REMR_REG_NO, inport(_Q2REMR) | 0x8000 );


/* Job No.2 : draw two solid rectangles */
        array[0] =  -50;     array[1] =  -(60);
        array[2] =   50;     array[3] =  -(60);
        array[4] =   50;     array[5] =  -(-60);
        array[6] =  -50;     array[7] =  -(-60);

        rad = PI * d / 180;
        for(i=0;i<=6;i+=2) {
                short tx =(short)  (array[i] * cos(rad)-  array[i+1] * sin(rad));
                short ty =(short)  (array[i] * sin(rad)+  array[i+1] * cos(rad));
                array[i]  =  tx;
                array[i+1] =  ty;
        }

        lcofs(0x0000, kj,100);
        polygon4c(0x0001,array,0xF800);                /* POLYGON4C */

        lcofs(0x0000, 320-kj,100);
        polygon4c(0x0001,array,0xF800);                /* POLYGON4C */

        lcofs(0x0000, 0,0);


/* Disable the RSAR */
        wpr( 0x0000, _Q2_REMR_REG_NO, inport(_Q2REMR) & 0x7fff );


        kj+=5;
```

```
                    /* Drawing start */
                            draw_end(DBmode,&first,quick);
                            draw_start();

                    } /* for */

            } /* for */

            return;                                                    /* EXIT */
    }


    void ginit(short DBmode)
    {               /* Q2SD */
      unsigned short hsw =  64;
      unsigned short xs  = 131;
      unsigned short xw  = 640;
      unsigned short hc  = 910;

      unsigned short vsw =   3;
      unsigned short ys  =  16;
      unsigned short yw  = 240;
      unsigned short vc  = 262;

            outport(_Q2SYSR, 0x4080);                  /* Initilaize Draw/Display  */
            outport(_Q2SRCR, 0xfe80);                  /* Clear SR register        */

            /* Set InterFace Control Registers */
            outport(_Q2IER ,   0x0000);
            outport(_Q2MEMR,   0x0031);
            outport(_Q2DSMR,   0x0005);
            outport(_Q2DSMR2,  0x0000);
            outport(_Q2REMR,   0x0041);
            outport(_Q2IEMR,   0x0000);

            /* Set Memory Control Regisers    */
            outport(_Q2DSX,       xw);   /* Display size of x                */
            outport(_Q2DSY,       yw);   /* Display size of y                */
            outport(_Q2DSA0,   0x0000);  /* Frame buffer 0 area start address(H) */
```

```
outport(_Q2DSA1,   0x0010);  /* Frame buffer 1 area start address(H) */
outport(_Q2DLSAH,  0x0019);  /* Display list area start address(H)   */
outport(_Q2DLSAL,  0x0000);  /* Display list area start address(L)   */
outport(_Q2SSAR,   0x0008);  /* Color area sorce start address(H)    */
outport(_Q2WSAR,   0x001F);  /* Work area start address(H)           */
outport(_Q2DMASH,  0x0000);  /* DMA transfer start address(H)        */
outport(_Q2DMASL,  0x0000);  /* DMA transfer start address(L)        */
outport(_Q2DMAWL,  0x0000);  /* DAM transfer word                    */

/* Display Contral Registers */
outport(_Q2HDS,              hsw+xs-11    );
outport(_Q2HDE,              hsw+xs-11+xw );
outport(_Q2VDS,              ys-2         );
outport(_Q2VDE,              ys-2+yw      );
outport(_Q2HSW,              hsw-1        );
outport(_Q2HC,               hc-1         );
outport(_Q2VSP,              vc-vsw-1     );
outport(_Q2VC,               vc-1         );
outport(_Q2DOR,              0x0000);
outport(_Q2DOG_DOB,          0x007C);
outport(_Q2CDR,              0x00FC);
outport(_Q2CDG_CDB,          0xFCFC);

/* Input Data Control Registers */
outport(_Q2ISAH,   0x0000);
outport(_Q2ISAL,   0x0000);
outport(_Q2IDSX,   0x0000);
outport(_Q2IDSY,   0x0000);
outport(_Q2IDE,    0x0000);

switch(DBmode) {                              /* Enable Draw/Display */
          case 0:
                 outport(_Q2SYSR, 0x2000);      /* Idle,Auto Change mode,No DMA */
                 break;
          case 1:
                 outport(_Q2SYSR, 0x2040);      /* Idle,Auto Renderring mode,No DMA */
                 break;
          default:
                 outport(_Q2SYSR, 0x2080);      /* Idle,manual change moe,No DMA */
```

```
        }
}

#include "Q2SDl.inc"
#include "MS7709.inc"
```

5.2.15 tst_brk.c のソースファイル

```
/*
                    Q2 Display List / SHC sample program
                    (Draw hexagon with breaking)

                                Only HD64413
                                Copyright(c) Hitachi Ltd. 1999

                    The break drawing is based on 5 operations ( From (1) to (5) ).
                    Kind of double buffer control is manual display change.

*/
#include "Q2SD_REG.h"
#include "Q2SD_mac.h"

#define  MAX_WORD  512

#define  DISPLIST2 0x005000L /* Store dislpay list for Back drawing  */

#define _Q2_REMR_REG_NO 0x006
#define _Q2_RTNH_REG_NO 0x04a
#define _Q2_RTNL_REG_NO 0x04b
#define _Q2_RSAR_REG_NO 0x04c


unsigned short *DISPLIST_ptr;



void  init_BSC(void);                                /* Initialize SH7709 bus state controller */
void  init_start(void);                              /* Initialize Display List Double Buffering (Only 1st) */
short _draw_start(void);                             /* Initialize Display List Address pointer */
void  change_com_buffer(void);
void  ginit(short DBmode);                           /* Graphics System Open  (Q2 Initialize) */
short _draw_end(short DBmode,char *first,char quick); /* Display List Execution */



void main(void)
{
```

```
            long  count;
            short array[MAX_WORD], poly[MAX_WORD];
            short DBmode;
            short fill_color = 0x0404;
            unsigned long display_list_adr;
            short adr_h, adr_l;
            short color = 0;
            short draw_buffer_area, display_buffer_area;
            unsigned long displying_area_address;
            unsigned long drawing_area_address;
            short st;
            short temp;
            unsigned long  CMD_StartAddress;

            unsigned short move_x,  move_y;
            unsigned short lcofs_x, lcofs_y;
            unsigned short uclip_xmin, uclip_ymin, uclip_xmax, uclip_ymax;
            unsigned short sclip_xmin, sclip_ymin;
            unsigned short rtn_address_h, rtn_address_l;
            unsigned short rsae_bit;

/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
            char  first,quick;

            DBmode = 2;
            first  = ON;
            quick  = OFF;

            init_BSC();                                 /* Initialize SH7709 bus state controller */

            ginit(DBmode);                              /* Initialize Q2 */

   /* Deside displaying area and drawing area */
{
   if( (inport( _Q2SR ) & 0x0100) != 0 ) {
```

```
        displying_area_address = _Q2DSA1;
        drawing_area_address   = _Q2DSA0;
    } else {
        displying_area_address = _Q2DSA0;
        drawing_area_address   = _Q2DSA1;
    }

    draw_buffer_area    = inport( drawing_area_address    );
    display_buffer_area = inport( displying_area_address );
}


/* ===  Create display list for back drawing  ========================== */

          /* Set display list start address */
          DISPLIST_ptr = (volatile unsigned short * const)(DISPLIST2 | UGMBASE);

          vbkem(0x0000, 0,0);                                /* vbkem */
          sclip(0x0000, 639,239);                            /* sclip */
          lcofs(0x0000, 0,0);                                /* lcofs */

          /* Set drawing start address */
          wpr( 0x0000, _Q2_RSAR_REG_NO, display_buffer_area );

          /* Enable the RSAR */
          wpr( 0x0000, _Q2_REMR_REG_NO, inport(_Q2REMR) | 0x8000 );

          for (color=0; color<127; color++){

                    lcofs(0x0000, color,color);

                    fill_color = ((color & 0xf) << 8) | (color & 0xf);

                    poly[ 0] =  40;         poly[ 1] =  10;
                    poly[ 2] =  130;        poly[ 3] =  poly[1];
                    poly[ 4] =  poly[2];    poly[ 5] =  100;
                    poly[ 6] =  poly[0];    poly[ 7] =  poly[5];
                    polygon4c(0x0000, poly, fill_color);                  /* polygon4c */

                    vbkem(0x0000, 0,0);
```

```
        } /* color */

        /* Disable the RSAR */
        wpr( 0x0000, _Q2_REMR_REG_NO, inport(_Q2REMR) & 0x7fff );

        trap(0);                                /* End of display list for back drawing */

        /* Change DLSAR           */
        CMD_StartAddress = DISPLIST2;
        adr_h = (CMD_StartAddress >> 16L) & 0xffff;
        adr_l =  CMD_StartAddress       & 0xffff;
        outport(_Q2DLSAH,adr_h);
        outport(_Q2DLSAL,adr_l);

        /* Transfer command from internal command buffer to UGM */
        outport(_Q2SRCR,0x0400);                /* Clear TRA bit   */
        outport(_Q2SYSR,0x2180);                /* Start drawing   */

/* ===  End of create display list for back drawing  ================= */

/* ===  Start drawing with breaking  ================================== */
        init_start();                           /* Initialize Transfer Procedure */

while(1){

 /*_/_/_/_/_/  (1). The operation of waiting internal update  _/_/_/_/_/*/
        outport( _Q2SRCR, 0x4000 );                                     /* Clear FRM bit */
        while( st=inport(_Q2SR), (st&0x4000)==0);                       /* FRM = 1 ? */

 /*_/_/_/_/_/  (2). The operation of breaking back drawing  _/_/_/_/_/*/
        outport(_Q2SRCR,0x0080);                                        /* Clear BRK bit */
        outport(_Q2SYSR,0x2480);                                        /* Stop drawing  */

        if ( st=inport(_Q2SR), (st&0x0400) != 0 ){                      /* Check TRA bit */
              break;
        } else {
           while( st=inport(_Q2SR), (st&0x0080)==0);   /* BRK = 1 ? */
```

```
            /* Get current pointer position */
            move_x = inport(_Q2XC);
            move_y = inport(_Q2YC);

            /* Get local offset */
            lcofs_x = inport(_Q2XO);
            lcofs_y = inport(_Q2YO);

            /* Get user clipping renge */
            uclip_xmin = inport(_Q2UXMIN);
            uclip_ymin = inport(_Q2UYMIN);
            uclip_xmax = inport(_Q2UXMAX);
            uclip_ymax = inport(_Q2UYMAX);

            /* Get system clipping renge */
            sclip_xmin = inport(_Q2SXMAX);
            sclip_ymin = inport(_Q2SYMAX);

            /* Get return address */
            rtn_address_h = inport(_Q2RTNH);
            rtn_address_l = inport(_Q2RTNL);

            /* Get RSAE bit */
            rsae_bit = inport(_Q2REMR) & 0x8000;

            /* Get command status */
            display_list_adr = inport(_Q2CSTH);
            display_list_adr <<= 16;
            display_list_adr |= inport(_Q2CSTL);
            adr_h = (short)( (display_list_adr >> 13) & 0x03FFL);
            adr_l = (short)(  display_list_adr         & 0x1FFFL);
        } /* else */


/*_/_/_/_/_/  (3). The oparetion of something drawing _/_/_/_/_/*/
/*_/_/_/_/_/              while breaking back drawing    _/_/_/_/_/*/

        _draw_start();
```

```
        /* Set drawing start address */
        wpr( 0x0000, _Q2_RSAR_REG_NO, display_buffer_area );

        /* Enable the RSAR */
        wpr( 0x0000, _Q2_REMR_REG_NO, inport(_Q2REMR) | 0x8000 );


        lcofs(0x0000, 0,0);

        /* ploygon4c ( clear screen ) */
        array[0] =   0;     array[1] =   0;   /* Left  Up   */
        array[2] = 639;     array[3] =   0;   /* Right Up   */
        array[4] = 639;     array[5] = 239;   /* Left  Down */
        array[6] =   0;     array[7] = 239;   /* Right Down */
        polygon4c(0x0000, array, 0x0000);     /* polygon4c  */

        array[0] =   0;     array[1] =   0;   /* Left  Up   */
        array[2] = 159;     array[3] =   0;   /* Right Up   */
        array[4] = 159;     array[5] = 119;   /* Left  Down */
        array[6] =   0;     array[7] = 119;   /* Right Down */
        polygon4c(0x0000, array, 0xffff);     /* polygon4c  */

        array[0] = 160;     array[1] = 120;   /* Left  Up   */
        array[2] = 319;     array[3] = 120;   /* Right Up   */
        array[4] = 319;     array[5] = 239;   /* Left  Down */
        array[6] = 160;     array[7] = 239;   /* Right Down */
        polygon4c(0x0000, array, 0xffff);     /* polygon4c  */


        /* Drawing start with chacking TRA bit */
        _draw_end(DBmode,&first,OFF);


/*_/_/_/_/_/  (4). The operation of continuity back drawing  _/_/_/_/_/*/

        _draw_start();

        /* Store RSAE bit */
        wpr( 0x0000, _Q2_REMR_REG_NO, rsae_bit | (inport(_Q2REMR) & 0x7fff) );

        /* Store return address */
```

```
            wpr( 0x0000, _Q2_RTNH_REG_NO, rtn_address_h );
            wpr( 0x0000, _Q2_RTNL_REG_NO, rtn_address_l );

            /* Store user clipping renge */
            uclip( 0x0000, uclip_xmin,uclip_ymin, uclip_xmax,uclip_ymax );

            /* Store system clipping renge */
            sclip( 0x0000, sclip_xmin,sclip_ymin );

            /* Store local offset before move command */
            lcofs( 0x0000, lcofs_x,lcofs_y );

            /* Store current pointer position */
            move( 0x0000, move_x,move_y );

            /* Go to next display list */
            jump(0x0000, adr_h,adr_l);

            /* Drawing start without chacking TRA bit */
            _draw_end(DBmode,&first,ON);


  /*_/_/_/_/_/  (5). The operation of frame change  _/_/_/_/_/*/

            /* Set display start address */
            outport( displying_area_address, display_buffer_area );

            /* Change display buffer */
            temp              = draw_buffer_area;
            draw_buffer_area    = display_buffer_area;
            display_buffer_area = temp;

} /* while(1) */

/* ===  End of drawing with breaking  ==================================== */

            return;           /* EXIT */
}

void ginit(short DBmode)
{               /* Q2SD */
```

```
unsigned short hsw =  64;
unsigned short xs  = 131;
unsigned short xw  = 640;
unsigned short hc  = 910;

unsigned short vsw =  3;
unsigned short ys  =  16;
unsigned short yw  = 240;
unsigned short vc  = 262;

        outport(_Q2SYSR, 0x4080);                  /* Initilaize Draw/Display  */
        outport(_Q2SRCR, 0xfe80);                  /* Clear SR register        */

        /* Set InterFace Control Registers */
        outport(_Q2IER ,   0x0000);
        outport(_Q2MEMR,   0x0031);
        outport(_Q2DSMR,   0x0005);
        outport(_Q2DSMR2,  0x0000);
        outport(_Q2REMR,   0x0041);
        outport(_Q2IEMR,   0x0000);

        /* Set Memory Control Regisers    */
        outport(_Q2DSX,       xw);    /* Display size of x                    */
        outport(_Q2DSY,       yw);    /* Display size of y                    */
        outport(_Q2DSA0,   0x0000);   /* Frame buffer 0 area start address(H) */
        outport(_Q2DSA1,   0x0010);   /* Frame buffer 1 area start address(H) */
        outport(_Q2DLSAH,  0x0019);   /* Display list area start address(H)   */
        outport(_Q2DLSAL,  0x0000);   /* Display list area start address(L)   */
        outport(_Q2SSAR,   0x0008);   /* Color area sorce start address(H)    */
        outport(_Q2WSAR,   0x001F);   /* Work area start address(H)           */
        outport(_Q2DMASH,  0x0000);   /* DMA transfer start address(H)        */
        outport(_Q2DMASL,  0x0000);   /* DMA transfer start address(L)        */
        outport(_Q2DMAWL,  0x0000);   /* DAM transfer word                    */

        /* Display Contral Registers */
        outport(_Q2HDS,              hsw+xs-11   );
        outport(_Q2HDE,              hsw+xs-11+xw );
        outport(_Q2VDS,              ys-2        );
        outport(_Q2VDE,              ys-2+yw     );
```

```
        outport(_Q2HSW,              hsw-1       );
        outport(_Q2HC,               hc-1        );
        outport(_Q2VSP,              vc-vsw-1    );
        outport(_Q2VC,               vc-1        );
        outport(_Q2DOR,              0x0000);
        outport(_Q2DOG_DOB,          0x007C);
        outport(_Q2CDR,              0x00FC);
        outport(_Q2CDG_CDB,          0xFCFC);

        /* Input Data Control Registers */
        outport(_Q2ISAH,   0x0000);
        outport(_Q2ISAL,   0x0000);
        outport(_Q2IDSX,   0x0000);
        outport(_Q2IDSY,   0x0000);
        outport(_Q2IDE,    0x0000);

        switch(DBmode) {                            /* Enable Draw/Display */
                case 0:
                        outport(_Q2SYSR, 0x2000);   /* Idle,Auto Change mode,No DMA */
                        break;
                case 1:
                        outport(_Q2SYSR, 0x2040);   /* Idle,Auto Renderring mode,No DMA */
                        break;
                default:
                        outport(_Q2SYSR, 0x2080);   /* Idle,manual change moe,No DMA */
        }
}

#include "Q2SDl.inc"


short _draw_start(void)                             /* Initialize Display List Address Pointer */
{
  long  CMD_StartAddress;
        if(DrawBuffer == 0)                         /* Display List Start Address */
                CMD_StartAddress = (long)(DISPLIST1+UGMBASE);   /* Execute Buffer0 -> Transfer Display List
to Buffer1 */
        else
                CMD_StartAddress = (long)(DISPLIST0+UGMBASE);   /* Execute Buffer1 -> Transfer Display List
```

```
to Buffer0 */

          DISPLIST_ptr = (unsigned short *)CMD_StartAddress;

          return( 0 );
}


short _draw_end(short DBmode,char *first,char quick)                    /* Execute Display List */
{
  unsigned short st, ah, al;
  unsigned long CMD_StartAddress;

          trap(0);                                                      /* Add 'trap' command */

          outport(_Q2SRCR,0x0400);                                      /* Clear TRA bit    */

          if(DrawBuffer == 0) {
                    DrawBuffer = 1;
                    CMD_StartAddress = DISPLIST1;
          }
          else {
                    DrawBuffer = 0;
                    CMD_StartAddress = DISPLIST0;
          }
          ah = (CMD_StartAddress >> 16L) & 0xffff;
          al =  CMD_StartAddress & 0xffffL;
          outport(_Q2DLSAH,ah);                                         /* Change DLSAR */
          outport(_Q2DLSAL,al);

          /* Renddering start */
          outport(_Q2SYSR,0x2180);

          /* Check RS bit */
          while( st=inport(_Q2SYSR), (st&0x0100)!=0 );                  /* RS = 0 ? */

          /* Manual Change */
          if(quick==OFF)
                    while(1){
                    st=inport(_Q2SR);
```

```
                    if ((st & 0x0400)!=0) break;                        /* Check TRA bit */
                    if ((st & 0x0200)!=0) {
                            outport(_Q2SRCR,0x0200);                    /* Clear CSF bit   */
                            break; /* CHeck CSF bit */
                    }
            }

            return( 0 );
}

#include "MS7709.inc"
```

5.2.16 elps.c のソースファイル

```
/*
                        Q2 Display List / SHC sample program
                       ( Draw filled ellipse)

                                        Copyright(c) Hitachi Ltd. 1999
*/
#include <machine.h>
#include <stdio.h>
#include <math.h>
#include "Q2SD_REG.h"
#include "Q2SD_mac.h"


unsigned short *DISPLIST_ptr;

void  init_BSC(void);                                /* Initialize SH7709 bus state controller */
void  init_start(void);                              /* Initialize Display List Double Buffering (Only 1st) */
short draw_start(void);                              /* Initialize Display List Address pointer */
short draw_end(short DBmode,char *first,char quick); /* Display List Execution */
void  change_com_buffer(void);
/* ------------------------------------------------------- */

void  _fill_ellipse( short xc, short yc, short rx, short ry, short color );
void  clrscrn(void);
void  ginit(short DBmode);                           /* Graphics System Open  (Q2 Initialize) */



void main(void)
{
          long  count;
          short DBmode;
/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
          char  first,quick;
```

```
          DBmode = 2;
          first  = ON;
          quick  = OFF;

          init_BSC();                              /* Initialize SH7709 bus state controller */

          ginit(DBmode);                           /* Initialize Q2 */
          init_start();                            /* Initialize Transfer Procedure */
          draw_start();                            /* Start Transfer Display List to UGM */
          sclip(0x0000, 639,239);
          lcofs(0x0000,0,0);

for( count = 1 ; count < 10 ; count++ ){
{
  short lp;
  short rx, ry;
  short x_center, y_center;

/* Define ellipse center */
          x_center = 100, y_center = 100;

          for (lp=1; lp<=100; lp+=count){

                    clrscrn();

/* Draw ellipse  ---------------------------- */
                    rx = lp;
                    ry = lp/2;
                    _fill_ellipse( x_center,y_center, rx,ry, 0xFFFF );

                    draw_end(DBmode,&first,quick);
                    draw_start();
          }
}

{
  short lp;
  short rx, ry;
  short x_center, y_center;
```

```
/* Define ellipse center */
          x_center = 100, y_center = 100;

          for (lp=1; lp<=100; lp+=count){

                    clrscrn();

/* Draw ellipse  ---------------------------- */
                    rx = lp/2;
                    ry = lp;
                    _fill_ellipse( x_center,y_center, rx,ry, 0xFFFF );

                    draw_end(DBmode,&first,quick);
                    draw_start();
          }
}
} /* for */

          return;                                   /* EXIT */
}


void clrscrn(void)
{
          short array[8];

/* ploygon4c (clear) ------------------------------------------------------ */
          array[0] =   0;     array[1] =   0;
          array[2] =  639;     array[3] =   0;
          array[4] =  639;     array[5] =  239;
          array[6] =   0;     array[7] =  239;
          polygon4c(0x0000, array, 0x0000);
}


/* Creates display list to draw ellipse with filling */
void _fill_ellipse( short xc, short yc, short rx, short ry, short color )
{
          if( rx<=0 || ry<=0 ) return;
```

```
if( rx > ry ){
  short x = rx;
  short r = rx;
  short y = 0;
  short poly[4];  /* 1st point  = ( poly[0], poly[1] ) */
                  /* 2nd point  = ( poly[2], poly[3] ) */

          /* Draw ellipse in work area */
          while( x >= y ) {
            short x1=(short)( (long)x * ry / rx );
            short y1=(short)( (long)y * ry / rx );

                  poly[0] = xc+x;  poly[2] = xc-x;
                  poly[1] = poly[3] = yc+y1;
                  line( 0x0000, color, 2, poly );
                  poly[1] = poly[3] = yc-y1;
                  line( 0x0000, color, 2, poly );

                  poly[0] = xc+y;  poly[2] = xc-y;
                  poly[1] = poly[3] = yc+x1;;
                  line( 0x0000, color, 2, poly );
                  poly[1] = poly[3] = yc-x1;;
                  line( 0x0000, color, 2, poly );

                  if ( (r -= (y++ << 1)-1) < 0) r += (x-- - 1) << 1;
          } /* while */
} /* if */
          else {
            short x = ry;
            short r = ry;
            short y = 0;
            short poly[4]; /* 1st point  = ( poly[0], poly[1] ) */
                           /* 2nd point  = ( poly[2], poly[3] ) */

                  /* Draw ellipse in work area */
                  while( x >= y ) {
                    short x1=(short)( (long)x * rx / ry );
                    short y1=(short)( (long)y * rx / ry );
```

```
                                poly[0] = xc+x1; poly[2] = xc-x1;
                                poly[1] = poly[3] = yc+y;
                                line( 0x0000, color, 2, poly );
                                poly[1] = poly[3] = yc-y;
                                line( 0x0000, color, 2, poly );

                                poly[0] = xc+y1; poly[2] = xc-y1;
                                poly[1] = poly[3] = yc+x;
                                line( 0x0000, color, 2, poly );
                                poly[1] = poly[3] = yc-x;
                                line( 0x0000, color, 2, poly );

                                if ( (r -= (y++ << 1)-1) < 0) r += (x-- - 1) << 1;
                        } /* while */
                    } /* else */
} /* _fill_ellipse */


void ginit(short DBmode)
{               /* Q2SD */
  unsigned short hsw =  64;
  unsigned short xs  = 131;
  unsigned short xw  = 640;
  unsigned short hc  = 910;

  unsigned short vsw =  3;
  unsigned short ys  =  16;
  unsigned short yw  = 240;
  unsigned short vc  = 262;

          outport(_Q2SYSR, 0x4080);                  /* Initilaize Draw/Display */
          outport(_Q2SRCR, 0xfe80);                  /* Clear SR register      */

          /* Set InterFace Control Registers */
          outport(_Q2IER ,   0x0000);
          outport(_Q2MEMR,   0x0031);
          outport(_Q2DSMR,   0x0005);
          outport(_Q2DSMR2,  0x0000);
```

```
outport(_Q2REMR,   0x0041);
outport(_Q2IEMR,   0x0000);

/* Set Memory Control Regisers    */
outport(_Q2DSX,       xw);    /* Display size of x                */
outport(_Q2DSY,       yw);    /* Display size of y                */
outport(_Q2DSA0,   0x0000);   /* Frame buffer 0 area start address(H) */
outport(_Q2DSA1,   0x0010);   /* Frame buffer 1 area start address(H) */
outport(_Q2DLSAH,  0x0019);   /* Display list area start address(H)   */
outport(_Q2DLSAL,  0x0000);   /* Display list area start address(L)   */
outport(_Q2SSAR,   0x0008);   /* Color area sorce start address(H)    */
outport(_Q2WSAR,   0x001F);   /* Work area start address(H)           */
outport(_Q2DMASH,  0x0000);   /* DMA transfer start address(H)        */
outport(_Q2DMASL,  0x0000);   /* DMA transfer start address(L)        */
outport(_Q2DMAWL,  0x0000);   /* DAM transfer word                    */

/* Display Contral Registers */
outport(_Q2HDS,              hsw+xs-11    );
outport(_Q2HDE,              hsw+xs-11+xw );
outport(_Q2VDS,              ys-2         );
outport(_Q2VDE,              ys-2+yw      );
outport(_Q2HSW,              hsw-1        );
outport(_Q2HC,               hc-1         );
outport(_Q2VSP,              vc-vsw-1     );
outport(_Q2VC,               vc-1         );
outport(_Q2DOR,              0x0000);
outport(_Q2DOG_DOB,          0x007C);
outport(_Q2CDR,              0x00FC);
outport(_Q2CDG_CDB,          0xFCFC);

/* Input Data Control Registers */
outport(_Q2ISAH,   0x0000);
outport(_Q2ISAL,   0x0000);
outport(_Q2IDSX,   0x0000);
outport(_Q2IDSY,   0x0000);
outport(_Q2IDE,    0x0000);

switch(DBmode) {                              /* Enable Draw/Display */
          case 0:
```

```
                        outport(_Q2SYSR, 0x2000);       /* Idle,Auto Change mode,No DMA */
                        break;
                case 1:
                        outport(_Q2SYSR, 0x2040);       /* Idle,Auto Renderring mode,No DMA */
                        break;
                default:
                        outport(_Q2SYSR, 0x2080);       /* Idle,manual change moe,No DMA */
        }
}

#include "MS7709.inc"
#include "Q2SDl.inc"
```

5.2.17 v_wind.c のソースファイル

```
/*_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/

           Video window ( window size is 320*240 pixel )

               Copyright(c) Hitachi Ltd. 1999

  _/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/*/


#include "Q2SD_REG.h"
#include "Q2SD_mac.h"

unsigned short *DISPLIST_ptr;

void  init_BSC(void);                                /* Initialize SH7709 bus state controller */
void  init_start(void);                              /* Initialize Display List Double Buffering (Only 1st) */
short draw_start(void);                              /* Initialize Display List Address pointer */
short draw_end(short DBmode,char *first,char quick); /* Display List Execution */
void  change_com_buffer(void);
void  ginit(short DBmode);                           /* Graphics System Open  (Q2 Initialize) */

void set_up_video_window( void );
void Stop_the_video_function( void );


#define OFF                         0
#define ON                          1


static void clrscrn(void);


void main(void)
{
  short array[8];
  long  count_n;
  char  first = ON;
  char  quick = OFF;                        short  DBmode = 1;
/*
```

```
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/

        init_BSC();                             /* Initialize SH7709 bus state controller */

        ginit(DBmode);                          /* Initialize Q2 */

        set_up_video_window();                  /* Setup Video window */

        init_start();                           /* Initialize Transfer Procedure */
        draw_start();                           /* Start Transfer Display List to UGM */
        sclip(0x0000, 639,239);
        lcofs(0x0000,0,0);


        for( count_n = 0 ; count_n < 500 ; count_n++ ){

                /* ploygon4c (clear)  */
                array[0] =   0;     array[1] =   0;
                array[2] =  639;     array[3] =   0;
                array[4] =  639;     array[5] =  239;
                array[6] =   0;     array[7] =  239;
                polygon4c(0x0000, array, 0x0000);

                draw_end(1,&first,quick);
                draw_start();
        }

        Stop_the_video_function();

        return;                                 /* EXIT */
}


void ginit(short DBmode)
{               /* Q2SD */
  unsigned short hsw =  64;
  unsigned short xs  = 131;
```

```
unsigned short xw  = 640;
unsigned short hc  = 910;

unsigned short vsw =  3;
unsigned short ys  =  16;
unsigned short yw  = 240;
unsigned short vc  = 262;

        outport(_Q2SYSR, 0x4080);                  /* Initilaize Draw/Display  */
        outport(_Q2SRCR, 0xfe80);                  /* Clear SR register        */

        /* Set InterFace Control Registers */
        outport(_Q2IER ,   0x0000);
        outport(_Q2MEMR,   0x0031);
        outport(_Q2DSMR,   0x0005);
        outport(_Q2DSMR2,  0x0000);
        outport(_Q2REMR,   0x0041);
        outport(_Q2IEMR,   0x0000);

        /* Set Memory Control Regisers    */
        outport(_Q2DSX,       xw);    /* Display size of x                */
        outport(_Q2DSY,       yw);    /* Display size of y                */
        outport(_Q2DSA0,   0x0000);   /* Frame buffer 0 area start address(H)  */
        outport(_Q2DSA1,   0x0010);   /* Frame buffer 1 area start address(H)  */
        outport(_Q2DLSAH,  0x0019);   /* Display list area start address(H)    */
        outport(_Q2DLSAL,  0x0000);   /* Display list area start address(L)    */
        outport(_Q2SSAR,   0x0008);   /* Color area sorce start address(H)     */
        outport(_Q2WSAR,   0x001F);   /* Work area start address(H)            */
        outport(_Q2DMASH,  0x0000);   /* DMA transfer start address(H)         */
        outport(_Q2DMASL,  0x0000);   /* DMA transfer start address(L)         */
        outport(_Q2DMAWL,  0x0000);   /* DAM transfer word                     */

        /* Display Contral Registers */
        outport(_Q2HDS,              hsw+xs-11    );
        outport(_Q2HDE,              hsw+xs-11+xw );
        outport(_Q2VDS,              ys-2         );
        outport(_Q2VDE,              ys-2+yw      );
        outport(_Q2HSW,              hsw-1        );
        outport(_Q2HC,               hc-1         );
```

```
        outport(_Q2VSP,              vc-vsw-1     );
        outport(_Q2VC,               vc-1         );
        outport(_Q2DOR,              0x0000);
        outport(_Q2DOG_DOB,          0x007C);
        outport(_Q2CDR,              0x00FC);
        outport(_Q2CDG_CDB,          0xFCFC);

        /* Input Data Control Registers */
        outport(_Q2ISAH,   0x0000);
        outport(_Q2ISAL,   0x0000);
        outport(_Q2IDSX,   0x0000);
        outport(_Q2IDSY,   0x0000);
        outport(_Q2IDE,    0x0000);

        switch(DBmode) {                        /* Enable Draw/Display */
                case 0:
                        outport(_Q2SYSR, 0x2000); /* Idle,Auto Change mode,No DMA */
                        break;
                case 1:
                        outport(_Q2SYSR, 0x2040); /* Idle,Auto Renderring mode,No DMA */
                        break;
                default:
                        outport(_Q2SYSR, 0x2080); /* Idle,manual change moe,No DMA */
        }
}


void set_up_video_window( void )
{
  unsigned long Video0_area_start_address = 0x07D000L;
  unsigned long Video1_area_start_address = 0x105000L;
  unsigned long Video2_area_start_address = 0x17D000L;


/* Initialize register for voide function */
{
  unsigned short temp;

        /* Hide the video window */
```

```
temp = inport(_Q2DSMR2);
temp &= 0xFFFE;
outport(_Q2DSMR2, temp);  /* VWE=0 */

/* Stop the capture for the video window */
temp = inport(_Q2VIMR);
temp &= 0xFFFE;
outport(_Q2VIMR, temp);                                          /* VIE=0 */


{
  short lp;
  for ( lp=0; lp<2; lp++ ) {
          outport( _Q2SRCR, 0x4000 );                            /* Clear FRM flag */
          while( (inport(_Q2SR) & 0x4000) == 0 );                /* Wait FRM flag */
  }
}


outport(_Q2HVP, 320);             /* Video window horizpntal display position (dot) */
outport(_Q2VVP, 0);               /* Video window vertical display position  (dot) */

/* Video0 area start address(From A22 to A10 ) */
outport(_Q2VSAH0, Video0_area_start_address >> 16   );
outport(_Q2VSAL0, Video0_area_start_address & 0xFC00);

/* Video1 area start address(From A22 to A10 ) */
outport(_Q2VSAH1, Video1_area_start_address >> 16   );
outport(_Q2VSAL1, Video1_area_start_address & 0xFC00);

/* Video2 area start address(From A22 to A10 ) */
outport(_Q2VSAH2, Video2_area_start_address >> 16   );
outport(_Q2VSAL2, Video2_area_start_address & 0xFC00);

/* Set 'video windows size' and 'voide area size' */
outport(_Q2VSIZEX, 320);
outport(_Q2VSIZEY, 240);

outport(_Q2VIMR, 0x0022); /* (VIZ4,3,2,1,0)=00001,  H = HACTIVE * 1/2, V = VACTIVE * 1    */
                          /* ( Caution )
```

```
                                        HACTIVE(dot/HSYNC) and VACTIVE(line/VSYNC) are defined
                                                                        by external caputure LSI.
                                        In this sample program requires HACTIVE=640 and VACTIVE=240 */
                                    /* VINM=0                                          */
                                    /* (ODEN1,ODEN0) =(0,0), 1/60 caputuring           */
                                    /* RGB=1, If RGB=1, have to set VWRY to 0.         */
                                    /* VIE=0                                           */

          /* Set RGB bit */
                    temp = inport(_Q2VIMR);
                    outport(_Q2VIMR, temp | 0x0002);       /* RGB =1 */

          /* Start the capture for the video stream */
                    temp = inport(_Q2VIMR);
                    outport(_Q2VIMR, temp | 0x0001);       /* VIE=1 */

          /*  Set VWRY and Dispaly the video window */
                    temp = inport(_Q2DSMR2);
                    outport(_Q2DSMR2, temp | 0x0001);      /* VWRY=0, VWE=1 */
 }
}


void Stop_the_video_function( void )
{
  /* Stop the voide function */
  unsigned short temp;

          /* Hide the video window */
          temp = inport(_Q2DSMR2);
          temp &= 0xFFFE;
          outport(_Q2DSMR2, temp);                         /* VWE=0 */

          /* Stop the capture for the video window */
          temp = inport(_Q2VIMR);
          temp &= 0xFFFE;
          outport(_Q2VIMR, temp);                          /* VIE=0 */
}
```

```
#include "MS7709.inc"

#include "Q2SDl.inc"
```

5.2.18 init_bt.c のソースファイル

```
/*_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/

            Copyright(c) Hitachi Ltd. 1999

              Initialize for Bt829.

  _/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/*/


#include "Q2SD_REG.h"
#include "Q2SD_mac.h"

void init_BSC(void);                          /* Initialize SH7709 bus state controller */
void init_Bt829B(void);                       /* Initialize VideoDecoder Bt829B */


/* -------------- Bt829B register address ----------------- */
#if 0
/* Area2 */
#define II2_Data                    0xA8900000L
#define II2_Control                 0xA8900002L
#else
/* Area5 */
#define II2_Data                    0xB4900000L
#define II2_Control                 0xB4900002L
#endif


/* BT829B Register Data ( Video In Size = 640 × 480 ) */
short BTdata[] = {
         0x0a, /* IFORM       (0x01) MUX3 -> MUXOUT, Auto XT, NTSC */
         0x00, /* TDEC        (0x02) */
         0x12, /* CROP        (0x03) */
         0x16, /* VDEALY_LO   (0x04) */
         0xe0, /* VACTIVE_LO  (0x05) */
         0x78, /* HDEALY_LO   (0x06) */
         0x80, /* HACVIVE_LO  (0x07) */
         0x02, /* HSCALE_HI   (0x08) */
         0xAC, /* HSCALE_LO   (0x09) */
```

```
        0xf0, /* BRIGHT      (0x0A) */
        0x20, /* CONTROL     (0x0B) */
        0xd8, /* CONTRAST_LO (0x0C) */
        0xfe, /* SAT_U_LO    (0x0D) */
        0xb4, /* SAT_V_LO    (0x0E) */
        0x10, /* HUE         (0x0F) */

        0x02, /* OFORM       (0x12) 8bit stream */
        0x60, /* VSCALE_HI   (0x13) */
        0x00, /* VSCALE_LO   (0x14) */
        0x01, /* TEST        (0x15) */
        0x00, /* VPOLE       (0x16) */

        0x68, /* ADEALY      (0x18) */
        0x5d, /* BDEALY      (0x19) */
        0x82  /* ADC         (0x1A) */
        };


void main(void)
{
        init_BSC();                             /* Initialize SH7709 bus state controller */

        init_Bt829B();                          /* Initialize VideoDecoder */
}


void init_Bt829B(void)                          /* Initiallize VideoDecoder Bt829A */
{
  short  lp;                                    /* loop counter*/
  short  regno;                                 /* regno counter */

        /* Initiallize IIC-bus controller */

        do{
                outport(II2_Control, 0x80 << 8);

                /* Loads into S0 */
                outport(II2_Data,   0x55 << 8);
```

```
            /* Loads into S1 */
            outport(II2_Control, 0xA0 << 8);

            /* Loads into S2 CLK=8MHz */
            outport(II2_Data,   0x18 << 8);

            outport(II2_Control, 0xC1 << 8);

}while( ( inport(II2_Control) & 0x8100 ) != 0x8100 );


/* Set up BT829B Register */
regno = 0x01;

      for(lp=0; lp<23; regno++, lp++){

            /* Check BB */
            while( ( inport(II2_Control) & 0x0100 ) == 0 );

            /* Chip address( write mode ) */
            outport(II2_Data,  0x88 << 8);

            /* START II2 */
            outport(II2_Control, 0xc5 << 8);

            /* Check PIN */
            while( ( inport(II2_Control) & 0x8000 ) != 0 );
            if( ( inport(II2_Control) & 0x0800 ) != 0 ) {

                  /* Stop */
                  outport(II2_Control, 0xc3 << 8);
                  return;
            }

            /* SUB ADDR */
            if( regno == 0x10 ){
                  regno = 0x12;
            }
            if( regno == 0x17 ){
                  regno = 0x18;
```

```
                }
                outport(II2_Data, (regno << 8) );

                /* Check PIN */
                while( ( inport(II2_Control) & 0x8000 ) != 0 );
                if( ( inport(II2_Control) & 0x0800 ) != 0 ) {

                        /* Stop */
                        outport(II2_Control, 0xc3 << 8);
                        return;
                }

                /* Data */
                outport(II2_Data, BTdata[ lp ] << 8);

                /* Check PIN */
                while( ( inport(II2_Control) & 0x8000 ) != 0 );
                if( ( inport(II2_Control) & 0x0800 ) != 0 ) {

                        /* Stop */
                        outport(II2_Control, 0xc3 << 8);
                        return;
                }

                                /* Stop */
                                outport(II2_Control, 0xc3 << 8);

        }
}

#include "MS7709.inc"
```

6. 描画性能

図 6.1～**図** 6.3 に HD64413A の描画性能をグラフで示します。グラフは、320（H）×240（V）の範囲を描画するのに要する時間を示しています。

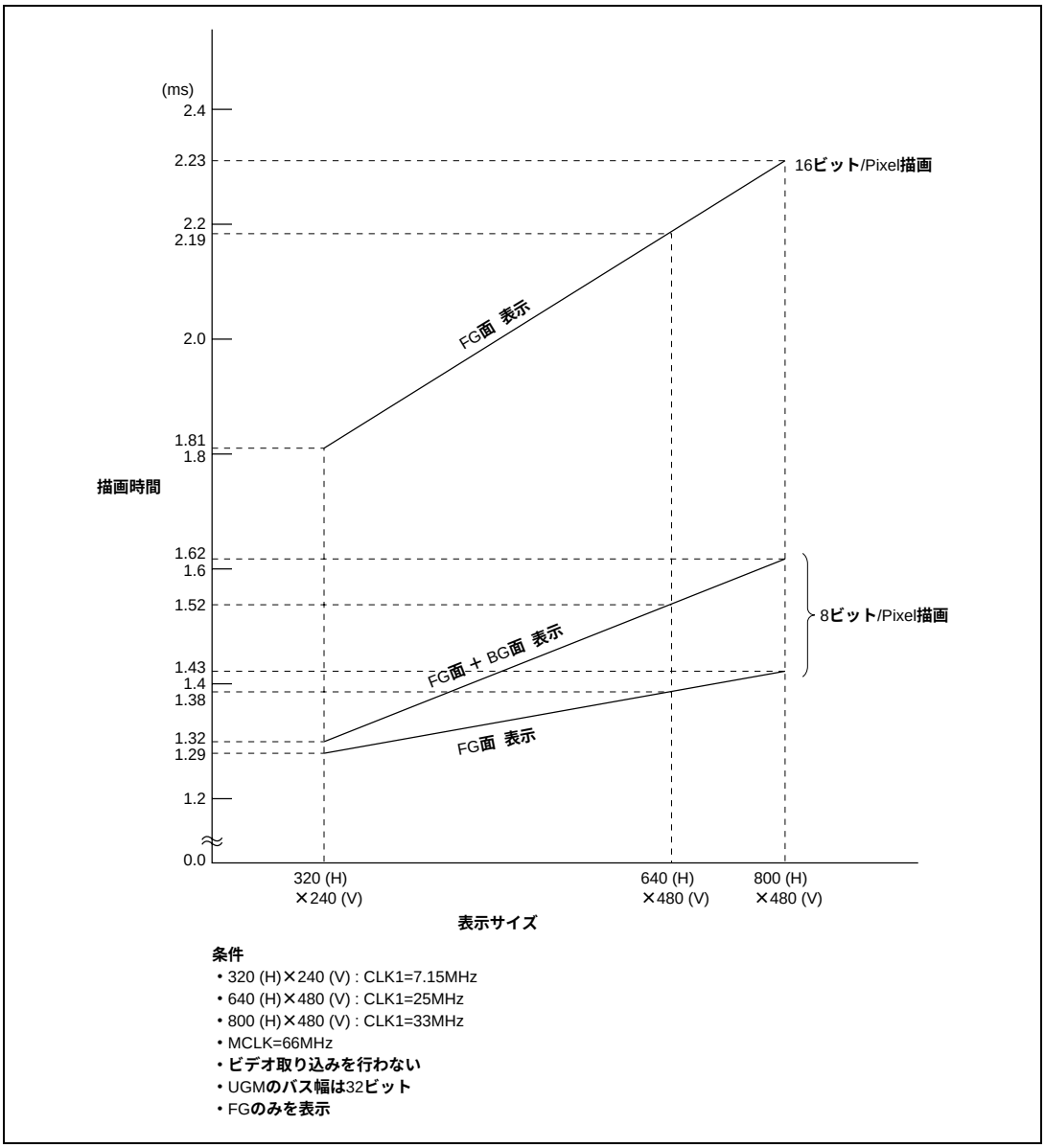


図 6.1　FST＝0 **の時の** POYLGON4C **の描画性能（描画範囲：**320（H）×240（V））

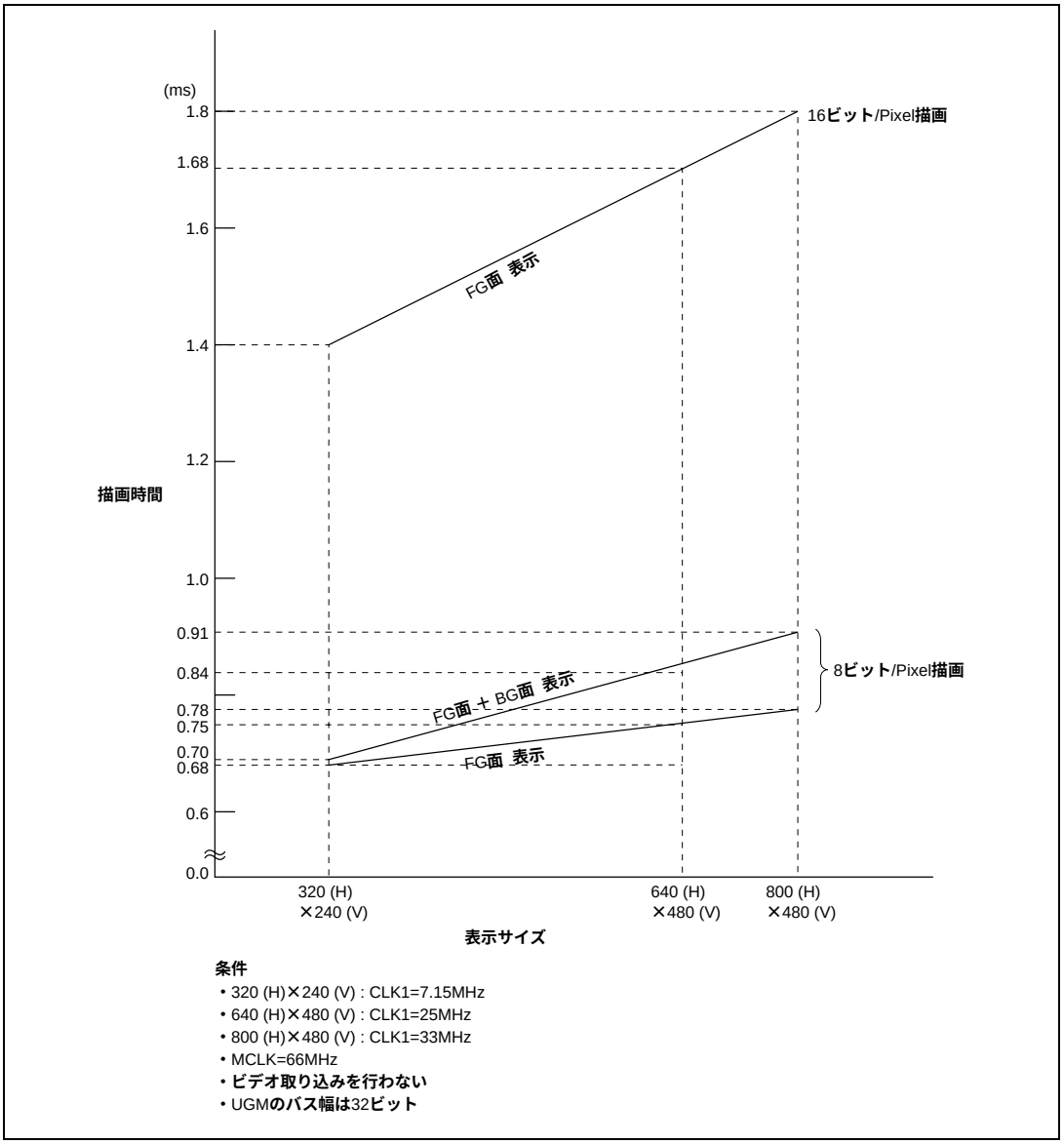


図 6.2　FST＝1 **の時の** POYLGON4C **の描画性能（描画範囲：**320（H）×240（V））

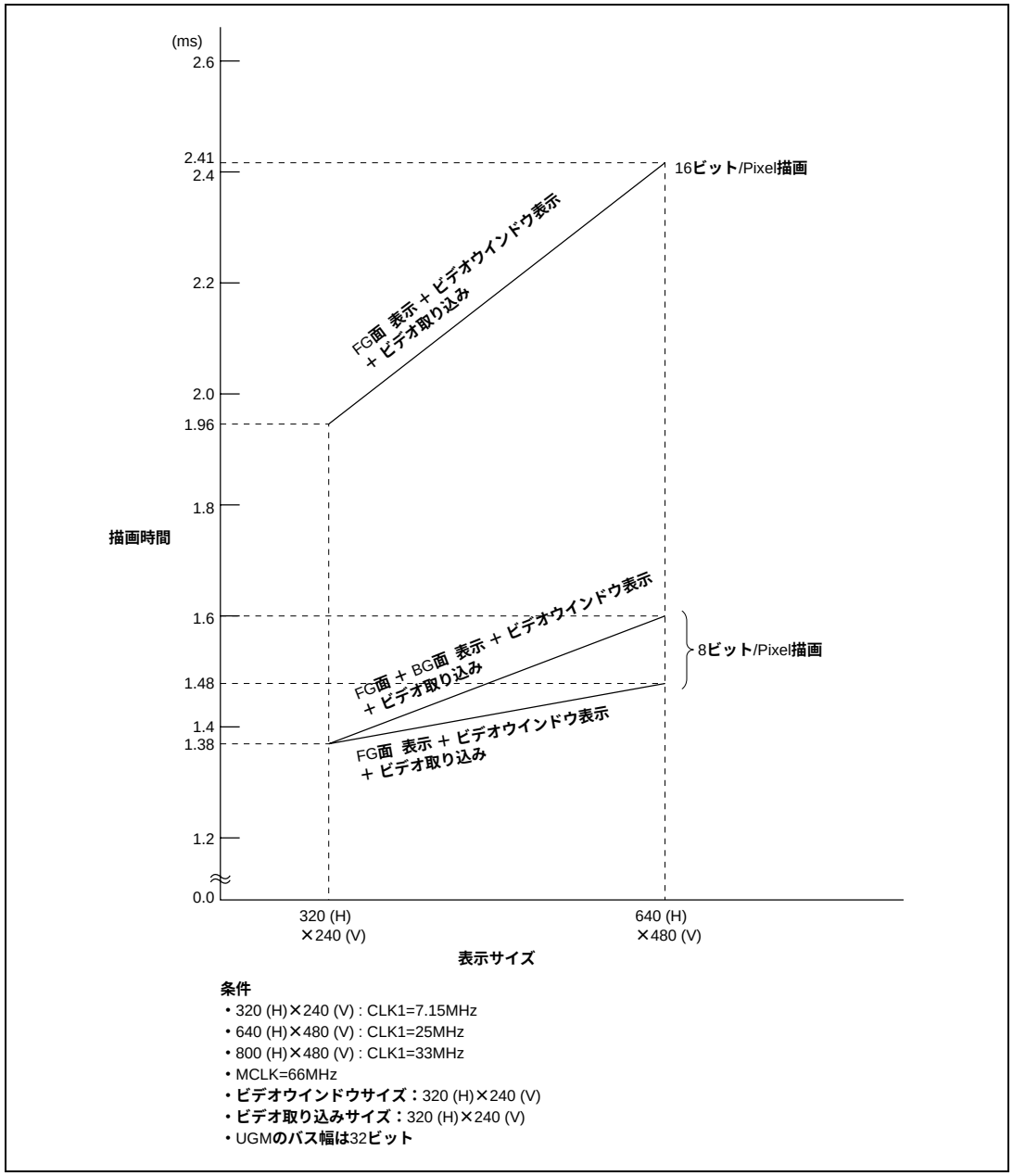


図 6.3　FST＝0 **の時の** POYLGON4C **の描画性能（描画範囲：**320（H）×240（V））

付録

A. MS4413DB01 の仕様

表 A.1 に MS4413DB01 のシステム仕様を示します。本アプリケーションではエリア 5 を使用します。

表 A.1　MS4413DB01 仕様

No.	機能	仕様
1	使用アドレスエリア	MS7709RP01：エリア 2 またはエリア 5（SW 切替） UGM エリア：H'A8000000～H'A87FFFFF（エリア 2 使用） H'B4000000～H'B47FFFFF（エリア 5 使用） Q2I レジスタエリア：H'A8800000～A88002FF（エリア 2 使用） H'B4800000～B48002FF（エリア 5 使用） I2C バスコントローラ（PCF8584）：H'A9000000～H'A9000002（エリア 2 使用） H'B5000000～H'B5000002（エリア 5 使用）
2	CLK0（描画）入力	OSC3：66MHz（max）、OSC1：14.31818MHz、MS7709RP01：CKIO から選択可
3	内部 CLK 逓倍機能	CLK0：Q2SD 設定により逓倍 OFF、ON（1・2・4 逓倍）可（最大 66MHz）
4	CLK1（表示）入力	OSC2：14.31818MHz
5	UGM メモリ	64Mb（×32）SDRAM：μPD4564323G5（NEC）×1
6	UGM 容量	8MB
7	表示	Q2SD：アナログ RGB 出力→NTSC－Video 生成（OSC・ジャンパー設定変更で PAL 対応）
	NTSC エンコーダ	CXA2075M（SONY）使用
8	表示色	26 万色中同時 256 色（カラーパレット内蔵）
9	対象ディスプレイ	NTSC ビデオ出力 TV／アナログ RGB 出力 TV
	解像度	標準 480×240（最大 640×480）
	インターフェース	アナログ RGB（セパレート SYNC）／NTSC（コンポジット・ビデオ）
10	VIDEO 入力部	VIDEO デコーダ：　Bt829B（ROCKWELL） コントロール部 I/F 方式：I2C バス I2C バスコントローラ：PCF8584T（PHILIPS） 外部入力 CLK：8MHz I2C－bus CLK90KHz/max
11	CPU I/F	3.3V レベル
12	割込み出力	SolutionEngine の IRQ1～8 を選択可
13	DMA 転送	SolutionEngine の DMAC の内、Ch0、Ch1 を選択可
14	WAIT 出力	SolutionEngine の WAIT0～3 を選択可
15	電源	＋3.3V・＋5V・A＋5V、拡張スロットより供給
16	ボードサイズ	128×182mm

図 A.1 に MS4413DB01 のブロック図を示します。

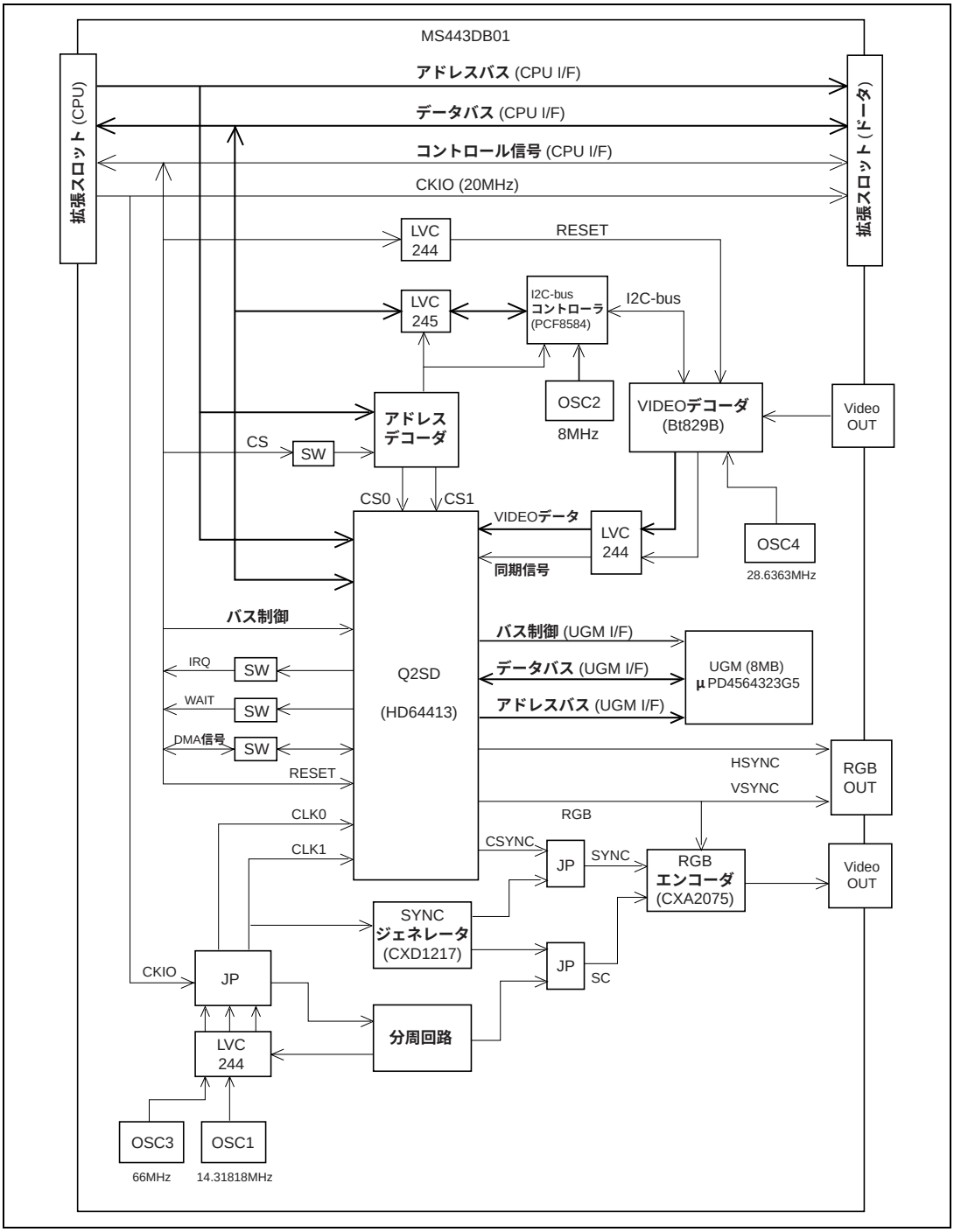


図 A.1　MS4413DB01 **ブロック図**

図 A.2～**図** A.7 に MS4413DB01 の回路図を示します。

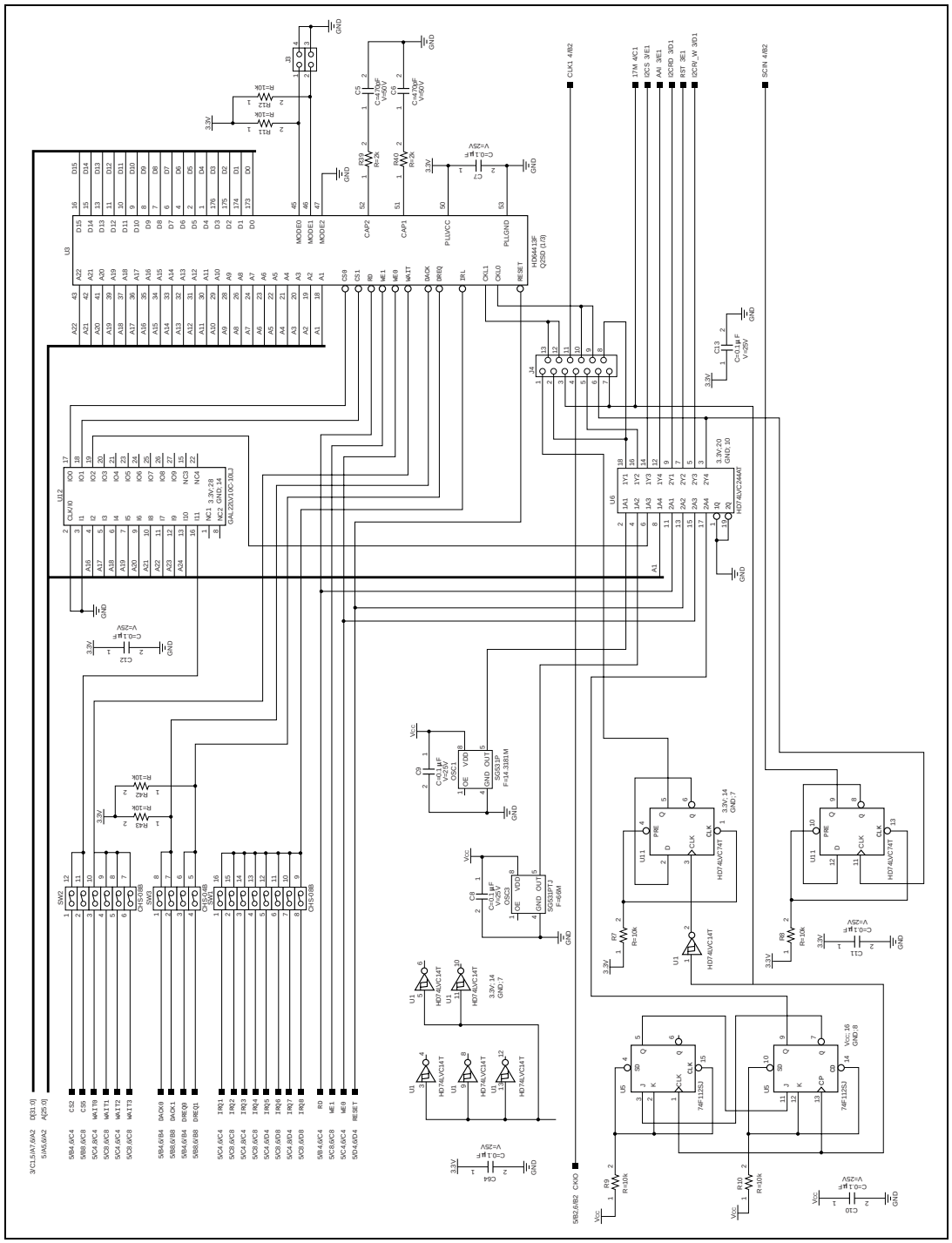


図 A.2　MS4413DB01 **回路図（1）**

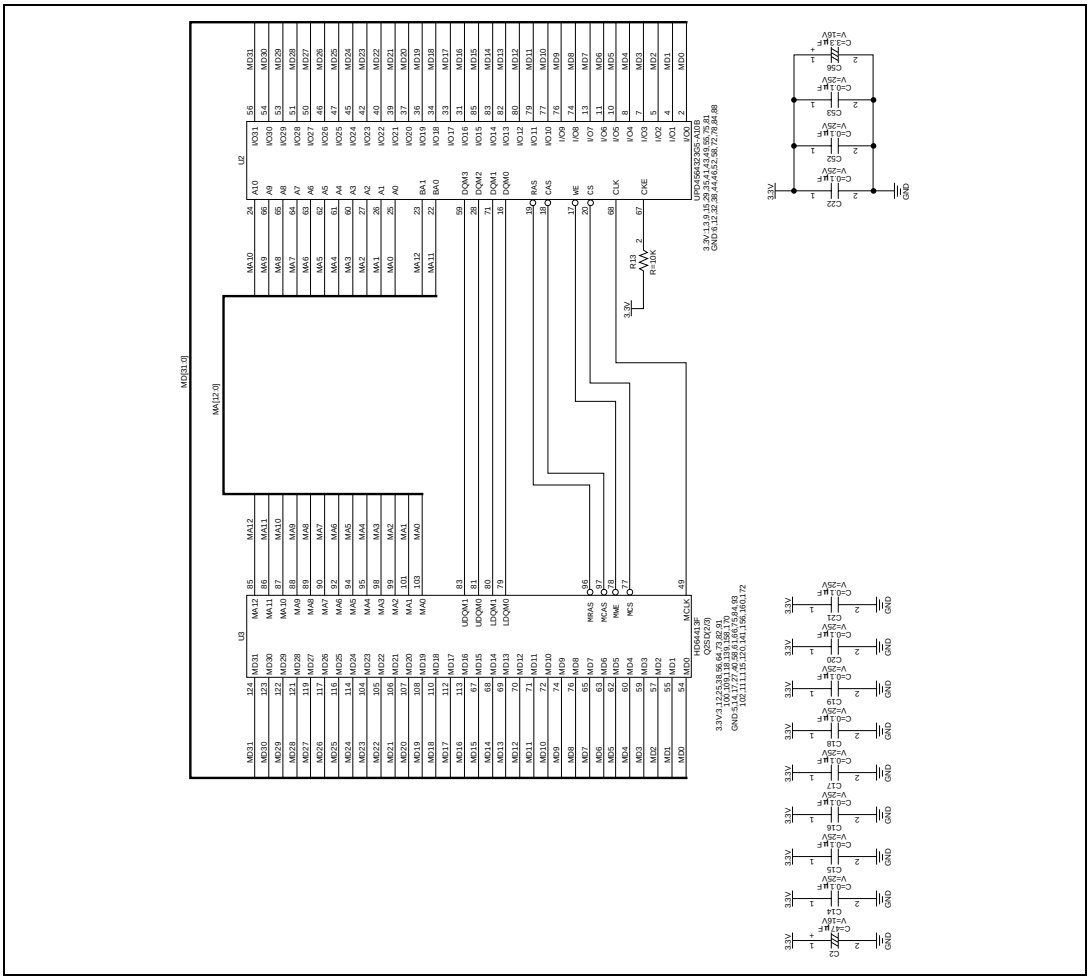


図 A.3　MS4413DB01 **回路図（2）**

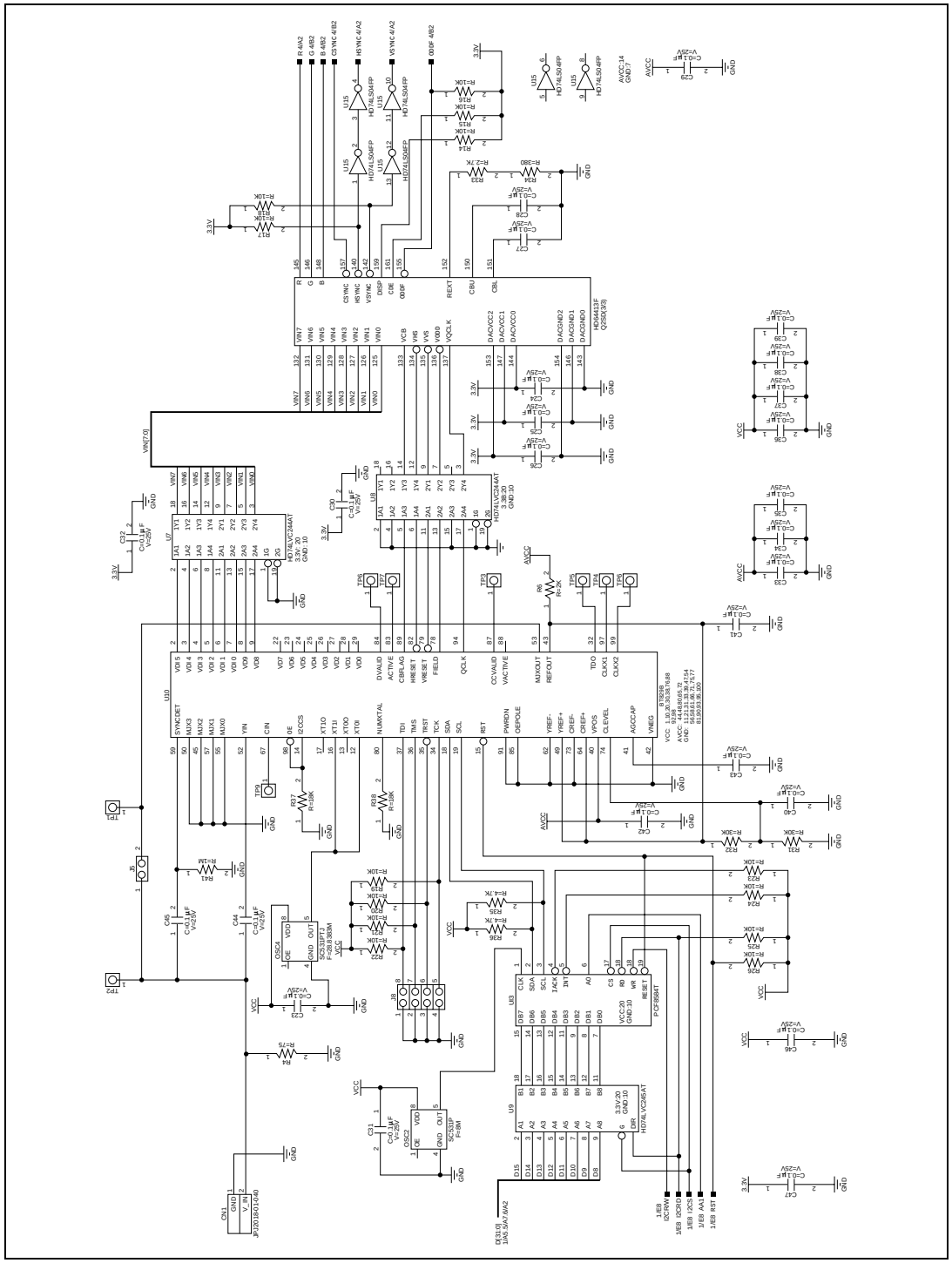


図 A.4　MS4413DB01 **回路図（3）**

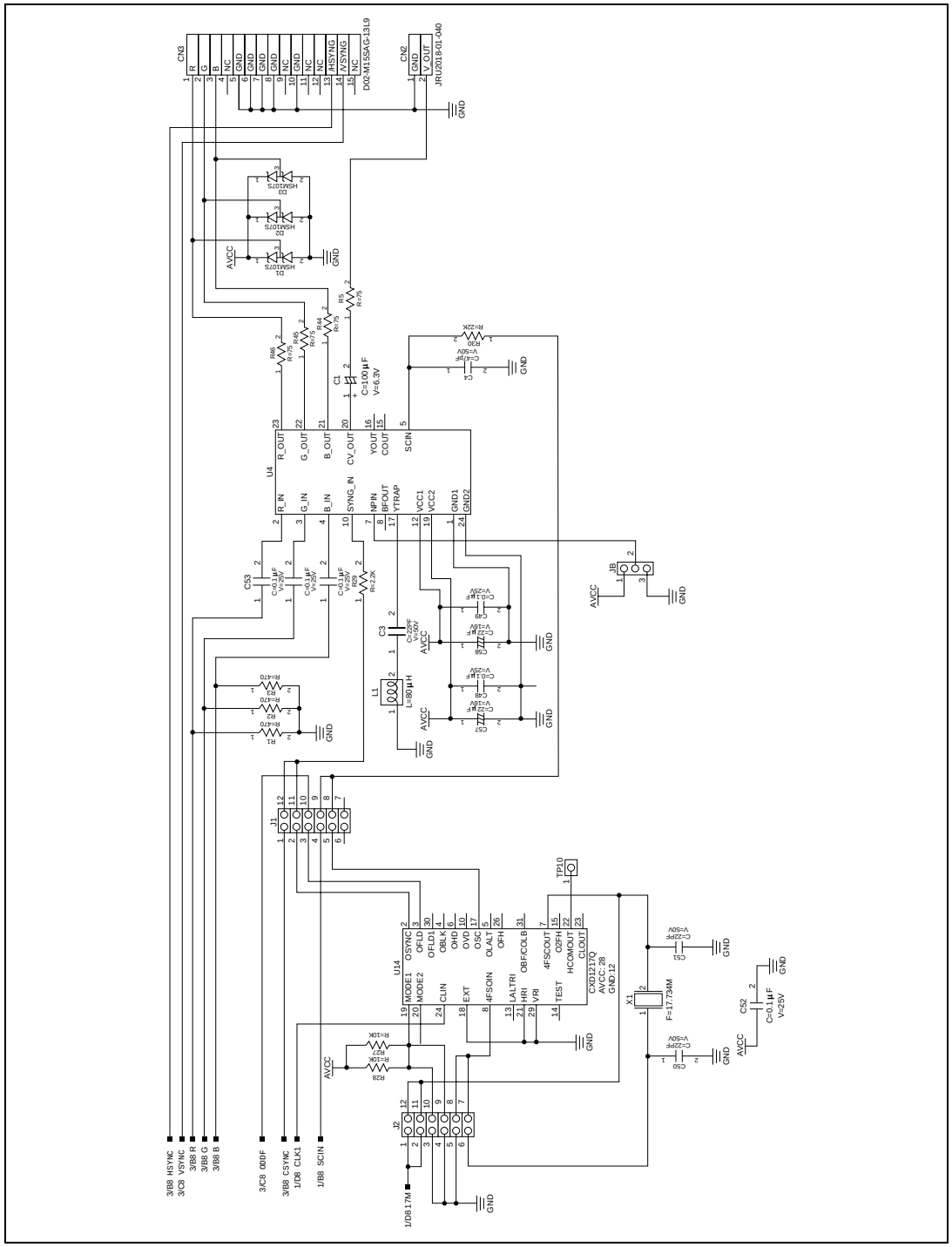


図 A.5　MS4413DB01 **回路図（4）**

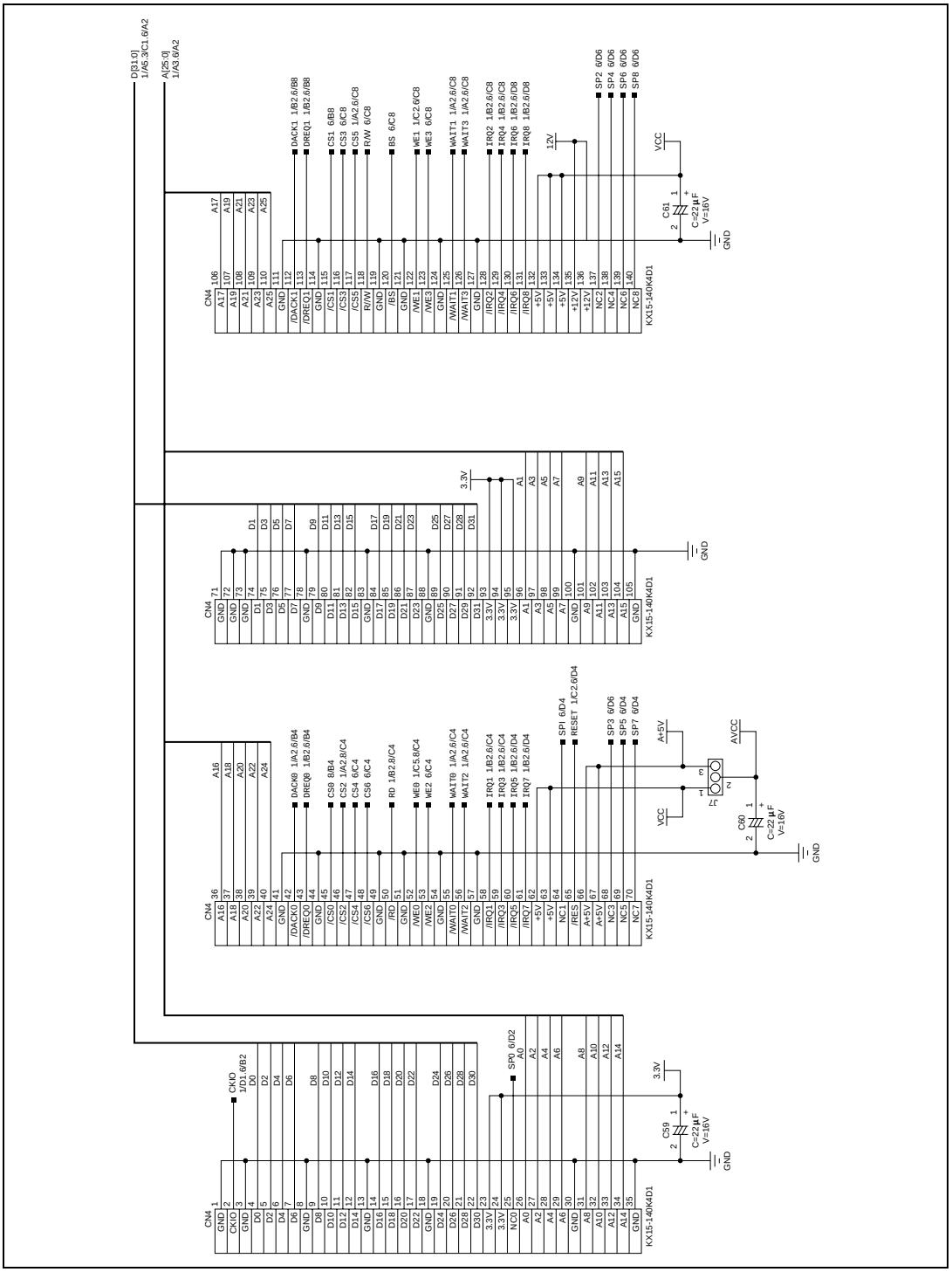


図 A.6　MS4413DB01 **回路図**（5）

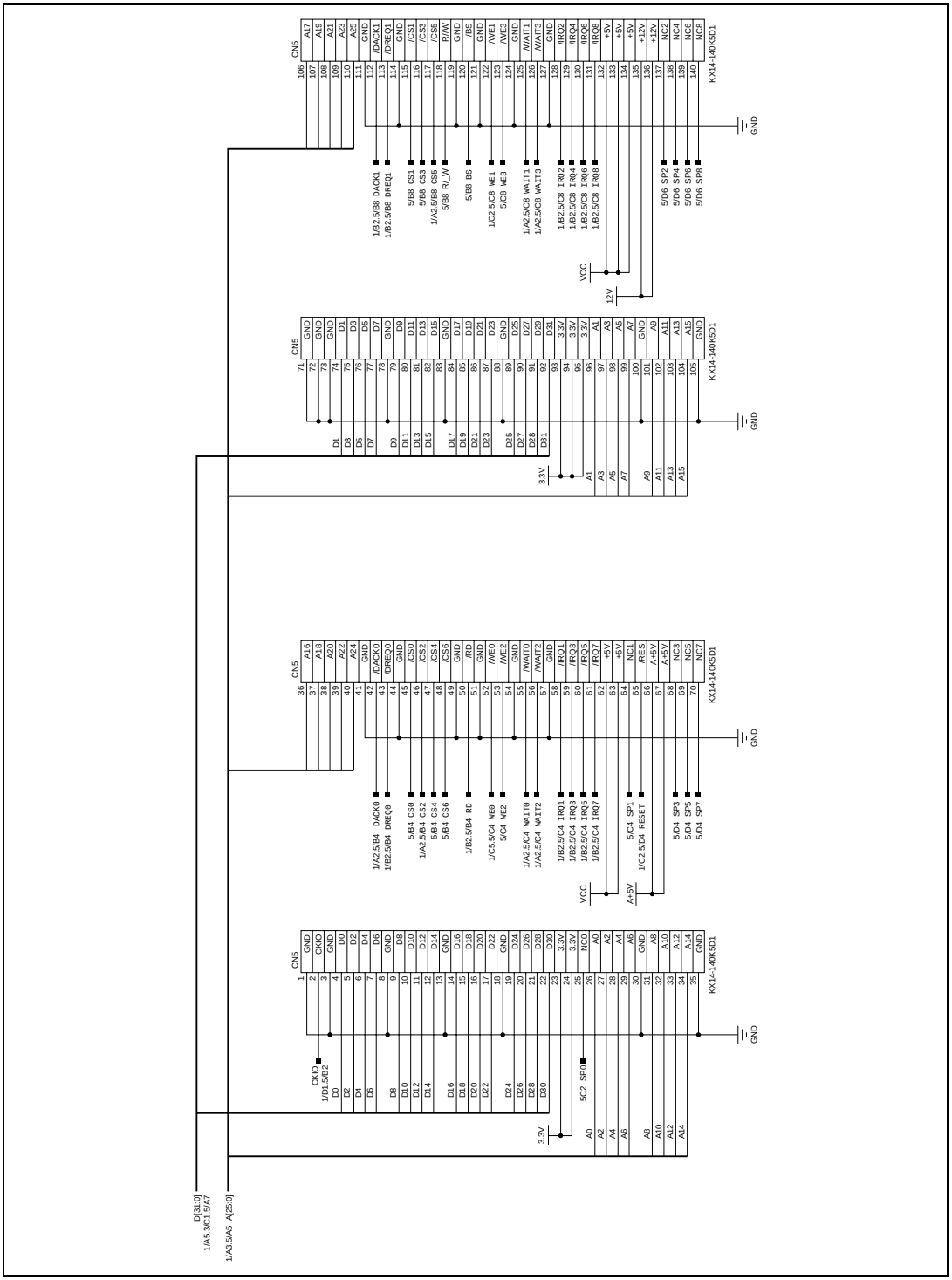


図 A.7　MS4413DB01 **回路図**（6）

HD64413A Q2SD アプリケーションノート

発行年月　平成11年9月　第１版
発　　行　株式会社　日立製作所
　　半導体グループ電子統括営業本部
編　　集　株式会社　超Lメディア
　　技術ドキュメントグループ

HD64413A Q2SD
アプリケーションノート

ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668

ADJ-502-079