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。

お客様各位

資料中の「日立製作所」、「日立XX」等名称の株式会社ルネサス テクノロジへの変更について

2003年4月1日を以って三菱電機株式会社及び株式会社日立製作所のマイコン、ロジック、アナログ、ディスクリート半導体、及びDRAMを除くメモリ(フラッシュメモリ・SRAM等)を含む半導体事業は株式会社ルネサス テクノロジに承継されました。従いまして、本資料中には「日立製作所」、「株式会社日立製作所」、「日立半導体」、「日立XX」といった表記が残っておりますが、これらの表記は全て「株式会社ルネサス テクノロジ」に変更されておりますのでご理解の程お願い致します。尚、会社商標・ロゴ・コーポレートステートメント以外の内容については一切変更しておりませんので資料としての内容更新ではありません。

ルネサステクノロジ ホームページ (http://www.renesas.com)

2003年4月1日
株式会社ルネサス テクノロジ
カスタマサポート部

RENESAS

ご注意

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ (http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。

Application Note

HD64411　Q2
Quick 2D Graphics Renderer

Q2 アプリケーションノート

SuperH RISC engine 周辺LSI

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　1997.12

はじめに

弊社では、多方面的に描画システムを構築できるよう、弊社の 32 ビット RISC マイコンである SuperH を中核としたソフトウェアによるグラフィック描画システムの提案、および、SuperH のシリーズ化を行ってまいりました。そのような中で、新たなシステム提案として、グラフィック描画にて、リアルタイムな動作を表現することで、今までに無かった新たなグラフィック描画の世界を表現し、ユーザが要求する多種多様な表現を行えるようにする必要があると考えてまいりました。

弊社では、SuperH で行っていたソフトウェアによる描画の一部を、ハードウェア処理に置き換え、描画処理にて多種多様な表現を行う方法を考えました。そのハードウェア描画処理を担当する LSI が、HD64411(Q2:Quick　2D Graphics Renderer)です。

本書では、HD64411(Q2)を使用するうえで、SuperH とのインタフェースを行う時の要点、およびサンプルプログラムを掲載し、ソフトウェア設計を行う際のご参考として役立てていただける情報をまとめました。

また、HD64411(Q2)は、SuperH のチップセットであり、Q シリーズ(Quick シリーズ)の第一弾が HD64411(Q2)であります。

ADJ-502-064

SuperH RISC engine **周辺** LSI

HD64411　Q2
Quick 2D Graphics Renderer

Q2　　HD64411

HD64411　Q2 **アプリケーションノート**

発行年月日　　**平成** 9 **年** 12 **月**　**第** 1.0 **版**

発行　　**株式会社　日立製作所**

半導体事業部

編集　　**株式会社日立マイコンシステム**

技術情報センタ

はじめに

弊社では、多方面的に描画システムを構築できるよう、弊社の 32 ビット RISC マイコンである SuperH を中核としたソフトウェアによるグラフィック描画システムの提案、および、SuperH のシリーズ化を行ってきました。そのような中で、ユーザが要求する多種多様な表現を実現するために、新たなシステム提案としてグラフィック描画にてリアルタイムな動作を表現し、今までに無かった新たなグラフィック描画の世界を展開する必要性が生じてきました。

そこで、SuperH で行っていたソフトウェアによる描画の一部を、ハードウェア処理に置き換え、描画処理にて多種多様な表現を行う方法を考えました。そのハードウェア描画処理を担当する LSI が、 Q シリーズ（Quick シリーズ）の第一弾、HD64411（Q2: Quick　2D Graphics Renderer）です。

Q シリーズは、SuperH ファミリとのチップセットによりグラフィックス表示システムに必要なジオメトリ処理、レンダリング処理、表示処理をコンパクトなシステムで高速に実現するもので、「シンプル」「リアルタイム」「アップグレード」をコンセプトとし、SuperH ファミリに対応したレンダリング処理と表示処理を行います。

本書では、HD64411（Q2）を使用するうえで、SuperH とのインタフェースを行う時の要点、およびサンプルプログラムを掲載し、ソフトウェア設計を行う際に参考として役立てていただける情報をまとめました。

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。

・SuperH の条件

SH7604（HD6417604SF28）を使用

CKIO の周波数　　：28.7MHz

メインメモリの容量　：4MB（MH5116160×2 個）

・Q2 の条件

Q2（HD64411）を使用

CLK0 の周波数　　：33MHz

CLK1 の周波数　　：14.32MHz

DCLK の周波数　　：CLK1/2　（sample7.c のみ、CLK1 を使用）

FCLK の周波数　　：CLK1/4　（sample7.c のみ、CLK1/2 を使用）

表示サイズ　　：320×240 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　　：2MB（MH5118165 の 60ns 品×1 個）

DA コンバータ　　：HD153510CP を使用

NTSC エンコーダ　　：CXA1645M

・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　　：日立製作所　SH シリーズ C コンパイラ

SH SERIES C Compiler Ver. 3.0F

リンケージエディタ：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

総目次

第 5 章　描画方法

第 6 章　サンプルプログラム集

1. システム構成例

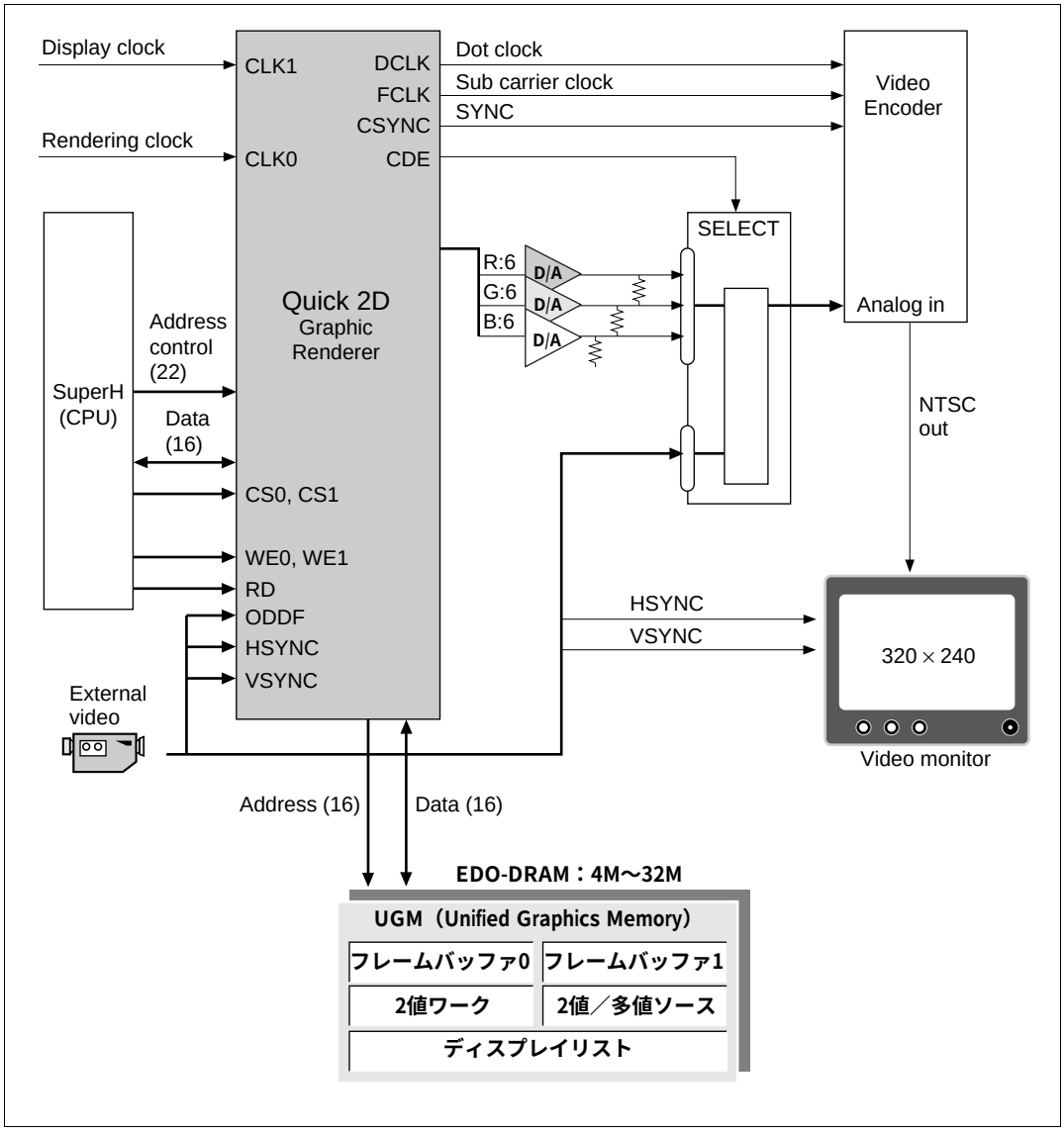
1. システム構成例

HD64411（Q2：Quick 2D Graphics Renderer）は、SuperH 用のグラフィックスアクセラレータ LSI で、基本的には SuperH と EDO ページモード DRAM と直結して使用することができます。

表示用のドットクロック（DCLK）の発生源となる CLK1、および HD64411 の動作クロック（CLK0）は、CLK0 で Q2 の逓倍数を N とすると、お互いの周波数に対し CLK0×N ≧2×DCLK の関係が成立する範囲で、非同期のクロックを使用できます。

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、HD64411 がノンインタレースで動作する時の表示サイズは、320×240、および 400×240 程度になり、インタレースシンク&ビデオで動作する時の表示サイズは、640×480 程度になります。

また、HD64411 を TV 同期モードにし、外部の映像信号から HSYNC, VSYNC, ODDF および CLK1 を供給することで、外部の映像信号と表示合成を行うことが可能です。



2. システム構築

第 2 章　目次

2.1 表示サイズの適合性の判定

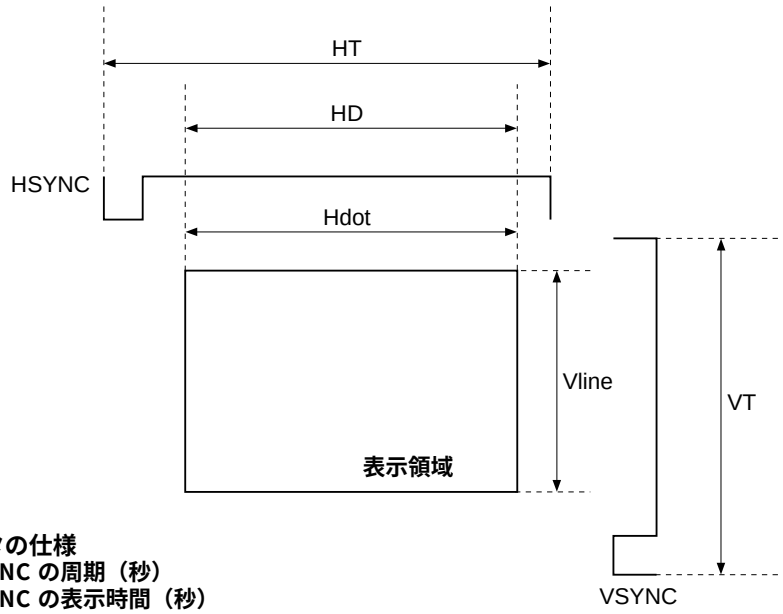
HD64411で表示させたいサイズが使用可能かを判定する方法について説明します。

判定を行うには、表示モニタのHSYNCおよびVSYNCの規定を知っている必要があります。

下記に示す式を満たせば、HD64411で表示できます。

$$\frac{HD}{Hdot} \leq \frac{CLK0 \times N}{2}$$

ただし、
・NはQ2の逓倍数（1、2または4）
・CLK0はQ2のCLK0端子に入力するクロック（Hz）



表示モニタの仕様
HT: HSYNC の周期（秒）
HD: HSYNC の表示時間（秒）
Hdot: 垂直方向の表示画素数（画素）
VT: VSYNC の周期（秒）
Vline: 垂直方向の表示ライン数（ライン）
【注】 VTおよびVlineは、Q2をノンインタレースモードで動作させたとき、1フレーム当たりの値を指定します。同様に、Q2をインタレースシンクモード、または、インタレースシンク&ビデオモードで動作させたときは、1フィールド当たりの値を指定します。

図 2.1 表示モニタの仕様

2.2 メモリ割り当て

2.2.1 HD64411 のメモリマッピング

HD64411 のアドレスマップドレジスタおよび UGM は、Super H のメモリ空間のキャッシュスルー空間にマッピングして、使用します。

図 2.2 に、UGM に 16M ビットの EDO ページモード DRAM メモリを使用したときのマップ例を示します。

HD64411 の A22～A1 端子には、UGM のアドレスを直接入力させる必要があります。この例では A1～A20 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ています。このため UGM は、残りの A21～A22 端子が Low レベルになるメモリ空間に配置をします。**図** 2.2 では、SuperH が UGM をアクセスした際に、A21～A22 端子が常に Low レベルになるように、H'22000000 から UGM を配置しています。

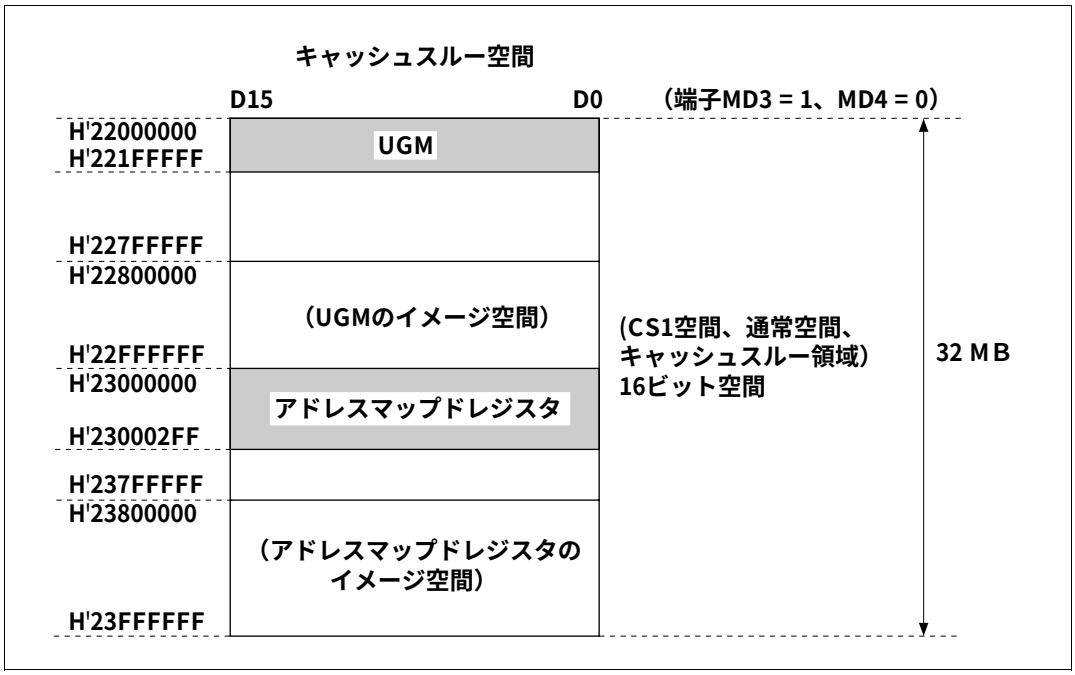


図 2.2　UGM **に** 16M **ビット** EDO **ページモード** DRAM **メモリを使用したときのマップ例**

2.2.2 UGM 領域配置

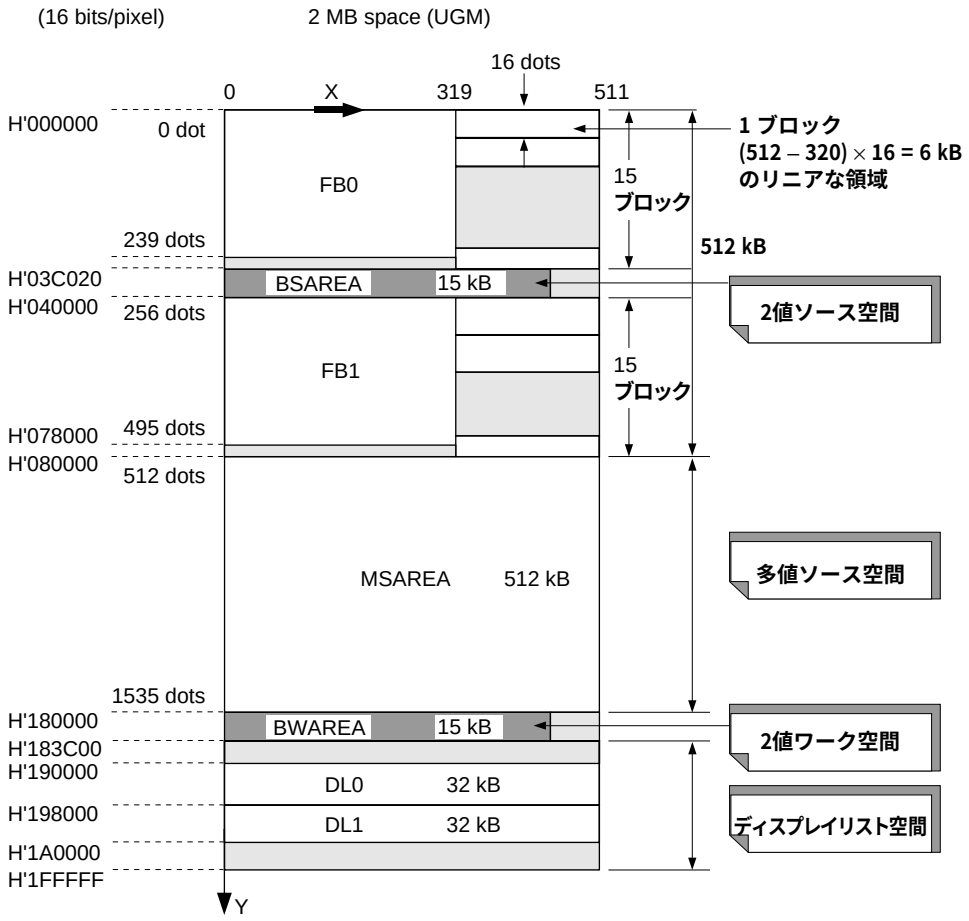
UGM 内の領域配置例を以下に示します。なお、表示サイズは 320×240 とします。

FB0、FB1 は、ダブルバッファ制御時に、表示領域および描画領域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は、Y 軸に接する 128 ドットおきの位置に設定します。設定した各々のアドレスは、描画を行うときの座標の原点になります。また、描画を行うときの座標をレンダリング座標といいます。

BWAREA は、ワーク座標として使用する領域です。ワーク座標の X 軸の最大画素数は、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［バイト］になります。

MSAREA は、自然画等の多値パターンを格納するための領域、BSAREA は、0 と 1 の組み合わせでつくられた 2 値パターンを格納するための領域です。また、DL0 および DL1 は、ディスプレイリストを格納するための領域です。DL0 と DL1 をソフトウェア制御にて、交互に、一方の領域を HD64411 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 Super H 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。このソフトウェア制御方法をコマンドダブルバッファ制御といいます。

また HD64411 では、Y 軸方向の 16 画素ごとに、アドレスが連続しているメモリのブロックを構成できます。**図** 2.3 の場合、FB0 の横に 6k バイトのブロックを 240/16=15 ブロック配置可能です。またブロックごとに、2 値ソース空間または、ディスプレイリストを格納するための領域としても使用できます。なお、各ブロックの開始位置は、Y 軸方向に 16 ライン毎の位置で、かつ X 軸方向において GBM=0 のときは 32 画素単位、GBM=1 のときは 16 画素単位の位置ごとになります。



FB0: Frame Buffer 0 area
FB1: Frame Buffer 1 area
BSAREA: Binary Source area
MSAREA: Multivalued Source area
BWAREA: Binary Work area
DL0: Display List 0 area
DL1: Display List 1 area

【注】 SH-2から見るとUGM アドレスはH'22000000 を加えた値となります。
たとえば、2値ワーク空間の先頭アドレスはH'22180000となります。

図 2.3 UGM **領域配置例**

3. SuperH とのインタフェース

第 3 章　目次

3.1 クロックの決定

HD64411 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。

CLK0 に使用可能なクロックの種類として、以下の(1)～(3)に示すいずれかのクロックが使用可能です。

(1) CPU が SH-1 または SH-2 のとき、CPU の CKIO 端子から出力されるクロックを直接 CLK0 端子に入力する場合。
このとき、CKIO から出力されるクロックと同じ周波数で、かつ同じ位相のクロックを用いるために、HD64411 を逓倍 OFF で使用してください。
また、このとき SuperH と HD64411 間のインタフェースは、クロック同期インタフェースになります。

(2) CPU が SH-3 のとき、CPU の CKIO 端子から出力されるクロック以外のクロックを CLK0 端子に入力する場合。
このとき SuperH と HD64411 間のインタフェースは、クロック非同期インタフェースになります。

(3) CPU に関係なく、CPU の CKIO 端子から出力されるクロック以外のクロックを CLK0 端子に入力する場合。
このとき SuperH と HD64411 間のインタフェースは、クロック非同期インタフェースになります。

また、CLK1 に入力するクロックは N を Q2 の MODE0～2 端子で設定する逓倍数とすると、N×CLK0[Hz]＞2×DCLK[Hz]を満足するクロックを入力してください。なお、CLK0=2×DCLK の条件で使用する場合、Q2 の逓倍を OFF にし、CLK0 および CLK1 は、同じ発生源のクロックから生成するようにします。これは、周囲の温度等でクロックの周波数が変化しても、 CLK0=2×DCLK の条件を保てるようにするための方法です。

Q2 は、FCLK 端子から、CLK1 の半分の周波数のクロックを出力しますが、FCLK を NTSC エンコーダのサブキャリアとして使用する場合、描画した図形の水平方向の明るさの変化が、FCLK の周波数の整数倍でないような図形を描画してください。

例えば、1 画素おきに、白と黒の点を描画させると、この色の変化が CLK1 の周波数と同じになり、FCLK の整数倍という条件を満します。このため、この色の変化がサブキャリアに干渉して、この点線近くで色ずれが発生します。通常、サブキャリア周波数は別クロックから発生させてください。

3.2 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 の動作周波数（CKIO）と HD64411 の動作周波数（CLK0 または N×CLK0、N は Q2 の逓倍数）の関係で決まります。

このため、HD64411 が出力する WAIT 信号を SuperH が見つけられるように、SuperH の AC タイミングと HD64411 の AC タイミングの両方を考慮した上で、ソフトウェアウェイトを設定してください。

図 3.1 に、SH-2 を使用し、CKIO = CLK0 = 28.63636MHz で使用する場合の例を挙げます。SuperH のソフトウェアサイクル（T_w）を 6 サイクルにすることで、SuperH の WAIT 端子の規定である t_{WTS} および t_{WTH} の規定を守れるようになり、SuperH と HD64411 間のハードウェアサイクル（T_{wx}）を確定できるようになります。（t_{WAS1} = 4tcyc0 ns (max.), t_{WAD} = 25 ns (max.))

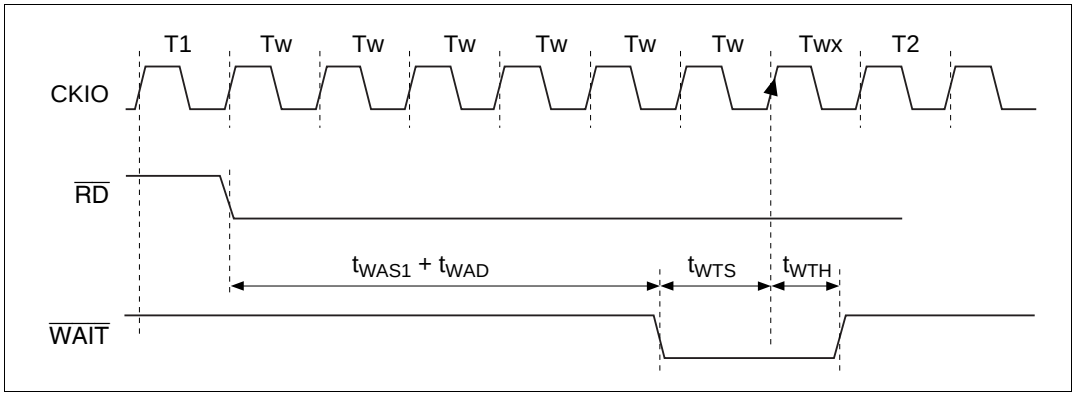


図 3.1 ソフトウェアウェイトサイクル例（SH-2、CKIO = CLK0 = 28.63636MHz **のとき）**

3.3 アドレスマップドレジスタの初期化手順

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。

次の（1）～（4）の順番に従って設定してください。

（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。
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、SuperH から UGM へのアクセスは行わないでください。

（2）レジスタアドレス 000～025 間のレジスタに初期値を設定します。

（3）レンダリングモードレジスタ内の GBM に 0 を設定した場合にのみ、カラーパレットレジスタに初期値を設定します。

（4）システム制御レジスタに SRES=0, DRES=0, DEN=1 または、SRES＝0, DRES＝0, DEN＝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、はじめて SuperH の UGM へのアクセスが可能になります。
なお、HD64411 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サンプルプログラム例を disp_on.c に示します。本プログラムでは、Q2 を表示オフへ移行させ、その間に UGM にリフレッシュを行うように、DSY レジスタに yw を設定しています。

3.4 UGM へのデータ転送

Super H 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、まず HD64411 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ます。これではじめて SuperH と UGM 間のデータ伝送が可能になります。

なお、表示同期動作を行っていないときに、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ときには UGM へのアクセスは行わないでください。

また、UGM にアクセスできるバスマスタは、ひとつだけです。このため、HD64411 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、HD64411 が描画処理を行っている最中であっても、転送可能です。

3.5 WAIT 信号について

Q2 が出力する WAIT 信号は以下のように動作しますので、SuperH と Q2 を直結せずに外付けインタフェース回路を設計する場合には注意が必要です。

Q2 が内部的に発生するウェイト要求は、SuperH のバスシーケンスとは無関係に発生するため、WAIT の Low レベル出力は現在実行されている SuperH のバスシーケンスに間に合う場合（**図** 3.2、**図** 3.4）と、そうでない場合（**図** 3.3、**図** 3.5）があります。これは、ソフトウェアウェイト数と動作クロックの関係によって決まりますが、後者の場合、Q2 は SuperH が WAIT を受け付けるタイミングの後に一時的に WAIT 信号＝Low を出力し、一旦 High 出力の後、次の SuperH のバスシーケンスで SuperH の WAIT 信号サンプリング位置に間に合うような WAIT 信号＝Low を出力します。

SuperH と Q2 を直結せずに外付けインタフェース回路を設計する場合には、このようなタイミングが発生しないように AC 特性を満足した回路設計を行い、適切なソフトウェアウェイト数を SuperH に設定する必要があります。

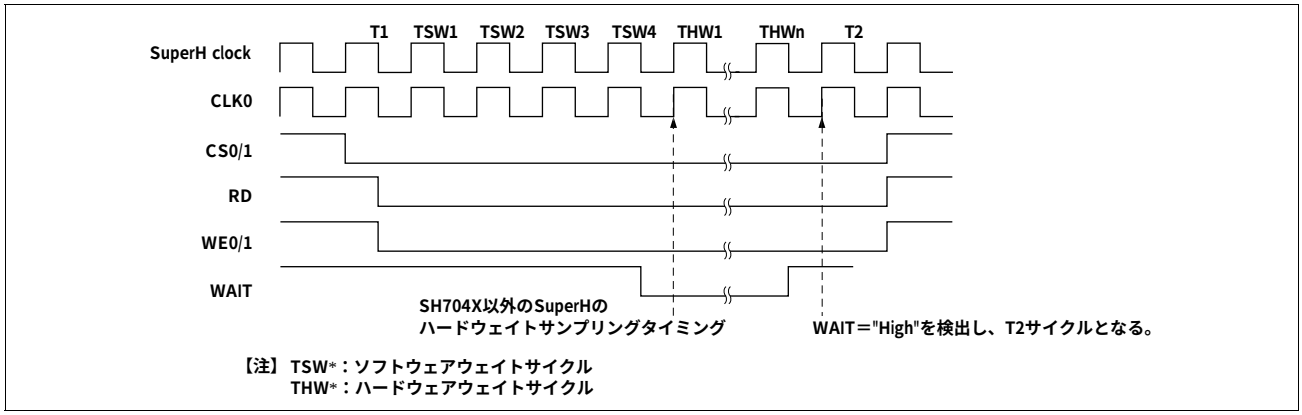


図 3.2　SH704X **以外の** SuperH **使用時でウェイト信号が間に合う場合**
（ソフトウェアウェイト＝4**、**SuperH clock **と** CLK0 **が同期）**

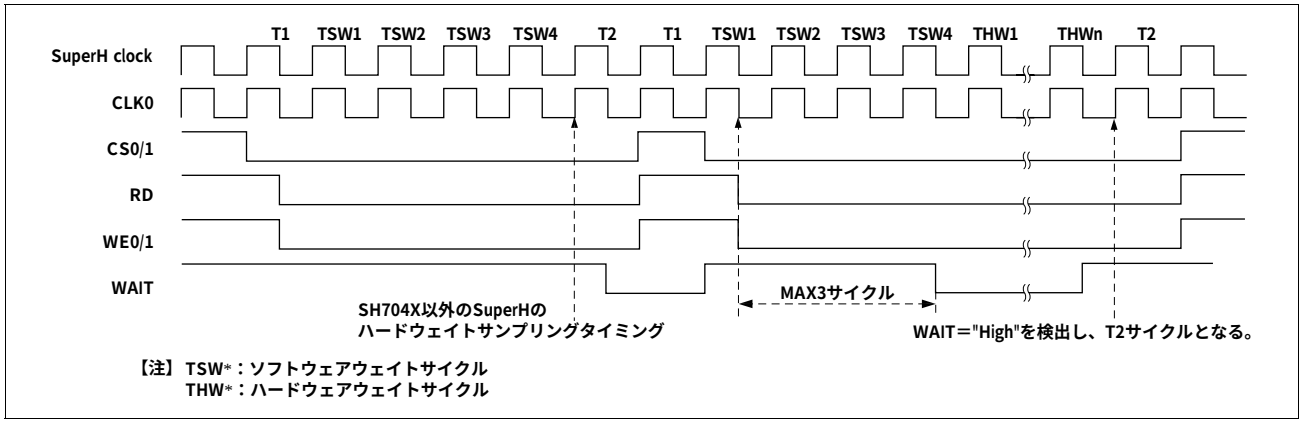


図 3.3　SH704X **以外の** SuperH **使用時でウェイト信号が間に合わない場合**
（ソフトウェアウェイト＝4**、**SuperH clock **と** CLK0 **が同期）**

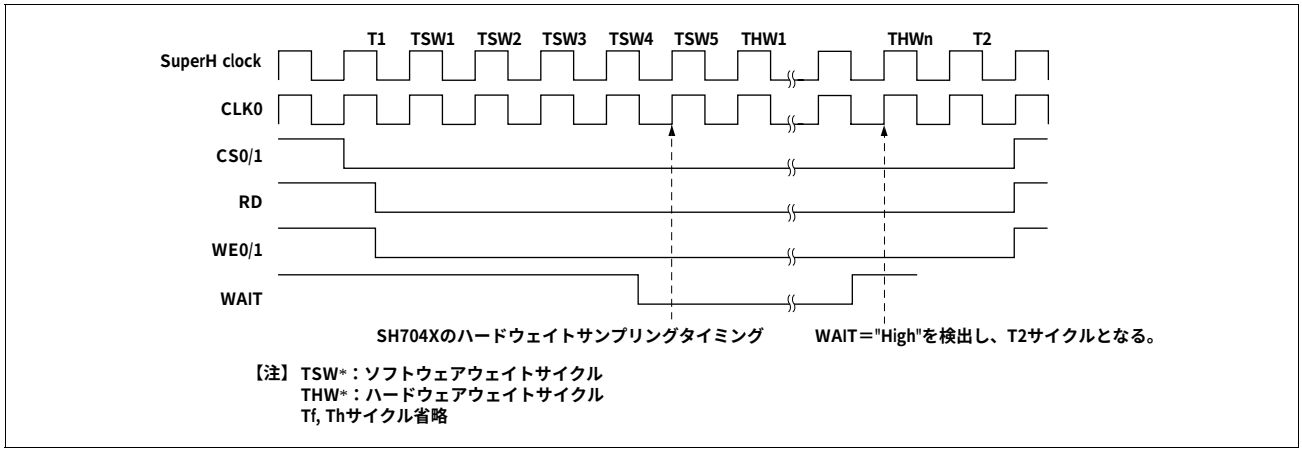


図 3.4　SH704X **使用時でウェイト信号が間に合う場合**
（ソフトウェアウェイト＝5、SuperH clock **と** CLK0 **が同期）**

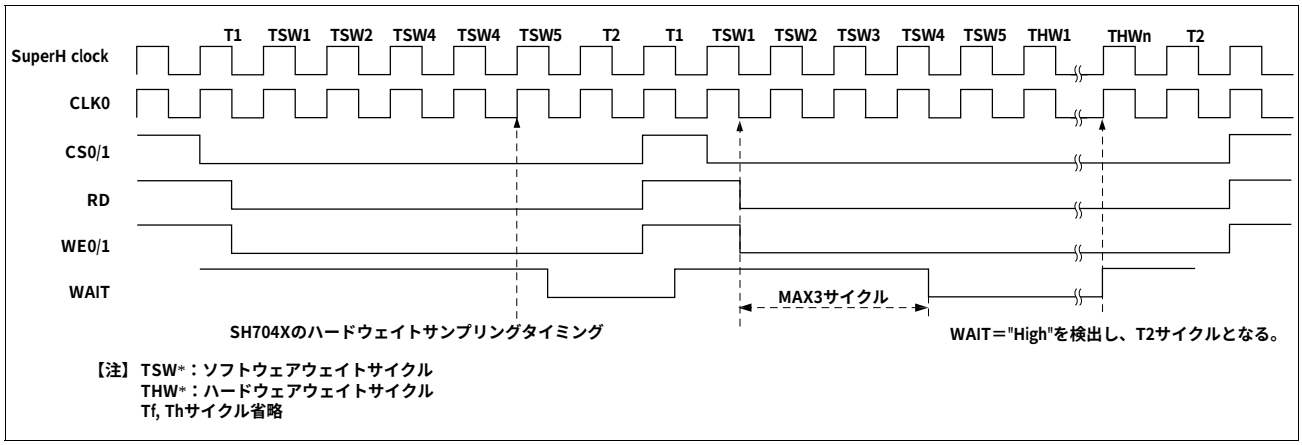


図 3.5　SH704X **使用時でウェイト信号が間に合わない場合**
（ソフトウェアウェイト＝5、SuperH clock **と** CLK0 **が同期）**

4. 表示制御方法

第 4 章　目次

4.1 カラーパレットの設定方法

HD64411 のカラーパレットは、GBM=0 のときにのみ有効です。SuperH が、カラーパレットにアクセスを行う前には、あらかじめ GBM が 0 であることを確認してください。

また、カラーパレットは 2 ワード連続アクセスで、カラーパレットのリードまたはライトを行う仕様になっています。このため、以下の（1)、(2）に示すことを行わないようにしてください。

(1) カラーパレットをリードするときの、2 ワード連続アクセス間におけるカラーパレットへのライト

(2) カラーパレットをライトするときの、2 ワード連続アクセス間におけるカラーパレットへのリード

なお、GBM が 1 の時に、SuperH がカラーパレットにアクセスを行うと、HD64411 は SuperH に対し WAIT 信号を出力し続けます。

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、大きく分けて以下に示す２種類の方法が考えられます。

（1）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。
（2）内部更新でフレームチェンジを行う方法
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ます。

FB0 および FB1 以外の領域にも、描画を行いたい場合、（2）の方法が有効です。これを行うには、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かを判定する必要があります。DBF=0 のとき、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 のとき、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** DSA0, DSA1 **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） VBK ビットをクリアし、VBK ビットが 1 になるのを待ちます。これを行うことで、内部更新期間が終了したことを確認できます。
（2）DSA0 に、次の内部更新で表示させたい位置の表示開始アドレスを設定します。
なお、設定を行っても、表示開始アドレスを管理するレジスタは、内部更新で設定値が有効になるため、次の内部更新が発生するまで有効な値として反映されません。
（3）DSA1 に、描画開始アドレスを設定します。
（4）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）～（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。また、DSA1 を変更することで、FB0 および FB1 以外の領域にも、描画を行うことが可能になります。
内部更新によるフレームチェンジを使用したサンプルプログラムを SAMPLE5.C に示します。

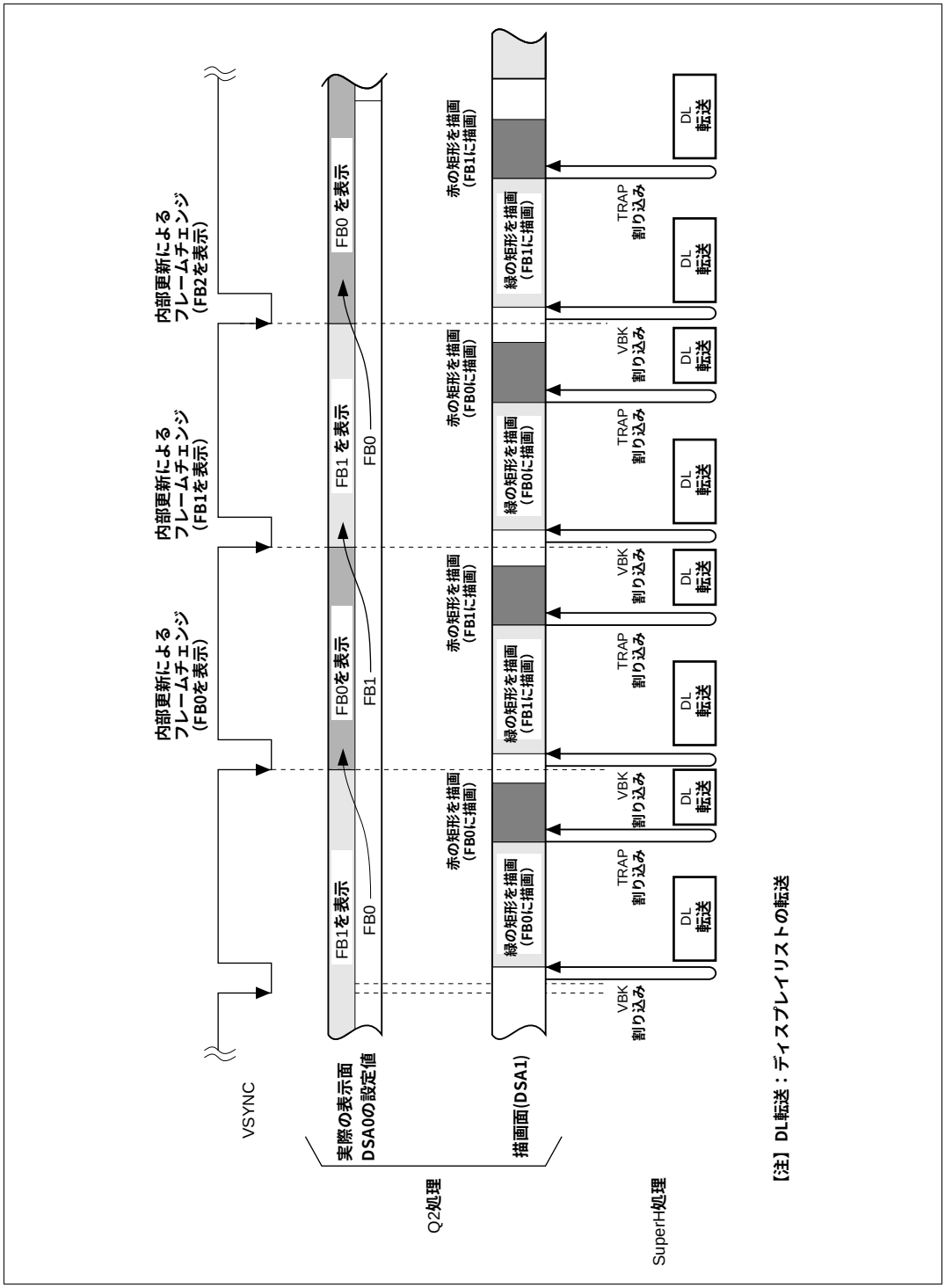


図 4.1　SAMPLE5.C の描画タイミングチャート

4.3 HD64411 に関する各モードへの移行方法

4.3.1 表示オフへの移行手順

DRES=0 かつ DEN=0 の場合、HD64411 は表示オフに移行し、表示オフ時出力レジスタに設定された色を画面全体に表示します。このモードに移行する際、DSY 設定値の値が VDE－VDS 未満であると、UGM のリフレッシュを行いません。このため、DRES=0 かつ DEN=0 を設定したい場合には、内部更新が始まる前に DSY に VDE－VDS 以上の値を設定し、続けて DRES=0 かつ DEN=0 を設定してください。なお、基本的に表示オフへの移行は、DRES=0, DEN=1 の状態からの移行を前提とします。

以下に手順を示します。

(1) VBK ビットをクリアし、VBK ビットが 1 になるのを待ちます。これを行うことで、内部更新期間が終了したことを確認できます。
(2) DSY に VDE - VDS 以上の値を設定します。なお、DSY は内部更新が行われるまで、以前に設定した値を内部的に有効な値とします。
(3) DRES = 0, DEN = 0 を設定します。内部更新により、HD64411 は表示オフへ移行します。

また、表示オフから DRES=0、DEN=1 の状態には、以下の手順で戻します。

(4) VBK ビットをクリアし、VBK ビットが 1 になるのを待ちます。これを行うことで、内部更新期間が終了したことを確認できます。
(5) DSY に VDE－VDS－1 の値を設定します。
(6) DRES = 0、DEN = 1 を設定します。内部更新により、HD64411 は表示開始アドレスで示されるアドレスから表示を行います。

4.3.2 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は、TVM ビットに 01 を設定することで移行できます。また、同期方式切り換えモードのとき、HD64411 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 HD64411 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してください。以下に手順を示します。

(1) DRES = 1, DEN = 0 を設定します。
(2) TVM=01 を設定します。HD64411 は、同期方式切り換えモードへ移行します。

なお、 DRES = 1, DEN = 0 が内部更新で有効になっている間は、SuperH は UGM をアクセスできません。

5.　描画方法

第５章　目次

5.1 描画の開始方法

HD64411 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、HD64411 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ことを目的とします。

（2）HD64411 に描画を行わせるために、（1）で配置したディスプレイリストに続けて、SuperH にて、ユーザが使用したい描画コマンドをディスプレイリストとして UGM に配置します。

（3）ディスプレイリストの終了を示すために、（2）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。

（4）レンダリング開始アドレスの設定を行った後、SuperH にて UGM のダミーリードを行って、Q2 内部の FIFO をフラッシュし続けて、RS ビットに 1 を設定してください。この設定を行うことで、HD64411 に描画を行わせることができます。なお、通常、 RS ビットに 1 を設定する場合、VBK ビットをクリアし、VBK ビットが 1 になるのを待ってから、RS ビットに 1 を設定してください。

POLYGON4 系コマンドを使用したプログラム例として、sample.c, sample2.c および sample3.c を掲載していますので、御参考ください。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く、それ以降は必要に応じて LCOFS, SCLIP コマンドをディスプレイリストとして使用してください。

5.2 任意形状のパターンの描画方法

5.2.1 多角形を描画する方法

HD64411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ひとつであるワーク参照および、ワーク座標を使用します。

以下にHD64411が行う描画手順を示します。

(1) CLRWコマンドを実行します。
(2) FTRAPコマンドで描画させたい多角形の形を、ワーク座標に描画します。
(3) POLYGON4Cコマンドのワーク指定を有効にし、ワーク座標に描画済みの多角形を参考にして、描画を行います。
(4) LINEコマンドにて、多角形の縁取りを描画します。

実際には、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HD64411が描画を行います。

プログラム例として、sample4.cを御参考ください。

5.2.2 任意の形のパターンを描画させる方法

HD64411で、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2通りの方法が考えられます。

(1) ワーク座標に2値の任意形状のパターンをSuperHで描画し、それをワーク参照を有効にしたPOLYGON4Cコマンドで、レンダリング座標に描画を行う方法。
プログラム例として、sample9.cを御参考ください。
sample9.cは、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、それを2値の任意な形のパターンとして使用したプログラム例です。
また、2値の任意パターンをワーク座標に配置する前に、ワーク座標をゼロクリアする必要があります。SuperHが直接、ワーク座標に2値パターンを描画する場合、ワーク座標のゼロクリアは、ディスプレイリスト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。
この方法は、SuperHがワーク座標を直接管理する際の方法として有効な方法です。

(2) 多値ソース座標に任意形状のカラーパターンをSuperHで配置し、それを透過指定を有効にしたPOLYGON4Aコマンドで、レンダリング座標に 描画を行う方法。
プログラム例として、sample6.cを御参考ください。

sample6.c は、16×16 画素のカラーパターンを 10 個、多値ソース座標に配置しておき、それを任意な形のパターンとして使用したプログラム例です。なお、sample6.c に、カラーパターンのデータは存在しません。
sample6.c を実行する前に、カラーパターンを多値ソースに転送しておく必要があります。

5.3 カレントポインタの設定手順

カレントポインタは MOVE コマンドで指定し、現在のローカルオフセット値を加算した値が設定されます。プログラム中で RLCOFS コマンドでローカルオフセットを変更しただけではカレントポインタは正しく設定されません。このような場合は必ず、RLCOFS コマンドの後に MOVE コマンドを発行し、カレントポインタをアップデートしてください。

（正しく動作する例）

```
LCOFS    XO＝0,YO＝0
MOVE     XC＝0,YC＝0
     :
     :
RLCOFS   XO＝10,YO＝-5
MOVE     XC＝0,YC＝0
RLINE    …
```

（誤った使用方法）

```
LCOFS    XO＝0,YO＝0
MOVE     XC＝0,YC＝0
     :
     :
RLCOFS   XO＝10,YO＝-5
RLINE    …←カレントポインタを指定していないので、ローカルオフセット値が加算されない
```

5.4 相対系コマンドの使用方法

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ときは、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、相対系コマンド以外のコマンドを挿入しないでください。

5.5 2値ソースを使用する際の注意

HD64411 が UGM に配置された 2 値ソースを使用する場合、HD64411 内部に存在するソースバッファに 2 値ソースを取り込み、これを使用して描画を行います。また、このソースバッファは 16 ワードの容量をもち、HD64411 は、UGM のアドレスが 16 ワードの境界を越えるごとに 16 ワードずつソースバッファに 2 値ソースを格納します。このため、16 ワード以内の 2 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、HD64411 に描画を行わせる必要があります。16 ワード以内の 2 値ソースを使用する場合、案として以下の方法が考えられます。

(1) コマンド毎に異なる 2 値ソースアドレスを指定する。
例えば、POLYGON4B コマンドで 16 ワード以内の 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。

(2) 透過指定を使用する。
16 ワードを越える 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。

(3) 2 値ソースをディスプレイリストの中に含ませる。
例えば、POLYGON4B コマンドを使用する場合、次の (a) ～ (c) に示すような方法でディスプレイリストを配置し描画を行うことで、ソースバッファを更新することが可能です。
(a) POLYGON4B コマンドをディスプレイリストとして UGM に配置します。
(b) (a) に続けて JUMP コマンドをディスプレイリストとして UGM に配置します。
JUMP コマンドの分岐先は、次項目 (c) で配置する 2 値ソースの最終アドレスの次のアドレスを指定します。
(c) (b) に続けて 2 値ソースを UGM に配置します。

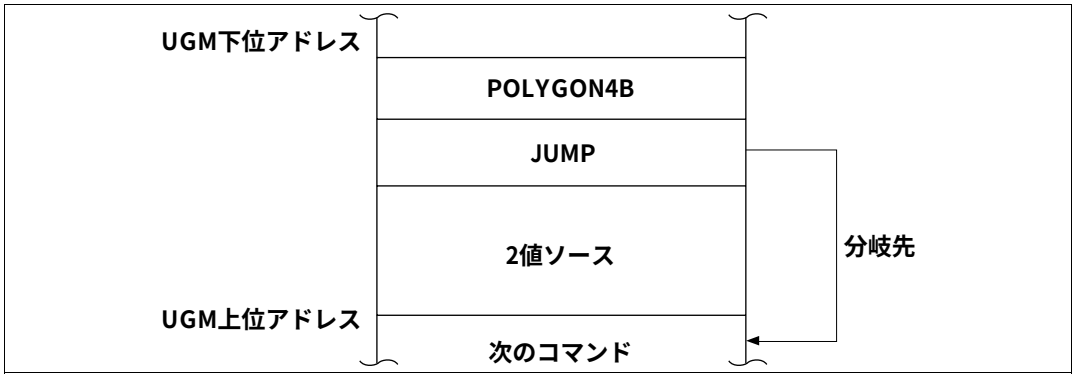


図 5.1 2 値ソースを含んだディスプレイリスト (POLYGON4B)

5.6 3次元空間を表現する方法

3次元空間を表現したサンプルプログラム例として、sample8.cを掲載しています。

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次元の座標値をもち、それらは、配列変数 coordindex で決定されます。

本サンプル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、ひと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、それをもとに決定します。

（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、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 UGM 上に生成します。

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、HD64411に描画を行わせた結果、3次元空間を表現できます。

6. サンプルプログラム集

第 6 章　目次

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、縮小および拡大を行います。なお、SuperH のメインメモリおよび HD64411 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'6000000～H'63FFFFF
HD64411 の UGM	:H'22000000～
HD64411 のアドレスマップドレジスタ	:H'23000000～

■レジスタの設定手順例

・disp_on.c：表示同期動作を開始するサンプルプログラム。
HD64411 のアドレスマップドレジスタに値を設定して、初期化を行います。

■ライブラリ例

・q2_mac.c : sample.c から sample8.c で使用した HD64411 のコマンドのマクロ関数のサンプル。
・q2L.c　：コマンドダブルバッファ制御のソースライブラリ。

■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、移動をさせます。
・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、移動をさせます。
・sample3.c: POLYGON4A を使用したサンプルプログラム。
自然画の文字を拡大、縮小、回転、移動をさせます。なお、MSAREA 内に 320×240 画素の自然画が配置されていることを前提とします。

■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内では、 五角形を描画させます。
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：内部更新によってダブルバッファ制御を行うサンプルプログラム
サンプルプログラム内では、1VSYNC 間に、2 回描画開始アドレスを変更して、矩形を描画しています。

・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: 640×480 の表示を行うサンプルプログラム。
・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 面で構成されます。
・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。

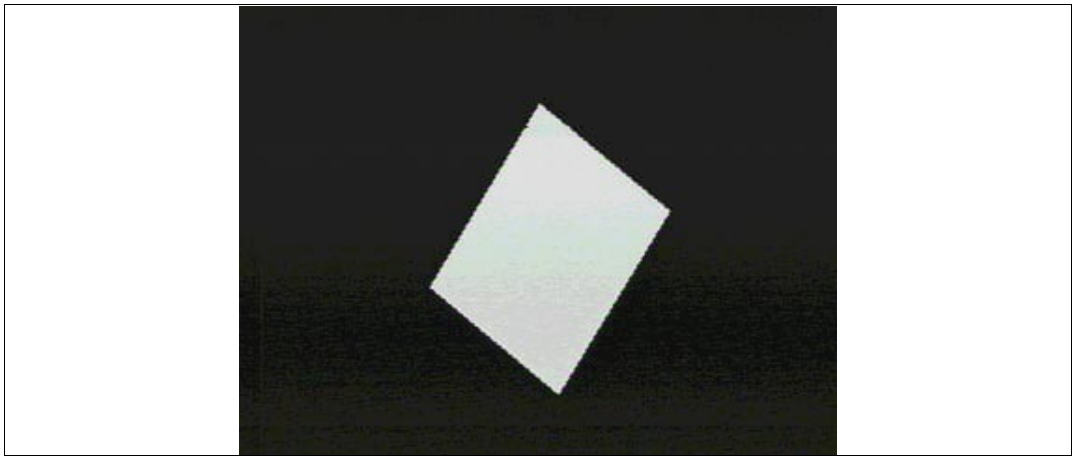


図 6.1　sample.c **の描画例**

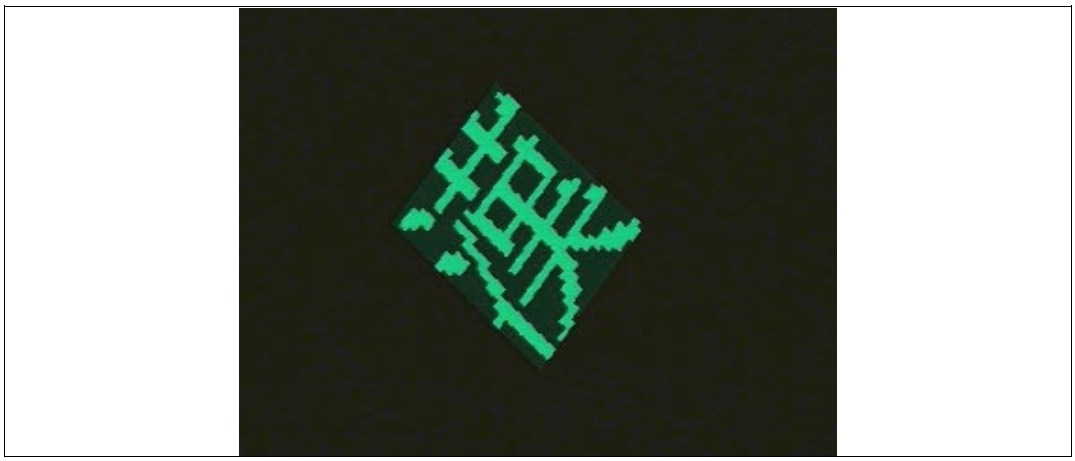


図 6.2　sample2.c **の描画例**

図 6.3　sample3.c **の描画例**

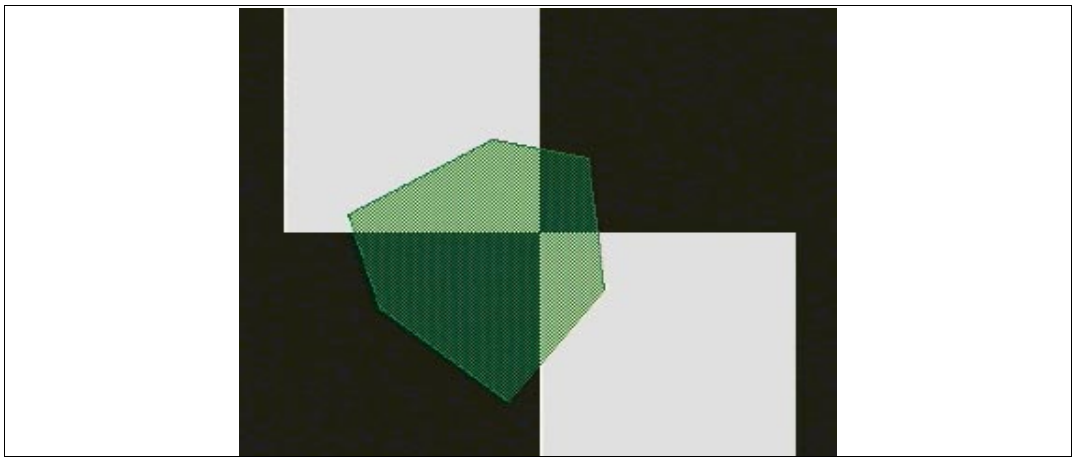


図 6.4　sample4.c **の描画例**

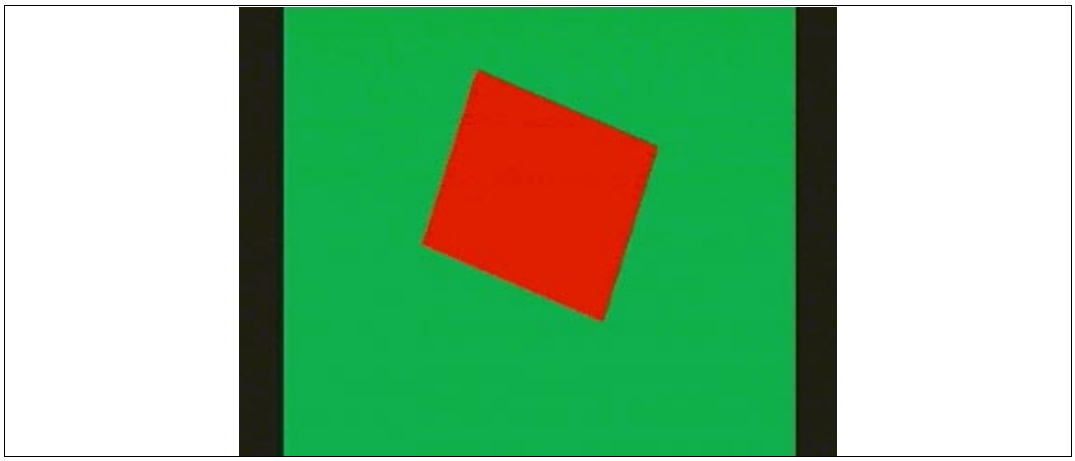


図 6.5　sample5.c **の描画例**

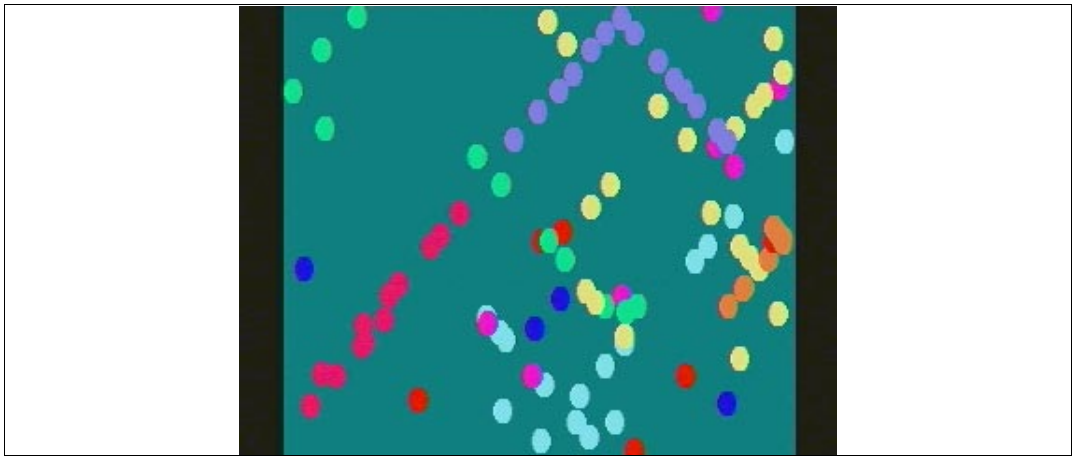


図 6.6　sample6.c **の描画例**

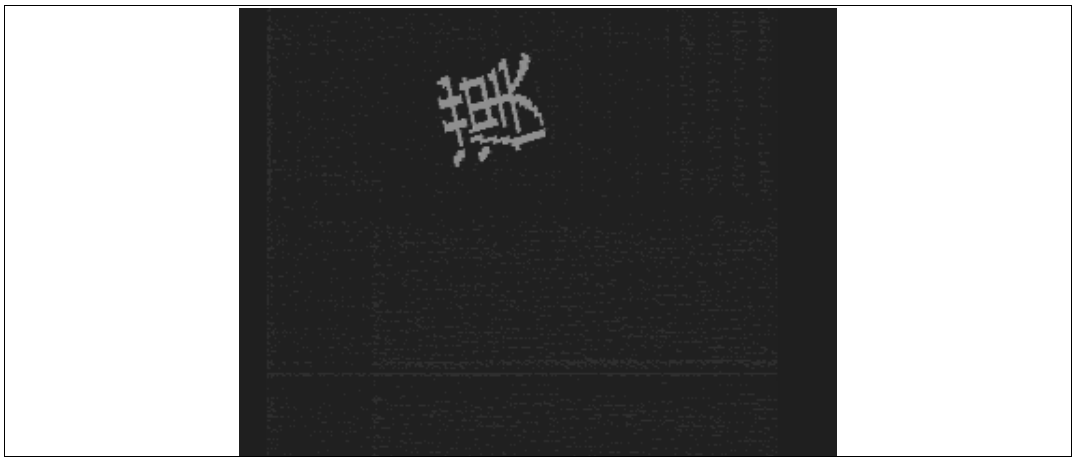


図 6.7　sample7.c **の描画例**

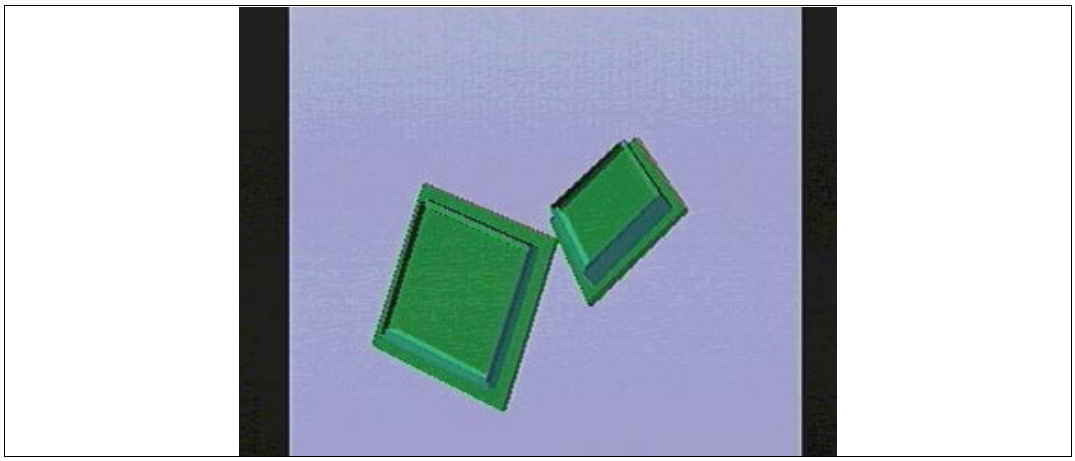


図 6.8　sample8.c **の描画例**

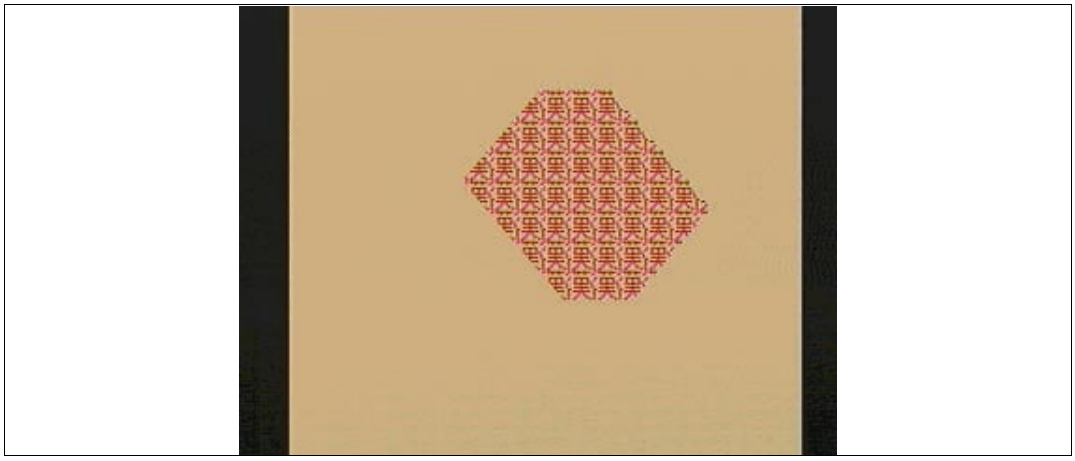


図 6.9　sample9.c **の描画例**

6.1 サンプルプログラムの説明

6.1.1 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、フレームチェンジ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、DISPLIST0、DISPLIST1で定義されます。

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

/*-------------------------------------------------------
(1). 関数のプロトタイプを記述します。
-------------------------------------------------------------*/
/* ------------ Q2 Display List Functions ---------------- */
void polygon4c(short draw_mode, short poly[], short color1);
void lcofs(short draw_mode,short xo,short yo);
void sclip(short draw_mode,short xmax,short ymax);
void trap(short draw_mode);
void   ginit(short DBmode);      /* Graphics System Open   (Q2 Initialize) */
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);

/*-------------------------------------------------------------------------------
(2). 使用したい Q2 のコマンドのマクロ関数を記述します。
     なお、 DISPLIST_ptr ポインタ変数は、draw_start, draw_end 関数で用いる
     予約変数です。 DISPLIST_ptr は、ディスプレイリストを生成する際に用います。
-------------------------------------------------------------------------------*/
/* ----- Q2 function macro ------------------------------ */
#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 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 trap( draw_mode )\
{\
        *DISPLIST_ptr++ = 0xf800 ¦ draw_mode;          /* TRAP */\
}


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;

/*-------------------------------------------------------------------------------
(3). ginit 関数にて、Q2 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 */

/*-------------------------------------------------------------------------------
(4). 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 */


/*-------------------------------------------------------------------------------
(5). 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。
-------------------------------------------------------------------------------*/
        draw_start();               /* Start Transfer Display List to UGM */

/*-------------------------------------------------------------------------------
(6). Q2 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);

/*-------------------------------------------------------------------------------
(7). draw_end 関数にて、Q2 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、Q2 がレンダリング（描画）を行います。
-------------------------------------------------------------------------------*/
                draw_end(DBmode,&first,quick);

/*-------------------------------------------------------------------------------
(8).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、Q2 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);

/*-------------------------------------------------------------------------------
(9).サンプルプログラムは、ソフトウェア割り込みにて、プログラムを終了させることを
       前提としています。このため、trapa(40)関数を記述しています。
       ソフトウェア割り込みでの終了を前提としない場合には、trapa(40)関数を記述する
       必要はありません。
-------------------------------------------------------------------------------*/
                trapa(40);
}
```

6.1.2 記述を行う時の要点

SuperH と HD64411 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。また、SuperH は内部にキャッシュを内蔵しているものもあり、これを使用する場合は、1 回の VSYNC 間に呼び出す関数の数を減らすようにプログラムを作成します。これを行うことで、キャッシュのミスヒットを減らせるようになります。

本書に記載しているサンプルプログラムは、この点を考慮し、POLYGON4A 等の呼び出す頻度の高いものは、#define を使用して、直接、プログラムコードを埋め込むようにしています。

また、ディスプレイリストを管理するための draw_start 関数、draw_end 関数は、1 回の VSYNC 間に 1〜2 回程度に頻度でしか呼び出さないので、これらは、関数として定義しています。

6.2 サンプルプログラムのソースリスト

6.2.1 disp_on.c のソースファイル

（1）disp_on.c のメークバッチファイル

disp_on.c maker batch file

```
SHC /DEBUG /NOLISTFILE /OPT=1 /CPU=7600 disp_on.c  >disp_on.tag
LNK disp_on /OUTPUT=disp_on /LIB=c:\SHC\LIB\SHCLIB.LIB /FORM=A /START=P(6000000)
CNVS disp_on
DEL disp_on.abs
DEL disp_on.obj
```

(2) disp_on.c のソースリスト

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

            Q2 Display List / SHC sample program
              ( Disp on. )

                Copyright(c) Hitachi Ltd. 1997

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

#include <machine.h>
#include <stdio.h>
#include <stdlib.h>

#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 -----*/
#define  outport(add,data) ( *VU_SHORT(add) ) = ( (unsigned short)(data) )

#define BASE_ADDRESS   0x23000000L          /* Byte address (SH series)*/

#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 _Q2SSAH    0x01CL | BASE_ADDRESS     /* No.00E */
#define _Q2WSAH    0x01EL | BASE_ADDRESS     /* No.00F */
#define _Q2DMASH   0x020L | BASE_ADDRESS     /* No.010 */
```

```
#define _Q2DMASL    0x022L | BASE_ADDRESS     /* No.011 */
#define _Q2DMAW     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 _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 */

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


void main(void)
{
  char  first = ON;
  char  quick = OFF;

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

       ginit(DBmode);     /* Initialize Q2 */

       trapa(40);
}

void ginit(short DBmode)
{
  unsigned short hsw =  32;
  unsigned short xs  =  65;
  unsigned short xw  = 320;
  unsigned short hc  = 455;
```

```
  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,          0x0024);
       outport(_Q2DSMR,          0x0105);
       outport(_Q2REMR,          0x0001);
       outport(_Q2IEMR,          0x0000);

       /* Set Memory Control Regisers    */
       outport(_Q2DSX,    xw-1);    /* Display size of x                */
       outport(_Q2DSY,    yw  );    /* Display size of y                */
       outport(_Q2DSA0,   0x0000);  /* Frame buffer 0 area start address(H) */
       outport(_Q2DSA1,   0x0004);  /* 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(_Q2SSAH,   0x0008);  /* Color area sorce start address(H)    */
       outport(_Q2WSAH,   0x0018);  /* Work area start address(H)          */
       outport(_Q2DMASH,  0x0000);  /* DMA transfer start address(H)       */
       outport(_Q2DMASL,  0x0000);  /* DMA transfer start address(L)       */
       outport(_Q2DMAW,   0x0000);  /* DAM transfer word                 */

       /* Display Contral Registers */
       outport(_Q2HDS,        hsw+xs-3   );
       outport(_Q2HDE,        hsw+xs-3+xw);
       outport(_Q2VDS,        ys        );
       outport(_Q2VDE,        ys+yw     );
       outport(_Q2HSW,        hsw-1     );
       outport(_Q2HC,         hc-1      );
       outport(_Q2VSP,        vc-vsw    );
       outport(_Q2VC,         vc        );
       outport(_Q2DOR,        0x0000);
       outport(_Q2DOG_DOB,    0x007C);
       outport(_Q2CDR,        0x00FC);
       outport(_Q2CDG_CDB,    0xFCFC);

       outport(_Q2SYSR, 0x0080);      /* Start display */
}
```

6.2.2 q2_mac.c のソースファイル

```
/*

                  Q2 command macro sample

               Copyright(c) Hitachi Ltd. 1997
*/

extern unsigned short *DISPLIST_ptr;    /* Display List Address pointer */

/* ------------- 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);


/* ----- 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++ = 0x7000 | 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 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++ = 0x7800 | 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;\
}
```

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

6.2.3 q2L.c のソースファイル

```
/*
                Command double buffer control for Q2

               ((for 16bit/Pixel 65536 Color mode))
          Copyright(c) Hitachi Ltd. 1997
*/
#include <machine.h>


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


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

#define BASE_ADDRESS  0x23000000L    /* Q2 internal register base address */
                              /* Byte address (SuperH series)*/

#define UGMBASE      0x22000000L    /* UGM base address */
                              /* Byte address (SuperH series)*/

#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 _Q2SSAH      0x01CL | BASE_ADDRESS         /* No.00E */
#define _Q2WSAH      0x01EL | BASE_ADDRESS         /* No.00F */
#define _Q2DMASH     0x020L | BASE_ADDRESS         /* No.010 */
#define _Q2DMASL     0x022L | BASE_ADDRESS         /* No.011 */
#define _Q2DMAW      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 _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  DISPLIST0          0x190000L
#define  DISPLIST1          0x198000L

#define _DUMMY     0x000000L | UGMBASE

#define OFF               0
#define ON                1


/* ------------- Q2 Display List Functions ---------------- */
extern void trap(short draw_mode);
void  ginit(short DBmode);     /* Q2 Initialize */
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=0;  /* Display List Double Buffering Destination Number */
unsigned short *DISPLIST_ptr;   /* Display List Address pointer */
short         flag;

/* ------------------------------------------------------------------- */
void ginit(short DBmode)       /* Initialize Graphic System */
/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
{
  unsigned short hsw =  32;
```

```
unsigned short xs  =  65;
unsigned short xw  = 320;
unsigned short hc  = 455;

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,   0x0024);
     outport(_Q2DSMR,   0x0105);
     outport(_Q2REMR,   0x0001);
     outport(_Q2IEMR,   0x0000);

     /* Set Memory Control Regisers    */
     outport(_Q2DSX,   xw-1);   /* Display size of x                */
     outport(_Q2DSY,   yw-1);   /* Display size of y                */
     outport(_Q2DSA0,  0x0000); /* Frame buffer 0 area start address(H) */
     outport(_Q2DSA1,  0x0004); /* 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(_Q2SSAH,  0x0008); /* Color area sorce start address(H)    */
     outport(_Q2WSAH,  0x0018); /* Work area start address(H)          */
     outport(_Q2DMASH, 0x0000); /* DMA transfer start address(H)       */
     outport(_Q2DMASL, 0x0000); /* DMA transfer start address(L)       */
     outport(_Q2DMAW,  0x0000); /* DAM transfer word                 */

     /* Display Contral Registers */
     outport(_Q2HDS,              hsw+xs-3   );
     outport(_Q2HDE,              hsw+xs-3+xw);
     outport(_Q2VDS,              ys        );
     outport(_Q2VDE,              ys+yw     );
     outport(_Q2HSW,              hsw-1     );
     outport(_Q2HC,               hc-1      );
     outport(_Q2VSP,              vc-vsw    );
     outport(_Q2VC,               vc        );
     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(DMmode){		/* Enable Draw/Display */
		case 0:
			outport(_Q2SYSR, 0x2000);
	/* Idle,Auto Change mode,No DMA,No Cache,7604 */
			break;
		case 1:
			outport(_Q2SYSR, 0x2040);
	/* Idle,Auto Renderring mode,No DMA,No Cache,7604 */
			break;
		default:
			outport(_Q2SYSR, 0x2080);
	/* Idle,manual change moe,No DMA,No Cache,7604 */
	}
}


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;
	return( 0 );
}


short draw_end(short DBmode,char *first,char quick)  /* Execute Display List */
{
  unsigned short st, ah, al;
  short count = 1;
  unsigned short d;

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

```
switch(DBmode) {

	case 2:		/* Manual Change */
		change_com_buffer();

		/* Transfer command from internal command buffer to UGM */
		inport(_DUMMY);

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

		do{
			outport( _Q2SRCR, 0x0800 );
/* Clear VBK bit */
			while( d=inport(_Q2SR), (d&0x0800)==0);
				/* VBK = 1 ? */
		}while(count++<=0);

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

		break;

	case 1:	/* Auto Renderring */
		change_com_buffer();

		outport(_Q2SRCR,0x0800);	/* Clear VBK bit */
		while(st = inport(_Q2SR),(st & 0x0800) == 0); /*Wait VBK*/

		/* Transfer command from internal command buffer to UGM */
		inport(_DUMMY);

		/* Renddering start */
		outport(_Q2SYSR,0x2140);
		break;

	default:	/* Auto Change */
		change_com_buffer();

		outport(_Q2SRCR,0x0800);	/* Clear VBK bit */
		while(st = inport(_Q2SR), (st & 0x0800) == 0);
					/* Wait VBK */

		/* Transfer command from internal command buffer to UGM */
		inport(_DUMMY);
```

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


void change_com_buffer(void)
{
  unsigned short  ah,al;
  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;
        }
        ah = (CMD_StartAddress >> 16L) & 0xffff;
        al =  CMD_StartAddress & 0xffffL;
        outport(_Q2DLSAH,ah);                 /* Change DLSAR */
        outport(_Q2DLSAL,al);
}
```

6.2.4 sample.c のソースファイル

（1）sample.c のメークバッチファイル

sample.c maker batch file

```
SHC /DEBUG /NOLISTFILE /OPT=1 /CPU=7600 sample.c  >sample.tag
LNK sample /OUTPUT=sample /LIB=c:\SHC\LIB\SHCLIB.LIB /FORM=A /START=P(6000000)
CNVS sample
DEL sample.abs
DEL sample.obj
```

（2）sample.c のソースリスト

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

                Copyright(c) Hitachi Ltd. 1997
*/
#include <machine.h>
#include <stdio.h>
#include <math.h>


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


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

#define BASE_ADDRESS 0x23000000L    /* Q2 internal register base address */
                              /* Byte address (SuperH series)*/

#define UGMBASE    0x22000000L     /* UGM base address */
                              /* Byte address (SuperH series)*/

#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 _Q2SSAH     0x01CL | BASE_ADDRESS     /* No.00E */
#define _Q2WSAH     0x01EL | BASE_ADDRESS     /* No.00F */
#define _Q2DMASH    0x020L | BASE_ADDRESS     /* No.010 */
#define _Q2DMASL    0x022L | BASE_ADDRESS     /* No.011 */
#define _Q2DMAW     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 _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  DISPLIST0        0x190000L
#define  DISPLIST1        0x198000L

#define _DUMMY    0x000000L | UGMBASE

short         DrawBuffer=0;
unsigned short *DISPLIST_ptr;
short         flag;


#define PI            3.14159
#define MAX_WORD       512
#define OFF           0
#define ON            1

/* ------------- Q2 Display List Functions ---------------- */
```

```
void polygon4c(short draw_mode, short poly[], short color1);
void lcofs(short draw_mode,short xo,short yo);
void sclip(short draw_mode,short xmax,short ymax);
void trap(short draw_mode);

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

/* ----- Q2 function macro ------------------------------- */
#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 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 trap( draw_mode )\
{\
       *DISPLIST_ptr++ = 0xf800 | draw_mode;      /* TRAP */\
}


void main(void)
{
```

```
	short array[MAX_WORD];
	short d, i, h, j, k, del;
	short mn, kj;
	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;

	ginit(DBmode);		/* Initialize Q2 */
	init_start();		/* Initialize Transfer Procedure */
	draw_start();		/* Start Transfer Display List to UGM */
	sclip(0x0000, 319,239);
	lcofs(0x0000,0,0);

	del = 4;

	while(1) {

/* 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+160;
                        array[i+1] =  ty+115;
                }

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

/* Move --------------------------------------- */
        h=260;
        j=110;
        mn=160;
        kj=50;
        for(d=0;d<=360;d+=5) {
                drawclrscrn();

                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+h;
                        array[i+1] =  ty+j;
                    }

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

        }
        trapa(40);
}

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

/* ploygon4c (clear) ------------------------------------------------- */
        array[0] =   0;   array[1] =   0;
        array[2] =  319;  array[3] =   0;
        array[4] =  319;  array[5] =  239;
        array[6] =   0;   array[7] =  239;
        polygon4c(0x0000, array, 0x0000);
}

void ginit(short DBmode)
{
  unsigned short hsw =  32;
  unsigned short xs  =  65;
  unsigned short xw  = 320;
  unsigned short hc  = 455;

  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,  0x0024);
        outport(_Q2DSMR,  0x0105);
        outport(_Q2REMR,  0x0001);
        outport(_Q2IEMR,  0x0000);
```

```
	/* Set Memory Control Regisers    */
	outport(_Q2DSX,  xw-1);    /* Display size of x                */
	outport(_Q2DSY,  yw-1);    /* Display size of y                */
	outport(_Q2DSA0, 0x0000);  /* Frame buffer 0 area start address(H) */
	outport(_Q2DSA1, 0x0004);  /* 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(_Q2SSAH,  0x0008); /* Color area sorce start address(H)    */
	outport(_Q2WSAH,  0x0018); /* Work area start address(H)          */
	outport(_Q2DMASH, 0x0000); /* DMA transfer start address(H)       */
	outport(_Q2DMASL, 0x0000); /* DMA transfer start address(L)       */
	outport(_Q2DMAW,  0x0000); /* DAM transfer word                */

	/* Display Contral Registers */
	outport(_Q2HDS,     hsw+xs-3   );
	outport(_Q2HDE,     hsw+xs-3+xw);
	outport(_Q2VDS,     ys        );
	outport(_Q2VDE,     ys+yw     );
	outport(_Q2HSW,     hsw-1     );
	outport(_Q2HC,      hc-1      );
	outport(_Q2VSP,     vc-vsw    );
	outport(_Q2VC,      vc        );
	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,7604 */
			break;
		case 1:
			outport(_Q2SYSR, 0x2040);
/* Idle,Auto Renderring mode,No DMA,No Cache,7604 */
			break;
		default:
			outport(_Q2SYSR, 0x2080);
/* Idle,manual change moe,No DMA,No Cache,7604 */
	}
}
```

```
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;
       return( 0 );
}

short draw_end(short DBmode,char *first,char quick)/* Execute Display List */
{
  unsigned short st, ah, al;
  short count = 1;
  unsigned short d;

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

       switch(DBmode) {

              case 2:         /* Manual Change */
                     change_com_buffer();

                     /* Transfer command from internal command buffer to UGM */
                     inport(_DUMMY);

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

                     do{
                            outport( _Q2SRCR, 0x0800 );  /* Clear VBK bit */
                            while( d=inport(_Q2SR), (d&0x0800)==0);
                                                  /* VBK = 1 ? */
                     }while(count++<=0);

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

```
                }

                break;

        case 1:         /* Auto Renderring */
                change_com_buffer();

                outport(_Q2SRCR,0x0800);            /* Clear VBK bit */
                while(st = inport(_Q2SR), (st & 0x0800) == 0);
                                                    /* Wait VBK */

                /* Transfer command from internal command buffer to UGM */
                inport(_DUMMY);

                /* Renddering start */
                outport(_Q2SYSR,0x2140);
                break;

        default:      /* Auto Change */
                change_com_buffer();

                outport(_Q2SRCR,0x0800);            /* Clear VBK bit */
                while(st = inport(_Q2SR), (st & 0x0800) == 0);
                                                   /* Wait VBK */

                /* Transfer command from internal command buffer to UGM */
                inport(_DUMMY);

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


void change_com_buffer(void)
{
  unsigned short  ah,al;
  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;
        }
        ah = (CMD_StartAddress >> 16L) & 0xffff;
        al =  CMD_StartAddress & 0xffffL;
        outport(_Q2DLSAH,ah);                    /* Change DLSAR */
        outport(_Q2DLSAL,al);
}
```

6.2.5 sample2.c のソースファイル

（1）sample2.c のメークバッチファイル

```
SHC /DEBUG /NOLISTFILE /OPT=1 /CPU=7600 sample2.c  >sample2.tag
LNK sample2 /OUTPUT=sample2 /LIB=c:\SHC\LIB\SHCLIB.LIB /FORM=A /START=P(6000000)
CNVS sample2
DEL sample2.abs
DEL sample2.obj
```

（2）sample2.c のソースリスト

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

                Copyright(c) Hitachi Ltd. 1997
*/
#include <machine.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


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


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

#define BASE_ADDRESS 0x23000000L       /* Q2 internal register base address */
                                  /* Byte address (SuperH series)*/

#define UGMBASE    0x22000000L         /* UGM base address */
                                  /* Byte address (SuperH series)*/

#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 _Q2SSAH     0x01CL | BASE_ADDRESS          /* No.00E */
#define _Q2WSAH     0x01EL | BASE_ADDRESS          /* No.00F */
#define _Q2DMASH    0x020L | BASE_ADDRESS          /* No.010 */
#define _Q2DMASL    0x022L | BASE_ADDRESS          /* No.011 */
#define _Q2DMAW     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 _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  DISPLIST0        0x190000L
#define  DISPLIST1        0x198000L

#define _DUMMY      0x000000L | UGMBASE

short          DrawBuffer=0;
unsigned short *DISPLIST_ptr;
short          flag;


#define PI          3.14159
#define MAX_WORD     512
#define OFF         0
#define ON          1
```

```
/* ------------- Q2 Display List Functions ---------------- */
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 lcofs(short draw_mode,short xo,short yo);
void sclip(short draw_mode,short xmax,short ymax);
void trap(short draw_mode);

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

/* ----- Q2 function macro ------------------------------- */
#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 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 trap( draw_mode )\
{\
	*DISPLIST_ptr++ = 0xf800 | draw_mode;    /* TRAP */\
}


void main()
{
	short array[MAX_WORD];
	short ah,al;
	short d, i, h, j, k, del;
	short mn, kj;
	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;

	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 = 0x3c020L + UGMBASE;
	       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, 319,239);
	lcofs(0x0000,0,0);

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

	while(1) {

/* 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(0x0000, ah,al, 24,24, array,
                                backcolor,forecolor); /* polygon4b */

                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(0x0000, ah,al, 24,24, array,
                                backcolor,forecolor); /* polygon4b */

                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+160;
                        array[i+1] =  ty+115;
                }

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

```
/* Move ---------------------------------------- */
             h=260;
             j=110;
             mn=160;
             kj=50;
             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+h;
                           array[i+1] =  ty+j;
                    }
                    polygon4b(0x0000, ah,al, 24,24, array,
                           backcolor,forecolor);    /* polygon4b */
                    h-=5;
                    draw_end(DBmode,&first,quick);
                    draw_start();
             }

      }
      trapa(40);
}

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

/* ploygon4c (clear) ------------------------------------------------------ */
      array[0] =   0;  array[1] =   0;
      array[2] =  319;  array[3] =   0;
      array[4] =  319;  array[5] =  239;
      array[6] =   0;  array[7] =  239;
      polygon4c(0x0000, array, 0x0000);
}

void ginit(short DBmode)
{
  unsigned short hsw =  32;
  unsigned short xs  =  65;
  unsigned short xw  = 320;
```

```
unsigned short hc  = 455;

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,  0x0024);
     outport(_Q2DSMR,  0x0105);
     outport(_Q2REMR,  0x0001);
     outport(_Q2IEMR,  0x0000);

     /* Set Memory Control Regisers    */
     outport(_Q2DSX,   xw-1);    /* Display size of x                */
     outport(_Q2DSY,   yw-1);    /* Display size of y                */
     outport(_Q2DSA0,  0x0000);  /* Frame buffer 0 area start address(H) */
     outport(_Q2DSA1,  0x0004);  /* 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(_Q2SSAH,  0x0008);  /* Color area sorce start address(H)    */
     outport(_Q2WSAH,  0x0018);  /* Work area start address(H)          */
     outport(_Q2DMASH, 0x0000);  /* DMA transfer start address(H)       */
     outport(_Q2DMASL, 0x0000);  /* DMA transfer start address(L)       */
     outport(_Q2DMAW,  0x0000);  /* DAM transfer word                */

     /* Display Contral Registers */
     outport(_Q2HDS,    hsw+xs-3   );
     outport(_Q2HDE,    hsw+xs-3+xw);
     outport(_Q2VDS,    ys        );
     outport(_Q2VDE,    ys+yw     );
     outport(_Q2HSW,    hsw-1     );
     outport(_Q2HC,     hc-1      );
     outport(_Q2VSP,    vc-vsw    );
     outport(_Q2VC,     vc        );
     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,7604 */
			break;
		case 1:
			outport(_Q2SYSR, 0x2040);
/* Idle,Auto Renderring mode,No DMA,No Cache,7604 */
			break;
		default:
			outport(_Q2SYSR, 0x2080);
/* Idle,manual change moe,No DMA,No Cache,7604 */
	}
}

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;
	return( 0 );
}


short draw_end(short DBmode,char *first,char quick)/* Execute Display List */
{
  unsigned short st, ah, al;
  short count = 1;
  unsigned short d;

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

	switch(DBmode) {

		case 2:         /* Manual Change */
```

```
                change_com_buffer();

                /* Transfer command from internal command buffer to UGM */
                inport(_DUMMY);

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

                do{
                        outport( _Q2SRCR, 0x0800 );  /* Clear VBK bit */
                        while( d=inport(_Q2SR), (d&0x0800)==0);
                                                /* VBK = 1 ? */
                }while(count++<=0);

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

                break;

        case 1:         /* Auto Renderring */
                change_com_buffer();

                outport(_Q2SRCR,0x0800);            /* Clear VBK bit */
                while(st = inport(_Q2SR), (st & 0x0800) == 0);
                                                /* Wait VBK */

                /* Transfer command from internal command buffer to UGM */
                inport(_DUMMY);

                /* Renddering start */
                outport(_Q2SYSR,0x2140);
                break;

        default:     /* Auto Change */
                change_com_buffer();

                outport(_Q2SRCR,0x0800);            /* Clear VBK bit */
                while(st = inport(_Q2SR), (st & 0x0800) == 0);
                                                /* Wait VBK */

                /* Transfer command from internal command buffer to UGM */
                inport(_DUMMY);

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

```
       return( 0 );
}


void change_com_buffer(void)
{
  unsigned short  ah,al;
  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;
       }
       ah = (CMD_StartAddress >> 16L) & 0xffff;
       al =  CMD_StartAddress & 0xffffL;
       outport(_Q2DLSAH,ah);                /* Change DLSAR */
       outport(_Q2DLSAL,al);
}
```

6.2.6 sample3.c のソースファイル

（1）sample3.c のメークバッチファイル

sample3.c maker batch file

```
SHC /DEBUG /NOLISTFILE /OPT=1 /CPU=7600 sample3.c  >sample3.tag
LNK sample3 /OUTPUT=sample3 /LIB=c:\SHC\LIB\SHCLIB.LIB /FORM=A /START=P(6000000)
CNVS sample3
DEL sample3.abs
DEL sample3.obj
```

（2）sample3.c のソースリスト

```
/*
                  Q2 Display List / SHC sample program (3)
            (Zoom/Shrink/Rotate/Move 'Image')

          Must be load "child2.bmp" to (txs,tys) = (80,288), before.
          SSAR = 0x80000

                       Copyright(c) Hitachi Ltd. 1997
*/
#include <machine.h>
#include <stdio.h>
#include <math.h>


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


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

#define BASE_ADDRESS  0x23000000L    /* Q2 internal register base address */
                              /* Byte address (SuperH series)*/

#define UGMBASE      0x22000000L    /* UGM base address */
                              /* Byte address (SuperH series)*/

#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 _Q2SSAH     0x01CL | BASE_ADDRESS      /* No.00E */
#define _Q2WSAH     0x01EL | BASE_ADDRESS      /* No.00F */
#define _Q2DMASH    0x020L | BASE_ADDRESS      /* No.010 */
#define _Q2DMASL    0x022L | BASE_ADDRESS      /* No.011 */
#define _Q2DMAW     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 _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  DISPLIST0   0x190000L
#define  DISPLIST1   0x198000L

#define _DUMMY     0x000000L | UGMBASE

short         DrawBuffer=0;
unsigned short *DISPLIST_ptr;
short         flag;


#define  PI         3.14159
#define  MAX_WORD    512
#define OFF         0
```

```
#define ON         1

/* ------------- Q2 Display List Functions ---------------- */
void polygon4a(short draw_mode, short txs,short tys,short tdx,short tdy, short poly[]);
void polygon4c(short draw_mode, short poly[], short color1);
void lcofs(short draw_mode,short xo,short yo);
void sclip(short draw_mode,short xmax,short ymax);
void trap(short draw_mode);

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

/* ----- 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 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 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 trap( draw_mode )\
{\
	*DISPLIST_ptr++ = 0xf800 | draw_mode;	/* TRAP */\
}


void main(void)
{
	short array[MAX_WORD];
	short d, i, h, j, k, del;
	short mn, kj;
	double rad;
	short tx,ty;
	short txs, tys;
	short size_x, size_y;
	short DBmode;
/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
	char  first,quick;

	DBmode = 1;
	first  = ON;
	quick  = OFF;
	j  = 110;
	mn = 160;

	ginit(DBmode);	/* Initialize Q2 */

/* Bitmap Location of UGM ----------------------------------------------------
	>BMPLOAD child2.bmp 0 768
------------------------------------------------------------------------------ */
	txs  = 80;
	tys  = 288;
	size_x = 144;
		/* Must be load "child2.bmp" to (txs,tys) = (80,288), before */
	size_y = 185;

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

```
        draw_start();    /* Start Transfer Display List to UGM */
        sclip(0x0000, 319,239);
        lcofs(0x0000,0,0);

        del = 2;

        while(1) {

/* 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,size_x,size_y,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,size_x,size_y,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+160;
                          array[i+1] =  ty+115;
                    }

                    polygon4a(0x0000,txs,tys,size_x,size_y,array);
                                                    /* polygon4a */
                    draw_end(DBmode,&first,quick);
                    draw_start();
              }

/* Move ---------------------------------------- */
              h=260;
              j=110;
              mn=160;
              kj=50;
              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+h;
                          array[i+1] =  ty+j;
                    }
                    polygon4a(0x0000,txs,tys,size_x,size_y,array);
                                                    /* polygon4a */
                    h-=5;
                    draw_end(DBmode,&first,quick);
                    draw_start();
              }

      }
}

void clrscrn(void)
{
```

```
        short array[MAX_WORD];

/* ploygon4c (clear) ------------------------------------------------- */
        array[0] =   0;   array[1] =   0;
        array[2] =  319;  array[3] =   0;
        array[4] =  319;  array[5] =  239;
        array[6] =   0;   array[7] =  239;
        polygon4c(0x0000, array, 0x0000);
}

void ginit(short DBmode)
{
  unsigned short hsw =  32;
  unsigned short xs  =  65;
  unsigned short xw  = 320;
  unsigned short hc  = 455;

  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,  0x0024);
        outport(_Q2DSMR,  0x0105);
        outport(_Q2REMR,  0x0001);
        outport(_Q2IEMR,  0x0000);

        /* Set Memory Control Regisers    */
        outport(_Q2DSX,     xw-1);  /* Display size of x                */
        outport(_Q2DSY,     yw-1);  /* Display size of y                */
        outport(_Q2DSA0,  0x0000);  /* Frame buffer 0 area start address(H) */
        outport(_Q2DSA1,  0x0004);  /* 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(_Q2SSAH,  0x0008);  /* Color area sorce start address(H)    */
        outport(_Q2WSAH,  0x0018);  /* Work area start address(H)          */
        outport(_Q2DMASH, 0x0000);  /* DMA transfer start address(H)       */
        outport(_Q2DMASL, 0x0000);  /* DMA transfer start address(L)       */
        outport(_Q2DMAW,  0x0000);  /* DAM transfer word                */

        /* Display Contral Registers */
        outport(_Q2HDS,     hsw+xs-3   );
        outport(_Q2HDE,     hsw+xs-3+xw);
        outport(_Q2VDS,     ys        );
```

```
	outport(_Q2VDE,    ys+yw     );
	outport(_Q2HSW,    hsw-1     );
	outport(_Q2HC,     hc-1      );
	outport(_Q2VSP,    vc-vsw    );
	outport(_Q2VC,     vc        );
	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,7604 */
			break;
		case 1:
			outport(_Q2SYSR, 0x2040);
/* Idle,Auto Renderring mode,No DMA,No Cache,7604 */
			break;
		default:
			outport(_Q2SYSR, 0x2080);
/* Idle,manual change moe,No DMA,No Cache,7604 */
	}
}

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;
```

```
	return( 0 );
}


short draw_end(short DBmode,char *first,char quick)/* Execute Display List */
{
  unsigned short st, ah, al;
  short count = 1;
  unsigned short d;

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

	switch(DBmode) {

		case 2:       /* Manual Change */
			change_com_buffer();

			/* Transfer command from internal command buffer to UGM */
			inport(_DUMMY);

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

			do{
				outport( _Q2SRCR, 0x0800 );   /* Clear VBK bit */
				while( d=inport(_Q2SR), (d&0x0800)==0);
                                        /* VBK = 1 ? */
			}while(count++<=0);

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

			break;

		case 1:     /* Auto Renderring */
			change_com_buffer();

			outport(_Q2SRCR,0x0800);             /* Clear VBK bit */
			while(st = inport(_Q2SR), (st & 0x0800) == 0);
                                        /* Wait VBK */

			/* Transfer command from internal command buffer to UGM */
			inport(_DUMMY);

			/* Renddering start */
			outport(_Q2SYSR,0x2140);
```

```
                    break;

             default:    /* Auto Change */
                    change_com_buffer();

                    outport(_Q2SRCR,0x0800);            /* Clear VBK bit */
                    while(st = inport(_Q2SR), (st & 0x0800) == 0);
                                                  /* Wait VBK */

                    /* Transfer command from internal command buffer to UGM */
                    inport(_DUMMY);

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


void change_com_buffer(void)
{
  unsigned short  ah,al;
  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;
      }
```

```
        ah = (CMD_StartAddress >> 16L) & 0xffff;
        al =  CMD_StartAddress & 0xffffL;
        outport(_Q2DLSAH,ah);                /* Change DLSAR */
        outport(_Q2DLSAL,al);
}
```

6.2.7 sample4.c のソースファイル

（1）sample4.c のメークバッチファイル

```
SHC /DEBUG /NOLISTFILE /OPT=1 /CPU=7600 sample4.c  >sample4.tag
LNK sample4 /OUTPUT=sample4 /LIB=c:\SHC\LIB\SHCLIB.LIB /FORM=A /START=P(6000000)
CNVS sample4
DEL sample4.abs
DEL sample4.obj
```

（2）sample4.c のソースリスト

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

             Copyright(c) Hitachi Ltd. 1997
*/
#include <machine.h>
#include <stdio.h>

#define MAX_WORD    512
#define OFF         0
#define ON          1


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


/*====================   DEFINE TYPES   ========================*/
/*----- Define I/O function -----*/
#define outport(add,data) ( *VU_SHORT(add) ) = ( (unsigned short)(data) )
#define inport(add)      ( *VU_SHORT(add) )

#define BASE_ADDRESSS   0x23000000L    /* Q2 internal register base address */
                                /* Byte address (SuperH series) */

#define UGMBASE        0x22000000L    /* UGM base address */
                                /* Byte address (SuperH series) */

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 _Q2SSAH    0x01CL | BASE_ADDRESS    /* No.00E */
#define _Q2WSAH    0x01EL | BASE_ADDRESS    /* No.00F */
#define _Q2DMASH   0x020L | BASE_ADDRESS    /* No.010 */
#define _Q2DMASL   0x022L | BASE_ADDRESS    /* No.011 */
#define _Q2DMAW    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 _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  DISPLIST0   0x190000L
#define  DISPLIST1   0x198000L

#define _DUMMY     0x000000L | UGMBASE

short          DrawBuffer=0;
unsigned short *DISPLIST_ptr;
short          flag;


/* ------------- Q2 Display List Functions ---------------- */
void polygon4c(short draw_mode, short poly[], short color1);
void line(short draw_mode,short color,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 lcofs(short draw_mode,short xo,short yo);
void sclip(short draw_mode,short xmax,short ymax);
void trap(short draw_mode);

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

/* ----- Q2 function macro ------------------------------- */
#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 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 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 line( draw_mode, color, n, array )\
{\
  short *_q2_arrary = 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 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 trap( draw_mode )\
{\
	*DISPLIST_ptr++ = 0xf800 | draw_mode;	/* TRAP */\
}


void main(void)
{
	short array[MAX_WORD], poly[MAX_WORD];
	short DMmode;
	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 = 2;
	first  = ON;
	quick  = OFF;

	ginit(DBmode);	/* Initialize Q2 */

	init_start();	/* Initialize Transfer Procedure */
	draw_start();	/* Start Transfer Display List to UGM */
	sclip(0x0000, 319,239);
```

```
        lcofs(0x0000,0,0);

/* ploygon4c ( clear screen ) ------------------------------------- */
        array[0] =   0; array[1] =   0;  /* Left Up    */
        array[2] =  319; array[3] =   0;  /* Right Up   */
        array[4] =  319; 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);

        trapa(40);
}

void ginit(short DBmode)
{
  unsigned short hsw =  32;
  unsigned short xs  =  65;
  unsigned short xw  = 320;
  unsigned short hc  = 455;

  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,  0x0024);
        outport(_Q2DSMR,  0x0105);
        outport(_Q2REMR,  0x0001);
        outport(_Q2IEMR,  0x0000);

        /* Set Memory Control Regisers    */
        outport(_Q2DSX,     xw-1);  /* Display size of x               */
        outport(_Q2DSY,     yw-1);  /* Display size of y               */
        outport(_Q2DSA0,  0x0000);  /* Frame buffer 0 area start address(H) */
        outport(_Q2DSA1,  0x0004);  /* 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(_Q2SSAH,  0x0008);  /* Color area sorce start address(H)   */
        outport(_Q2WSAH,  0x0018);  /* Work area start address(H)         */
        outport(_Q2DMASH, 0x0000);  /* DMA transfer start address(H)       */
        outport(_Q2DMASL, 0x0000);  /* DMA transfer start address(L)       */
        outport(_Q2DMAW,  0x0000);  /* DAM transfer word                */

        /* Display Contral Registers */
        outport(_Q2HDS,     hsw+xs-3   );
        outport(_Q2HDE,     hsw+xs-3+xw);
        outport(_Q2VDS,     ys        );
```

```
	outport(_Q2VDE,    ys+yw     );
	outport(_Q2HSW,    hsw-1     );
	outport(_Q2HC,     hc-1      );
	outport(_Q2VSP,    vc-vsw    );
	outport(_Q2VC,     vc        );
	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,7604 */
			break;
		case 1:
			outport(_Q2SYSR, 0x2040);
/* Idle,Auto Renderring mode,No DMA,No Cache,7604 */
			break;
		default:
			outport(_Q2SYSR, 0x2080);
			/* Idle,manual change moe,No DMA,No Cache,7604 */
	 }
}

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;
```

```
	return( 0 );
}


short draw_end(short DBmode,char *first,char quick)/* Execute Display List */
{
  unsigned short st, ah, al;
  short count = 1;
  unsigned short d;

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

	switch(DBmode) {

		case 2:       /* Manual Change */
			change_com_buffer();

			/* Transfer command from internal command buffer to UGM */
			inport(_DUMMY);

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

			do{
				outport( _Q2SRCR, 0x0800 );   /* Clear VBK bit */
				while( d=inport(_Q2SR), (d&0x0800)==0);
                                        /* VBK = 1 ? */
			}while(count++<=0);

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

			break;

		case 1:     /* Auto Renderring */
			change_com_buffer();

			outport(_Q2SRCR,0x0800);            /* Clear VBK bit */
			while(st = inport(_Q2SR), (st & 0x0800) == 0);
                                        /* Wait VBK */

			/* Transfer command from internal command buffer to UGM */
			inport(_DUMMY);

			/* Renddering start */
			outport(_Q2SYSR,0x2140);
			break;
```

```
            default:    /* Auto Change */
                  change_com_buffer();

                  outport(_Q2SRCR,0x0800);            /* Clear VBK bit */
                  while(st = inport(_Q2SR), (st & 0x0800) == 0);
                                                /* Wait VBK */

                  /* Transfer command from internal command buffer to UGM */
                  inport(_DUMMY);

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


void change_com_buffer(void)
{
  unsigned short  ah,al;
  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;
      }
      ah = (CMD_StartAddress >> 16L) & 0xffff;
      al =  CMD_StartAddress & 0xffffL;
```

```
        outport(_Q2DLSAH,ah);                /* Change DLSAR */
        outport(_Q2DLSAL,al);
}
```

6.2.8 sample5.c のソースファイル

（1）sample5.c のメークバッチファイル

```
SHC /DEBUG /NOLISTFILE /OPT=1 /CPU=7600 sample5.c  >sample5.tag
LNK sample5 /OUTPUT=sample5 /LIB=c:\SHC\LIB\SHCLIB.LIB /FORM=A /START=P(6000000)
CNVS sample5
DEL sample5.abs
DEL sample5.obj
```

（2）sample5.c のソースリスト

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

            ( Do not use Q2's double buffer control )
            ( Move 'Solid' Polygon )

            Copyright(c) Hitachi Ltd. 1997
*/
#include <machine.h>
#include <stdio.h>
#include <math.h>


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


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

#define BASE_ADDRESS  0x23000000L      /* Q2 internal register base address */
                                /* Byte address (SuperH series)*/

#define UGMBASE      0x22000000L      /* UGM base address */
                                /* Byte address (SuperH series)*/

#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 _Q2SSAH    0x01CL | BASE_ADDRESS       /* No.00E */
#define _Q2WSAH    0x01EL | BASE_ADDRESS       /* No.00F */
#define _Q2DMASH   0x020L | BASE_ADDRESS       /* No.010 */
#define _Q2DMASL   0x022L | BASE_ADDRESS       /* No.011 */
#define _Q2DMAW    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 _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  DISPLIST0       0x190000L
#define  DISPLIST1       0x198000L

#define _DUMMY    0x000000L | UGMBASE

short         DrawBuffer=0;
unsigned short *DISPLIST_ptr;
short         flag;


#define PI        3.14159
#define MAX_WORD   512
#define OFF       0
#define ON        1
```

```
/* ------------- Q2 Display List Functions ---------------- */
void polygon4c(short draw_mode, short poly[], short color1);
void lcofs(short draw_mode,short xo,short yo);
void sclip(short draw_mode,short xmax,short ymax);
void trap(short draw_mode);

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

/* ----- Q2 function macro ------------------------------- */
#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 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 trap( draw_mode )\
{\
       *DISPLIST_ptr++ = 0xf800 | draw_mode;    /* TRAP */\
}


void main(void)
{
       short draw_buffer_area = 0x0000, display_buffer_area = 0x0004;
```

```
        unsigned long displying_area_address;
        unsigned long drawing_area_address;

        short array[MAX_WORD];
        short d, i, kj, st;
        double rad;
        short tx,ty;
        short temp;
        short DBmode;
/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
        char  first,quick;

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

        ginit(DBmode);    /* Initialize Q2 */

        /* Deside displaying area and drawing area */
{
  short st;
  short temp;

        displying_area_address = _Q2DSA0;
        drawing_area_address   = _Q2DSA1;

        st = inport( _Q2SR );
        if( (st & 0x0100) != 0 ) {
               displying_area_address = _Q2DSA1;
               drawing_area_address   = _Q2DSA0;
               temp                = draw_buffer_area;
               draw_buffer_area      = display_buffer_area;
               display_buffer_area    = temp;
        }
}

        init_start();    /* Initialize Transfer Procedure */
        draw_start();    /* Start Transfer Display List to UGM */
        sclip(0x0000, 319,239);
        lcofs(0x0000,0,0);

        while(1) {

        /* MOVE */
               kj = 50;
```

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

/* Wait VSYNC */
       outport( _Q2SRCR, 0x0800 );          /* Clear VBK bit */
       while( st=inport(_Q2SR), (st&0x0800)==0);  /* VBK = 1 ? */

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


/* Set drawing start address for job No.1 */
       outport( drawing_area_address,   display_buffer_area );

/* Job No.1 : clear screen */
       array[0] =    0;   array[1] =    0;
       array[2] =  319;   array[3] =    0;
       array[4] =  319;   array[5] =  239;
       array[6] =    0;   array[7] =  239;
       polygon4c(0x0000, array, 0x0606);

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


/* Set drawing start address for job No.2 */
       outport( drawing_area_address,   display_buffer_area );

/* Job No.2 : draw a solid ploygon */
       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+kj;
              array[i+1] =  ty+100;
       }

polygon4c(0x0000,array,0xF800);    /* POLYGON4C */
       kj+=5;

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


/* Change display buffer */
```

```
                    temp             = draw_buffer_area;
                    draw_buffer_area   = display_buffer_area;
                    display_buffer_area = temp;


             } /* for */

      } /* while */

      trapa(40);
}

void ginit(short DBmode)
{
  unsigned short hsw =  32;
  unsigned short xs  =  65;
  unsigned short xw  = 320;
  unsigned short hc  = 455;

  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,  0x0024);
      outport(_Q2DSMR,  0x0105);
      outport(_Q2REMR,  0x0001);
      outport(_Q2IEMR,  0x0000);

      /* Set Memory Control Regisers    */
      outport(_Q2DSX,   xw-1);   /* Display size of x               */
      outport(_Q2DSY,   yw-1);   /* Display size of y                */
      outport(_Q2DSA0,  0x0000);  /* Frame buffer 0 area start address(H) */
      outport(_Q2DSA1,  0x0004);  /* 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(_Q2SSAH,  0x0008);  /* Color area sorce start address(H)    */
      outport(_Q2WSAH,  0x0018);  /* Work area start address(H)          */
      outport(_Q2DMASH, 0x0000);  /* DMA transfer start address(H)       */
      outport(_Q2DMASL, 0x0000);  /* DMA transfer start address(L)       */
      outport(_Q2DMAW,  0x0000);  /* DAM transfer word                 */

      /* Display Contral Registers */
      outport(_Q2HDS,     hsw+xs-3   );
```

```
	outport(_Q2HDE,     hsw+xs-3+xw);
	outport(_Q2VDS,     ys        );
	outport(_Q2VDE,     ys+yw     );
	outport(_Q2HSW,     hsw-1     );
	outport(_Q2HC,      hc-1      );
	outport(_Q2VSP,     vc-vsw    );
	outport(_Q2VC,      vc        );
	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,7604 */
			break;
		case 1:
			outport(_Q2SYSR, 0x2040);
/* Idle,Auto Renderring mode,No DMA,No Cache,7604 */
			break;
		default:
			outport(_Q2SYSR, 0x2080);
/* Idle,manual change moe,No DMA,No Cache,7604 */
	}
}

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;
	return( 0 );
}


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

			/* Manual Change */
			change_com_buffer();

			/* Transfer command from internal command buffer to UGM */
			inport(_DUMMY);

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

	return( 0 );
}


void change_com_buffer(void)
{
  unsigned short  ah,al;
  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;
        }
        ah = (CMD_StartAddress >> 16L) & 0xffff;
        al =  CMD_StartAddress & 0xffffL;
        outport(_Q2DLSAH,ah);         /* Change DLSAR */
        outport(_Q2DLSAL,al);
}
```

6.2.9 sample6.c のソースファイル

(1) sample6.c のメークバッチファイル

```
SHC /DEBUG /NOLISTFILE /OPT=1 /CPU=7600 sample6.c  >sample6.tag
LNK sample6 /OUTPUT=sample6 /LIB=c:\SHC\LIB\SHCLIB.LIB /FORM=A /START=P(6000000)
CNVS sample6
DEL sample6.abs
DEL sample6.obj
```

(2) sample6.c のソースリスト

```
/*
               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. 1997
*/
#include <machine.h>
#include <stdio.h>
#include <math.h>


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


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

#define BASE_ADDRESS    0x23000000L     /* Q2 internal register base address */
                                  /* Byte address (SuperH series)*/

#define UGMBASE   0x22000000L          /* UGM base address */
                                  /* Byte address (SuperH series)*/

#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 _Q2SSAH    0x01CL | BASE_ADDRESS   /* No.00E */
#define _Q2WSAH    0x01EL | BASE_ADDRESS   /* No.00F */
#define _Q2DMASH   0x020L | BASE_ADDRESS   /* No.010 */
#define _Q2DMASL   0x022L | BASE_ADDRESS   /* No.011 */
#define _Q2DMAW    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 _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  DISPLIST0  0x190000L
#define  DISPLIST1  0x198000L

#define _DUMMY   0x000000L | UGMBASE

short         DrawBuffer=0;
unsigned short *DISPLIST_ptr;
short         flag;

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,
            };

#define PI       3.14159
#define MAX_WORD  512
#define OFF      0
#define ON       1

/* ------------- Q2 Display List Functions ---------------- */
void polygon4a(short draw_mode, short txs,short tys,short tdx,short tdy, short poly[]);
void polygon4c(short draw_mode, short poly[], short color1);
void lcofs(short draw_mode,short xo,short yo);
void sclip(short draw_mode,short xmax,short ymax);
void trap(short draw_mode);

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

/* ----- 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 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 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 trap( draw_mode )\
{\
       *DISPLIST_ptr++ = 0xf800 | draw_mode;    /* TRAP */\
}


void main(void)
{
   short block_count = 100;
   short lp;
   short dx[MAX_WORD], dy[MAX_WORD];
       short array[MAX_WORD][8];
       short tx,ty;
       short txs, tys;
       short size_x, size_y;
       short DBmode;
/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
       char  first,quick;

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

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

```
/* Bitmap Location of UGM ----------------------------------------------------
        >BMPLOAD bolls.bmp 0 832
------------------------------------------------------------------------------ */
       txs  = 0;
       tys  = 320;
       size_x = 16; /* Must be load "bolls.bmp" to (txs,tys)=(0,832), before */
       size_y = 16;

       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, 319,239);
       lcofs(0x0000,0,0);

       while(1) {

/* MOVE */
                     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];

                            /* polygon4a */
                            polygon4a(0x0200, txs+(abs(dx[lp])-1)*size_x,tys, size_x,size_y,
array[lp]);
                     }

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

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

/* ploygon4c (clear) ---------------------------------------------------- */
       array[0] =   0;  array[1] =   0;
       array[2] =  319;  array[3] =   0;
       array[4] =  319;  array[5] =  239;
       array[6] =   0;  array[7] =  239;
       polygon4c(0x0000, array, 0x040f);
}

void ginit(short DBmode)
{
  unsigned short hsw =  32;
  unsigned short xs  =  65;
  unsigned short xw  = 320;
  unsigned short hc  = 455;

  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,  0x0024);
       outport(_Q2DSMR,  0x0105);
       outport(_Q2REMR,  0x0001);
       outport(_Q2IEMR,  0x0000);

       /* Set Memory Control Regisers    */
       outport(_Q2DSX,    xw-1);  /* Display size of x                */
       outport(_Q2DSY,    yw-1);  /* Display size of y                */
       outport(_Q2DSA0,  0x0000);  /* Frame buffer 0 area start address(H) */
       outport(_Q2DSA1,  0x0004);  /* 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(_Q2SSAH,  0x0008);  /* Color area sorce start address(H)    */
       outport(_Q2WSAH,  0x0018);  /* Work area start address(H)          */
       outport(_Q2DMASH, 0x0000);  /* DMA transfer start address(H)       */
       outport(_Q2DMASL, 0x0000);  /* DMA transfer start address(L)       */
       outport(_Q2DMAW,  0x0000);  /* DAM transfer word                */

       /* Display Contral Registers */
```

```
	outport(_Q2HDS,    hsw+xs-3   );
	outport(_Q2HDE,    hsw+xs-3+xw);
	outport(_Q2VDS,    ys         );
	outport(_Q2VDE,    ys+yw      );
	outport(_Q2HSW,    hsw-1      );
	outport(_Q2HC,     hc-1       );
	outport(_Q2VSP,    vc-vsw     );
	outport(_Q2VC,     vc         );
	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,7604 */
			break;
		case 1:
			outport(_Q2SYSR, 0x2040);
/* Idle,Auto Renderring mode,No DMA,No Cache,7604 */
			break;
		default:
			outport(_Q2SYSR, 0x2080);
/* Idle,manual change moe,No DMA,No Cache,7604 */
	}
}

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;
       return( 0 );
}


short draw_end(short DBmode,char *first,char quick)  /* Execute Display List */
{
  unsigned short st, ah, al;
  short count = 1;
  unsigned short d;

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

       switch(DBmode) {

              case 2:       /* Manual Change */
                     change_com_buffer();

                     /* Transfer command from internal command buffer to UGM */
                     inport(_DUMMY);

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

                     do{
                            outport( _Q2SRCR, 0x0800 ); /* Clear VBK bit */
                            while( d=inport(_Q2SR), (d&0x0800)==0);
                                                  /* VBK = 1 ? */
                     }while(count++<=0);

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

                     break;

              case 1:        /* Auto Renderring */
                     change_com_buffer();

                     outport(_Q2SRCR,0x0800);             /* Clear VBK bit */
                     while(st = inport(_Q2SR), (st & 0x0800) == 0);
                                                   /* Wait VBK */

                     /* Transfer command from internal command buffer to UGM */
                     inport(_DUMMY);
```

```
                    /* Renddering start */
                    outport(_Q2SYSR,0x2140);
                    break;

             default:    /* Auto Change */
                    change_com_buffer();

                    outport(_Q2SRCR,0x0800);            /* Clear VBK bit */
                    while(st = inport(_Q2SR), (st & 0x0800) == 0);
                                                  /* Wait VBK */

                    /* Transfer command from internal command buffer to UGM */
                    inport(_DUMMY);

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


void change_com_buffer(void)
{
  unsigned short  ah,al;
  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;
        }
        ah = (CMD_StartAddress >> 16L) & 0xffff;
        al =  CMD_StartAddress & 0ffffL;
        outport(_Q2DLSAH,ah);                /* Change DLSAR */
        outport(_Q2DLSAL,al);
}
```

6.2.10 sample7.c のソースファイル

(1) sample7.c のメークバッチファイル

```
SHC /DEBUG /NOLISTFILE /OPT=1 /CPU=7600 sample7.c  >sample7.tag
LNK sample7 /OUTPUT=sample7 /LIB=c:\SHC\LIB\SHCLIB.LIB /FORM=A /START=P(6000000)
CNVS sample7
DEL sample7.abs
DEL sample7.obj
```

(2) sample7.c のソースリスト

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

                    Copyright(c) Hitachi Ltd. 1997

                CLK0 = 33.00 MHz
                CLK1 = 14.32 MHz
                DCLK = 14.32 MHz

                Color depth  ; 16bit/pixel
                Display size : 640 pixel X 480 line
                Scan mode    : Interrace sync & video (SCM = 11)
                Blank check  : Use VBK bit.

                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>


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


/*====================    DEFINE I/O FUNCTIONS    ======================*/
/*----- Define I/O function -----*/
```

```
#define  outport(add,data) ( *VU_SHORT(add) ) = ( (unsigned short)(data) )
#define  inport(add)      ( *VU_SHORT(add) )

#define BASE_ADDRESS  0x23000000L      /* Q2 internal register base address */
                                /* Byte address (SuperH series)*/

#define UGMBASE      0x22000000L      /* UGM base address */
                                /* Byte address (SuperH series)*/

#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 _Q2SSAH    0x01CL | BASE_ADDRESS   /* No.00E */
#define _Q2WSAH    0x01EL | BASE_ADDRESS   /* No.00F */
#define _Q2DMASH   0x020L | BASE_ADDRESS   /* No.010 */
#define _Q2DMASL   0x022L | BASE_ADDRESS   /* No.011 */
#define _Q2DMAW    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 _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 DISPLIST0  0x0F0000L             /* (0,480) */
#define DISPLIST1  0x1F0000L             /* (0,992) */

#define _DUMMY    0x000000L | UGMBASE

short         DrawBuffer=0;
unsigned short *DISPLIST_ptr;
short         flag;



#define PI       3.14159
#define MAX_WORD  512
#define OFF      0
#define ON       1

/* ------------- Q2 Display List Functions ---------------- */
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 lcofs(short draw_mode,short xo,short yo);
void sclip(short draw_mode,short xmax,short ymax);
void trap(short draw_mode);

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

/* ----- Q2 function macro ------------------------------- */
#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 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 trap( draw_mode )\
{\
       *DISPLIST_ptr++ = 0xf800 | draw_mode;    /* TRAP */\
}


void main()
{
       short array[MAX_WORD];
       short ah,al;
       short d, i, h, j, k, del;
       short mn, kj;
       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;

       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 );

        while(1) {

/* 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+160;
                        array[i+1] =  ty+115;
                }

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

/* Move ---------------------------------------- */
        h=600;
        j=110;
        mn=160;
        kj=50;
        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+h;
                        array[i+1] =  ty+j;
                }

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

    }
```

```
        trapa(40);
}

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

/* 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)
{
  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,  0x0024);
        outport(_Q2DSMR,  0x0035);
        outport(_Q2REMR,  0x0041);
        outport(_Q2IEMR,  0x0000);

        /* Set Memory Control Regisers    */
        outport(_Q2DSX,   xw-1);    /* Display size of x                */
        outport(_Q2DSY,   yw-1);    /* 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(_Q2SSAH,  0x0008);  /* Color area sorce start address(H)    */
        outport(_Q2WSAH,  0x0018);  /* Work area start address(H)          */
        outport(_Q2DMASH, 0x0000);  /* DMA transfer start address(H)       */
        outport(_Q2DMASL, 0x0000);  /* DMA transfer start address(L)       */
        outport(_Q2DMAW,  0x0000);  /* DAM transfer word                 */
```

```
	/* Display Contral Registers */
	outport(_Q2HDS,    hsw+xs-3   );
	outport(_Q2HDE,    hsw+xs-3+xw);
	outport(_Q2VDS,    ys         );
	outport(_Q2VDE,    ys+yw      );
	outport(_Q2HSW,    hsw-1      );
	outport(_Q2HC,     hc-1       );
	outport(_Q2VSP,    vc-vsw     );
	outport(_Q2VC,     vc         );
	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,7604 */
			break;
		case 1:
			outport(_Q2SYSR, 0x2040);
/* Idle,Auto Renderring mode,No DMA,No Cache,7604 */
			break;
		default:
			outport(_Q2SYSR, 0x2080);
/* Idle,manual change moe,No DMA,No Cache,7604 */
	}
}

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;
	return( 0 );
}


short draw_end(short DBmode,char *first,char quick)  /* Execute Display List */
{
  unsigned short st, ah, al;
  short count = 1;
  unsigned short d;

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

	switch(DBmode) {

		case 2:		/* Manual Change */
			change_com_buffer();

			/* Transfer command from internal command buffer to UGM */
			inport(_DUMMY);

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

			do{
				outport( _Q2SRCR, 0x0800 );  /* Clear VBK bit */
				while( d=inport(_Q2SR), (d&0x0800)==0);
								/* VBK = 1 ? */
			}while(count++<=0);

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

			break;

		case 1:		/* Auto Renderring */
			change_com_buffer();

			outport(_Q2SRCR,0x0800);		/* Clear VBK bit */
			while(st = inport(_Q2SR), (st & 0x0800) == 0);
								/* Wait VBK */
```

```
                    /* Transfer command from internal command buffer to UGM */
                    inport(_DUMMY);

                    /* Renddering start */
                    outport(_Q2SYSR,0x2140);
                    break;

             default:    /* Auto Change */
                    change_com_buffer();

                    outport(_Q2SRCR,0x0800);            /* Clear VBK bit */
                    while(st = inport(_Q2SR), (st & 0x0800) == 0);
                                                /* Wait VBK */

                    /* Transfer command from internal command buffer to UGM */
                    inport(_DUMMY);

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


void change_com_buffer(void)
{
  unsigned short  ah,al;
  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;
	}
	ah = (CMD_StartAddress >> 16L) & 0xffff;
	al =  CMD_StartAddress & 0xffffL;
	outport(_Q2DLSAH,ah);                /* Change DLSAR */
	outport(_Q2DLSAL,al);
}
```

6.2.11 sample8.c のソースファイル

(1) sample8.c のメークバッチファイル

```
SHC /DEBUG /NOLISTFILE /OPT=1 /CPU=7600 sample8.c  >sample8.tag
LNK sample8 /OUTPUT=sample8 /LIB=c:\SHC\LIB\SHCLIB.LIB /FORM=A /START=P(6000000)
CNVS sample8
DEL sample8.abs
DEL sample8.obj
```

(2) sample8.c のソースリスト

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

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

                Copyright(c) Hitachi Ltd. 1997

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

#include <machine.h>
#include <stdio.h>
#include <stdlib.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 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 -----*/
#define  outport(add,data) ( *VU_SHORT(add) ) = ( (unsigned short)(data) )
#define  inport(add)      ( *VU_SHORT(add) )

#define BASE_ADDRESS   0x23000000L     /* Q2 internal register base address */
                               /* Byte address (SuperH series)*/

#define UGMBASE       0x22000000L     /* UGM base address */
                               /* Byte address (SuperH series)*/

#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 _Q2SSAH    0x01CL | BASE_ADDRESS    /* No.00E */
#define _Q2WSAH    0x01EL | BASE_ADDRESS    /* No.00F */
#define _Q2DMASH   0x020L | BASE_ADDRESS    /* No.010 */
#define _Q2DMASL   0x022L | BASE_ADDRESS    /* No.011 */
#define _Q2DMAW    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 _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  DISPLIST0  0x190000L
#define  DISPLIST1  0x198000L

#define _DUMMY     0x000000L | UGMBASE

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
};


short         DrawBuffer=0;
unsigned short *DISPLIST_ptr;
short         flag;


/* ------------- Q2 Display List Functions ---------------- */
void polygon4c(short draw_mode, short poly[], short color1);
void lcofs(short draw_mode,short xo,short yo);
void sclip(short draw_mode,short xmax,short ymax);
void trap(short draw_mode);

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

```
/* 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 clrscrn(void);
void set_poly(short array[12], short dz);
void set_polygon_sample(short array[N][12], short point[N][3],
                                short coordIndex[N][5], short _n_point);


/* ----- Q2 function macro ------------------------------- */
#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 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 trap( draw_mode )\
{\
       *DISPLIST_ptr++ = 0xf800 | draw_mode;    /* TRAP */\
}


void main(void)
```

```
{
  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;

       ginit(DBmode);    /* Initialize Q2 */
       init_start();     /* Initialize Transfer Procedure */
       draw_start();     /* Start Transfer Display List to UGM */
       sclip(0x0000, 319,239);
       lcofs(0x0000,0,0);

       while(1) {

       {
       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();
            }
       }

       } /* while */
       trapa(40);
}

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

/* ploygon4c (clear) ------------------------------------------------------ */
       array[0] =   0;  array[1] =   0;
       array[2] =  319;  array[3] =   0;
       array[4] =  319;  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)
{
  unsigned short hsw =  32;
  unsigned short xs  =  65;
  unsigned short xw  = 320;
  unsigned short hc  = 455;

  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, 0x0024);
	outport(_Q2DSMR, 0x0105);
	outport(_Q2REMR, 0x0001);
	outport(_Q2IEMR, 0x0000);

	/* Set Memory Control Regisers    */
	outport(_Q2DSX,    xw-1);  /* Display size of x                */
	outport(_Q2DSY,    yw-1);  /* Display size of y                */
	outport(_Q2DSA0,  0x0000);  /* Frame buffer 0 area start address(H) */
	outport(_Q2DSA1,  0x0004);  /* 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(_Q2SSAH,  0x0008);  /* Color area sorce start address(H)   */
	outport(_Q2WSAH,  0x0018);  /* Work area start address(H)         */
	outport(_Q2DMASH, 0x0000);  /* DMA transfer start address(H)      */
	outport(_Q2DMASL, 0x0000);  /* DMA transfer start address(L)      */
	outport(_Q2DMAW,  0x0000);  /* DAM transfer word                */

	/* Display Contral Registers */
	outport(_Q2HDS,    hsw+xs-3   );
	outport(_Q2HDE,    hsw+xs-3+xw);
	outport(_Q2VDS,    ys        );
	outport(_Q2VDE,    ys+yw     );
	outport(_Q2HSW,    hsw-1     );
	outport(_Q2HC,     hc-1      );
	outport(_Q2VSP,    vc-vsw    );
	outport(_Q2VC,     vc        );
	outport(_Q2DOR,    0x0000);
	outport(_Q2DOG_DOB, 0x007C);
	outport(_Q2CDR,    0x00FC);
	outport(_Q2CDG_CDB, 0xFCFC);

	switch(DBmode) {                /* Enable Draw/Display */
		case 0:
			outport(_Q2SYSR, 0x2000);
	/* Idle,Auto Change mode,No DMA,No Cache,7604 */
			break;
		case 1:
			outport(_Q2SYSR, 0x2040);
	/* Idle,Auto Renderring mode,No DMA,No Cache,7604 */
			break;
		default:
			outport(_Q2SYSR, 0x2080);
	/* Idle,manual change moe,No DMA,No Cache,7604 */
	}
}

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;
       return( 0 );
}


short draw_end(short DBmode,char *first,char quick)   /* Execute Display List */
{
  unsigned short st, ah, al;
  short count = 1;
  unsigned short d;

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

       switch(DBmode) {

              case 2:         /* Manual Change */
                     change_com_buffer();

                     /* Transfer command from internal command buffer to UGM */
                     inport(_DUMMY);

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

                     do{
                            outport( _Q2SRCR, 0x0800 );    /* Clear VBK bit */
                            while( d=inport(_Q2SR), (d&0x0800)==0);
                                                   /* VBK = 1 ? */
                     }while(count++<=0);

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

```
                }

                break;

        case 1:             /* Auto Renderring */
                change_com_buffer();

                outport(_Q2SRCR,0x0800);           /* Clear VBK bit */
                while(st = inport(_Q2SR), (st & 0x0800) == 0);
                                                /* Wait VBK */

                /* Transfer command from internal command buffer to UGM */
                inport(_DUMMY);

                /* Renddering start */
                outport(_Q2SYSR,0x2140);
                break;

        default:       /* Auto Change */
                change_com_buffer();

                outport(_Q2SRCR,0x0800);            /* Clear VBK bit */
                while(st = inport(_Q2SR), (st & 0x0800) == 0);
                                                 /* Wait VBK */

                /* Transfer command from internal command buffer to UGM */
                inport(_DUMMY);

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


void change_com_buffer(void)
{
  unsigned short  ah,al;
  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;
        }
        ah = (CMD_StartAddress >> 16L) & 0xffff;
        al =  CMD_StartAddress & 0xffffL;
        outport(_Q2DLSAH,ah);                    /* Change DLSAR */
        outport(_Q2DLSAL,al);
}


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;
            }
      }
```

6.2.12 sample9.c のソースファイル

(1) sample9.c のメークバッチファイル

```
SHC /DEBUG /NOLISTFILE /OPT=1 /CPU=7600 sample9.c  >sample9.tag
LNK sample9 /OUTPUT=sample9 /LIB=c:\SHC\LIB\SHCLIB.LIB /FORM=A /START=P(6000000)
CNVS sample9
DEL sample9.abs
DEL sample9.obj
```

(2) sample9.c のソースリスト

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

                ( Do not use Q2's double buffer control )
        ( Move 'Solid' Polygon with pattern which is in work area )

                Copyright(c) Hitachi Ltd. 1997
*/
#include <machine.h>
#include <stdio.h>
#include <math.h>


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


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

#define BASE_ADDRESS   0x23000000L    /* Q2 internal register base address */
                               /* Byte address (SuperH series)*/

#define UGMBASE     0x22000000L     /* UGM base address */
                               /* Byte address (SuperH series)*/

#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 _Q2SSAH    0x01CL | BASE_ADDRESS   /* No.00E */
#define _Q2WSAH    0x01EL | BASE_ADDRESS   /* No.00F */
#define _Q2DMASH   0x020L | BASE_ADDRESS   /* No.010 */
#define _Q2DMASL   0x022L | BASE_ADDRESS   /* No.011 */
#define _Q2DMAW    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 _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  DISPLIST0  0x190000L
#define  DISPLIST1  0x198000L

#define _DUMMY     0x000000L | UGMBASE

short         DrawBuffer=0;
unsigned short *DISPLIST_ptr;
short         flag;


#define PI       3.14159
#define MAX_WORD  512
#define OFF      0
#define ON       1
```

```
/* ------------- Q2 Display List Functions ---------------- */
void polygon4c(short draw_mode, short poly[], short color1);
void lcofs(short draw_mode,short xo,short yo);
void sclip(short draw_mode,short xmax,short ymax);
void trap(short draw_mode);

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

/* ----- Q2 function macro ------------------------------- */
#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 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 trap( draw_mode )\
{\
      *DISPLIST_ptr++ = 0xf800 | draw_mode;            /* TRAP */\
}
```

```
void main(void)
{
      short draw_buffer_area = 0x0000, display_buffer_area = 0x0004;
      unsigned long displying_area_address;
      unsigned long drawing_area_address;

      short array[MAX_WORD];
      short d, i, kj, st;
      double rad;
      short tx,ty;
      short temp;
      short DBmode;
/*
 DBmode: 0 ... Auto Change
 DBmode: 1 ... Auto Renderring
 DBmode: 2 ... Manual Change
*/
      char  first,quick;

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

      ginit(DBmode);       /* Initialize Q2 */

      /* Deside displaying area and drawing area */
{
  short st;
  short temp;

      displying_area_address = _Q2DSA0;
      drawing_area_address   = _Q2DSA1;

      st = inport( _Q2SR );
      if( (st & 0x0100) != 0 ) {
            displying_area_address = _Q2DSA1;
            drawing_area_address   = _Q2DSA0;
            temp                = draw_buffer_area;
            draw_buffer_area      = display_buffer_area;
            display_buffer_area   = temp;
      }
}


/* Clear work area */
{
  unsigned short x,y;
  unsigned short *wk  = VU_SHORT (0x180000L | UGMBASE);
      for(y=1;y<=240;y++){
```

```
            for(x=1;x<=512/16;x++){
                  *wk = 0x0000;
                  ++wk;
            }
      }
}


/* Transfer Font Data "äø" at work area ---------------------------------- */
{
  unsigned short work_area_width = 512 / 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(_Q2WSAH), 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, 319,239);
      lcofs(0x0000,0,0);

      while(1) {

      /* MOVE */
            kj = 50;
            for(d=0;d<=360;d+=5) {

            /* Wait VSYNC */
                  outport( _Q2SRCR, 0x0800 );    /* Clear VBK bit */
```

```
      while( st=inport(_Q2SR), (st&0x0800)==0);  /* VBK = 1 ? */

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


/* Set drawing start address for job No.1 */
      outport( drawing_area_address,   display_buffer_area );

/* Job No.1 : clear screen */
      array[0] =    0;  array[1] =    0;
      array[2] =  319;  array[3] =    0;
      array[4] =  319;  array[5] =  239;
      array[6] =    0;  array[7] =  239;
      polygon4c(0x0000, array, 0x0606);

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


/* Set drawing start address for job No.2 */
      outport( drawing_area_address,   display_buffer_area );

/* Job No.2 : draw a solid ploygon */
      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+kj;
            array[i+1] =  ty+100;
      }

polygon4c(0x0001,array,0xF800);      /* POLYGON4C */
      kj+=5;

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


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

```
            } /* for */

      } /* while */

      trapa(40);
}

void ginit(short DBmode)
{
  unsigned short hsw =  32;
  unsigned short xs  =  65;
  unsigned short xw  = 320;
  unsigned short hc  = 455;

  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, 0x0024);
      outport(_Q2DSMR, 0x0105);
      outport(_Q2REMR, 0x0001);
      outport(_Q2IEMR, 0x0000);

      /* Set Memory Control Regisers    */
      outport(_Q2DSX,   xw-1);   /* Display size of x                */
      outport(_Q2DSY,   yw-1);   /* Display size of y                */
      outport(_Q2DSA0,  0x0000); /* Frame buffer 0 area start address(H) */
      outport(_Q2DSA1,  0x0004); /* 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(_Q2SSAH,  0x0008); /* Color area sorce start address(H)    */
      outport(_Q2WSAH,  0x0018); /* Work area start address(H)          */
      outport(_Q2DMASH, 0x0000); /* DMA transfer start address(H)       */
      outport(_Q2DMASL, 0x0000); /* DMA transfer start address(L)       */
      outport(_Q2DMAW,  0x0000); /* DAM transfer word                 */

      /* Display Contral Registers */
      outport(_Q2HDS,    hsw+xs-3   );
      outport(_Q2HDE,    hsw+xs-3+xw);
      outport(_Q2VDS,    ys        );
      outport(_Q2VDE,    ys+yw     );
      outport(_Q2HSW,    hsw-1     );
```

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

      switch(DBmode) {            /* Enable Draw/Display */
            case 0:
                  outport(_Q2SYSR, 0x2000);
/* Idle,Auto Change mode,No DMA,No Cache,7604 */
                  break;
            case 1:
                  outport(_Q2SYSR, 0x2040);
/* Idle,Auto Renderring mode,No DMA,No Cache,7604 */
                  break;
            default:
                  outport(_Q2SYSR, 0x2080);
/* Idle,manual change moe,No DMA,No Cache,7604 */
      }
}

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;
      return( 0 );
}


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

                  /* Manual Change */
```

```
                change_com_buffer();

                /* Transfer command from internal command buffer to UGM */
                inport(_DUMMY);

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

      return( 0 );
}


void change_com_buffer(void)
{
  unsigned short  ah,al;
  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;
      }
      ah = (CMD_StartAddress >> 16L) & 0xffff;
      al =  CMD_StartAddress & 0xffffL;
      outport(_Q2DLSAH,ah);                  /* Change DLSAR */
      outport(_Q2DLSAL,al);
}
```

7. 描画性能

以下に HD64411 の描画性能をグラフで示します。グラフは、描画範囲内を POLYGON4C コマンドで塗りつぶした時の時間を示しています。

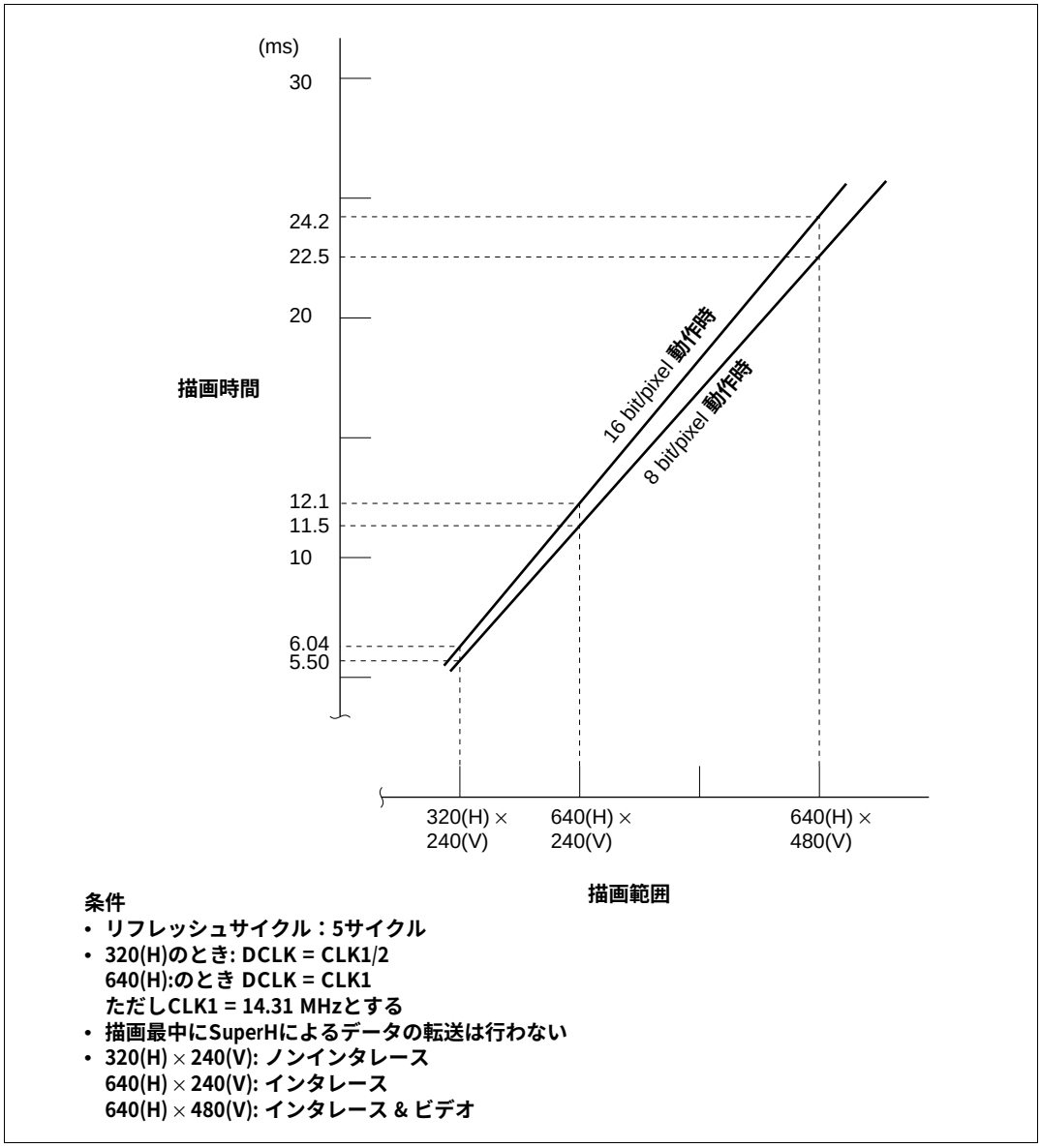


図 7.1　POLYGON4C **の描画時間**（CLK0＝33MHz）

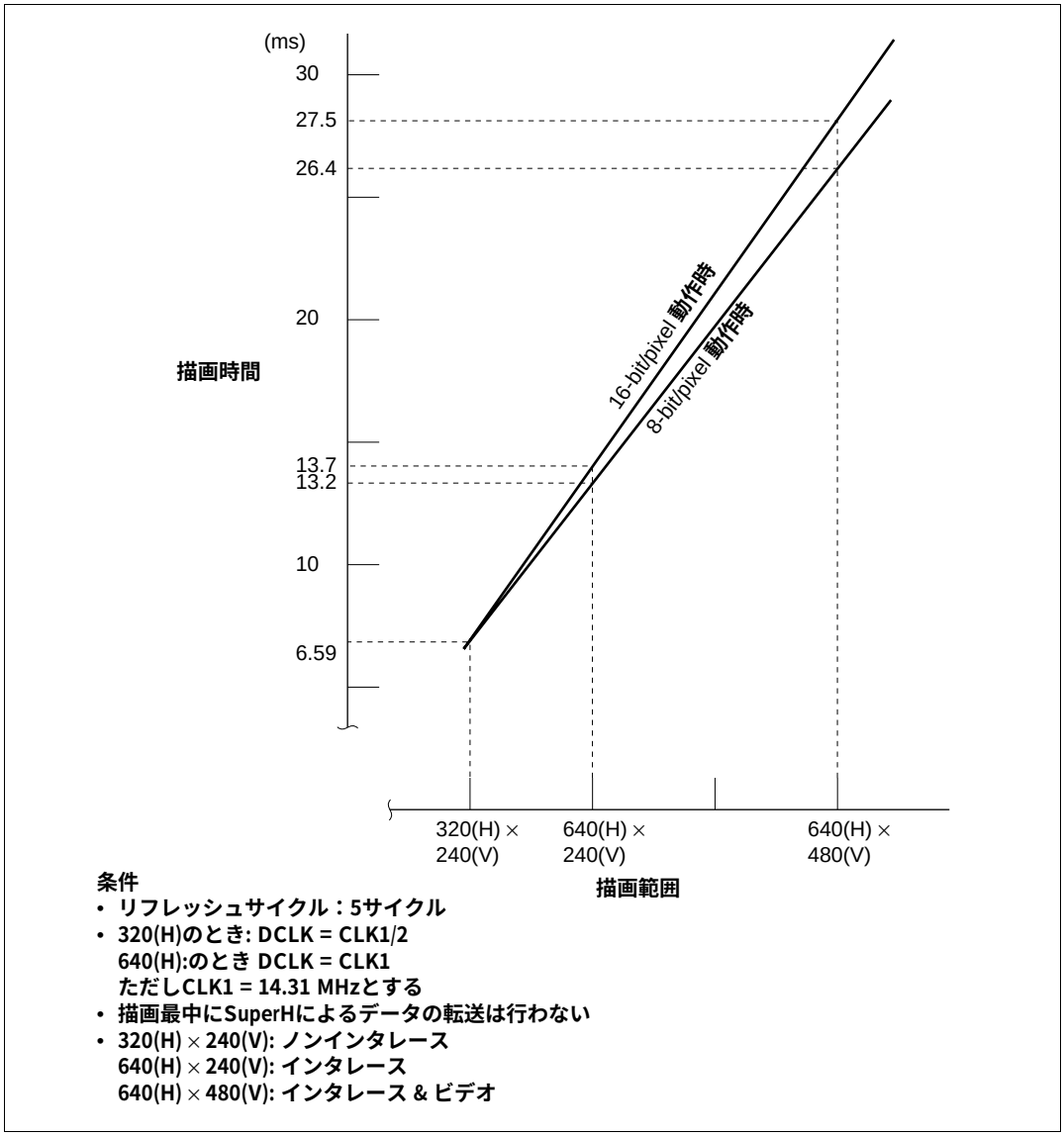


図 7.2　POLYGON4C **の描画時間**（CLK0＝28.7MHz）

8. 付録

付録として SH-2 と HD64411 を使用した時の回路例を掲載します。なお回路例は、パーソナルコンピュータに挿入して使用することを前提としています。この回路例ではパーソナルコンピュータにて、SH-2 のメインメモリにプログラムを転送し、転送終了後、SH-2 を起動して HD64411 を制御させるようにしています。以下に主な仕様を挙げます。

■回路例の主な仕様

番号	機　能		仕　様
CPU 部仕様			
1	ホストシステム		IBM-PC/AT 互換機、ISA バスを備えるもの。
2	制御部		ボード上の SuperH から、Q2 を制御する。
		CPU	SH-2（HD6417604SF28）、Q2 を SH-2 の I/O として制御
		周波数	28.7MHz
		主記憶メモリ	HM5116160ATT-6×2（4MB）
3	DMA		Q2 による DMA 転送をサポート
		レベル	DREQ0、DREQ1 を選択可能
		モード	サイクルスチル（バーストは不可）
		用途	Q2 に対するライト DMA 転送に用いる。ただし、データ幅 16bit 固定。

番号	機　　能		仕　　様
以下アクセラレータ（Q2）部仕様			
4	CLK0（描画）入力		OSC1：33 MHz（max）、もしくは OSC2：14.31818 MHz、SH-2 の CKIO を選択可能。
5	CLK0→内部 CLK 逓倍機能		Q2 により 1、2、4 逓倍が可能
6	CLK1（表示）入力		OSC2：14.31818 MHz、dotclk＝7.15909 MHz（2fSC NTSC）、ただし DOT ビット＝1 の時
7	UGM メモリ		16Mb（×16 構成）EDO-DRAM：HM5118165-6 相当×1
8	UGM 容量		2MB
9	表示		Q2 の DD0～DD17 出力（18bit）から、アナログ RGB、NTSC-Video を生成する。
		DAC	HD153510CP（8bit×3、パレットなし）RGB 各 8bit 入力のうち、上位各 6 bit を使用。
		NTSC エンコーダ	Sony CXA1645M
10	表示色		
		16bit/pixel	同時 65536 色（R：5、G：6、B：5）固定
		8bit/pixel	262144 色中同時 256 色（Q2 内部のパレットを使用）
11	対象ディスプレイ		NTSC ビデオ出力 TV/アナログ RGB 出力 TV モニタ
		解像度	標準 320H×240V
		水平同期	HSYNC＝15.7KHz（63.5uS）
		垂直同期	VSYNC＝60Hz（16.7mS＝263H）
		ドットクロック	DCLK＝7.16MHz、ただし、DOT ビット＝1 の時
		インタフェース	アナログ RGB（セパレート SYNC）
			NTSC（コンポジット・ビデオ）

HD64411 Q2 Quick 2D Graphics Renderer
Q2 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-502-064