16ビット乗除算／レジスタ間演算／ブロック転送が可能

超8ビット級MPU 63C09の拡張機能をさぐる

63C09解析委員会UNO

6809のマイナーチェンジ版に，63C09というLSIがあります。ハードに強い一部のユーザの間では，その高速性を買われ，本体の改造に使われたりしてきました。ところが，最近この63C09に各種の拡張機能が隠されていたことがわかりました。ここでは，それらの機能を発見し，探索してきた「63C09解析委員会」の方にその概要を報告していただきます。

なお，本体の改造，ことにCPUの差し換えは，メーカーの修理は保証されず，ほかの周辺LSI，周辺機器も交換しなければならない場合もありえ，おまけに63C09だと従来のソフトの一部（あるいは多数）が動作しなくなる危険性がありますので，「私は自作したプログラムしか使わない！」という，よほど腕に自信のある方以外にはお勧めしかねます。

6809の高速版 63C09

パソコンの楽しみ方にはいろいろありますが，その筋の兵（つわもの）だけに許された遊びとして，ハードの改造があります。その昔からいろいろな改造が行われてきましたが，中でもCPUの高速化は，処理能力の向上が著しいことと，比較的簡単に行えることから，市販ソフトに頼着しないプログラム自作派の間では広く試みられてきました。古くはFM-8に積まれた68A09（1.2MHz）を68B09(2MHz)に変えて，FM-7並みの処理速度を与えたこと，最近ではFM-11を中心にCPUをクロックアップして2.5～4MHzで動かすことが，その代表的なところです。

「えっ，2.5～4MHzだって，そんなに速い6809ってあったっけ」と思われる方もおられるでしょうが，実は存在するのです。秋葉原や日本橋のチップ屋さんで売っている日立の63C09というMPUがそれで，3MHzで動作する，6809のC-MOS版です。ノーマルに使うなら従来の68B09（2MHz）の1.5倍の処理能力になり，選別して規格外の4MHzで動くものを使えば2倍の処理能力となります。

この63C09を使って高速化を行うわけで，基本的にはCPUとクロックを差し換えるだけの簡単なものですが，それだけでは済まないこともあります。FM-8，7，77D1/D2/L2/L4，77AV，11のようにCPUがソケットに差さっている機種では単純に差し換えるだけですが，FM-NEW7，77AV20/40/20EX/40EXのように基板に直接ハンダづけされている機種ではかなりの腕がないとCPUが引っこ抜けません。また，クロックアップした場合周辺LSIや周辺機器が追い付けないこともあり，その場合はそれらも交換しなければなりませんが，AV系のようにカスタムLSIが多用されている場合は難しいでしょう。7/AV系のサブシステムのように微妙なタイミングで動いているものでは，クロックアップはかなり困難で，しかも，クロックアップしたが最後，プロテクトやその他の処理に内蔵タイマやソフトウェア的なタイミングを使った市販アプリケーションソフトの多くは全く使いものにならなくなってしまいます。

さて，これら様々な困難を伴った高速化ですが，そのもたらす結果は苦労を補って余りあるものです。

ちなみにFM-11の場合を例にとると，CPUの差し換えだけだと2.5～3MHzぐらいが限界のようで，それ以上を望むと一部周辺LSIの差し換え等が必要なようです。11ではサブシステムの高速化も可能で，手を加えれば4MHzまでいけます。とくに，4MHz化されたFM-11のサブシステムの表示速度は目を見張るものがあり，漢字の表示速度が漢字VRAMをもったFM16βと大差ない速さになります。

拡張機能の発見

というわけで，私の周りの歴戦の勇士たちは次々とFM-11に高速化改造を行ったのですが，そこに1つ，奇妙な問題が発生しました。

コマスのワープロWPV3が動かなくなってしまったのです。最初は，前述したソフト的なタイミングの問題か何かだと思われたのですが，驚いたことに，クロックを2MHzに落としても動きません。

そこで，友人のGigo氏を中心に原因探究が始まったのですが，ほどなく6809の未定義命令で引っかかっていることがわかりました。未定義命令とは，メーカが発行しているマニュアルで定義されていない命令のことで，建て前上はそのような命令を使ってもなんの動作もしないことになっているのですが，実は隠し命令になっていることもままあります。一昔前のパソコン雑誌には，よく各社製CPUの隠し命令の解析記事が載っていたりしました。このよう

警告　CPUを63C09に交換した場合，かなりの市販アプリケーション（とくにゲーム）が動作しなくなります。

な隠し命令は，6809にもそう大したものではありませんでしたがありました。さて，未定義命令の扱いが63C09と6809とでは異なるということは，隠し命令も異なる可能性があるわけです。63C09の出荷開始時期(1985年秋)を考えても，何か機能が追加されても当然なくらいで，「もしかしたら」の期待がわきました。

引っかかっているコードの1つに「$1F，$62」というものがありました。命令自体はTFR（レジスタ間のデータ転送命令）でおなじみのものですが，未定義レジスタ番号からYレジスタへ転送するように指示されています。6809の場合は未定義なのでYレジスタに$FFFFが返りますが，63C09の場合はYレジスタにはめちゃくちゃな値が入ります。試みに，Yレジスタから未定義レジスタ番号にデータ転送してから，未定義レジスタ番号からYレジスタへ戻してやると，元のデータがちゃんと残っていました……。つまり，63C09の未定義レジスタ番号は，番号が余って未定義となっていたのではなく，実在するレジスタを指す番号だったのです！

この隠しレジスタを発見したGigo氏は，狂喜してかたっぱしから友人に電話をかけまくりました。そして，私のところにも夜も丑三つ時過ぎにかかってきました……。話の内容は，「63C09にはレジスタが余計にある。レジスタがあるからには命令もあるはずだ。みんなで手分けして調べよう」というもので，それから隠し命令を探す日日が始まり，ディスアセンブル表の割り当てのないコードをデバッガでメモリ上に書き，ブレークポイントを設定してTFRでレジスタに値をセットして実行させ，レジスタの内容を見るという単調な作業が繰り返されました。その日わかった結果を，パソコン通信を通じて情報交換するうちに，いつとはなしに，FM-11でOS-9をやっている，それもほとんど病気に近いマニアが集まり，「63C09解析委員会」なる集団が自然発生しました。

マニアの執念は恐ろしいもので，ほどなく63C09の拡張機能の大筋が判明しました。概略を述べると，

・3種類のレジスタが増設されており，そのうちの1つはアキュムレータとして，またインデックスレジスタとして使える
・32÷16ビット除算，16÷8ビット除算，16×16ビット乗算，レジスタ間演算，ビット操作，ブロック転送等の命令が拡張されている。
・未定義の命令を検出した場合トラップがかかる
・6809コンパチのモードと，63C09本来のモードの2種類の動作モードをもつ

といったところで，今までの6809で不便であった部分，弱点であった部分が相当改善されており，またとても8ビットMPUとは思えない強力な機能も含まれています。

これらの解析結果はNANNO-NETを皮切りに，いくつかのBBSにアップされました。多くのネットワーカーの方から多大な反響を得ましたが，より多くの方に「8ビットを超える8ビットMPU」63C09の全貌を知っていただくために，Oh！FMの誌上をお借りしてご報告します。

63C09化のメリット／デメリット

日立から販売されているHD63C09は，モトローラのMC6809とピンコンパチの8ビットMPUです。MPUの仕様は6809に拡張機能を付け加えた形のもので，6809の上位コンパチになっています（未定義命令を除く）。日立から公式発表はされていませんが，63C09の拡張機能を活用すると，6809パソコンの処理能力を大幅に向上させることができます。

6809パソコンのMPUを63C09に差し換えた場合のメリットは処理能力の向上につきます。その要因としては以下の3点が考えられます。

1　高速クロック
2　拡張命令／拡張レジスタ
3　ネイティブモード

1は当然のことで，MPUの動作クロックを上げればソフトの実行速度は上がります。未定義命令トラップやソフトウェアタイマの関係で引っかかる一部を除けば，従来のソフトが高速に動かせます。動作クロックの上昇率はハードにより異なり，場合によってはほとんど上げられないこともあります。

2は，新規にソフトを書き起こすか，従来のソフトにパッチを当てたときに効果があります。従来の6809だとアセンブラのマクロ機能で表現していた処理の相当が，63C09の1命令で書けるようになり，マシンサイクルを短縮できます。また，拡張レジスタを活用すると，スタックへの退避回数を減らすことができるなど，かなり柔軟にプログラムを組むことができるようになります。

3は，63C09固有のモードに切り換えて使うと，通常のエミュレーションモードよりも命令の実行サイクルが短くなることにより生じるものです。このモードを使うと，同じ動作クロックでも，通常より実行速度が最大20～35%上がります（アドレッシングモードにより効果が違う）。ただ，スタックや割り込み関係で動作が異なる点があるので，ネイティブモードを利用するにはF-BASICなりOS-9なりのシステムの一部を書き換える必要があります。

逆にデメリットとしては，まず，6809の未定義命令を使ったソフトに限らず，多くの内蔵タイマを使った市販ソフトその他が動作不良を起こしてしまうであろうこと，また，63C09の拡張機能を活用するにはそれらを活用するための開発ツールを自作できるくらいのそれなりの腕が必要で，ノービスには難しいことが難点といえるでしょう。

メリットとデメリットを比較すると，現状では自分でプログラムを書くだけの人ならその恩恵を受けることができるが，ごく一般のゲームユーザやアプリユーザは決して手を出さないほうがよい，といったところでしょう。

それでは，63C09の拡張機能について以下順番に解説していきます。

拡張レジスタ

63C09では6809よりレジスタの数が3つ増えています（図1）。そのうち2つは16ビットのレジスタで，もう1つは8ビットのモードステータスレジスタです。

Wレジスタ[16ビット]

アキュムレータとしても，インデックスレジスタとしても使用できる16ビットレジスタです。

アキュムレータとして使うときは，16ビ

ットレジスタとしてのほか，2つの8ビットレジスタ（E/Fレジスタ）に分割して使うこともできます。ちょうど，既存のD/A/Bレジスタがもう1組増えたようなものです。ただし，AND/OR等のW/E/Fレジスタでは使えない命令もあります。

また，既存のDレジスタと連結して32ビットレジスタ（Qレジスタ）として使うことができ，乗除算のときに利用します。

インデックスレジスタとして使うときは，既存のX/Yレジスタと同様に利用します。この場合，6809でポストバイトに用いられていないビットパターンを使用します。Wレジスタをインデックスレジスタとして使用したときの，アキュムレータオフセットと5ビットオフセット，8ビットのコンスタントオフセットはありません。

また，特殊な使い方として，ブロック転送でのカウンタレジスタとして使う方法があります。

Vレジスタ[16ビット]

Vレジスタを使う命令は，レジスタ間演算命令やTFRなどに限られています。Vレジスタの特徴は，MPUをリセットしてもレジスタの値が変化しないことです。このレジスタをOSなどで定数等を保持するような目的に使うと便利でしょう。

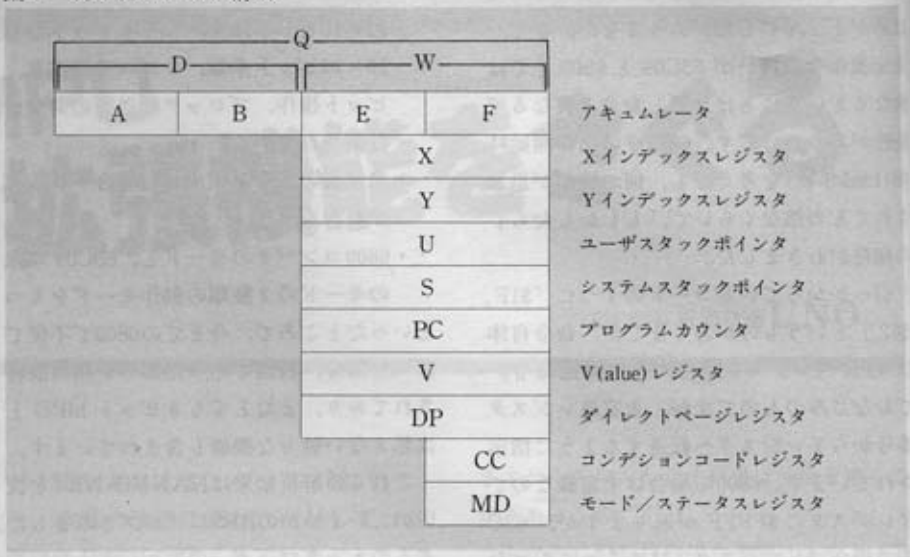
MDレジスタ[8ビット]

モード/ステータスビットレジスタの略で，除算実行時のエラー検出や未定義命令トラップの作動チェック，動作モードの設定など，63C09になって増えたモードやステータスの表示に用いられます。各ビットの意味は次のとおりです。

- ビット7　R　除算で0で割ったときに1がセットされる
- ビット6　R　未定義命令をフェッチしたときに1がセットされる
- ビット1　W　FIRQ時のレジスタの退避モード設定ビット
  0→FIRQ時，PCとCCのみスタックに退避
  1→FIRQ時，すべてのレジスタを退避
- ビット0　W　動作モード設定ビット
  0→エミュレートモード
  1→ネイティブモード

なお，リセット時にはすべてのビットは0になります。

図1　63C09レジスタ構成



動作モード

63C09には2つの動作モードがあります。1つは6809とのコンパチビリティを考えたエミュレートモードで，もう1つは63C09の本来の機能を引き出すネイティブモードです。

と，いうと，「拡張レジスタや拡張命令が使えるのがネイティブモードで，使えないのがエミュレートモードだな」と思われる方もいるでしょうが，それはハズレです。63C09では，拡張レジスタと拡張命令をどちらのモードでも使えます。

エミュレートモードとネイティブモードの違いは，インタラプト時のスタックの扱いの違いです。インタラプトがかかったときレジスタの内容はスタックに退避されますが，そのとき従来からのレジスタだけを退避させるのがエミュレートモードで，拡張レジスタのWレジスタも退避させるのがネイティブモードです。

63C09をリセットした直後はエミュレートモードに設定されています。このモードでは6809のソフトが問題なく動作する代わり，マルチタスクで拡張レジスタを使うときに気を付けなければなりません。たとえば，拡張レジスタのWレジスタをカウンタに使うブロック転送命令TFMを使ったケースを考えます。タスクAでTFM命令を使用中に，インタラプトをかけて，タスクBに移ったとしましょう。そのときにタスクBでWレジスタを使用したとしたら，また元のタスクAに戻ったときにWレジスタの中身が変更されていますので，誤動作を起こします（もちろんシングルタスクで使用するときや，マルチタスクでも1つのタスクでしか拡張レジスタを使用しないときは問題ありません）。よって，エミュレートモードでは，拡張レジスタを使用するときはいちいちインタラプトを禁止して，さらにこのレジスタを一度スタックにセーブしてからでないと，別タスクに切り換えてはなりません。

これでは，高速ゲームやOS-9から拡張レジスタを使いづらいうえ，使いにくい拡張命令も発生します。そのために，拡張レジスタと拡張命令を使うことを前提にしたモード，ネイティブモードが存在します。このモードでのインタラプトはPC, U, Y, X, DP, W, D, CCの順にスタックにレジスタを退避して割り込み処理に入ります。ここで注意してほしいのは，WはDPとDの間にあるということです。これは，DとWの32ビットレジスタペアQとしてスタックに退避するという意味です。

ネイティブモードの特徴としては，もう1つ，命令のマシンサイクル短縮があげられます。その結果，アドレッシングモードによって20～35%高速に動作します。

とくにダイレクト，エクステンド，インヘラントで顕著にその効果が現れます。

なお，ネイティブモードでもVレジスタとMDレジスタはその性格上退避されませんので注意してください。

エミュレートモードからネイティブモードに移行するには，新設されたMDレジスタのビット0（LSB）に1を書き込むことによって実現します。

さて，63C09のモードには上の2つのほかに，FIRQのスタック退避モードが用意されています。ご存じのように，6809ではFIRQをフェッチすると，PCとCCのみをスタックに退避してインタラプト処理ルーチンへ分岐します。しかし，制御用のボードマイコンの場合，FIRQよりIRQがもう1つあったほうが便利なケースがあります。しかし，63C09は6809とピンコンパチを謳っていますので，足の配置を変えるわけにはいきません。そこで，FIRQをIRQとして使用できるように，スタックの退避をすべてのレジスタが行うようにモードをソフトで切り換えられるようになっています。FIRQをIRQとして使用する場合はMDレジスタのビット1に1を書くことによって実現します。

トラップ

63C09は以下の現象が発生したときにトラップがかかります。

1　未定義命令がフェッチされたとき
2　除算命令のDIV命令で0で割ったとき

トラップがかかると，エミュレートモードではPC，U，Y，X，DP，B，A，CCの順に，ネイティブモードではPC，U，Y，X，DP，W，B，A，CCの順にSレジスタにレジスタをプッシュした後，$FFF0のアドレスに書いてあるベクタに分岐します（$FFF0は6809ではRESERVE）。このトラップはリセットの次の割り込み優先度があります。なお，未定義命令かゼロディバイドかを判定する命令としてBITMD命令があります。

このトラップのため，未定義命令を使っている6809のソフトが動作しなくなりますが，代わりにOS-9/68000等で使われているトラップライブラリを組めるようになります。たとえば，未定義命令に浮動小数点演算プロセッサの呼び出しを割り当てておくと，その命令を未定義命令トラップに引っかけ，処理ルーチンに飛ばすことが可能になります。このトラップライブラリを利用すると，オブジェクトのサイズをかなり縮められるので便利でしょう。

拡張命令

63C09の拡張命令には，既存の命令の対応レジスタを増やした追加命令と，新規に設けられた新設命令に分けられます。

新設命令としては，レジスタ間演算命令や，ブロック転送命令，乗算／除算命令，ビット操作命令，ビット演算／転送命令等の命令があります。

追加命令

63C09では，既存の命令も拡張されていて，対応するレジスタが増えています。

たとえば，今までありそうでなかったTSTD，ADCDなどが追加されています。これらは従来でもアセンブラ上でマクロを使って表現できましたが，これらを使うことによりマシンサイクルを短縮できます。

また，ADDやSUBなどの命令では，E/F/Wレジスタが増えたことにより，それに対応する命令が増えています。いわば，A/B/Dレジスタがもう1組増えたようなもので，プログラミングの柔軟性が増します。ただし，A/B/Dレジスタで使える命令がすべて対応しているわけではありませんので注意してください（図2）。

既存の命令の中で追加の度合いが大きいのはTFRとEXG命令でしょう。TFR，EXG命令では，対象レジスタの指定にポストバイトのビットパターンを用いています。63C09ではレジスタが増えていますので，そのビットパターンの組み合わせも増えていて（0110→W，0111→V，1110→E，1111→F），レジスタアドレッシングとでもいったらよい状態になっています。このレジスタアドレッシングは新設命令のレジスタ間演算でも使用しています。ここで注意しなければいけないのは，本当の未定義レジスタ番号を指定した場合，63C09と6809とでは動作が異なるということです。

図2　アキュムレータで行える処理

	A	B	E	F	D	W	Q
CLR	○	○	○	○	○	○	
INC	○	○	○	○	○	○	
DEC	○	○	○	○	○	○	
TST	○	○	○	○	○	○	
COM	○	○	○	○	○	○	
NEG	○	○			○		
SEX	○*	○*			○*	○*	
ASL/LSL	○	○			○		
ASR	○	○			○		
LSR	○	○			○	○	
ROL	○	○			○	○	
LD	○	○	○	○	○	○	○
ST	○	○	○	○	○	○	○
ADD	○	○	○	○	○	○	
SUB	○	○	○	○	○	○	
CMP	○	○	○	○	○	○	
ADC	○	○			○		
SBC	○	○			○		
AND	○	○			○		
OR	○	○			○		
EOR	○	○			○		
BIT	○	○			○		
MUL	○*	○*			○		
DIV					○		○

＊　ワークとして使用

レジスタ間演算命令

6809での演算は，ほとんどレジスタ対メモリないしイミディエイト値で行われていました。そのためAレジスタとBレジスタの値のANDをとりたい場合は，どちらかのレジスタをメモリ上にストアしてから演算（この場合はAND）を行わなければなりませんでした。63C09ではこれが解決されていてレジスタ同士の演算が可能になりました。これらはTFRやEXGと同じレジスタアドレッシングを用います。

レジスタ間演算命令には以下のようなものがあります。

ADDR，ADCR，SUBR，SBCR，
ANDR，ORR，EORR，CMPR

ブロック転送命令

6809でメモリ上のデータを移動させるときは，一度そのデータをレジスタにロードしてきては，それをセーブするということを繰り返して行っていました。これはこれでよいのですが，問題はその処理にかかる時間です。そこでZ80や8086などにもあるブロック転送命令が，63C09にも設けられています。

ブロック転送命令では，転送元アドレス（ソース），転送先アドレス（ディスティネーション）の指定に16ビットレジスタのD/X/Y/U/Sレジスタの中から1～2使います。レジスタの指定にはポストバイトを使い，その形式はレジスタアドレッシングの形式をとります。また，ソースとディスティネーションを同じレジスタでも指定できます。転送するバイト数のカウントにはWレジスタを使います。

転送方法には4種類あり，正方向（TFMr0+，r1+）/逆方向（TFMr0−，r1−）の通常のブロック転送のほか，I/Oポート等のアドレスにデータを次々と流し込むもの（TFMr0+，r1），指定ブロックを指定値で塗りつぶすもの（TFMr0，r1+）があります。

乗算／除算命令

6809にはMULという8×8ビットの乗算命令がありましたが，これはAレジスタとBレジスタの値を掛け合わせるだけのものでした。63C09で設けられた16×16ビット乗算命令（MULD）では，いろいろなアドレッシングモードが使え，追加というよりは新設に近いものです。

また，63C09にはそれに加えて16÷8ビット除算（DIVD），32÷16ビット除算（DIVQ）が設けられて，これらも，いろいろなアドレッシングモードが使えるようになっています。

ビット操作命令（6301コンパチ命令）

日立のHD6301には，6801の拡張命令としてビット操作命令が新設されていましたが，同じ63シリーズの63C09にも同じ命令があります。これらの命令はイミディエイトデータとメモリの内容を論理演算して，結果をメモリに戻したり，関連コンディションコードを変化させてしまうので，ビットパターンを操作するときなどに重宝します。

行える論理演算には，論理積（AIM），論理和（OIM），排他的論理和（EIM），論理積コンディションコード（TIM）があります。オブジェクトの構成は，

〈命令コード〉，〈ビットの位置〉，〈オペランド〉

の順になっています。

これらの命令を使うと，6301の命令はマクロアセンブラを用いれば63C09上で実行可能になります。つまりOASYS Lite等の組み込みプログラムをFM上で動かすことができるかもしれないわけで，そういう意味でもおいしい命令なのです（もっとも，その前に根性でROMを逆アセンブルしなければなりませんが）。

ビット演算/転送命令

63C09には，多分にI/Oを意識したビット演算／転送命令が存在しています。これらの命令は，アドレッシングモードにダイレクトモードしかサポートしていない難はありますが，使い慣れれば便利に使えるでしょう。動作は，ダイレクトページのLABELのビットnとREGレジスタのビットmを論理演算して，REGレジスタに入れるものがほとんどです。ビット演算／転送命令には以下のようなものがあります。

BAND，BOR，BEOR，BIAND，BIOR，BIEOR，LDBT，STBT

オブジェクトの構成は

〈命令コード（$11，$xx）〉，〈ポストバイト〉，〈オペランド〉

と，かなり変則な構成をとります。オペランドはダイレクトアドレッシングのみです。また，ポストバイトは特殊な形式をとります。

その他の命令

その他の命令としては，まずモード切り換え命令があります。といっても，エミュレートモードからネイティブモードへの移行は，MDレジスタのビット0（LSB）に1を書き込むことによって行われますので，MDレジスタに対する普通のLD命令を使います。

次にトラップがかかったとき，未定義命令でかかったのか除算のエラーで起こったのかを調べる命令BITMDがあります。これは，MDレジスタのステータスビット（ビット7 or 6）を調べ，どちらでトラップがかかったのかを知らせます。ただし，この命令を実行するとMDレジスタのステータスビット（ビット7 and 6）はクリアされますので，未定義コードトラップか，Divide by Zeroトラップかは，一度きりしか調べることはできません。

そして，スタックに関するものがあります。63C09では，レジスタが増設されていますが，現在のPSHS / PSHUではそれらをスタックにセーブすることはできません。なぜなら，PSHS / PSHUおよびPULS / PULUのポストバイトが，すでにすべて割り当てられていて追加の余地がないということです。そこで，63C09では増設されたレジスタへのスタック操作は別命令のPSHSW，PULSW，PSHUW，PULUWを使います。ただし，これはWレジスタに対するもののみしかありません。よって，これらはポストバイトをもたないインヘレントアドレッシングのみです。

おわりに

以上63C09に隠されていた機能の大筋を説明してきました。6809に＋αで追加してほしかった機能がほぼ盛り込まれており，6809派にはひさびさの好ニュースといえます。ただ，惜しむらくは登場時期が遅かったことで，そのため活躍の場がパソコンの改造か，産業用ワンボードマイコン程度に限られてしまったことです。ゲームパソコンも68000系や8086系，65816といった16ビットCPUを使い始めている現状では，63C09を積んだパソコンをメーカーが出荷することは，まずありえないことでしょう。当面は市販アプリに依存しないFM-11等の改造にしか使えないというのは残念なことです。

なお，この稿をまとめるにあたっては，私が解析した資料のほか，63C09解析委員会の仲間（とくにGigo氏とMiyazaki氏）が解析された資料を参照させていただきました。63C09解析委員会関係者のご協力に感謝いたします。

〈参考文献〉

- 63C09解析委員会，「お年玉プレゼント 63C09に隠し機能があった」等，NANNO-NET，1988年1月1日～
- モトローラ，「MC6809-MC6809Eマイクロプロセッサプログラミングマニュアル」，CQ出版社，1982年
- 「6809 インストラクションポケットブック」，Oh!FM 1983年第4号，日本ソフトバンク
- 水谷隆太，「6809の未定義命令」，I/O 1985年5月号，工学社
- 原進，「FM-11のクロックを3MHzに」，パソコンワールド 1987年1月号，ピーシーワールドジャパン

図3　63C09で増えた命令(灰色に塗られた部分)

(横の列は，上からプリバイトなし，プリバイト＄10付き，プリバイト＄11付きの順に並んでいる。また，各項目中の下段左側の数値はサイクル数で，カッコ内がネイティブモード時の値。右側は命令長。)

	DIRECT 0000xxxx 0x	0001xxxx 1x	0010xxxx 2x	REL 0011xxxx 3x	ACC A/D/E 0100xxxx 4x	ACC B/W/F 0101xxxx 5x	INDEXD 0110xxxx 6x	EXTEND 0111xxxx 7x	IMMED 1000xxxx 8x	DIRECT 1001xxxx 9x	INDEXD 1010xxxx Ax	EXTEND 1011xxxx Bx	IMMED 1100xxxx Cx	DIRECT 1100xxxx Dx	INDEXD 1110xxxx Ex	EXTEN 1111xxxx Fx
0000 0 (なし)	NEG 6(5),2	(PRE) (BYTE1)	BRA 3,2	LEAX 4+,2+	NEGA 2(1),1	NEGB 2(1),1	NEG 6+,2+	NEG 7(6),3	SUBA 2,2	SUBA 4(3),2	SUBA 4+,2+	SUBA 5(4),3	SUBB 2,2	SUBB 4(3),2	SUBB 4+,2+	SUBB 5(4),3
($10)				addr 4,3	negd 3(2),2				subw 5(4),4	subw 7(5),3	subw 7+(6+),3+	subw 8(6),4				
($11)				band 7(6),4					sube 3,3	sube (4),3	sube 5+,3+	sube 6(5),4	subf 3,3	subf 5(4),3	subf 5+,3+	subf 6(5),4
0001 1 (なし)	oim 6,3	(PRE) (BYTE2)	BRN 3,2	LEAY 4+,2+			oim 7+,3+	oim 7,4	CMPA 2,2	CMPA 4(3),2	CMPA 4+,2+	CMPA 5(4),3	CMPB 2,2	CMPB 4(3),2	CMPB 4+,2+	CMPB 5(4),3
($10)			LBRN 5,4	adcr 4,3					cmpw 5(4),4	cmpw 7(5),3	cmpw 7+(6+),3+	cmpw 8(6),4				
($11)				biand 7(6),4					cmpe 3,3	cmpe 5(4),3	cmpe 5+,3+	cmpe 6(5),4	cmpf 3,3	cmpf 5(4),3	cmpf 5+,3+	cmpf 6(5),4
0010 2 (なし)	aim 6,3	NOP 2(1),1	BHI 3,2	LEAS 4+,2+			aim 7+,3+	aim 7,4	SBCA 2,2	SBCA 4(3),2	SBCA 4+,2+	SBCA 5(4),3	SBCB 2,2	SBCB 4(3),2	SBCB 4+,2+	SBCB 5(4),3
($10)			LBHI 5/6(5),4	subr 4,3					sbcd 5(4),4	sbcd 7(5),3	sbcd 7+(6+),3+	sbcd 8(6),4				
($11)				bor 7(6),4												
0011 3 (なし)	COM 6(5),2	SYNC 2,1	BLS 3,2	LEAU 4+,2+	COMA 2(1),1	COMB 2(1),1	COM 6+,2+	COM 7(6),3	SUBD 4(3),3	SUBD 6(4),2	SUBD 6+(5+),2+	SUBD 7(5),3	ADDD 4(3),3	ADDD 6(4),2	ADDD 6+(5+),2+	ADDD 7(5),3
($10)			LBLS 5/6(5),4	sbcr 4,3	comd 3(2),2	comw 3(2),2			CMPD 5(4),4	CMPD 7(5),3	CMPD 7+(6+),3+	CMPD 8(6),4				
($11)				bior 7(6),4	come 3(2),2	comf 3(2),2			CMPU 5(4),4	CMPU 7(5),3	CMPU 7+(6+),3+	CMPU 8(6),4				
0100 4 (なし)	LSR 6(5),2	sexw 4,1	BHS/BCC 3,2	PSHS 5+(4+),2	LSRA 2(1),1	LSRB 2(1),1	LSR 6+,2+	LSR 7(6),3	ANDA 2,2	ANDA 4(3),2	ANDA 4+,2+	ANDA 5(4),3	ANDB 2,2	ANDB 4(3),2	ANDB 4+,2+	ANDB 5(4),3
($10)			LBHS/BCC 5/6(5),4	andr 4,3	lsrd 3(2),2	lsrw 3(2),2			andd 5(4),4	andd 7(5),3	andd 7+(6+),3+	andd 8(6),4				
($11)				beor 7(6),4												
0101 5 (なし)	eim 6,3		BLO/BCS 3,2	PULS 5+(4+),2			eim 7+,3+	eim 7,4	BITA 2,2	BITA 4(3),2	BITA 4+,2+	BITA 5(4),3	BITB 2,2	BITB 4(3),2	BITB 4+,2+	BITB 5(4),3
($10)			LBLU/BCS 5/6(5),4	orr 4,3					bitd 5(4),4	bitd 7(5),3	bitd 7+(6+),3+	bitd 8(6),4				
($11)				bieor 7(6),4												
0110 6 (なし)	ROR 6(5),2	LBRA 5(4),3	BNE 3,2	PSHU 5+(4+),2	RORA 2(1),1	RORB 2(1),1	ROR 6+,2+	ROR 2,2	LDA 7(6),3	LDA 4(3),2	LDA 4+,2+	LDA 5(4),3	LDB 2,2	LDB 4(3),2	LDB 4+,2+	LDB 5(4),3
($10)			LBNE 5/6(5),4	eorr 4,3	rord 3(2),2	rorw 3(2),2			ldw 4,4	ldw 6(5),3	ldw 6+,3+	ldw 7(6),4				
($11)				ldbt 7(6),4					lde 3,3	lde 5(4),3	lde 5+,3+	lde 6(5),4	ldf 3,3	ldf 5(4),3	ldf 5+,3+	ldf 6(5),4
0111 7 (なし)	ASR 6(5),2	LBSR 9(7),2	BEQ 3,2	PULU 5+(4+),2	ASRA 2(1),1	ASRB 2(1),1	ASR 6+,2+	ASR 7(6),3		STA 4(3),2	STA 4+,2+	STA 5(4),3		STB 4(3),2	STB 4+,2+	STB 5(4),3
($10)			LBEQ 5/6(5),4	cmpr 4,3	asrd 3(2),2					stw 6(5),3	stw 6+,3+	stw 7(6),4				
($11)				stbt 8(7),4						ste 5(4),3	ste 5+,3+	ste 6(5),4		stf 5(4),3	stf 5+,3+	stf 6(5),4
1000 8 (なし)	ASL/LSL 6(5),2		BVC 3,2		ASLA/LSLA 2(1),1	ASLB/LSLB 1(2),1	ASL/LSL 6+,2+	ASL/LSL 7(6),3	EORA 2,2	EORA 4(3),2	EORA 4+,2+	EORA 5(4),3	EORB 2,2	EORB 4(3),2	EORB 4+,2+	EORB 5(4),3
($10)			LBVC 5/6(5),4	pshsw 5/6,2	asld 3(2),2				eord 5(4),4	eord 5(5),3	eord 7+(6+),3+	eord 8(6),4				
($11)				tfm(r+,r+) 6+3n,3												
1001 9 (なし)	ROL 6(5),2	DAA 2(1),1	BVS 3,2	RTS 5(4),1	ROLA 2(1),1	ROLB 2(1),1	ROL 6+ 2+	ROL 2,2	ADCA 7(6),3	ADCA 4(3),2	ADCA 4+,2+	ADCA 5(4),3	ADCB 2,2	ADCB 4(3),2	ADCB 4+,2+	ADCB 5(4),3
($10)			LBVS 5/6(5),4	pulsw 6,2	rold 3(2),2	rolw 3(2),2			adcd 5(4),4	adcd 7(5),3	adcd 7+(6+),3+	adcd 8(6),4				
($11)				tfm(r-,r-) 6+3n,3												
1010 A (なし)	DEC 6(5),2	ORCC 3(2),2	BPL 3,2	ABX 3(1),1	DECA 2(1),1	DECB 2(1),1	DEC 6+,2+	DEC 7(6),3	ORA 2,2	ORA 4(3),2	ORA 4+,2+	ORA 5(4),3	ORB 2,2	ORB 4(3),2	ORB 4+,2+	ORB 5(4),3
($10)			LBPL 5/6(5),4	pshuw 6,2	decd 3(2),2	decw 3(2),2			ord 5(4),4	crd 7(5),3	ord 7+(6+),3+	ord 8(6),4				
($11)				tfm(r+,r) 6+3n,3	dece 3(2),2	decf 3(2),2										
1011 B (なし)	tim 4,3		BMI 3,2	RTI 6/15(17),1			tim 5+,3+	tim 5,4	ADDA 2,2	ADDA 4(3),2	ADDA 4+,2+	ADDA 5(4),3	ADDB 2,2	ADDB 4(3),2	ADDB 4+,2+	ADDB 5(4),3
($10)			LBMI 5/6(5),4	puluw 4,3					addw 5(4),4	addw 7(5),3	addw 7+(6+),3+	addw 8(6),4				
($11)				tfm(r,r+) 6+3n,3					adde 3,3	adde 5(4),3	adde 5+,3+	adde 6(5),4	addf 3,3	addf 5(4),3	addf 5+,3+	addf 6(5),4
1100 C (なし)	INC 6(5),2	ANDCC 3,2	BGE 3,2	CWAI 20(22),2	INCA 2(1),1	INCB 2(1),1	INC 6+,2+	INC 7(6),3	CMPX 4(3),3	CMPX 6(4),2	CMPX 6+(5+),2	CMPX 7(5),3	LDD 3,3	LDD 5(4),2	LDD 5+,2+	LDD 6(5),3
($10)			LBGE 5/6(5),4		incd 3(2),2	incw 3(2),2			CMPY 5(4),4	CMPY 7(5),3	CMPY 7+(6+),3+	CMPY 8(6),4		ldq 8(7),3	ldq 8+,3+	ldq 9(8),4
($11)				BITMD 4,3	ince 3(2),2	incf 3(2),2			CMPS 5(4),4	CMPS 7(5),3	CMPS 7+(6+),3+	CMPS 8(6),4				
1101 D (なし)	TST 6(4),2	SEX 2(1),1	BLT 3,2	MUL 11(10),1	TSTA 2(1),1	TSTB 2(1),1	TST 6+(5+),2+	TST 7(5),3	BSR 7(6),2	JSR 7(6),2	JSR 7+(6+),2+	JSR 8(7),3	ldq 5,5	STD 5(4),2	STD 5+,2+	STD 6(5),3
($10)			LBLT 5/6(5),4		tstd 3(2),2	tstw 3(2),2								stq 8(7),3	stq 8+,3+	stq 9(8),4
($11)				ldmd 5,3	tste 3(2),2	tstf 3(2),2			divd 25,3	divd 27(26),3	divd 27+,3+	divd 28(27),4				
1110 E (なし)	JMP 3(2),2	EXG 8(5),2	BGT 3,2				JMP 3+,2+	JMP 4(3),3	LDX 3,3	LDX 5(4),2	LDX 5+,2+	LDX 6(5),3	LDU 3,3	LDU 5(4),2	LDU 5+,2+	LDU 6(5),3
($10)			LBGT 5/6(5),4						LDY 4,4	LDY 6(5),3	LDY 6+(6+),3+	LDY 7(6),4	LDS 4,4	LDS 6(5),3	LDS 6+(6+),3+	LDS 7(6),4
($11)									divq 34,4	divq 36(35),3	divq 36+,3+	divq 37(36),4				
1111 F (なし)	CLR 6(5),2	TFR 6(4),2	BLE 3,2	SWI 19(21),1	CLRA 2(1),1	CLRB 2(1),1	CLR 6+,2+	CLR 7(6),3		STX 5(4),2	STX 5+,2+	STX 6(5),3		STU 5(4),2	STU 5+,2+	STU 6(5),3
($10)			LBLE 5/6(5),4	SWI2 20(22),2	clrD 3(2),2	clrw 3(2),2				STY 6(5),3	STY 6+(6+),3+	STY 7(6),4		STS 6(5),3	STS 6+(6+),3+	STS 7(6),4
($11)				SWI3 20(22),2	clre 3(2),2	clrf 3(2),2			muld 28,4	muld 30(29),3	muld 30+,3+	muld 31(30),4				