X386/XFree86 のビデオ情報調整を自分で行なうための手引き

(または趣味と実益のためのモニター調整の手引き)

日本語訳　岡本　一幸 Kazuyuki Okamoto <ikko-@pacific.rim.or.jp>

目 次

1 序文

改良のための率直なご意見、ご批評とご提案は*esr@snark.thyrsus.com*宛てに送ってください。[訳注: 勿論、本書の日本語版に対する改良のための率直なご意見、ご批評とご提案は*<ikko-@pacific.rim.or.jp>*`(mailto:ikko-@pacific.rim.or.jp)`宛てに送って下さい。]

ユーザーはXFree86サーバーを自分のビデオに合わせてハードウェアを使いこなすように構成出来ます。この手引き書は自分自身のビデオカードとモニタの最適な時間調節の数値を、如何に生成するか勉強する助けになるでしょう。

ここではまずXFree86サーバーを何とか動かすための方法を提示し、それをベースに実験をしながらいろいろと設定を変更して自分の好みに合わせるためのやり方を説明します。

ほとんど動作しているモードを持っている場合は（特に、前もって設定しているVESAモードは左右に片寄ったり、小さすぎたり、大きすぎたりするでしょうが、安定しているでしょう）、さっそく問題修正の節から始めてください。この節は調整数値をちょうど良い数値に近づける方法を教えてくれるでしょう。

XFree86はホットキーでXF86Configファイルに定義されている異なるモード間を移動できます（詳細についてはXF86Config.manを参照してください）。これはとても便利な機能です。新しいモードを試してみたくなったら、新しいモードを重複しないモードラベルを付けてホットキーリストの*end*に追加してください。新しいモードが動かなかったときの保険に、既に動いているモードは標準値として残しておいてください。「二つの例題」章の最後のXconfig節を読めば、すみやかに問題を解決するためにはどのように実験を記録すればよいか、参考になるでしょう。

最初に`lib/X11/doc`にある`Monitors`ファイルを確認して自分のモニターがデータベースの中にあれば、多分この文書の残りの部分は飛ばして構いません!データベースに自分のカードが使用するモードが無かった場合、いくつかの時間調節の数値を測定する必要がありますが、簡単な作業です。

2 ビデオディスプレイの動作原理

ビデオディスプレイの動作原理を知ることはXconfigファイルの色々な場所にどんな数字を入れるかを理解する上で重要です。これらの値はXFree86サーバーがディスプレイをハードウェアレベルで制御するのに使用されます。

ディスプレイはドットの集まりから画像を表示します。このドットを左から右へ並べて線を作ります。この線を上から下へ並べて画像を作ります。ドットはディスプレイ内部からの電子ビームで叩かれたとき発光します。それぞれのドットに均一に電子ビームを当てるために、一定のパターンでディスプレイ上を走査します。

そのパターンはスクリーンの左上から始まり、右へ真直ぐスクリーンを横切るように走り、スクリーンの右端で一旦停止します。そして電子ビームはディスプレイの左端に移動し、一本下に移動します。新しい線はディスプレイの左から右へ最初の線を引いたように走査します。このパターンをディスプレイの一番下まで繰り返します。それから電子ビームがディスプレイの左上から右下まで移動したら、また最初から繰り返します。

フレームの始まりはディスプレイの左上の電子ビームの開始点です。電子ビームがディスプレイの右下隅まで届いてから左上隅に再び戻ってくる時がフレームの終了点です。フレームはディスプレイの一番上から一番下までの全ての電子ビームの線で出来ています。

もし電子ビームがフレームを移動している間ずっと「オン」だったら、ディスプレイの全てのドットは点灯してしまい、ディスプレイの縁には黒い部分はなくなるでしょう。そしてディスプレイの縁では、電子ビームの制御が難しいので画像が歪曲してしまうでしょう。この歪みを減らすため、ディスプレイの縁のドットには、電子ビームが届いても、ドットが輝かないようになっています。ディスプレイの実際に表示される領域が小さくなっているのは、こういうわけなのです。

もう一つ理解してほしいのは、表示される領域を描画していない時に電子ビームがどうなっているかということです。ディスプレイの横端を描画するために使われるはずだった時間は、電子ビームを右の端から左の端まで戻し、一つ下の次のラインに移動するために使われます。ディスプレイの上端および下端を描画するために掛かるはずだった時間は、電子ビームをディスプレイの右下隅から左上隅まで移動するために使われます。

アダプタカードはディスプレイの電子銃を点灯させ全てのドットで画像を生成するような信号を作成します。また、カードは電子ビームが右から左に移動して一本下がるときに水平同期信号と呼ばれる信号を作成します。すべてのラインの最後に一つの水平同期信号が発生します。さらに、アダプタカードは電子ビームをディスプレイの左上隅に移動するための垂直同期信号も生成します。垂直同期信号は全てのフレームの終わり近くに作成されます。

ディスプレイには、電子ビームの位置を安定させるため、水平同期信号と垂直同期信号の前後に短時間の余裕が必要です。電子ビームの安定化が出来ないと、画像がしっかりしません。

以降の節で、定義、公式と例題の補助のために、これらの基本に戻るでしょう。

3 ディスプレイとアダプタについての基礎知識

Xconfig の設定項目をさわる前に、以下の基礎的な事項を知っておく必要があります。

1. モニターの水平同期信号と垂直同期信号の周波数
2. ビデオアダプタの動作クロック周波数、または”ドットクロック”
3. モニターの周波数帯幅

です。

モニターの同期周波数：

モニターの水平同期周波数は、そのモニターが 1 秒間に書ける水平走査線の数のことで、これはモニターについて最も重要な統計値です。垂直同期周波数は、そのモニターが 1 秒間に電子ビームを縦方向に通過させることのできる回数のことです。

同期周波数は普通、モニターのマニュアルの仕様の頁に一覧になっています。垂直同期周波数の数値は一般的に Hz（秒当たりの単位周期）で、水平同期周波数は KHz（秒当たりの千単位周期）で計測されています。通常の範囲は垂直については 50 から 80Hz、水平については 31 から 135KHz 程度です。

マルチスキャンモニターの場合、その周波数は幅のある値として表示されています。ローエンドのものに多いのですが、複数の固定した周波数を持っているモニターもあります。このようなモニターも普通のモニターと同様に設定は出来ますが、モニターの持つ特徴に厳しく制限されてしまうでしょう。最高の解像度が得られるような最も高い水平同期と垂直同期周波数の組み合わせを選択してください。そして、固定周波数モニターでは設計値より高い周波数を与えるとモニターを痛めるおそれがあるので注意してください。

カードの動作周波数：

ビデオアダプタの仕様書にカードのドットクロックがあります（ドットクロックとは画面へ 1 秒間に点を表示できる総数です）。この情報が無い場合は、X サーバーがそれを取ってきます。X がモニターを固めてしまった場合でも、クロックとその他の情報を標準出力に吐き出します。この情報をファイルにリダイレクトした場合、コンソールへ戻って再立ち上げしなければならなくなってもファイルに保存されます。

SGCS X を使っている場合は、次の例のような行をローカルバス接続の S3 アダプタの Swan から収集します。XFree86 はちょっと難しい複数行にわたる書式を使っています。

```
WGA: 86C911 (mem: 1024k clocks: 25 28 40 3 50 77 36 45 0 0 79 31 94 65 75 71)
---  ------        -----          ------------------------------------------
 |      |             |                       動作可能な周波数を MHz で表わします。
 |      |            +-- ボード上のフレームバッファメモリの大きさ
 |      +-- チップの型式
 +-- サーバーの種類
```

注意: なるべくこの作業はマシンの負荷が低い時に行なって下さい。X はアプリケーションですから、ディスクの動作と時間調節のループが衝突すると、上記の数字は不正確になります。何回か繰り返し実行し、数字が大きく変動しないことを確かめて下さい。もし変動が大きい場合には、安定するまでプロセスを殺してみてください。SVr4 (システム V リリース 4) を使用している人へ: mousemgr (マウスマネージャー) プロセスは特に混乱の元です。

このような不正確さを避けるため、得られたクロックの数字をそのまま Clocks プロパティの値として Xconfig に取り込んで下さい。これは時間調節のループを抑止し、X が試してみることのできるクロックの値の正確な一覧を与えるためです。上記の例のデータを使うと、次のようになります。

```
wga
        Clocks  25 28 40 3 50 77 36 45 0 0 79 31 94 65 75 71
```

高く変わりやすい負荷が掛かったシステムでは、この方法は X の起動時に陥ることがある不思議な失敗を回避する助けになるでしょう。X が起動した時システムの負荷のせいで間違った値を得てしまい、config データベースから丁度いいドットクロックを見つけることが出来なかったり、間違ったものを見つけてしまうことがあり得るのです。

The monitor's video bandwidth:

モニターのビデオ信号帯域幅 :

モニターのビデオ信号帯域幅を知っておくことは、使用できる最も高いドットクロックのおおよその値を知るのに有用です。しかし、この値は柔軟性に富んでいます。例えば、名目上の帯域幅の 30% 増しで動作できるモニターもあります。

帯域幅を知ることは可能な構成定義からより賢い選択ができるようになります。あなたのディスプレイの表示品質 (特に高精細のためのシャープさ) に影響を及ぼします。

モニターのビデオ信号帯域幅はマニュアルの仕様の頁に載っています。無かった場合は、モニターの最も高い解像度のところを見てください。解像度から帯域幅 (つまり使用できるドットクロックの大まかな上限値) を推定するための経験則を下に示します。

640x480	25
800x600	36
1024x768	65
1024x768 interlaced	45
1280x1024	110

ところで、この表は絶対的なものではありません。これらの数字は標準的な XFree86 モードデータベースでの解像度毎の最も低いドットクロックです。モニターの帯域幅は一番上の解像度に要求される最小の帯域幅より高いでしょうから、恐れずにドットクロックを数 MHz 高めに試してみてください。

また、ドットクロックが 65MHz 位より低い場合には帯域幅はほとんど問題にならないことに注意してください。SVGA やほとんどの高解像度のモニターでは、これはモニターのビデオ信号帯域幅の限界よりもはるかに低い周波数ですから。次に例を示します。:

ブランド名	ビデオ信号帯域幅
NEC 4D	75Mhz
Nano 907a	50Mhz
Nano 9080i	60Mhz
Mitsubishi HL6615	110Mhz
Mitsubishi Diamond Scan	100Mhz

IDEK MF-5117	65Mhz
IOCOMM Thinksync-17 CM-7126	136Mhz
HP D1188A	100Mhz
Philips SC-17AS	110Mhz
Swan SW617	85Mhz

一番下のクラスのモニターでも、解像度に関して非常にビデオ信号帯域幅に制約を受けることはありません。NEC マルチシンク II が良い例です（仕様によれば 800x600 は表示出来ない）。そのディスプレイは 800x560 のみ表示できます。このような低解像度の場合は、高いドットクロックや大きなビデオ信号帯域幅を必要としなく、多分 32Mhz か 36Mhz で十分で、両方の周波数ともモニターのビデオ信号帯域幅である 30Mhz からそれ程かけ離れた値ではありません。

これら 2 つの動作周波数では、ディスプレイの持っている性能よりくっきりと表示しないかもしれませんが、でもかなりの品質だと言い切ってもいいでしょう。勿論、NEC マルチシンク II がもっと高い、例えば 36MHz ビデオ信号帯域幅を持っているに越したことはありません。しかし、大きく画像が歪む程周波数がかけ離れていなければ、文章を編集する等の一般的な作業には問題はありません。（もし画像の歪みがあまりにも大きい場合には、目で見てすぐわかるでしょう）。

どうやって制御するか：

モニターの同期信号帯域幅は、ビデオアダプタのドットクロックと共に、表示できる最高の解像度を決定します。しかしハードウェアの性能を引き出すのはドライバーです。どんなに優れたビデオアダプタやモニターでも、良いデバイスドライバーがなければ宝の持ち腐れになってしまいます。一方、有能でないハードでも多目的に使用できるデバイスドライバがあれば十分役に立ちます。これが XFree86 の設計哲学です。

4 基本仕様の読み方

この節では仕様が何を意味するかと、その他知らなければならないことを説明します。先ず最初に、定義をします。次に計算をする時に使う変数名を括弧内で示します。[訳注：この定義だけは対訳風にします。]

horizontal sync frequency (HSF)

Horizontal scans per second (see above).

水平同期周波数 (HSF)

毎秒の水平走査数（上記参照）。

vertical sync frequency (VSF)

Vertical scans per second (see above). Mainly important as the upper limit on your refresh rate.

垂直同期周波数 (VSF)

毎秒の垂直走査数（上記参照）。主に再描画速度の上限として重要。

dot clock (DCF)

More formally, ‘driving clock frequency’; sometimes loosely called ‘bandwidth’. The frequency of the crystal or VCO on your adaptor — the maximum dots-per-second it can emit.

ドットクロック (DCF)

より正式には、‘駆動クロック周波数’; 時々適当に ‘帯域幅’ と呼ぶ。アダプタの発信子または VCO の周波数 — 毎秒描画可能ドット数の最大。

video bandwidth (VB)

The highest frequency at which your monitor's video signal can change. This constrains the highest dot clock you can use and the overall sharpness of fine details in the video image.

ビデオ信号帯域幅 (VB)

モニターのビデオ信号が変化できる最高の周波数。これは最大のドットクロックと画像全体に詳細表示した時のシャープさを制約します。

frame length (HFL, VFL)

Horizontal frame length (HFL) is the number of dot-clock ticks needed for your monitor's electron gun to scan one horizontal line, *including the inactive left and right borders*. Vertical frame length (VFL) is the number of scan lines in the *entire* image, including the inactive top and bottom borders.

フレーム長 (HFL, VFL)

水平フレーム長 (HFL) はモニターの電子銃が１つの＊使われていない左右の境界を含む＊水平線を走査するのに必要なドットクロックの数。垂直フレーム長 (VFL) は使われていない上と下の境界を含む＊完全な＊画面の走査線の数です。

screen refresh rate (RR)

The number of times per second your screen is repainted. Higher frequencies are better, as they reduce flicker. 60Hz is good, VESA-standard 72Hz is better. Compute it as

```
RR = DCF / (HFL * VFL)
```

画面再描画速度 (RR)

毎秒の画面再描画回数。高いほうがちらつきを低減します。60Hz で良く、VESA 標準の 72Hz の方がより良いでしょう。計算はこのようにします。

```
RR = DCF / (HFL * VFL)
```

Note that the product in the denominator is *not* the same as the monitor's visible resolution, but typically somewhat larger. We'll get to the details of this below. 分母にある積はモニターに表示される解像度では＊なく＊、幾らか大きいことに注意してください。これについては以降で詳細に説明します。

帯域幅について：

モニター製造会社は帯域幅が画面の迫力と色変化のシャープさを制約するので高帯域幅であることを宣伝します。帯域幅が大きいほど、より細かい画像を表示することができます。

モニターは電気信号を用いて画像を表示します。信号は一旦デジタルからアナログへと変換されると、つねにアナログ波形として取り扱われます。それは多くの、固定した周波数の単純な波形の組合せであると考えられ、それらの多くは MHz の範囲で例えば 20MHz、40MHz、さらに 70MHz だったりします。モニターのビデオ信号帯域幅は事実上歪みが無く扱える高周波のアナログ信号です。

私達の目的のためには、帯域幅は主に使用可能なドットクロックのおおよその上限として重要です。

同期周波数と再描画速度：

画面上の水平走査線はフレーム長走査の中で実際に表示される部分です。それぞれの瞬間には本当はたった一つの点が画面に輝いているのですが、再描画速度が十分速いので目には絶え間無く全ての画像が”見える”という訳です。

ここでいくつかの絵で解説します：

```
 -----------------------
|                       |
|->->->->->->->->->->->  |
|                      )|
|<-----<-----<-----<---  |
|                       |
|                       |
|                       |
|                       |
|_______________________|
```

水平フレーム長は電子ビーム
がこのようなパターンを描く
時間をドットクロック単位で
表したものです。

```
 -----------------------
|        ^              |
|       ^ |             |
|       | v             |
|       ^ |             |
|       | |             |
|       ^ |             |
|       | v             |
|       ^ |             |
|_______|_v_____________|
```

垂直フレーム長は電子ビーム
がこのようなパターンを描く
時間をドットクロック単位で
表したものです。

実際のラスター走査はとても細かいジグザグ型のパターンをしていて、左右に電子ビームが動いて同時に上下にも動いています。

さて、ドットクロックとフレームの大きさは再描画速度に関係があるということが分かります。定義上、１ヘルツ (hz) は１秒に１周期です。それから、水平フレーム長を HFL とし垂直フレーム長を VFL とした場合に全ての画面を覆うには (HFL * VFL) 回ドットクロックが必要です。カードは定義から毎秒 DCF 回信号を出していますので、明らかにモニターの電子銃は左から右、戻って、下から上へ、戻ってを毎秒 DCF / (HFL * VFL) 回、画面上を動き回れる訳です。これは毎秒何回画面を描き直しているかを表わしているので、画面の再描画速度なのです。

解像度とちらつきの関係がトレードオフの関係にあるので、自分の要求に応じて設定を行なうためにこの概念を理解する必要があります。

5　システムの構成におけるトレードオフ

前に示した公式は、このように変形できます。

```
DCF = RR * HFL * VFL
```

つまり、ドットクロックが一定だとすると、一秒間にこれだけのドット数を再描画速度、水平解像度または垂直解像度に振り分けることができるということです。これらの数字の一つを増やすと他の数字を減らさなければなりません。

しかし、再描画速度はモニターの最大垂直同期周波数を超えることは出来ませんので、注意してください。従って与えられたドットクロックで与えられたモニターでは、自分で強制出来ないフレーム長の積の最小以下になります。

自分の設定を選ぶ時に、RR が低すぎる場合、画面のちらつきで顔をしかめることを覚えてください

多分、再描画速度を 60Hz 以下に下げたくは無いでしょう。蛍光燈のちらつく速さですので、ちらつきに敏感な場合は VESA 人間工学標準の 72MHz に保つ必要があります。

ちらつきは大変目に辛いものですが、しかし人間の目は適応し人々のちらつきに対する耐性はかなり広範囲です。画面の 90% が見える角度でモニターに向き合っている場合に、暗い背景と良いコントラストの色を前景に使い、輝度を低から中に調整するならば、＊たぶん＊ 45Hz 位に小さくても快適でしょう。

厳密なテストのやり方は次の通りです : xterm -bg white -fg black で真っ白な背景に黒の前景の xterm を開いて、表示可能な領域全てを隠すぐらいの大きさにして下さい。そしてモニターの明かる

さを最大の設定値の 3/4 に設定して、モニターから顔をそむけて、モニターを横目で覗いてみて下さい（これは、より敏感な視野周辺部の細胞を働かせるためです）。なんにもちらつきを感じない場合若しくは許せる範囲ならば、あなたにとってその再描画速度は丁度良いのです。もしそうでなかったら、一見大丈夫なように見えても、明らかには判からないようなちらつきによってひどく目が疲労し頭痛を起こしますので、普通に見える様になる迄、より高い再描画速度に調整して下さい。

このようにして最小の許容できる再描画速度を選ぶことができます。HFL と VFL を選択するためには、多少の作戦の余地があるでしょう。

6 要求メモリ量

自分が実現しようとしているカラーやグレースケールディスプレイの解像度は使用可能なフレームバッファメモリの量で制限されます。2 色（白黒）でグレースケールでない物の場合は、制限を受けることは多分ありません。

256 色のディスプレイの場合は、ビデオメモリのバイト数は表示されるドットの数だけ必要です。このバイト数は赤緑青から成る集合の点を１点とした数です。必要なメモリ量を得るには、線１本当たりに表示される点の数に表示される線の数を掛けて下さい。800x600 の解像度を持つディスプレイの場合は、ディスプレイに表示する点の数は 800 x 600 = 480,000 になります。また、１ドットが１バイトに成るので、アダプタカードに同じバイト数のビデオメモリが必要です。

従って、メモリ量は、一般に (HR * VR)/1024 をキロバイト単位に切り上げた物が必要です。例を挙げれば、(936 * 702)/1024 = 642K バイトとなります。従って 1M バイトのメモリがあれば、余りを仮想スクリーンに割り当ててスクロール出来ます。

ところが、ボードに 512K しか無い場合はこの解像度は使えません。良いモニターを持っていたとしても、十分なビデオメモリが無しにはモニターの性能を活かすことは出来ません。また一方で、SVGA カードが 1M バイトのメモリを持っていたとしても、モニターが最高で 800x600 しか表示できないならば、これ以上の高解像度はどうやっても無理というものです。

要求量より多いメモリを持っていても心配しないで下さい。XFree86 は余ったメモリで表示領域をスクロールできるようにします（仮想スクリーンの大きさのパラメータについての Xconfig ファイルの文書を参照して下さい）。また、512K バイトのメモリを搭載したカードは 512,000 バイトではなく、本当は 512 x 1024 = 524,288 バイトのメモリが搭載されていることを覚えていて下さい。

S3 カードで SGCS X が動作していて、かつ 16 色（1 ピクセル当たり 4 ビット）の場合、Xconfig ファイルで 深さ 4 と設定すれば、カードは倍の解像度を使えます。例えば、S3 カードは通常 1024x768x256（1024x768 の解像度で 256 色）ですが、深さ 4 にすれば 1280x1024x16（1280x1024 の解像度で 16 色）が使えるようになります。[訳注：深さとは色数やグレースケールを表現するビット数です。]

7 フレームサイズの計算

警告：この方法はマルチシンクモニターのために開発しました。多分、この方法は固定周波数モニターでうまく行くかも知れませんが、保証できません。

最初に最大の使用可能な HSF で DCF を割って、１秒あたりの水平走査可能回数を計算して下さい。

例えば、65MHz のドットクロックの Sigma Legend SVGA カードと、55KHz の水平走査周波数のモニターを使っていると仮定します。DCF / HSF を計算すると 1181 という量が得られます。

さあ、最初の黒魔術の技の片鱗を使いましょう。この式の答えをもっとも近い 8 の倍数に丸めて下さい。これは 8 ビットレジスタを持ち、左に 3 ビットずらして 11 ビットの値を得るような SVGA と S3 の VGA 制御装置において有効です。ATI 8514/A のような他のカードではこのような要求はないかも知れませんが、我々は正確な知識を持ち合わせてはいませんし、丸めによって不都合なことが生じることはありません。従って１秒あたりの水平走査可能回数を 1176 に丸めます。

この数字（DCF / HSF を 8 の倍数に丸めたもの）は最小の HFL として用いることが出来ます。より低い HSF で同期信号を設定すれば、もっと長い HFL（つまり、多分より多くの画面の水平方向のドット数）が得られます。しかし、遅くてより多く見えるちらつきの速度に復讐されるでしょう。

経験的な法則では、水平フレーム長の 80% が水平解像度として使用可能で、水平走査線の表示される部分（これは大体、境界と電子ビームが画面の右端から次の走査線の左端へ戻ってくる時間を差し引いたもの）です。この例では、944 になります。

さて、通常の画面のアスペクト比（横縦比）4:3 を得るため、今計算した水平解像度の 3/4 になるように垂直解像度を設定しましょう。この例では、708 になります。実際の VFL を得るには、1.05 を掛けて 743 になります。

この 4:3 について — 表示上の幅と高さが 4:3 の比率は大体黄金分割比の値、(1 + sqrt(5))/2 に近い値です。人間はこの種の矩形を見て好ましいと思うようにできているようです。そのためブラウン管や 800x600, 640x480 そして 1024x768 という標準的な解像度はみなこの黄金分割比の近似値になっています。しかしこれは心理学的な要求であって、技術的な要求ではありません。もしその方が画面の大きさを有効に活用できるのであれば、非黄金分割比を使うことをためらう理由は何もありません。

結局、HFL=1176 と VFL=743 としました。65MHz をこの 2 つの数字の積で割ると、十分に高くて健康に良い 74.4Hz の再描画速度が得られます。素晴らしい！VESA 標準より立派でしょう！おまけに、おそらく予想していた 800x600 よりも高い 944x708 という解像度が得られたのです。本当に素晴らしい！

さらに（大体 76 Hz 迄）再描画速度を改良することも可能です。これには、定格よりも 2 KHz くらい高い水平同期周波数でも動くモニターが多いという事実と、VFL を幾らか下げる（つまり、上の例で言うと 944 の 75% よりも小さくする）ことを利用します。しかしこの "オーバードライブ" 作戦を試してみる前に、あなたのモニターの電子銃が垂直同期を 76 Hz 以上でできることを * 確認 * してください。（例えば、人気のある NEC 4D ではこれは出来ません。これは 75 Hz 迄の VSF のみが利用できます。）

以上が、ラスター表示についての単純な計算と基本的な事柄のほとんどです。ほとんど黒魔術でも何でも無いですね。

8 黒魔術と同期信号

さあ、自分で選んだドットクロック対応に計算した HFL/VFL の数字があり、無難な再描画速度が見つかり、十分な VRAM（ビデオメモリ）があることを確認しました。これからが本当の黒魔術です – いつどこで同期信号を出すかを知る必要があります。

同期信号が実際にモニターの水平及び垂直の走査周波数を制御しています。仕様表から引っぱり出してきた HSF と VSF は定格上の最大同期周波数の推定値です。アダプタカードからの信号の中の同期信号はモニターにどれくらい速く実際に動作するか伝えます。

上記の 2 つの絵を思い出して下さい。目に見える画像（あなたの選んだ解像度）を表示するために使われるのは、フレームをラスター走査するために必要な時間の一部だけです。

水平同期：

前の定義によれば、１本の水平走査線をたどるのには HFL 分の時間掛かります。表示される部分のクロック回数（水平スクリーン解像度）を HR と呼びましょう。その時は明らかに、定義から HR ¡ HFL となります。具体的に両方が同時に開始したと仮定して次に示します：

```
|--- -- -- -- -- -- -- -- -- -- -- -- --
|- - - - - - - - - - - -                |
|_______________________|_______________|_____
0                       ^               ^   単位: 1 クロック
                        |   ^       ^   |
                        HR  |       |  HFL
```

```
|   |<----->|   |
|<->|  HSP  |<->|
 HGT1        HGT2
```

ここで、上にあるように表示データのクロック終了とフレーム全体のクロック終了の間に HSP の同期信号長を配置します。何故そうするのか ? それはこうすると、画面表示が左右に移動しなくなるからです。表示をスクリーン上で表示されるべき場所、つまりモニターの表示可能領域内にきっちりとおさめるためです。

その上、同期信号の両側に ”防御時間” として約 30 クロック必要です。HGT1 と HGT2 で表わしています。一般的には HGT1 は HGT2 と等しくありませんが、しかし真っさらの状態から設定を行うならば、2 つを等しくして実験を始めたら良いでしょう (それは同期信号を中央に置くことになります)。

同期信号の置き違えの症状は、1 つの境界が極端に広くなって画像の他の側が画面の端から回り込んだり、白い端の線と”お化け画像”の帯になったり、画面の画像表示のずれとして現われます。垂直同期信号抜けは垂直保持が調節不備になっている TV の様に実際に縦スクロールします (事実、同じ現象が起こります)。

幸運ならば、モニターの同期信号の幅がその仕様書に記載されているでしょう。記載がない場合には、ここからが本当の黒魔術の始まり、、、

ここでは少し試行錯誤を行う必要があるでしょう。しかしほとんどの場合には、同期信号を約 3.5 から 4.0 マイクロ秒と仮定すれば安全です。

具体的には、HSP を 3.8 マイクロ秒にしましょう (この値は実験を始めるに当たっては悪い値ではありません)。

さて、先ほど 65Mhz をクロックに使いましたから、HSP を 247 クロック分と等しくすればよいことがわかります。(247 = 65x10**6 * 3.8 *10**(-6)) [メガ=10**6, マイクロ=10**(-6) である事を思い出して下さい]

垂直同期 :

前の絵に戻って、247 クロック分を絵の中でどのように置いたらいいでしょう ?

この例では、HR は 944 で HFL は 1176 です。この 2 つの例の差は 1176-944=232 < 247 です! 明らかにこの違いを調整しなければいけません。何ができるでしょうか ?

最初に 1176 は 1184 へ上げて、 944 は 936 へ下げてください。さて、違いは 1184-936= 248 になりました。う~ん、近づきましたね。

次は HSP を計算するのに 3.8 の代わりに、3.5 を使うようにすると、65*3.5=227 となります。かなり良くなりました。しかし 248 は 227 よりそれほど大きくありません。HR と SP の開始点の間と SP の終了点と HFL の間に 30 かそれぐらいあける必要があります。そして、それらは 8 の倍数にしなければなりません !我々はここで行き詰まってしまったのでしょうか ?

いいえ !こうしてみましょう、936% 8==0 です、また (936+32)% 8==0 です。しかし、936+32=968、968+227=1195、1195+32=1227 となります。ふむふむ、そんなに悪くはありません。しかし、8 の倍数にはなっていませんので、1232 に丸めて下さい。

しかし、同期信号を HR と HFL のちょうど真ん中に置くことが出来ないという潜在的な困難があります。幸いにも、1232-32=1200 も 8 の倍数である事と、(1232-32)-968=232 は 3.57 マイクロ秒 (232/65) という妥当な長さの同期信号に対応することが計算でわかります。

さらに、936/1232 は大体 0.76 つまり 76%, ですが、80% からそう遠くないのでまあ大丈夫でしょう。

その上、現在の水平フレーム長を使うなら、基本的にはモニターがその能力内で 52.7KHz(=65MHz/1232) において同期が取れるかどうか調べましょう。問題は無いでしょう。

経験則から、上記の 936*75%=702 を新しい垂直解像度にしましょう。702*1.05=737 が新しい垂直フレーム長になります。

画面再描画速度は 65Mhz/(737*1232)=71.6 Hz になります。それでも素晴らしいですね。

同様に垂直同期信号の配置を図解します：

```
|___ -- -- -- -- -- -- -- -- -- -- -- --
|_ _ _ _ _ _ _ _ _ _ _ _ _              |
|_______________________|_______________|_____
0                       VR              VFL     単位:1 垂直クロック
                        ^   ^       ^

                        |   |       |
                        |<->|<----->|
                         VGT    VSP
```

垂直表示データの終了時きっかりに同期信号を発信します。VGT は同期信号の垂直防御時間です。ほとんどのモニターは VGT 無し（防御時間無し）で快適に動作し、例題でも防御時間無しです。2,3 クロックの防御時間が必要なものもありますが、防御時間を入れて不都合が生じることは普通ありません。

例題に戻りましょう：フレーム長の定義から、垂直クロックは全＊水平＊フレームを辿る時間ですので、例題では 1232/65Mhz=18.95 マイクロ秒になります。

実験によれば垂直同期のパルス幅は 50 マイクロ秒から 300 マイクロ秒の範囲に設定すべきです。例題では 8 垂直クロック分、150 マイクロ秒を使用します (150 マイクロ秒 / 18.95 マイクロ秒　8)。

9 全体のまとめ

Xconfig ファイルのビデオモード（Video Modes）表は数行の数列で出来ていて、それぞれの行は X サーバーが使用する１つのモードの完全な仕様を表わしています。その項目は名称節、ドットクロック節、水平節、垂直節の 4 つの節に分けられます。

名称節は１項目で、行の残りで指定しているビデオモードの名称です。この名称は、Xconfig ファイルのグラフィックドライバ設定節の "Modes" 行で参照されます。現在の行が前の行と同じ名称ならば名称は省略出来ます。

ドットクロック節はビデオモード行の項目に DCF と呼んでいたドットクロックのみを書きます。次の節で作成する数字をドットクロックとしてこの項目に書きます。

水平節はそれぞれの水平線を画面の上でどのように作成するかを 4 項目で書きます。最初の項目は HR と呼んでいた映像を構成する輝く１本が何ドットであるかを書きます。 2 番目の項目はどのドットから水平同期信号が始まるかを示します。3 番目の項目はどのドットで水平同期信号が終わるかを示します。 4 番目の項目は全水平フレーム長 (HFL) を指定します。

垂直節も 4 項目で書きます。最初の項目は画面上に表示される走査線の数を書きます (VR)。 2 番目の項目は垂直同期信号が何番目の線から始まるかを書きます。3 番目の項目は垂直同期信号が何番目の線で終わるかを書きます。4 番目の項目は全垂直フレーム長を書きます。

例：

```
#名称節  モードクロック節  水平節 (1 から 4)  垂直節 (1 から 4)
#Modename    clock  horizontal timing  vertical timing

"752x564"     40    752 784  944 1088  564 567 569 611
              44.5  752 792  976 1240  564 567 570 600
```

（注意：標準的な X11R5 そのままでは 小数のドットクロックは使えません。）

Xconfig では、線上で輝いているドットの数、輝いているドットから同期信号の開始点までのドット数、同期信号の持続時間分のドット数と同期信号の終了点から後のドット数、これらを足し合わせると１本当たりのドット数が計算できます。水平方向のドット数は一律に 8 で割り切れる数値でなければいけません。

例：

水平方向の数値：800 864 1024 1088

この例は輝いているドット数 (800)、続いて同期開始点のドット (864)、その次が同期終了点のドット (1024)、次は水平線の最後のドット (1088) となります。

全ての水平方向の数字 (800、864、1024 と 1088) は 8 で割り切れる事にもう一度注意をして下さい。これは垂直方向の数字には当てはまりません。

画面の上から下までの線の数がフレームを構成します。フレームの基本的な時間調整は線で行ないます。線の多くは画像を表示するために使われます。最後の発光する線を表示した後で、数本分遅延が挿入され、その後垂直同期信号が生成されます。そして同期信号は数本分だけ持続し、フレームの最後に同期信号の後で必要な遅延が生成されます。この動作モードを規定する数値は、次の例のように入力します。

例：

垂直方向の数値：600 603 609 630

この例はディスプレイに 600 本の線が表示され、603 番目の線から垂直同期が始まり 609 番目で終わる事、そして全部で 630 本の線を使用することを表わしています。

垂直方向の数値は 8 で割り切れる値でなくても構いません。

例題に戻りましょう。上記によって、Xconfig へ書き込む必要な全ての値は次の様です:

```
{$<$} name {$>$}   DCF     HR   SH1 SH2   HFL   VR  SV1 SV2 VFL
```

ここで SH1 は水平同期信号の開始クロックで、SH2 はその終了クロックになり、同様に SV1 は垂直同期信号の開始クロックで、SV2 はその終了クロックです。

```
#名称節  モードクロック節  水平節 (1 から 4)  垂直節 (1 から 4)  オプション
#name     clock    horizontal timing   vertical timing    flag
936x702   65       936 968 1200 1232   702 702 710 737
```

特別なオプションは必要無く、これはノンインターレースモードで動作します。これで完了しました。

10 質疑応答

Q. ここにある例題は標準的な画面サイズではありませんが、使ってもよいですか ?

A. 勿論です。なにがなんでも 640x480, 800x600 とか 1024x768 を使わなければならない理由はありません。XFree86 ドライバはハードウェアを自由に設定できるようになっています。一般に正しい設定は 2 、3 分でできるようになります。正しい設定を当てるのに重要な物は高い再描画速度と妥当な表示領域です。高解像度だけを追求して涙目を誘うちらつきに代償を払うことのないようにしてください。

Q. 65Mhz のドットクロックと 55Khz の HSF で得られる解像度はこれ * だけ * ですか ?

A. 絶対そんなことはありません ! 一般的な手順に従って作業を行い、本当に好みの設定にたどり着くため、少しの試行の繰り返しをお勧めします。 実験はとっても楽しいものです。 ほとんどの設定はビデオ表示が乱れてちゃんと設定できないかもしれませんが、マルチシンクモニターを (帯域幅よりもはるかに高い周波数のクロックを出力させようとしない限りは — モニターの説明書に書かれている最大解像度に近い値を守っている場合は) 痛めることはありません。固定周波数のモニターには注意してください。周波数調整は固定周波数のモニターを痛める * かも * しれません。

Q. 二つの標準的な解像度を記載してますね。 Xconfig の中では色々標準的な解像度がありますが、時間調節の数値を調整する場合のポイントを教えてください。

A. 勿論教えましょう ! 現在の Xconfig にある ”標準的な” 640x480 を例題に取り上げます。この場合、動作周波数が 25MHz 、フレーム長が 800 と 525 の時に約 59.5Hz の画面再描画周波数が使えます。 悪く無いでしょ ? しかし一般的に多くの SVGA ボードの動作周波数は 28MHz を使えます。

28MHz で 640x480 を使う場合、以前に説明した手順に従えば、フレーム長は 812 と 505 に設定できます。ここで再描画周波数は 68MHz まで引き上げられ、標準値よりかなり向上しました。

Q. ところでインターレースとノンインターレースについてはどうですか ?

A. 固定ドットクロックでは、ノンインターレースのちらつきよりインターレースのちらつきの方が酷いので、インターレースは市場から消えつつあります。フリッカーが増える代わりに、遅いドットクロックで高解像度を実現することができます。DCF が十分速い場合には (例えば 90MHz とかそれ以上) インターレースでもちらつきは発生しませんが、現在のスピードではインターレースモニターは X に向いていません。

Q. 今までの議論をまとめてもらえますか ?

A. 要約して言えば :

1. 任意の固定の動作周波数では、最高の解像度に高めれば再描画速度を下げなければならず、したがってちらつきが増えるでしょう。
2. 高解像度を希望しモニターがそれをサポートするときは、それに合うドットクロック、つまり DCF を供給する SVGA カードを手に入れて下さい。ドットクロックは高いに越したことはありません。

11 二つの例題

ここで別の仮定をします。:

あるアダプターカードは 40 MHz のクロックで動作します。あるディスプレイは 30 KHz から 37 KHz の範囲の水平同期信号で動作します。この時の 1 つの線当たりのドットの最小数は 40,000,000/37,000 = 1,081.081 より、約 1,081 ドット/線となります。

この 1 本当たりのドット数をこれからの計算で使います。だから、ディスプレイ上のそれぞれの線は最小で 1081 ドットあります。この数値を 8 で割り切れる数に丸めて 1088 に切り上げます。ここでは水平同期信号長に 3.8 マイクロ秒を使うことを仮定しています。 3.8 マイクロ秒の信号を作るのに何ドット必要なのかを見つける必要があります。先ず 1 ドットが何マイクロ秒なのかを探しましょう。毎秒 40,000,000 ドットなので、1/40,000,000 がドット当たりの秒数です。つまり

1/40,000,000 = .000000025 = ドット当たり .025 マイクロ秒

となります。

この時 3.8 マイクロ秒の同期信号を構成するドット数 D は

3.8 マイクロ秒 = D ドット x .025 マイクロ秒/ドット

つまり

D ドット = (3.8 マイクロ秒) / (.025 マイクロ秒/ドット) = 152 ドットです。

線当たり 1088 ドットの時、152 ドットを同期信号に使用して、800 ドットが輝いて見えているわけです。 (152 は 8 で割り切れることに注意して下さい。割り切れない場合は割り切れるように切り上げて下さい。) ここでディスプレイのために必要な同期信号の前後の時間を計算する仕事が残っています。

経験則により、防御時間の約 30 単位が必要です。詳しく述べると、32 ドットを割り当てると都合が良いのは、他の全ての数値が 8 で割り切れているからです。表示領域に 800 ドット、同期信号に 152 ドット使った結果、1088 - (800 + 152) = 136 となり、136 ドットが 2 つの時間の間を分けています。136 の半分の 68 ドットが輝いているドットと同期信号の間に置かれて、68 ドットが同期信号の後に置かれています。Xconfig ファイルの水平方向の数値は次のように書きます。

800 (800+68) (800+68+152) (800+68+152+68)

つまり

800 868 1020 1088 となります。

今度は垂直方向の数値を計算しましょう。当初から垂直方向の数値にドットだとかドット当たりのマイクロ秒とかの単位を使わずに線の数を使っていた事を思い出して下さい。よって、1 本の線を表示するのにどれくらいの時間が掛かるかを計算する必要があります。この計算はそれぞれの線が 1088 ドットでそれぞれのドットは.025 マイクロ秒なので簡単です。従って、それぞれの線は

(1088 ドット/線) x (.025 マイクロ秒/ドット) = 27.2 マイクロ秒/線と計算できます。

線当たり 800 ドット表示可能を選択したので、アスペクト比が 4 対 3 になるように線の数を選択しましょう。この時、800 は 4 x 200 なので縦方向の線の数は 3 x 200 = 600 になるでしょう。目標解像度は 800x600 です。

垂直同期信号の範囲は 50 から 300 マイクロ秒であることが分かっています。代表的な同期信号として 150 マイクロ秒を選択した場合に、150 マイクロ秒を 27.2 マイクロ秒で割れば 1 同期信号当たりの本数が計算できます。

(150 マイクロ秒/同期信号) / (27.2 マイクロ秒/本) = 5.51 本/同期信号

これを 6 本/同期信号に切り上げることで（切り捨てはしないでください）、垂直同期信号幅が求まりました。

フレームの線の（輝いている線と周囲にある輝いていない線を加えた）合計本数を（"ビデオ調整..." から）、表示可能な線の本数より 5% 増しの合計本数として予測できます。よって、線の合計本数は次のように計算します。

(600 本) x (1.05) = 630 (フレーム当たりの合計本数)

さてここで、輝いている線 (以下、輝線) の終わりとフレームの終わりの間に同期信号を置かないといけません。合計で 630 本、輝線が 600 本、同期信号に 6 本あるので、

630 - 600 - 6 = 24 本残ります。

幾つかのディスプレイは輝線の直後に同期信号を発行しても問題ありませんが、他の幾つかのディスプレイでは、輝線の最後と同期信号の最初の間に 1 本か 2 本分間隔を置くべきでしょう。安全のために余裕を持って、輝線の最後と同期信号の最初の間に 3 本分追加します。残りの本数を同期信号の後ろに追加します。垂直同期調整の数値は次のようになります。

600 (600+3) (600+3+6) (600+3+6+21)

つまり

600 603 609 630

とにかく作業を行なう前に、線 1 本当たり 27.2 マイクロ秒の時にフレーム当たり 630 本をディスプレイが表示できることを確認しなければいけません。設定しようとしている構成で毎秒何フレーム表示するか計算し、ディスプレイのマニュアルの垂直同期速度に関する記載と見比べることによって確認できます。線 1 本当たり 27.2 マイクロ秒の時にフレーム当たり 630 本表示するためには、630 x 27.2 = 17,136 マイクロ秒/フレーム 必要です。17,136 マイクロ秒/フレーム は 0.017136 秒/フレーム つまり 1/0.017136 フレーム/秒 です。

1 / (0.017136 秒/フレーム) = 58.4 フレーム/秒

マニュアルに垂直同期速度が 58.4 Hz と書かれていた場合、若しくは 58.4 Hz がディスプレイの垂直同期可能な周波数帯域にある場合は問題ありません。ディスプレイがこの速度を扱えない場合は、全ての数字を比例させてフレーム当たりの線の本数を変更しなければいけません。

さて Xconfig のテスト構成を作成するために解像度とクロックを合わせて水平と垂直の調整数値を組合せましょう。定義する行は次のようになります。

"800x600" 40 800 868 1020 1088 600 603 609 630

これで XFree86 の構成が出来上がりました。ここでいくつか仮定したことがとてもふさわしくなくてうまく動作しない場合があるかもしれませんが、ほとんどの場合少なくとも安定した表示ができるようになります。さあ、快適にするためにちょっとした実験をやってみましょう。

実際の計算

私のアダプターカードが 40 MHz の水晶発信子を搭載しているので、周波数を 40 MHz から始めま

しょう。私のディスプレイの最高水平同期周波数が 37 KHz ならば、線当りの最小ドットクロックは 40,000,000/37,000 = 1081 になります。私のディスプレイの垂直同期周波数は 50 Hz から 90 Hz までの範囲です。

私のディスプレイのマニュアルによれば最大の水平同期信号は 3.92 マイクロ秒です。ドット当り 0.025 マイクロ秒の場合、信号は

(3.92 マイクロ秒) / (.025 マイクロ秒/ドット) = 156.8 ドットになります。

これを最も近い 8 の倍数に切り上げて、160 ドットにしてください。

またマニュアルによれば最後に輝いたドットと同期信号の始まりの間に最低 0.67 マイクロ秒間隔を置くべきと解説しています。 40 MHz の時 0.67 マイクロ秒で出せるドット数 D （40 MHz は .025 マイクロ秒/ドットであることを覚えていてください）は

D ドット = (0.67 マイクロ秒) / (.025 マイクロ秒/ドット) = 26.8 ドットになります。

26.8 は 8 の倍数ではないので、32 ドットに切り上げて下さい。

私のディスプレイのマニュアルによれば同期信号の後の間隔を 3.56 マイクロ秒かそれ以上置くべきであると書いています。3.56 マイクロ秒でのドット数 D は

D ドット = (3.56 マイクロ秒) / (.025 マイクロ秒/ドット) = 142.4 ドット

8 の倍数になるように 142.4 を 144 に切り上げて下さい。

水平線のために 800 の輝くドット、輝くドットと同期信号の間に 32 ドット、同期信号のための 152 ドット、同期信号の後に 144 ドットがあります。

800 + 32 + 160 + 144 = 1136

1 本の線に 1136 ドット使っています。これは前に計算した 1088 より大きいですが、1088 は線を作るドット数の *最小の* 数であることを思い出して下さい。ですから線当り 1136 ドットは第一歩としてはいいところでしょう。

Xconfig に記入する数値は、次のように

”800x?” 40 800 (800+32) (800+32+160) (800+32+160+144)...

または

”800x?” 40 800 832 992 1136... としましょう。

.025 マイクロ秒/ドットで 1136 ドットの線を引くということは 1136 x .025 = 28.4 マイクロ秒掛かることを表しています。

水平解像度に 800 ドット/本 を決めたので、垂直解像度に 600 本/フレームを決めましょう。

私のディスプレイのマニュアルによると垂直同期信号は最低 64 マイクロ秒必要と書いています。 64 マイクロ秒で線を引くと本数は

(64 マイクロ秒/同期信号) / (28.4 マイクロ秒/本) = 2.25 本/同期信号となります。

垂直同期信号のために 2.25 を 3 本に切り上げて下さい。

マニュアルによれば最後に表示した線と同期信号の開始点の間に最低 318 マイクロ秒、同期信号の終了後の遅延を最低 630 マイクロ秒の間隔を置くように書かれています。それぞれの区間で何本、線が引けるかを次に計算します。

(318 マイクロ秒) / (28.4 マイクロ秒/本) = 11.20 本

(630 マイクロ秒) / (28.4 マイクロ秒/本) = 22.18 本

同期信号の前の間隔を 12 本、後の間隔を 23 本に丸めます。これで垂直調整の数値が決定しました。

600 (600+12) (600+12+3) (600+12+3+23)

すなわち

600 612 615 638 となります。

このフレーム周波数がディスプレイの能力に合っていることを確かめるため、フレーム周波数が、28.4 マイクロ秒/本の時 638 本/フレームは 18,119 マイクロ秒/ フレームであり、55.19 フレーム/秒になることを確認してください。私のディスプレイは 50 Hz から 90 Hz までの全ての周波数を扱えるので問題ありません。

解像度、クロック、水平方向と垂直方向の調整数値を Xconfig ファイルのビデオモードの行に次の様に記入して下さい。

”800x600” 40 800 832 992 1136 600 612 615 638

これが私が試した最初のビデオモードです。ちらつきがひどすぎて、とても満足できるものではないので止めました。この設定の調整数値を上げたり下げたりする例を次から説明します。ちらつきと解像度の間の折り合いをつけて最終的に”784x614” に設定しました。

ほとんど全てのクロックの周波数は 40 MHz であることに注意してください。実験を通してより高い周波数は私のアダプターカードの能力を超えており、より低い周波数は私の望む解像度を与えるものではないということが分かりました。

例：私が試した数値です：

```
# 次の行は動くけれど中心から右に寄っています。
"752x564"     40  752  784  944 1088  564 567 569 611
#           44.5  752 792  976 1240  564 567 570 600
#
# この行は前の行であった問題を修正したものです。
#"752x564"     40  752  816  976 1088  564 567 569 611
#
# 垂直方向の表示の大きさを増やしてみましょう、これは動作します。
#"752x614"     40  752  816  976 1088  614 617 619 661
#
# 水平方向の表示の大きさを増やしてみましょう、これは動作します。
#"784x564"     40  784  816  976 1088  564 567 569 611
#
# 次も動きますが、中心から右に寄っています。
#"784x614"     40  784  816  976 1088  614 617 619 661
#
# 直前の例の中心にきていない問題を修正したものです。
"784x614"     40  784  848 1008 1088  614 617 619 661
#
# 表示の大きさを増やしてみましょう
# 次も動きますが、中心からわずかに左に寄っています。
#"800x614"     40  800  864 1024 1088  614 617 619 661
#
# 直前の設定の中心にきていない問題を修正したものです。
"800x614"     40  800  864 1024 1104  614 617 619 661
#
# 表示の大きさを増やしてみましょう、これは動作します。
"816x614"     40  816  880 1040 1120  614 617 619 661
#
# 表示の大きさを増やしてみましょう、これは動作します。
"800x620"     40  800  864 1024 1104  620 623 625 661
#
# 表示の大きさを増やしてみましょう、これは動作します。
"816x620"     40  816  880 1040 1120  620 623 625 661
#
# 表示の大きさを増やしてみましょう、これは動作します。
"832x630"     40  832  896 1056 1136  630 633 635 661
#
# 表示の大きさを変えました、動きますが激しくちらつきます。
"848x618"     40  848  912 1072 1152  618 621 623 661
```

12 表示の問題修正

さあ、X の構成定義の数値を手に入れました。Xconfig に その数値をテスト中のコメントをつけて書きました。X を立ちあげ、ホットキーで新しいモードに切り替えてみました ... しかし画像が変です。こういう場合、どうすればいいのでしょう ?一般的な問題と解決策をここに挙げます。

同期信号の調整を変えて画像を *移動* してみてください。フレーム長を変えて画像を *拡大縮小* して下さい (相対的な位置をそのままに同期信号を移動する必要があります。そうしないと拡大縮小と同時に画像の移動も起こってしまいます)。もう少し個別の対処方法を次に示します。:

水平と垂直の表示位置は独立しています。これは画像を水平に移動させても垂直位置には影響がなく、また逆も同じということです。ところが拡大縮小では必ずしもそうではありません。水平方向の大きさを変えても垂直方向の大きさには何も影響を与えず、逆も同じですが、縦横両方の変更量の合計は限定されています。特に、縦横に大き過ぎた場合はより高いドットクロックに変更して修正する必要があるでしょう。使用可能な解像度を引き上げることになるため、これはめったに問題とはなりません。

画像が左か右にずれている場合

これを修正するために水平同期信号を移動しましょう。それは水平同期信号の開始端と終了端を定義している水平調整部分の 2 つの数値の真ん中の数値を (8 の倍数ずつ) 増減させて行いましょう。

画像が左にずれている (右の縁の部分が大きすぎて、右へ画像を移動したい) 時は、数値を増やしてください。画像が右にずれている (左の縁の部分が大きすぎて、左へ画像を移動したい) 時は、同期信号を減らしてください。

画像が上下に動いている場合

これを修正するために垂直同期信号を移動してください。それは垂直同期信号の開始端と終了端を定義している垂直調整部分の 2 つの数値の真ん中の数値を増減させて行いましょう。

画像が上にずれている (下の縁の部分が大きすぎて、下に画像を移動したい) 時は、数値を減らしてください。画像が下にずれている (上の縁の部分が大きすぎて、上へ画像を移動したい) 時は、数値を増やしてください。

画像が水平方向に広すぎる (狭すぎる) 場合

これを修正するためには水平フレーム長を増やし (減らし) てください。それは最初の調整部分の 4 番目の数値を変えて行います。画像が移動するのを回避するため、同期信号 (2 番目と 3 番目の数値) もその半分だけ移動し、同じ相対位置を保存しておいてください。

画像が垂直方向に膨らんでいる (痩せている) 場合

これを修正するためには垂直フレーム長を減らし (増やし) てください。それは 2 番目の調整部分の 4 番目の数値を変えて行います。画像が移動するのを回避するため、同期信号 (2 番目と 3 番目の数値で) もその半分だけ移動し、同じ相対位置を保存しておいてください。

これらの技術の組合せでも取れない他の歪みは多分もっと基本的な部分が違っている、例えば計算違いとかモニターで使えない高いドットクロックを使っているような場合があります。

最後に、フレーム長のどちらかの数値を増やせば再描画速度が低下してしまうこと、またその逆も言えることを覚えておいて下さい。

```
$XConsortium: VidModes.sgml,v 1.3 95/01/27 16:14:33 kaleb Exp $
Generated from XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/VidModes.sgml,v 3.6 1995/07/21
```

このファイルは xc/programs/Xserver/hw/xfree86/doc/sgml/VidModes.sgml,v 3.6 1995/07/21 14:40:58 を、岡本 一幸 (Kazuyuki Okamoto <*ikko-@pacific.rim.or.jp*> (`mailto:ikko-@pacific.rim.or.jp`)) が XFree86 3.1.2 を日本でインストールする人向けに和訳したものです。ここがおかしいとか、ここはこうしたほうがいいといったご意見がありましたら、電子メールでお知らせ下さい。原文の著作権は XFree86 プロジェクト社にあります。この和訳の著作権は XFree86 プロジェクト社と岡本 一幸にありますが、この和訳の不具合は私に、電子メールで送って下さい。

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/Japanese/VidModes.sgml,v 3.3 1995/08/31 07:49:5
```

[EOF]