XFree86 Video Timings HOWTO

RaymondEric Steven

Thyrsus Enterprises (http://www.tuxedo.org/~esr/)

esr@thyrsus.com

도운이

김현종

polsci@hitel.net

$Date: 2001/12/16 23:44:59 $

본 문서는 원칙적으로 구버전의 **XFree86** 을 대상으로 하며 이제는 시대에 뒤진 문서가 되버렸습니다**. XFree86 4.0.1** 버전부터는 **X** 설정화일의 **Modes Section** 부분에 원하는 해상도를 적어주기만 하면 자동으로 최적의 모드라인으로 설정됩니다**.**

이 문서에서는 비디오 카드와 모니터의 조합에 알맞도록 XFree86 의 모드라인(mode line)을 설정하는 요령을 설명합니다. 현재의 XFree86 배포본에는 모드라인 값을 가장 표준적인 조합으로 설정하는데 필요한 쓸만한 기능들이 포함되어 있습니다; 본 문서는 주로 초고성능 모니터내지는 희귀한 하드웨어라 수동으로 모드라인(mode line)을 설정해야만 하는 분들께 유용할 것입니다. 또한, kvideogen 을 써서 모드라인을 생성한다거나 표준모드가 자신의 모니터에 딱 들어맞지 않아서 xvidtune 으로 표준모드를 미세하게 수정해야 할 때에도 도움이 될 것입니다.

고친 과정

고침 6.12001-10-29고친이 esr
Note that VESA modes top out at 1920x1440.
고침 6.02001-08-09고친이 esr
Clearer explanation of DDC and EDID. This HOWTO is now basically obsolete.

고침 5.02000-08-22고친이 esr
First DocBook version.

1. 면책

본 문서의 내용에 대한 실행 여부는 전적으로 당신 책임입니다. 제조사가 보증하는 제원을 넘어서는 범위에서 모니터를 작동시킬 경우 모니터 및 당신 자신에게 피해를 입힐수도 있습니다. 자세한 주의사항은 모니터의 오버 부분을 읽어보십시요. 모니터의 오버로 인해 당신 내지 당신 모니터에 발생할 수 있는 피해는 전적으로 당신의 책임이며 필자는 책임질 수 없습니다.

본 HOWTO 문서의 최신 버전은 리눅스 문서 프로젝트 (http://www.linuxdoc.org/LDP) 웹 페이지에서 찾으실 수 있습니다.

감상이나 비평, 개선해야 할 점 등은 <esr@snark.thyrsus.com> 으로 보내주십시요. 청컨대 당신의 특수한 모니터 문제를 해결해달라는 메일은 보내지 말아주십시요. 그런 질문은 당신을 실망시킬 뿐이고 저로서도 시간낭비일 뿐입니다. 이 주제에 대해 제가 아는 모든 바는 본 문서속에 다 담겨있습니다.

2. 본 HOWTO 문서가 시대에 뒤처지게 된 이유

XFree86 버전 4.0.0 이후부터는 더이상 당신이 모드라인을 생성시킬 필요가 없어졌습니다. 대신, 이제 모드라인들은 당신이 XFree86 설정화일 속의 Screen 섹션내의 Modes 부분에 적어준 해상도 및 모니터에 대한 X 서버의 EDID 질의로 얻어진 모니터의 능력치에 기초해, 서버 기동시 서버의 의해 자동으로 계산됩니다.

화면 해상도와 색상수를 바꾸려면 그저 Display 섹션에 적어주면 됩니다. 한 예로, 다음은 필자의 랩탑 컴퓨터의 X 설정화일의 Screen 부분입니다.

```
Section "Screen"
Identifier    "Screen0"
        Device       "ATI Rage Mobility"
        Monitor      "Monitor0"
DefaultDepth 16

Subsection "Display"
         Depth       16
                Modes        "1024x768"
EndSubsection

EndSection
```

보통의 경우, 그저 Mode 항목에다가 숫자만 바꾸어 주면 됩니다. 그러면 X 가 나머지 일을 다 처리할 것입니다. 만일 불가능한 해상도를 적는다면, 모니터측이 지원가능하다고 알려온 EDID 데이터 중 그와 가장 가까운 값으로 설정될 것입니다.

따라서, 이제부터 설명될 내용은 다음 경우에만 유용할 것입니다. (a) EDID 능력이 없는 구형모니터인 경우, (b) 그래픽 카드의 드라이버가 모니터에 대한 EDID 질의 기능을 지원하지 않는 경우, (c) 옛날 버전의 XFree86 을 사용하는 경우(이 경우는 XFree86 을 업그레이드함으로써 해결하십시요), (d) 모니터와 그래픽 카드의 조합이 1920 x 1440 을 초과하는 해상도를 지원할 경우(1920 x 1440 해상도는 XFree86 의 표준 모드라인값으로 지원할수 있는 최대 해상도입니다)

3. 소개

XFree86 서버에서는 사용자가 직접 자신의 비디오 시스템을 설정할 수 있으므로 잘만 세팅하면 하드웨어의 능력을 최대한으로 이끌어낼 수 있습니다. 본 문서는 자신의 비디오카드와 모니터에 최적화된 타이밍 수치를 계산하는 방법을 설명합니다.

우선 비디오 시스템을 일단 작동시키는 방법을 설명한 후, 그 기본세팅을 바탕으로 입맛에 딱 맞는 최적화된 세팅이 될때까지 계속 실험해 나가는 방법을 택하겠습니다.

현재 X 윈도우를 특정 모드에서 잘 쓰고 있기는 한데 화면에 약간의 문제가 있는 경우라면(특히, 미리 설정된 VESA 모드하에서 안정적인 화면이 나오기는 하는데 화면이 상하좌우로 치우쳤다든지 너무 크거나 작은 경우), 바로 이미지와 관련된 문제해결 절로 건너뛰시기 바랍니다. 그 절에서는 타이밍 수치를 미세조정하여 원하는 효과를 얻는 방법을 설명합니다.

X 서버 설치후 처음 동작시켜보니 첫 화면이 엉망이라고해서 바로 모드 튜닝방법을 몽땅 공부할 필요는 없습니다; 모드라인 값들은 대부분 산업표준의 값들입니다. 따라서, 화면이 엉망이 되었다면 아마도 재수없게도 당신의 하드웨어가 그 디폴트 값들 중 하나와 궁합이 잘 안맞았기 때문일 것입니다. 튜닝방법을 몽땅 공부하는 대신, CTRL-ALT-(키패드 키의)+ 키를 써서 당신이 설치한 모든 모드들을 하나씩 시험해 보십시요. 만일 그 중 어떤 모드들이 잘 동작한다면 이제 다른 모드들은 모두 주석처리 해두고 640x480 모드만 남겨 잘 동작하는지 확인하세요. 잘 동작한다면 이제 800x600 이나 1024x768 등과 같이 당신 모니터가 처리할 수 있는 범위의 주파수의 모드들을 하나하나 체크해 나가십시요.

4. 모드라인을 자동으로 계산해주는 도구들

당신 모니터가 1996 년 이후에 제작된 것이라면 아마도 EDID (http://www.vesa.org/dload/summary/sumeedidrar1.htm) 사양을 지원할 것입니다. EDID 능력을 가진 모니터(마이크로소프트식 용어로는 플러그앤 플레이(PnP) 모니터라고 부르기도 합니다)들은 자신의 능력치를 컴퓨터에게 알려줄 수 있습니다

XFree86 4.0 에 포함된 드라이버중 많은 것들은 DDC,VESA Display Data Channel facility (http://www.vesa.org/dload/summary/sumeddcv1.htm)를 지원합니다. DDC 가능한 그래픽 카드 모듈은 모니터에게 그 EDID 능력표를 건네줄 것을 요청하고, 그 건네받은 데이터를 기초로 자신을 설정합니다. 따라서 XFree86 4.0 이상과 근래 생산된 모니터를 가지고 있다면 대부분의 경우 따로 모드라인을 설정할 필요가 없을 것입니다.

만일 그래픽 카드 모듈은 DDC 능력이 없는데 모니터는 EDID 가능하다면 read-edid 프로그램을 써서 모니터에게 그 데이터를 얻어낸 후 거기서 모드라인을 계산하는 방법이 가능합니다. http://altern.org/vii/programs/linux/read-edid/ 를 보십시요(역주: 이곳 (http://web.onetel.net.uk/~elephant/john/programs/linux/read-edid/)으로 옮겨진 듯 하군요).

XFree86 3.2 부터는 **XF86Setup** 프로그램이 같이 들어있습니다. 이 프로그램을 쓰면 사용자가 대화식으로 쉽게 모니터 모드를 설정할 수 있으므로 손수 비디오 타이밍 숫자를 계산하느라 골머리를 썩힐 필요가 없습니다. 따라서 사실상 대부분의 경우에 있어 기본 모니터 모드를 당신이 직접 계산할 필요는 없는 것입니다. 하지만 불행하게도 **XF86Setup** 에는 몇가지 한계가 있습니다; 이 프로그램은 1280x1024 까지의 표준 비디오 모드만을 설정할 수 있습니다. 만일 1600x1200 이상의 초고성능 모니터를 가지고 있다면 그때는 모니터의 기본모드를 손수 계산해야만 합니다.

KVideoGen (http://paranoia.rulez.org/videogen/) 이라는 KDE 의 프로그램은 모니터와 카드의 기본수치를 바탕으로 모드라인을 계산해줍니다. 필자는 시험삼아 KVideoGen 을 써서 모드라인 값들을 얻어봤습니다만, 얻어진 값들을 실제로 적용시켜 본 적은 없습니다. 이 프로그램의 수평 수직 'refresh rate' 파라메터들은 아래에서 설명할 동기 주파수 HSF 및 VSF 에 해당합니다. 'horizontal sync pulse' 수치는 마이크로세컨드 단위의 동기펄스 폭 HSP 에 해당하는 듯 합니다(KVideoGen 은 'front porch' HGT1 과 'back porch' HGT2 값을 고정된 값으로 가정합니다). 만일 당신이 'horizontal sync pulse' 수치를 모른다면 디폴트 값을 쓰는 것이 안전할 것입니다.

XFree86 용 모드라인을 계산해주는 곳으로는 또 여기 (http://zaph.com/Modeline)가 있습니다. 파이썬 스크립트를 다운받거나 제공되는 CGI 폼을 사용하면 됩니다.

최근 버전의 XFree86 에 들어있는 **xvidtune** 이라는 프로그램을 쓰면 편리하게 모니터 모드를 테스트하고 튜닝할수 있습니다. 이 프로그램은 첫 화면에서 잘못 사용했을 때의 결과에 대해 무시무시한 경고를 하고있습니다만 이 문서를 충분히 숙독해서 xvidtune 의 각 항목의 숫자들이 의미하는 바를 확실히 알게된다면 자신있게 xvidtune 프로그램을 효과적으로 사용할 수 있을 것입니다.

xvidtune 을 사용하면 X 설정 화일을 수정하거나 X 서버를 다시 부팅하지 않고서도 바로 새로운 모드를 시험해볼 수 있습니다. 또다른 방법으로는, X 윈도우 내에서 Xconfig 화일에 정의된 각각의 모드간을 핫키로 전환할 수도 있습니다(자세한 것은 XFree86.man 을 보십시요). 이 기능을 써서 화면이 먹통이 된 상황을 빠져나오세요! 새로운 모드를 테스트할 때는 그 모드에 따로 적당한 이름을 붙여서 핫키 리스트 끝에 추가시키십시요. 이미 잘 동작하는 모드를 디폴트로 놓아둔 후, 테스트 모드가 실패할 경우 핫키를 써서 디폴드 모드로 돌아오면 됩니다.

이 문서의 끝부분에는 'modeplot' 스크립트가 적혀있습니다. 이를 이용하면 사용가능한 모드의 그래프를 그려볼 수 있습니다. 이 스크립트가 직접 모드라인을 생성해 주지는 않습니다만 모드라인의 원리를 이해하는데 도움이 될 것입니다.

5. 비디오 디스플레이의 동작원리

Xconfig 화일의 각 항목에 어떤 수치를 써넣어야 하는지를 알고싶다면 먼저 디스플레이의 동작원리를 이해해야 합니다. 이 수치들은 XFree86 서버가 가장 저수준에서 디스플레이를 컨트롤할 때 사용됩니다.

디스플레이는 래스터 도트들을 연속적으로 나열하여 화면을 만듭니다. 도트들은 왼쪽에서 오른쪽으로 배열되면서 선을 형성하고, 이러한 선들이 위에서부터 아래로 배열됨으로써 화면을 형성합니다. 화면의 반대편에는 각각 삼원색중 한가지를 담당하는 세개의 전차총들이 있어 전자빔들을 방출합니다. 디스플레이내의 전자빔들이 도트에 부딪치는 순간, 빛이 나게 됩니다. 각 도트들을 같은 시간 만큼씩 때리기 위해 빔들은 일정한 패턴으로 화면을 가로지릅니다. 이를 래스터라고 합니다.

윗 문단에서 "도트를 연속으로 나열하여" 라고 설명했는데, 사실 이 "래스터 도트"는 실제의 인(phosphor) 도트를 뜻하는 것은 아닙니다. 모니터 브라운관의 안쪽 표면에 도포되어있는 인(phosphor) 도트는 래스터 도트보다 훨씬 작습니다. -- 반드시 작아야만 합니다. 만일 그렇지 않다면 화면에 심각한 모아레 효과가 나타날 것입니다. 래스터 도트란 사실상 아날로그 구동 신호의 샘플링들로서, 그 신호의 산과 마루가 미세한 간격으로 규칙적으로 발생함으로써 도트들의 격자가 생성되어 화면을 이룹니다.

빔의 패턴은 스크린의 왼쪽 꼭대기부터 시작해서 직선으로 오른쪽으로 이동합니다. 이 직선은 아주 약간 오른쪽이 낮게 경사져 있습니다만 그냥 수평이라 생각해도 큰 상관이 없을 정도입니다. 빔이 스크린의 오른쪽 끝에 도달하면 한줄 아래 왼쪽으로 돌아가서 다시 오른쪽으로 주사됩니다. 이 과정이 화면 맨 밑의 오른쪽 끝에 도달할때까지 반복되며, 최하단 오른쪽 끝에 도달하면 빔은 다시 왼쪽 최상단으로 돌아갑니다(이때 시간이 약간 걸립니다). 이러한 패턴이 계속 되풀이되는 것입니다.

이러한 방식의 한가지 변용이 있는데 이는 인터레이싱 방식이라는 것입니다; 인터레이싱 방식에서는 반 프레임 동안 한줄씩 건너뛰며 라인이 주사되고 다음 반 프레임 동안 나머지 빈줄들이 한 라인씩 채워져 나갑니다.

화면의 왼쪽 최상단부터 시작하는 것을 프레임의 시작이라고 합니다. 한 프레임은 빔들이 오른쪽 최하단에 도달했다가 다시 원래의 처음 위치인 왼쪽 최상단으로 돌아온 순간 끝납니다. 즉, 한 프레임이란 화면의 상단에서 하단까지 빠짐없이 주사된 모든 라인들로 구성되는 것입니다.

만일 전자빔들이 프레임을 횡단하는 동안 줄곧 켜져있다면 화면상의 모든 도트들이 빛을 낼 것입니다. 그렇게 된다면 화면 가장자리의 검은 테두리부분도 없을 것입니다. 이 경우, 가장자리 부분의 화면에는 왜곡이 생기게 됩니다. 디스플레이의 가장자리 부분에서는 빔을 컨트롤하는 것이 상당히 어렵기 때문입니다. 이러한 왜곡을 줄이기 위해 화면의 가장자리 부분에서는 빔을 꺼서 도트들이 발광하지 않게 만듭니다. 이런 이유로 디스플레이의 가시화면부분은 모니터 표면 전 영역보다 작은 것입니다.

이해하셔야할 또다른 중요한 점은 빔이 가시화면의 끝부분에 도착했다면 그다음 어떻게 되는가 하는 것입니다. 계속 진행했다면 디스플레이의 가장자리를 비추게 되었을 나머지 시간은 빔이 왼쪽 끝으로 돌아가는 시간으로 사용됩니다. 마찬가지로 화면 상하단의 검은 가장자리를 비출 시간이 빔이 오른쪽 최하단에서 왼쪽 최상단으로 돌아오는데에 사용됩니다.

(역주 : 나중에 모드라인 수치를 보면 아시겠지만 한 라인에 할당되는 시간의 일부만이 화면을 만드는데 이용됩니다. 빔이 화면 끝에 도달하면 나머지 잔여 시간은 빔을 원위치로 되돌리는데 사용됩니다)

비디오 카드는 전자빔을 (원하는 색상으로) 발생시키라는 신호를 디스플레이 장치에 보내고 디스플레이 장치는 이를 받아 각 도트를 주사하여 화상을 만듭니다. 비디오 카드는 또한 수평 동기 펄스라는 신호를 보내어 빔이 오른쪽 끝에서 한줄아래 왼쪽 처음으로 돌아오게 합니다.

수평동기 펄스는 각 라인의 끝에서마다 한개씩 발생합니다. 또, 수직동기 펄스를 보내면 빔이 왼쪽 최상단으로 돌아오게 됩니다. 수직동기 펄스는 각 프레임의 끝부분에서 발생합니다.

디스플레이의 전자빔들의 위치가 안정적으로 확보되기 위해서는 수평, 수직동기펄스의 앞뒤로 짧으나마 약간의 시간이 필요합니다. 빔들이 안정되지 못하면 화면이 불안정해집니다.

더 자세한 내용은 TV 와 모니터의 편향 시스템 (http://fribble.cie.rpi.edu/~repairfaq/REPAIR/F_deflfaq.html)을 참고하십시요.

다음 절에서는 이런 기초적인 사항들에 대한 정의, 공식, 예제 등을 다루겠습니다.

6. 디스플레이와 어댑터에 대한 기본적인 사항

Xconfig 의 각 항목을 해킹하기 전에 다음의 기본적인 사항들을 알고 있어야만 합니다.

- 모니터의 수평, 수직 동기주파수 옵션
- 모니터의 대역폭(bandwidth)
- 비디오 어댑터의 구동 클럭 주파수, 즉 "도트 클럭"

6.1. 모니터의 동기주파수

수평 동기주파수는 모니터가 일초에 수평 스캔라인을 몇번이나 주사할수 있는가를 뜻합니다; 이 수치야 말로 모니터에 있어 가장 중요한 수치입니다. 수직 동기 주파수는 일초에 빔이 수직으로 몇번이나 횡단할수 있는가입니다.

동기주파수들은 보통 모니터 설명서의 제원부분에 나와있습니다. 수평 동기 주파수는 주로 KHz, 수직 동기주파수는 주로 Hz 단위(1 초간의 반복횟수)로 표기합니다. 수직 동기주파수는 보통 50-150 Hz 이내이고 수평 동기주파수는 보통 31-135 KHz 이내입니다.

멀티싱크 모니터의 경우 이 동기주파수들은 고정된 수치가 아닌 어떤 범위로 표시됩니다. 반면 일부 모니터들, 특히 저가형 모니터들에서는 이 주파수들은 몇개의 고정된 값들을 가질 뿐입니다. 이 값들 역시 설정을 바꾸는 것이 가능하긴 합니다만 모니터 자체가 이에 맞춰 설정되었기 때문에 선택의 범위는 멀티싱크 모니터에 비해 크게 줄어드니다. 좋은 해상도를 원한다면 가장 높은 수평 수직주파수 쌍을 선택하세요 --- 주의할 점은 주파수가 고정된 모니터에서는 원래 디자인된 제원보다 높은 스피드로 모니터 클럭을 구동시킬 경우 모니터가 쉽게 손상될 수도 있다는 것입니다.

본 문서의 초기버전에서는 상당히 용감하게 멀티싱크 모니터를 오버시켰었습니다. 더 나은 성능을 얻기 위해 정격 수직 동기주파수의 최대값보다도 더 높은 값으로 모니터를 구동시켰던 것입니다. 이에 대해 보다 조심할 필요가 있음을 지적받았습니다. 이 문제는 뒤의 모니터의 오버 부분에서 다루겠습니다.

6.2. 모니터의 비디오 대역폭

모니터의 비디오 대역폭은 모니터 설명서의 제원 부분에 나와있습니다. 만약 나와있지 않다면 그 모니터의 정격 최고 해상도 부분을 보세요. 경험칙입니다만 다음은 해상도를 대략적인 대역폭으로 환산한 것입니다(동시에 이는 사용가능한 도트클럭 값 각각의 대략적인 상한값이기도 합니다).

```
640x480              25 MHz
800x600              36 MHz
1024x768             65 MHz
1024x768 interlaced  45 MHz
1280x 1024          110 MHz
1600x1200           185 MHz
```

이 수치들은 무슨 마술이 아닙니다. XFree86 의 표준모드 데이터 베이스에 적혀있는 각 해상도별 가장 낮은 도트클럭일 뿐입니다(단, 마지막 줄은 필자가 삽입한 것입니다). 모니터의 실제 대역폭은 아마도 위의 최소값보다는 더 큰 값일 것입니다. 따라서 위의 값보다 몇 MHz 더 높은 도트클럭을 사용한다해서 그리 겁먹을 필요는 없습니다.

또한 대역폭은 65 MHz 이하의 도트클럭에서는 거의 문제가 되지 않습니다. 대부분의 고해상도 모니터의 경우, SVGA 카드는 모니터의 비디오 대역폭의 최고 한계값에 턱없이 못미치는 범위에서 사용되기 때문입니다. 예를 들어보지요.(역주: 도트클럭은 비디오 카드가 발생시키는 주파수이고 비디오 대역폭은 모니터가 받아들일수 있는 주파수임을 상기하세요)

```
모니터                                      비디오 대역폭
------------------------------------------------------------
NEC 4D                                            75 MHz
Nano 907a                                         50 MHz
Nano 9080i                                        60 MHz
Mitsubishi HL6615                                 110 MHz
Mitsubishi Diamond Scan                           100 MHz
IDEK MF-5117                                      65 MHz
IOCOMM Thinksync-17 CM-7126                       136 MHz
HP D1188A                                         100 MHz
Philips SC-17AS                                   110 Mhz
Swan SW617                                        85 MHz
Viewsonic 21PS                                    185 MHz
PanaSync/Pro P21                                  220 MHz
```

설사 저가형 모니터라 할지라도 대부분의 경우 표준해상도에서는 심각한 대역폭 부족을 일으키지는 않습니다. NEC Multsync II 모니터는 좋은 예입니다 --- 이 모니터의 제원을 보면 800x600 화면조차 지원하지 못하는 것으로 나와있습니다. 겨우 800x560 밖에 지원하지 못합니다. 이와같은 저해상도에서는 높은 도트클럭이나 높은 대역폭이 필요없는 것입니다. 아마도 당신이 선택할수 있는 것은 32Mhz 나 36 Mhz 정도일텐데 둘다 모니터의 정격 비디오 대역폭인 30 Mhz 에서 크게 벗어난 값은 아닙니다.

이 두개의 구동 주파수에서는 어쩌면 화질이 기대한만큼 선명하지 않을 수도 있습니다만 분명히 참고 쓸만한 정도는 될 것입니다. 물론 NEC Multisync II 가 36 MHz 보다 높은 비디오 대역폭을 지원했더라면 더 좋았을 것입니다. 하지만 이미지의 왜곡을 일으킬 정도로 화질이 나

쁜 것이 아니라면 텍스트 작업등의 일반적인 용도로 쓰는데는 별 문제가 없을 것입니다(만일 화질이 그 정도로 나쁘다면 당신의 눈이 제일 먼저 알려주겠지요).

6.3. 카드의 도트클럭

보통, 비디오 카드 설명서의 제원 부분에는 비디오 카드의 최대 (일초에 화면에 출력할수 있는 픽셀수의 최대값)이 나와있습니다.

당신이 이 수치를 모른다면 X 서버가 찾아낼수 있습니다. 최근 버전의 X 서버들은 모두 --probeonly 옵션이 있어서, 이를 쓰면 X 를 재가동하거나 비디오 모드를 바꾸지 않고도 필요한 정보를 출력해 줍니다.

만일 -probeonly 옵션이 없다해도 실망할 필요는 없습니다. 최악의 경우 X 서버로 인해 모니터가 먹통이 된다고 하더라도 X 서버는 클럭 및 그외 다른 정보들을 표준 에러로 출력할 수 있습니다. 이 표준 에러 출력을 화일로 출력하도록 해두면 모니터가 먹통이 되어 재부팅을 할수밖에 없더라도 알아낸 정보는 화일에 그대로 저장되게 됩니다.

probe 결과나 시작 메시지는 다음 예와 비슷합니다.

XFree86 를 사용하는 경우 :

```
    Xconfig: /usr/X11R6/lib/X11/Xconfig
    (**) stands for supplied, (--) stands for probed/default values
    (**) Mouse: type: MouseMan, device: /dev/ttyS1, baudrate: 9600
    Warning: The directory "/usr/andrew/X11fonts" does not exist.
             Entry deleted from font path.
    (**) Font-
Path set to "/usr/lib/X11/fonts/misc/,/usr/lib/X11/fonts/75dpi/"
    (--) S3: card type: 386/486 localbus
    (--) S3: chipset:   924
                        ---
        칩셋 -- 이 부분은 정확한 칩의 종류를 나타내고 있습니다.
                86C911 칩의 초기 버전이군요.

    (--) S3: chipset driver: s3_generic
    (--) S3: videoram:  1024k
                        -----
             이 부분은 비디오 보드상의 프레임 버퍼 램의 크기를 나타냅니다.

    (**) S3: clocks:  25.00  28.00  40.00   3.00  50.00  77.00  36.00  45.00
    (**) S3: clocks:   0.00   0.00  79.00  31.00  94.00  65.00  75.00  71.00
                      ----------------------------------------------------
--
                          MHz 단위로 나타낸 사용가능한 구동 주파수들입니다.

    (--) S3: Maximum allowed dot-clock: 110MHz
                                        ------
                                    대역폭(Bandwidth)입니다.
    (**) S3: Mode "1024x768": mode clock =  79.000, clock used =  79.000
```

```
(--) S3: Virtual resolution set to 1024x768
(--) S3: Using a banksize of 64k, line width of 1024
(--) S3: Pixmap cache:
(--) S3: Using 2 128-pixel 4 64-pixel and 8 32-pixel slots
(--) S3: Using 8 pages of 768x255 for font caching
```

SGCS 나 X/inside X 를 사용하는 경우 :

```
WGA: 86C911 (mem: 1024k clocks: 25 28 40 3 50 77 36 45 0 0 79 31 94 65 75 71)
---  ------       -----         ------------------------------------------
--
 |      |             |                    MHz 단위의 사용가능한 구동 주파수들
 |      |             +-- 비디오 보드의 프레임 버퍼 램의 크기
 |      +-- 칩의 종류
 +-- 서버의 종류
```

주의: 되도록 컴퓨터에 다른 부하가 없는 상태에서 실행하세요. 왜냐하면 X 는 일종의 응용 프로그램이기 때문에 X 의 타이밍 루프가 디스크 작업등과 동시에 이루어지거나 한다면 위의 수치들 자체가 부정확해질 수 있기 때문입니다. 수 회에 걸쳐 실행해 본 후 수치들이 안정된 값을 갖는지를 확인하십시요; 만약 값이 계속 들쭉날쭉하다면 안정된 값을 가질 때까지 다른 프로세스들은 죽여버리세요. 마우스 데몬 프로세스가 있다면 이를 죽여놓으세요. 특히나 마우스 데몬은 당신을 헷갈리게 만들기 쉽습니다(리눅스 사용자의 경우는 gpm, SVr4 사용자들은 mousemgr 입니다).

클럭 측정이 부정확해지는 것을 피하려면 위에서 측정한 수치를 당신의 Xconfig 화일의 Clocks 속성 부분에 써주어야 합니다 --- 이는 타이밍 루프를 억제시키고 X 가 정확한 클럭값을 사용하게 해줍니다. 위의 예에서 나온 데이터를 사용하면 다음과 같습니다.

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

작업부하의 변동이 심한 시스템의 경우, 알수없는 이유로 X 가 뜨지 않는 경우가 가끔 있는데 이렇게 클럭수치를 적어주면 이를 방지할 수 있습니다. 간혹, 시스템 부하로 인해서 X 가 잘못된 타이밍을 잡은 후, 설정 데이터베이스에서 그 타이밍에 해당하는 도트클럭을 찾지 못거나 잘못된 도트클럭을 찾는 수가 있기 때문입니다.

6.4. 이 기본 수치들은 무엇을 제어하는가

궁극적으로는 비디오 카드의 도트클럭과 모니터의 동기주파수 대역이 사용가능한 해상도를 결정짓습니다. 하지만 하드웨어의 잠재능력을 얼마만큼 이끌어낼수 있는가는 드라이버에 달려 있습니다. 고성능 하드웨어가 구비되었다 하더라도 그에 상응하는 좋은 디바이스 드라이버가 없다면 돈낭비에 불과합니다. 반면 융통성 좋은 디바이스 드라이버를 쓰면 상대적으로 떨어지는 성능의 하드웨어라도 원래의 능력이상으로 성능을 이끌어낼 수 있습니다. 이것이 XFree86 의 디자인 철학입니다.

당신이 사용할 도트클럭을 모니터의 비디오 대역폭에 매칭시켜야 합니다. 이 부분에서 상당한 선택의 여지가 있습니다. -- 어떤 모니터들은 정격 대역폭보다 30 % 이상 오버해서 구동시

킬 수도 있습니다. 그 위험성 여부는 모니터의 정격 수직 동기주파수를 얼마만큼 오버시킬수 있는가에 달려 있습니다. 이에 대해서는 뒤에서 보다 자세히 설명하겠습니다.

대역폭에 대해 알게되면 보다 현명한 설정을 할 수 있습니다. 이를 통해 디스플레이의 화질(특히, 세밀한 부분의 선명도)을 향상시킬수 있는 것입니다.

7. 기본적인 제원을 해석하기

이 절에서는 위에 언급된 제원(specification)의 의미 및 그외 알아야 할 것들을 설명합니다. 먼저 몇가지 정의를 설명하겠습니다. 괄호속은 앞으로 계산에 사용할 변수입니다.

수평 동기주파수(horizontal sync frequency, HSF)

초당 수평 스캔의 수(위에서 설명했음).

수직 동기주파수(vertical sync frequency, VSF)

초당 수직 스캔의 수(위에서 설명했음). 화면갱신율의 상한값이 되므로 중요합니다.

도트클럭(dot clock, DCF)

더욱 엄밀히 말하면 '구동 클럭 주파수(driving clock frequency)' 입니다. 비디오 카드상의 크리스탈이나 VCO 가 방출할 수 있는 초당 최대 도트수를 말합니다.

비디오 대역폭(video bandwidth, VB)

모니터에 입력했을 때 화면이 생성될 수 있는 주파수값의 최대치입니다. 인터레이스드 모드와 같이 비디오 카드가 온오프 패턴을 교차시키는 경우는 그 동작 주파수의 최소값은 DCF 의 절반값이 되므로, 이론적으로는 대역폭 역시 DCF/2 부터 시작해도 됩니다. 하지만 정교한 화면 디스플레이를 위해서는 대역폭 값을 DCF 값의 최고치보다 많이 낮추는 것은 바람직하지 않으며 되도록 높이 잡아 쓰는 것이 좋습니다.

프레임 길이 (frame length, HFL, VFL)

수평 프레임 길이(HFL)는 모니터의 전자총이 수평줄 한개를 스캔하는데 필요한 도트클럭 틱(tick)의 수를 말합니다. 여기에는 좌우측 경계에서의 비발광부분까지 포함됩니다. 수직 프레임길이(VFL)은 전체 화면에서의 스캔라인의 수를 말하는데 역시 상하단 경계부의 비발광 부분까지 포함된 것입니다.

화면 재생율(screen refresh rate, RR)

화면이 초당 몇번이나 되풀이되어 그려지는지를 뜻합니다. 다른 말로 프레임 레이트(frame rate)라 불리우기도 합니다. 주파수가 높을수록 플리커(깜박임)가 줄어들기 때문에 이 값이 높을수록 좋습니다. 60 Hz 정도면 괜찮고, VESA 표준인 72 Hz 정도면 더욱 좋습니다. 계산법은 다음과 같습니다.

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

> 주의할 것은 분모 부분의 두수의 곱은 모니터의 가시해상도와는 다른 것으로서 보통 그보다 큰값이 됩니다. 이에 대한 자세한 것은 아래에서 다루겠습니다.

인터레이스드 모드에서 표기되는 화면 재생률(예를 들면 87Hz interaced)은 실제로는 반프레임에 대한 재생률입니다. 전체 화면은 마치 다소간의 플리커가 낀 보통의 디스플레이처럼 보입니다만 각각의 라인은 같은 주파수의 넌인터레이스드 모드 때의 절반의 빈도로 갱신되는 것입니다.

계산의 편의를 위해 인터레이스된 화면은 풀프레임(refresh)레이트로 환산하여 계산하겠습니다. 즉, 87 Hz interlaced 는 43.5 Hz 입니다. 인터레이스드 모드의 화질은 동일한 풀프레임 레이트에서의 넌인터레이스드 모드의 화질보다 좋습니다만, 동일한 반프레임 레이트에서의 넌인터레이스드 모드의 화질보다는 분명히 떨어집니다(역주: 87 Hz 인터레이스드 모드는 43.5 Hz 넌인터레이스드 모드보다는 화질이 좋고 87 Hz 넌인터레이스드 모드보다는 화질이 나쁘다는 말입니다).

7.1. 대역폭에 대해

모니터 제조업체들은 대역폭이 높다는 점을 선전하기를 좋아하는데 그 이유는 대역폭이 화면의 intensity 와 컬러변화의 선명도를 좌우하기 때문입니다. 높은 대역폭은 보다 세밀한 표현을 가능하게 합니다.

모니터는 전자적인 신호을 사용하여 화면을 생성합니다. 그러한 신호는 디지탈 형태에서 아날로그 신호로 변환되면서 웨이브 형태가 되어 모니터에 입력됩니다. 이 신호는 각각의 일정 주파수의 신호들이 중첩되어 하나의 형태를 이룬 것입니다. 각각의 주파수신호 대부분은 MHz 단위의 일정 주파수를 가집니다. 예를 들면 20 MHz, 40 MHz, 70 MHz 들의 신호가 하나로 결합되는 것입니다. 모니터의 비디오 대역폭이란 사실상 모니터가 왜곡없이 처리할수 있는 가장 높은 주파수의 아날로그 신호인 것입니다.

비디오 대역폭(video bandwidth)은 사용가능한 가장 높은 도트클럭을 결정하는데 있어 상당히 중요한 역할을 합니다.

7.2. 동기주파수와 리프레쉬율

화면상의 각각의 수평스캔라인은 단지 한 프레임 스캔 길이 중 보이는 부분에 해당될 뿐입니다. 한순간을 딱 끊어서 본다고 치면 스크린상에는 사실상 한점만이 발광하고 있을 뿐입니다. 다만 그것이 충분한 빠른 리프레쉬 율로 재생되므로 당신 눈의 잔상효과로 인해 자신이 이미지 전체를 보고 있다고 "느끼는" 것입니다. 다음 그림이 도움이 되겠군요.

```
 __________________________
|                          |
|->->->->->->->->->->->    |
|                        ) |
|<-----<-----<-----<---    |
|                          |
|                          |
|                          |
```

```
|                         |
|_________________________|
```

수평 동기주파수란 모니터의 전자빔이 위의 그림처럼 좌우로 1초에 몇번이나 수평횡단하는가 입니다.

```
 _________________________
|         ^               |
|       ^ |               |
|       | v               |
|       ^ |               |
|       | |               |
|       ^ |               |
|       | v               |
|       ^ |               |
|_______|_v_______________|
```

수직 동기주파수란 모니터의 전자빔이 위의 그림처럼 상하로 1초에 몇번이나 수직횡단하는가 입니다.

실제의 래스터 스캔은 매우 조밀한 지그재그 패턴이 됩니다. 빔은 좌우로 움직이며 또 상하로도 이동합니다.

이제 우리는 도트 클럭과 프레임 길이가 리프레쉬 레이트와 어떤 관련이 있는지 알수 있습니다. 정의에 의하면 1 Hz 는 초당 한 회의 반복을 의미합니다. 수평 프레임길이를 HFL 이라 하고 수직프레임 길이를 VFL 이라 하면 전체 화면은 (HFL * VFL) 틱을 가지는 것입니다. 정의에 의해 비디오 카드는 초당 DCF 만큼의 틱을 모니터로 입력하므로 모니터의 전자빔은 화면의 왼쪽에서 오른쪽으로 주사하고 화면 하단까지 갔다가 다시 꼭대기로 돌아오는 동작들을 1 초에 DCF / (HFL * VFL) 만큼씩 반복하는 것입니다. 이것이 바로 화면의 리프레쉬 레이트(화면 갱신율)입니다. 왜냐하면 리프레쉬 레이트란 화면이 일초에 몇번이나 갱신되는가를 말하는 것이기 때문입니다.

이 개념을 잘 이해해야 플리커(화면깜박임)를 최대한 줄인 해상도를 설정할 수 있습니다.

말보다는 그림이 더 이해하기 쉽겠군요.

```
        RR                                      VB
         |   min HSF                     max HSF |
         |    |             R1        R2  |      |
max VSF -+----|------------/----------/---|------+----- max VSF
         |    |:::::::::::/::::::::::/:::::\     |
         |    \::::::::::/::::::::::/:::::::\    |
         |     |::::::::/::::::::::/:::::::::|   |
         |     |:::::::/::::::::::/::::::::::\   |
         |     \::::::/::::::::::/::::::::::::\  |
         |      \::::/::::::::::/::::::::::::::| |
         |       |::/::::::::::/:::::::::::::::| |
         |        \/::::::::::/:::::::::::::::::\|
         |        /\:::::::::/:::::::::::::::::::|
         |       /  \:::::::/::::::::::::::::::::|\
```

```
           |      /     |:::::/::::::::::::::::::::::| |
           |     /      \::::/:::::::::::::::::::::::| \
min VSF -+----/-------\--/-----------------------|--\--- min VSF
           |   /          \/                       |   \
           +--/----------/\-----------------------+----\- DCF
             R1         R2  \                      |     \
                             min HSF               |    max HSF
                                                   VB
```

이것은 일반적인 모니터 모드 다이어그램입니다. X 축은 클럭레이트(DCF)를, Y 축은 리프레쉬율(RR)을 나타냅니다. 다이어그램의 채워진 영역은 모니터의 능력을 나타냅니다: 영역내의 각 점은 가능한 비디오 모드를 의미합니다.(역주: 텍스트 포맷이라서 그림이 제대로 표현되지 않고 있습니다. DCF = RR * HFL * VFL 공식대로 RR 을 수직축, DCF 를 수평축으로 잡아 그림을 직접 그려보면 쉽게 이해가 갈 것입니다).

'R1'과 'R2'로 이름붙인 선은 특정 해상도(예를들면 640x480)를 나타냅니다; 이는 하나의 해상도는 여라가지의 도트클럭-리프레쉬율의 조합들로 만들어낼 수 있음을 나타냅니다. R2 라인은 R1 라인때보다 높은 해상도임을 나타내고 있습니다.

위아래 영역을 둘러싼 수평라인들은 경계선으로서 수직동기주파수의 한계값을 뜻하고 있습니다. 비디오 대역폭은 클럭레이트의 상한값이 되므로 모니터의 능력을 나타낸 영역의 오른쪽 끝 수직라인으로 표현되었습니다.

모니터의 성능을 그려보기 절에서는 각각의 모니터를 대상으로 이런 다이어그램을 그려주는 프로그램(X 상에서 그려보면 더욱 보기 좋습니다)을 소개하고 한가지 더 재미있는 것을 설명합니다; 수평 동기 주파수의 최대값들로부터 경계값을 유도하는 방법입니다.

8. 시스템 설정시의 타협

위에서 유도한 식을 다른 관점에서 해석해 볼수도 있습니다.

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

이 식에서 도트클럭은 고정된 값입니다. 초당 도트수를 가지고 리플레쉬 레이트와 수평 수직 해상도를 결정할 수 있습니다. 이중 한가지 값이 증가하면 다른 값들의 곱은 감소하게 되는 관계입니다.

하지만 주목해야 할 것은 리프레쉬 레이트는 절대로 모니터의 최대 수직 동기주파수 보다 클 수 없다는 것입니다. 따라서 어떤 모니터든 특정 도트클럭이 주어지면 프레임 길이들의 곱의 최소값이 존재하게 되며 이 값을 더이상 낮추는 것은 불가능한 것입니다.

세팅할 때는 다음 사항을 기억하십시요: RR은 너무 낮게 잡으면 이에 따른 스크린 플리커로 인해 얼굴을 찌푸리게 될것입니다. 60 Hz 이상이 되도록 하십시요. VESA 의 인체공학적 표준은 72 Hz 입니다. 참고로 미국에서의 형광등의 플리커레이트는 120 Hz 입니다(50 Hz 의 교류전력을 사용하는 유럽등지에서는 100 Hz 가 됩니다); 민감한 사람은 60 Hz 나 72 Hz 에서도 플리커를 느낍니다. 그런 경우라면 더 높은 값으로 설정하세요.

확실히 플리커는 눈에 심한 피로감을 줍니다만, 인간의 눈이라는 것이 적응성이 있고 사람에 따라 플리커에 민감한 정도도 천차만별입니다. 만일 당신이 모니터를 90 % 의 시야각으로 검은 바탕색과 이에 잘 대비되는 색상을 사용하면서 중간 이하 정도의 intensity 로 본다면 45 Hz 같은 낮은 주파수에서도 눈이 편안할 *수도* 있습니다.

다음과 같이 시험해 보면 됩니다; 순수한 백색 배경색의 Xterm 을 띄우고 foreground 를 검은색으로 설정합니다. ,**xterm -bg white -fg black** 명령을 내리면 됩니다. 그 다음 이 Xterm 창을 최대한 확대하여 가시화면 전체를 채웁니다. 이제 모니터의 intensity 를 최대값의 3/4 로 맞춘 후 모니터를 외면합니다. 모니터를 옆눈으로 흘겨봅니다(보다 민감한 말초시신경을 사용하자는 것입니다). 플리커를 느끼지 못하거나 견딜만 하다면 그 리프레쉬 레이트가 당신에게 알맞은 것입니다. 불편하다면 보다 높은 리프레쉬 레이트로 설정하세요. 그냥 보통으로 모니터를 볼때는 괜찮은 듯 해도 있는듯 없는듯한 플리커가 서서히 눈의 피로를 가중시켜 종내에는 두통을 일으킬 수 있습니다.

인터레이스드 모드에서의 플리커의 정도는 설정된 수직해상도나 화면상에 표현된 내용등에도 영향을 받습니다. 일단 실험을 해 보세요. 그래도 85Hz 하프 프레임레이트보다 낮게 잡아 쓰고싶지는 않을겁니다.

자, 이제 사용가능한 최소한의 리프레쉬 레이트를 잡았다고 가정합시다. HFL 과 VFL 을 택하는데 있어 다소간의 재량의 여지가 있습니다.

9. 필요한 메모리의 양

프레임 버퍼 램이 얼마만큼인가에 따라 컬러 혹은 흑백 디스플레이에서의 해상도가 제한됩니다. 대부분의 경우, 그레이 음영을 가지지 않는 흑백 디스플레이의 경우에는 메모리 양은 큰 문제가 되지 않습니다.

256 컬러의 디스플레이에서는 도트 하나를 표현하는 데에는 한 바이트의 비디오 메모리가 필요합니다. 256 컬러의 디스플레이에 있어 한 바이트는 그 도트 한개의 적, 녹, 청의 배합비에 관한 정보를 담습니다. 필요한 메모리의 총량을 구하기 위해서는 라인당 가시도트의 수와 가시라인의 수를 곱하면 됩니다. 1024x768 해상도의 디스플레이에서는 1024 x 768 = 786432 가 화면상의 가시도트의 갯수입니다. 또, 한 도트당 한 바이트가 필요하므로 786432 바이트 만큼의 비디오 메모리가 필요한 것입니다.

따라서, 일반적으로 올림해서 (HR * VR)/1024 Kbyte 의 비디오 램이 필요합니다(이 예제에서는 정확히 768K 가 됩니다). 필요한 정확한 양보다 더 많은 양의 메모리를 가지고 있다면 그 여분만큼은 가상 스크린 panning 에 쓸수 있습니다.

하지만 비디오 카드에 512k 만 있다면 1024x768 해샹도를 쓸수 없습니다. 설사 좋은 모니터를 가지고 있다고 하더라도 충분한 비디오 램이 없다면 모니터의 잠재능력을 다 이용하지 못하는 것입니다. 반면, SVGA 카드가 1 M 를 가지고 있어도 모니터가 기껏해야 800X600 밖에 지원하지 못한다면 마찬가지로 고해상도는 불가능한 것입니다(가능한 방법이 있기는 합니다. 비월주사(interaced mode)방식 편을 보세요).

필요한 만큼의 메모리보다 더 많은 양을 가지고 있다고 해서 걱정할 필요는 없습니다; XFree86 은 그 여분의 메모리를 사용하여 가상화면을 스크롤할 수 있습니다(Xconfig 화일 문서의 가상화면에 대한 파라메터 부분을 보세요). 또한, 512k byte의 메모리를 가진 카드라도

실제로는 512,000 byte 만 가진 것은 아니라는 점에 주의하십시요. 512 x 1024= 524288 byte 를 가진 것입니다.

만일 S3 카드를 가지고 있고 X/inside 를 쓴다면, 16 컬러(픽셀당 4bit)를 쓸 경우 Xconfig 에서 depth 4 로 설정해서 카드가 다룰수 있는 해상도의 2 배의 해상도를 쓸 수 있습니다. 예를 들어 S3 카드들은 보통 1024x768x256 을 사용합니다. 이를 depth 4 를 써서 1280x1024x16 로 사용할 수 있는 것입니다.

10. 프레임 크기의 계산

경고: 이 방법은 멀티싱크 모니터를 위한 것입니다. 이 방법을 주파수가 고정된 모니터에 적용해도 대체로 잘됩니다만 이에 대한 어떠한 보장도 하지 않습니다!

사용가능한 가장 높은 HSF 로 DCF 를 나누어 수평 프레임 길이를 얻는 것부터 시작합니다.

예를 들어 65 MHz 도트클럭의 Sigma Legend SVGA 를 가지고 있고 모니터는 수평 스캔주파수 55KHz 라 합시다. 이때 (DCF/HSF) 의 값은 1181 입니다(65MHz = 65000kHz : 65000/55=1181).

이제 첫번째 마술을 부려봅시다. 이 수치를 8 의 정수배중 가장 가까운 숫자로 반올림해야 합니다. 이는 SVGA 와 S3 카드에 사용되는 VGA 하드웨어 컨트롤러에 관련된 문제입니다; 컨트롤러는 왼쪽으로 3 bit 쉬프트된 8 bit 레지스터 하나를 사용하므로 결국 11 bit 입니다. ATI 8514/A 같은 일부 카드들은 굳이 8 의 배수일 필요는 없다고 하는데, 필자는 그에 대해서는 잘 모르며 독자의 교정을 바랍니다. 좌우지간 1181 을 반올림하면 사용가능한 수평 프레임길이는 1176 이 됩니다.

이 수치(DCF/HSF 의 값을 가까운 8의 배수로 반올림한 값)는 사용가능한 HFL 값 중 최소값입니다. 동기펄스를 세팅해서 HSF 값을 낮추면 HFL 을 더 길게 만들수도 있습니다(HFL 이 커지면 스크린의 수평도트수가 더 많아집니다). 하지만 이 댓가로 플리커가 더 증가하게 됩니다.

경험적으로 볼때, 수평프레임의 길이의 80 % 가 수평해상도, 즉 수평 스캔라인의 가시부분으로 사용될 수 있습니다(역주: 그러니까 수평 프레임길이가 모두 화면에 비추이는 것이 아니라 그 일부만이 수평해상도로 쓰이는 것입니다). 간단히 말하자면 수평스캔라인에는 경계부분과 스윕백 타임이 포함되는 것입니다(스윕백 타임이란 빔이 오른쪽 끝에 도달한 다음 다시 왼쪽 끝으로 돌아오는데 걸리는 시간을 말합니다). 이 예에서는 1176 의 80 % 인 940 틱이 가시부분이 됩니다.

이제 일반적인 스크린 비율(aspect ratio)인 가로세로 4:3 를 얻기 위해서는 수직 해상도를 방금 계산한 수평해상도의 3/4 값으로 설정해야 합니다. 이 예제에서는 705 틱이 됩니다. 여기에 1.05 를 곱하면 VFL 값 740 틱을 얻습니다(역주: 왜 꼭 1.05 인지는 잘 모르겠군요).

4:3 이라는 비율이 무슨 마법의 숫자는 아닙니다; 화면을 최대한 이용하기 위해 이와 다른 가로세로 비율을 사용하겠다고 해서 말리는 사람은 없습니다. 프레임 높이와 프레임 길이를 말할 때는 편의상 대각선비를 씁니다. 대각선 길이에 0.8 을 곱하면 가로길이를 얻고 0.6 을 곱하면 세로길이를 얻습니다.

이제, HFL = 1176 이고 VFL= 740 입니다. 65 MHz 를 두수의 곱으로 나누면 74.6 Hz 라는 눈에 아주 좋은 리프레쉬율을 얻습니다. 훌륭하지요! 이는 VESA 표준보다도 좋은 값입니다! 당신은 944x705 의 해상도를 쓰게 되고 이는 보통 많이들 쓰는 800x600 보다도 높은 값입니다. 전혀 문제될 것이 없습니다.(역주 : 944 x 705 가 아니라 940 x 705 라고 생각됩니다. 어쨌든 저자는 이하의 계산에서 계속 944 를 쓰고 있군요)

더 나아가 리프레쉬 레이트를 76Hz 까지 향상시킬수도 있습니다. 대개의 모니터들은 정격 수평 동기주파수보다 2 khz 가량 높은 주파수에서도 잘 동작하며, 또한 VFL 을 다소(즉, 위의 예에서는 940 의 75% 보다 낮은 값) 낮추어도 잘 동작하기 때문입니다. 하지만 이런 "오버"를 하시기 전에 가능하다면 모니터의 전자총이 수직으로 76 Hz 까지 동기(sync)할수 있는지 확인하시기 바랍니다(한 예로 많이들 쓰는 NEC 4D 의 경우는 불가능합니다. 이 모니터는 오직 최대 75 Hz 의 VSF 만이 가능합니다. 이에 대한 자세한 사항은 모니터의 오버 부분을 보세요).

지금까지의 내용은 간단한 산술과 래스터 디스플레이의 기초적인 사항뿐이었습니다. 마술은 어디에도 없었습니다!

11. 마술, 그리고 동기 펄스

좋습니다. 이제 당신이 선택한 도트클럭에 대한 HFL/VFL 수치를 계산했고 가능한 리프레쉬율을 알아내었으며, 이 해상도를 쓸수 있을만한 비디오램을 가지고 있는지도 확인했습니다. 이제 정말 마술을 부려 봅시다 -- 언제 어디에 동기펄스를 위치시켜야 하는지를 알아야 합니다.

모니터의 수평, 수직 스캔 주파수를 컨트롤하는 것은 사실 동기펄스입니다. 모니터 설명서에 나오는 HSF, VSF 값은 대표값(nominal value)으로서 동기주파수의 대략적인 최고값입니다. 비디오 카드에서 출력되는 동기펄스는 얼마나 빨리 빔을 주사해야 하는지를 모니터에 지시합니다.

아까의 두개의 그림을 기억하시나요? 한 프레임을 래스터스캐닝하는데 드는 시간중 단지 일부만이 가시화면(즉, 화면해상도)을 만드는데 사용됩니다.

11.1. 수평 동기

앞서의 정의에 의하면, 수평스캔라인이 주사되는데는 HFL 만큼의 틱이 필요합니다. 가시화면을 생성하는 틱들의 숫자를 따로 HR(horizontal screen resolution) 이라 합시다. 이제 정의에 의해 HR < HFL 임이 분명합니다. 정확한 이해를 위해 아래와 같이 같은 출발점에서 시작되는 그림을 그려보았습니다.

```
|___ __ __ __ __ __ __ __ __ __ __ __ __
|_ _ _ _ _ _ _ _ _ _ _ _                |
|_______________________|_______________|_____
0                       ^               ^     단위: ticks
                        |   ^       ^   |
                        HR  |       |  HFL
                        |   |<----->|   |
                        |<->|  HSP  |<->|
```

```
HGT1          HGT2
```

이제 우리는 HSP 길이 만큼의 동기펄스를 위 그림처럼 배치해야 합니다. 즉, 화면을 생성하는 데이터의 클럭 틱의 끝부분과 전체 프레임의 클럭 틱의 끝 사이의 어딘가에 HSP 길이의 동기 펄스를 배치해야 하는 것입니다. 왜냐구요? 이렇게 해야만 화면이 왼쪽이나 오른쪽으로 치우치지 않고 모니터의 가시화면 영역내의 제 위치에 제대로 자리잡히게 되는 것입니다.

이제, 동기펄스의 앞뒤로 약 30 틱의 "완충시간(guard time)"을 잡아줍시다. 이는 HGT1 과 HGT2 로 표기합니다. 일반적인 설정에서는 HGT1 와 HGT2 가 같은 길이일 필요는 없습니다만 지금은 시험단계이니 임시로 일단 같은 값으로 해두고 실험을 시작합시다(즉, 동기펄스를 중앙에 위치시킨 것입니다)

동기펄스를 잘못 위치시키면 이미지가 한쪽으로 치우쳐 버립니다. 한쪽 경계부분은 지나치게 넓고 반대쪽에서는 이미지가 스크린 끝에서 중첩되어버려서 백색으로 빛나는 경계부인 소위 "ghost image" 를 만들어 버립니다. 수직 동기주파수를 잘못 위치시키면 마치 V-hold 가 잘못 맞춰진 TV 처럼 화상이 말려 올라가 버립니다(사실상 TV 와 동일한 현상입니다).

운이 좋다면 모니터의 동기펄스의 폭이 제원부분에 나와있을 겁니다. 나와있지 않다면 이제부터 진짜 마술을 부려 봅시다...

이 부분에서는 시행착오를 감수해야 합니다. 하지만 대부분의 경우, 싱크펄스로 약 3.5-4.0 usec (microsecond) 정도의 길이를 가정하면 안전합니다.

정확히 하기위해서 HSP 를 3.8 usec 가정합시다(실험용으로 그리 나쁜 값은 아닙니다).

이제 위에서의 65 MHz 의 클럭 타이밍을 사용한다고 가정하면, HSP 가 247 (= 65 * 10^6 * 3.8 * 10^-6) 클럭 틱이 되는 것을 알수 있습니다(M = 10^6, micro = 10^-6] 입니다).

일부 제조업체에서는 수평 프레임 파라메터들을 도트 길이 대신 타이밍으로 표기합니다. 따라서 다음과 같은 용어들이 사용될 수도 있습니다.

active time (HAT)

HR 에 해당하는 용어지만 마이크로세컨드(microseconds)단위입니다. HAT * DCF = HR.

blanking time (HBT)

(HFL - HR) 에 해당하는 용어로서 마이크로세컨드 단위입니다. (HBT * DCF) = (HFL - HR).

front porch (HFP)

HGT1 과 같은 의미입니다.

sync time

HSP 와 같은 의미입니다.

back porch (HBP)

HGT2 와 같은 의미입니다.

11.2. 수직 동기

위의 그림으로 돌아가 봅시다. 어떻게 해야 247 클럭 틱을 윗그림처럼 위치시킬수 있을까요?

우리의 예에서는 HR 은 944 이고 HFL 은 1176 입니다 두 수의 차이는 1176 - 944 = 232 < 247 입니다! 분명 어딘가 수정이 필요하군요. 어디를 어떻게 고쳐야 할까요?

첫번째 할일은 1176을 1184 까지 높이고 944 를 936 으로 낮추는 것입니다. 이제 1184 - 936 = 248 이 되었습니다. 흠, 가까와 졌군요.

그 다음, HSP 를 계산하는데 있어 3.8 대신 3.5 를 사용합니다; 그러면 65 * 3.5 = 227 을 얻습니다. 훨씬 나아졌군요. 하지만 248은 227 보다 그리 높은 수치는 아니군요. 보통 약 30 클럭 내외의 틱이 SP 의 시작점과 HR 사이에 필요하고, 또 같은 만큼이 SP 의 끝과 HFL 사이에 필요합니다. 그리고 이 값들은 반드시 8 의 정수배여야 합니다! 막혀버린 걸까요?

아닙니다. 이렇게 해봅시다. 936 % 8 = 0, (936 + 32) % 8 = 0. 하지만 936 + 32 = 968, 968 + 227 = 1195, 1195 + 32 = 1227. 흠... 이 값은 그리 나쁘지 않군요. 하지만 이 값은 8 의 정수배가 아니므로 이것을 1232 로 반올림 해버립시다.

하지만 이제 우리는 잠재적인 문제를 안게 되었습니다. 동기 펄스는 더이상 h 와 H 사이의 정중앙에 정확히 위치하지 않게 되었습니다. 다행히도 계산기를 두드려보면 1232 - 32 = 1200 이고 이것 역시 8 의 배수입니다. (1232 - 32) - 968 = 232 는 3.57 usec 길이의 싱크펄스를 사용했다고 가정했을 때의 수치에 대응합니다. 3.57 역시 괜찮은 값이구요.

게다가 936/1232 = 0.76, 즉 76% 입니다. 80% 에서 그리 먼 값이 아니므로 이 값도 괜찮습니다.

더우기 현재의 수평 프레임 길이를 사용하면, 기본적으로 모니터로 하여금 52.7 Khz(= 65 MHz / 1232) 의 동기펄스를 발생시키도록 하는 것입니다. 이 값 역시 모니터의 능력범위 내의 값입니다. 아무런 문제가 없습니다.

위에서 언급했던 경험칙을 사용하면, 936 * 75% = 702, 이 값이 우리의 새로운 수직 해상도입니다. 702 * 1.05 = 737, 우리의 새로운 수직 프레임 길이입니다.

스크린 리플레쉬율은 65 MHz / (737 * 1232) = 71.6 Hz. 이것 역시 아주 좋은 값입니다.

수직동기 펄스의 그림을 그려보면 다음과 같습니다.

```
|___ __ __ __ __ __ __ __ __ __ __ __ __
|_ _ _ _ _ _ _ _ _ _ _ _ _              |
|________________________|______________|_____
0                       VR             VFL     단위: ticks
                         ^   ^         ^
                         |   |         |
                         |<->|<------->|
                          VGT    VSP
```

우리는 수직 디스플레이 데이터의 틱이 끝나자마자 수직펄스를 시작했습니다. VGT 는 수직 완충시간이라는 것으로서 동기펄스에 필요합니다. 대부분의 모니터들은 VGT 를 0 으로 잡더라도(즉 완충시간이 없어도) 안정적이므로 우리의 예에서는 그렇게 설정했습니다. 몇몇 모니터들은 완충시간으로 2-3 틱을 필요로 하는 수도 있으며 보통 2-3 틱 더해주어도 별 문제가 되지는 않습니다.

예제로 돌아 갑시다. 프레임 길이의 정의에 의해 수직 틱 하나는 수평 프레임 하나가 완전히 주사되는데 걸린 시간입니다. 따라서 우리의 예에서 이 값은 1232 / 65 MHz = 18.95 us 입니다.

경험적으로 보면 수직 동기 펄스는 50 - 300 us 범위 이내에 있어야 합니다. 예를들어 150 us 라 해봅시다. 이 값은 약 8 수직 클럭틱이 됩니다(150 us / 18.95 us = 8)

일부 제조업체들은 수직 타이밍 파라메터들을 도트길이 대신 타이밍으로 표기합니다. 다음과 같은 용어들이 사용될 수도 있습니다.

active time (VAT)

VR 에 해당하는 용어지만 밀리세컨드 단위입니다. VAT * VSF = VR.

blanking time (VBT)

(VFL -VR) 에 해당하는 용어지만 밀리세컨드 단위입니다. VBT * VSF = (VFL - VR).

front porch (VFP)

VGT 와 같은 의미입니다.

sync time

VSP 와 같은 의미입니다.

back porch (VBP)

수직 동기펄스 다음의 두번째 완충시간과 같습니다. 보통 0 입니다.

12. 종합

Xconfig 화일내에는 비디오 모드에 관한 수치들이 나열되어 있습니다. 각 라인은 X 서버의 동작모드 하나를 나타냅니다. 하나의 라인은 이름 섹션, 클럭주파수 섹션, 수평 섹션, 수직 섹션의 4 부분으로 구성됩니다.

이름 섹션은 그 비디오 모드의 이름 필드 하나로 되어 있습니다. 뒤이어 나오는 수치들이 만드는 비디오 모드의 이름이 되는 것입니다. 이 이름은 Xconfig 화일의 그래픽 드라이버 설정 부분의 "Modes" 줄에서 사용되게 됩니다. 만일 바로 앞줄의 이름이 현재 줄의 이름과 동일하다면 현재 줄의 이름 필드를 생략해도 좋습니다.

도트 클럭 섹션은 그 비디오 모드 라인의 도트 클럭(우리가 DCF 라 부른 것) 필드 하나로 되어있습니다. 이 숫자는 연이어 나오는 섹션의 숫자들을 생성하는데 쓰이는 도트클럭입니다.

수평 섹션은 4개의 필드로 구성되며 디스플레이의 수평 라인이 어떻게 생성되어야 하는지를 정합니다. 이 섹션의 첫번째 필드는 화면을 구성하는데 필요한 라인당 수평 도트의 수(우리가 HR 이라 부른 것)를 나타냅니다. 두번째 필드(SH1)는 몇번째 도트에서 수평 동기 펄스가 시작되는지를 나타냅니다. 세번째 필드(SH2)는 몇번째 도트에서 수평 동기 펄스가 끝나는지를 나타냅니다. 네번째 필드는 수평 프레임 길이 전체(HFL)를 나타냅니다.

수직 섹션 역시 4개의 필드로 구성됩니다. 첫번째 필드는 화면을 구성하는데 필요한 수평라인의 수(VR)를 나타냅니다. 두번째 필드(SV1)는 몇번째 라인에서 수직 동기펄스가 시작되는지를 나타냅니다. 세번째 필드(SV2)는 몇번째 라인에서 수직 동기펄스가 끝나는지를 나타냅니다. 네번째 필드는 수직 프레임 길이 전체(VFL)를 나타냅니다.

예 :

```
#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
```

(주의 : X11R5 에서는 소수점 표기된 도트클럭은 허용되지 않습니다)

위에서 방금 언급한(라인당 화면을 만드는 도트의 수, 화면의 도트와 동기펄스의 시작을 구분짓는 도트넘버, 펄스의 지속시간을 의미하는 도트넘버, 동기펄스가 끝난 후의 도트의 수) 수치들이 Xconfig 파일 속에 써넣어져서 한 라인의 도트들을 생성해냅니다. 수평 도트들의 숫자는 반드시 8 로 나누어 떨어져야 합니다.

수평 숫자들의 예 : 800 864 1024 1088

윗줄의 예에는 가시영역의 도트수(800), 동기펄스가 몇번째 도트에서 시작되는지(864), 동기펄스가 몇번째 도트에서 끝나는지(1024), 수평 라인이 몇번째 도트에서 끝나는지(1088)가 적혀있습니다.

수평 숫자들(800, 864, 1024, 1088)이 모두 8 로 나누어진다는 점에 다시한번 주의하십시요! 수직 숫자들은 8 로 나뉘어질 필요는 없습니다.

화면의 첫 줄부터 마지막 줄까지의 라인의 수가 프레임을 구성합니다. 한 프레임에 대한 타이밍 신호은 라인단위입니다. 다수개의 라인이 이미지를 그려내는 것입니다. 화면에 마지막 라인이 디스플레이된 후, 몇 라인 정도의 딜레이가 있고, 다음으로 수직 동기 펄스가 몇 라인 정도 지속되고, 최후로 프레임의 마지막 라인까지의 시간이 소요됩니다. 다음 예와 같이 수치들이 나열됩니다.

수직 숫자들의 예 : 600 603 609 630

이 예에서는 가시화면에 600 개의 라인이 있고 그다음 수직 동기펄스가 603 째 라인에서 시작해서 609 번째 라인에서 끝나며 총 630 개의 라인이 있음을 나타내고 있습니다.

수직 숫자들은 8 로 나뉘어질 필요가 없음에 유의하십시요!

이제 우리의 예로 돌아갑시다. 위의 내용대로 우리가 얻어낸 결과를 다음과 같은 형식으로 Xconfig 에 적어넣어야 합니다.

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

SH1 은 수평 동기펄스의 시작틱이고 SH2 는 수평동기 펄스가 끝나는 틱입니다; 마찬가지로, SV1 은 수직 동기 펄스의 시작틱이고 SV2 는 수직 동기 펄스가 끝나는 틱입니다.

위와 같은 형식으로 써넣기 위해 우리가 부려봤던 마술과 동기펄스를 기억해 봅시다. SH1 은 수평 동기 펄스의 리딩 엣지(leading edge, 한 펄스에서 마루에서 산으로 올라가는 부분)에서 시작됩니다. 따라서 SH1 = HR + HGT1 입니다. SH2 는 트레일링 엣지(trailing edge, 한 펄스

에서 산에서 마루로 떨어지는 부분)입니다. 따라서 SH2 = SH1 + HSP 입니다. 이와 유사하게 SV1 = VR + VGT 이고(단, VGT 는 보통 0 입니다) SV2 = SV1 + VSP 입니다.

```
#name     clock    horizontal timing    vertical timing    flag
936x702   65       936 968 1200 1232    702 702 710 737
```

특별한 flag 는 필요없습니다. 이것은 넌인터레이스드 모드니까요. 이제 정말 끝난 것입니다.

13. 모니터의 오버

만일 모니터가 고정 주파수 타입이라면 절대로 모니터의 스캔레이트를 초과하려는 시도를 하지 마십시요**.** 하드웨어에서 연기가 나는 수가 있습니다! 멀티싱크 모니터를 오버할 때는 다음과 같은 미묘한 잠재적인 문제들에 주의하십시요.

픽셀 클럭을 모니터의 대역폭의 최대값보다도 높이는 것은 오히려 덜 위험합니다. 정작 위험할 수 있는 일은 정격 최대 동기 주파수를 초과하는 것입니다. 몇몇 최신형 모니터들은 위험한 스캔 레이트가 되면 자동으로 모니터를 셧다운 시키는 보호회로를 내장하기도 합니다만 이를 너무 믿지는 마십시요. 특히 구형 멀티싱크 모니터들에서는(예를 들면 Multisync II) 수평 트랜스포머가 한개만 들어있습니다. 이런 모니터는 오버시에 보호가 잘 되지 않습니다. 비록 모니터가 분명히 고전압 정류회로를 가지고 있다해도(고정 주파수 모니터에는 이 회로가 없을수 있습니다) 이것이 반드시 모든 가능한 주파수 범위를 커버해주는 것은 아닙니다. 특히나 싸구려 모델에서는요. 이 경우 전자회로를 피로하게 만들고 또한 스크린에 도포된 인(phosphor)이 빨리 닳게되며 따라서 규격보다도 많은 전자파(X 레이 포함)가 모니터에서 방출됩니다.

하지만 정작 이 상황에서 근본적인 문제로 대두되는 것은 비디오 출력 드라이버의 slew rate(비디오 시그널의 경사도)입니다. 보통의 경우에는 이것과 픽셀 주파수와는 무관합니다만(비디오카드 제조사가 그러한 문제에 충분히 신경을 썼겠지만), 비디오카드의 픽셀 주파수가 최대값에 근접할수록 신호의 경사도가 무뎌지는 문제가 생길수 있습니다.

따라서 오버시에는 주의하시기 바랍니다...

14. 비월주사 방식(인터레이스드 모드)

(이 절은 주로 David Kastrup <`dak@pool.infomatik.rwth-aachen.de`> 씨가 보내주신 내용에 의거했습니다)

일정한 도트클럭하에서, 인터레이스드 화면은 모니터의 수직부 회로가 충분히 안정적으로 받쳐준다면 넌인터레이스드 화면에 비해 그다지 플리커가 심하지 않을 수도 있습니다. 역사적으로 볼때 인터레이스드 모드가 먼저 발명되었습니다.

인터레이스드 모드는 같은 수직 스캔 주파수(VSF, 이 값은 주로 광고에 많이 등장합니다)에서의 넌인테레이스드 모드보다도 뒤떨어진 성능을 보이기 때문에 악명이 높습니다만 분명 같은 수평 스캔 레이트에서 더 나은 화질을 보여줍니다. 따라서 보통 당신의 모니터/비디오카드의 성능의 최대 한계치는 인터레이스드 모드가 됩니다.

동일한 리프레쉬 레이트(혹은 하프 프레임 레이트, 즉 VSF)하에서 인터레이스드 화면은 넌인터레이스드 화면에 비해 플리커가 더 심합니다. 90 Hz 인터레이스드 화면은 90 Hz 의 넌인터레이스드 화면보다 나쁩니다. 하지만 겨우 절반의 비디오 대역폭과 절반의 수평 스캔 레이트만 있으면 됩니다. 만일 이것을 같은 도트클럭과 같은 스캔 레이트의 넌인터레이스드 모드와 비교해보면 인터레이스드 화면이 훨씬 좋다는 말입니다. 45 Hz 넌인터레이스드 화면은 견디기 힘들지요. 필자는 90 Hz 인터레이스드 모드로 1024x768 해상도에서 수년간 Multisync 3D 모니터를 사용해 왔고 매우 만족하고 있습니다. 제 생각에 당신이 같은 정도의 만족감을 느끼려면 적어도 70 Hz 의 넌인터레이스드 디스플레이를 사용하셔야 할 겁니다.

하지만 몇가지 점에 주의하셔야 합니다. 인터레이스드 모드는 오직 고해상도에서만 사용하세요. 그래야 교차적으로 발광하는 라인들이 서로 가까이에 위치하게 됩니다. 어쩌면 당신은 동기 펄스의 폭과 위치를 조정해서 가장 안정적인 라인위치를 얻으려 하실지도 모르겠군요. 만약 교번하는 라인들의 밝고 어두움의 대비가 뚜렷하다면 인터레이스 모드는 마치 요철이 있는 것처럼 보일 수도 있습니다. 필자는 메뉴의 배경으로 이러한 도트패턴을 사용한 프로그램을 하나 가지고 있습니다(XCept 라는 프로그램입니다. 다행히도 필자가 아는 한 이런 류의 다른 프로그램은 없는 것으로 압니다). 필자는 XCept 프로그램을 쓸때는 800x600 으로 설정합니다. 그러지않으면 정말 눈이 아프거든요.

같은 이유로 최소한 100 dpi 폰트를 쓰거나 최소한 가로선의 두께가 두줄이상인 폰트를 쓰세요.(고해상도에서는 어쨌거나 다른 선택의 여지가 없습니다).

그리고 당연한 이야기지만 하드웨어가 당신이 원하는 리프레쉬 레이트에서 넌인터레이스드 모드를 지원한다면야 인터레이스드 모드를 사용하지 마십시요.

하지만 당신이 모니터나 그래픽 카드의 능력의 최고치를 사용하는 해상도를 찾았는데, 플리커가 불만이라던가 아웃워시(outwashed, 대역폭을 초과한 것)된 디스플레이가 나온다면 인터레이스드 모드를 써서 그 해상도를 쓸 수 있을 것입니다. 물론 모니터의 VSF 가 이미 충분히 그 한계치에 가깝다면야 이 방법은 소용없겠지만요.

인터레이스드 모드를 디자인하는 것은 쉽습니다. 넌인터레이스드 모드와 똑같이 해주고 단지 두가지만 더 고려하면 됩니다. 수직 라인의 총 갯수(모드라인의 마지막 숫자)를 홀수로 해주어야 하고, "interlace" 라는 플레그를 설정할 경우 모니터의 실제 수직 프레임 레이트는 2배가 된다는 점입니다. 만일 당신이 설정한 모드가 "Interlace" 플래그가 없다고 쳤을때 45 Hz 모드로 설정한 것이라면, 모니터는 90 Hz 의 프레임 레이트를 지원해 주어야만 합니다.

한 예로서 필자의 1024x768 인터레이스드 모드라인을 제시합니다. 필자의 Multisync 3D 모니터는 수직 주파수 90 Hz 까지, 수평 주파수 38 KHz 까지를 지원합니다.

```
              Mode-
Line "1024x768" 45 1024 1048 1208 1248 768 768 776 807 Interlace
```

이 모드에서는 수평과 수직으로 거의 한계치에 가까이 설정되었습니다. 같은 모드를 "Interlaced" 플래그 없이 설정한다해도 거의 모니터의 수평 성능의 한계값이 될 정도입니다만(정확히 말하면 수직 스캔 레이트의 하한보다 약간 더 낮습니다), 넌인터레이스드 모드의 경우였다면 참을수 없을 정도로 플리커가 심한 화면이 됩니다.

기본적인 디자인 규칙은 다음과 같습니다: 만일 모니터의 수직 성능의 절반값 미만으로 모드를 디자인했다면, 총 수직 라인의 수를 홀수로 한 후 "Interlace" 플래그를 덧붙입니다. 대부분의 경우 화질이 큰폭으로 향상될 것입니다.

만일 어떤 넌인터레이스드 모드가 모니터 최대값의 약 30 % 내지 그 이하의 스캔레이트인데 인터레이스드 모드를 써서 모니터의 스펙의 최대치를 끌어낼 요량이라면, 직접 손으로 인터레이스드 모드를 디자인해서 훨씬 좋은 결과를 얻어낼 수도 있습니다. 하지만 보장은 못합니다.

15. 질문과 답변

질문 당신이 제시한 예는 표준적인 화면 크기가 아닌데 그렇게 써도 되는 건가요?

답변 왜 안됩니까? 왜 꼭 640x480, 800x600, 1024x768 등으로만 써야할 이유는 어디에도 없습니다. XFree86 서버는 하드웨어를 설정하는데 있어 많은 자유를 줍니다. 보통 괜찮은 설정에 이르기까지 2-3 번의 시도가 필요합니다. 중요한 것은 높은 리프레쉬 레이트로 괜찮은 크기의 가시화면을 얻는 것이지 눈물이 나오는 플리커를 댓가로 고해상도를 보자는 이야기는 아닙니다.

질문 65MHz 도트클럭에 55 KHz HSF 가 주어졌다면 오직 한가지 해상도만이 나오게 되는 것인가요?

답변 절대로 그렇지 않습니다! 위의 일반적인 절차를 따라서 몇번 시행착오를 거치면 정말 만족할만한 세팅을 찾을수 있습니다. 이렇게 실험해 보는 것은 매우 재미있는 작업일 수 있습니다. 대부분의 세팅은 고약한 비디오 화면을 보여주기도 할테지만 실상 최근의 멀티싱크 모니터는 그리 쉽게 파손되지 않습니다. 하지만 그 모드에서 오래 사용하기 전에 먼저 모니터가 당신 모드의 프레임 레이트를 지원할수 있는지 충분히 확인하시기 바랍니다.

고정 주파수 모니터의 경우에는 주의하세요! 이런 류의 해킹은 고정 주파수 모니터에서는 상당히 빨리 손상을 일으킬수 있습니다. 고정 주파수 모니터를 두고 실험할 때는 언제나 타당한 리프레쉬 레이트인지를 확인하세요.

질문 당신은 단지 두개의 표준 해상도를 언급했습니다. Xconfig 에서는 사용가능한 많은 표준 해상도가 있는데 굳이 타이밍을 놓고 고민해야할 이유가 있는지 말씀해 주십시요.

답변 필요가 분명히 있지요! 예를 들어 현재의 Xconfig 에 나열되어 있는 "표준" 640x480 해상도를 봅시다. 이것은 25 MHz 의 구동 주파수와 800, 525 의 프레임 길이, 리프레쉬 레이트 59.5 Hz 로 설정되어 있습니다. 이것도 그리 나쁘지는 않습니다. 하지만 대다수의 많은 SVGA 카드들은 구동 주파수로 28 MHz 가 가능합니다. 만일 우리가 앞에서 논한 절차에 따라서 설정하여 640x480 를 쓴다면 프레임 길이 812(버림해서 808로 잡아야함), 505 를 얻을수 있습니다. 이제 리프레쉬 레이트는 68 Hz 로 증가했고 이것은 표준보다도 상당히 개선된 것입니다.

질문 지금까지 논한 것을 정리해 주시겠습니까?

답변 아주 간략히 말해보지요.

- 어떤 고정된 구동 주파수하에서는, 최고 해상도를 높이는 것은 리프레쉬레이트를 낮추는 것을 감수해야만 하고 따라서 플리커가 더 심해지게 된다.
- 좋은 해상도를 원하고, 또 모니터가 이를 지원해 준다면, 이에 알맞은 도트클럭, 즉 DCF 를 갖는 SVGA 카드를 사용하라. 높을수록 더 좋은 것이다!

16. 이미지에 관련된 문제해결

좋습니다. 이제 당신은 X 설정을 위한 수치들을 얻었습니다. 그 수치들을 Xconfig 에 적고 테스트용으로 적당한 이름을 붙입시다. X 를 시작하고 핫키를 써서 그 새로운 모드로 들어갑니다. 그런데 이때 이미지가 똑바르지 않다면 어떻게 해야 할까요? 이 절에서는 일반적인 비디오 이미지의 왜곡의 예와 그에 대한 교정법을 설명합니다.

(이러한 자잘한 왜곡을 수정하는 데는 **xvidtune**(1) 이 정말 좋습니다).

동기 펄스 타이밍을 변화시키면 이미지가 이동합니다. 프레임 길이를 변화시키면 이미지가 확대/축소됩니다(동기 펄스를 상대적으로 같은 위치에 유지시켜야만 합니다. 그렇지 않으면 이미지가 확대/축소와 동시에 이동하게 됩니다). 다음은 몇가지 특정한 해결책입니다.

수평과 수직 위치는 서로 독립적입니다. 즉, 이미지를 수평으로 움직이는 것은 수직위치에 아무런 영향을 주지 않으며 그 반대도 마찬가지입니다. 하지만 확대/축소의 경우는 그렇지 않습니다. 수평크기를 바꾸어 주면서 수직크기를 바꾸어 주지 않는다면 전체적인 수정이 제한받을 수 있으며 반대 경우도 마찬가지입니다. 특히, 만일 이미지가 수평, 수직 양방향으로 너무 크다면 이를 교정하기 위해서는 틀림없이 더 높은 도트클럭을 사용해야만 합니다. 이는 사용가능한 해상도를 높이는 것이므로 거의 문제가 되지 않습니다!

16.1. 이미지가 좌우로 치우칠 때

이를 교정하려면 수평 동기 펄스를 이동시킵니다. 즉, 수평 타이밍 섹션부분의 중간의 두 숫자가 수평 동기 펄스의 리딩엣지와 트레일링 엣지를 규정하고 있으므로 이것을 8 의 배수만큼씩 증감시킵니다.

만일 이미지가 왼쪽으로 치우쳤다면(즉, 오른쪽 경계가 너무 커서 이미지를 오른쪽으로 이동시키려 할때) 그 숫자를 감소시킵니다. 만일 이미지가 오른쪽으로 치우쳤다면 (즉, 왼쪽 경계가 너무 커서 이미지를 왼쪽으로 이동시키려 할때) 동기 펄스를 증가시킵니다.

16.2. 이미지가 상하로 치우칠 때

이를 교정하려면 수직 동기 펄스를 이동시킵니다. 즉, 수직 타이밍 섹션 부분의 중간의 두 숫자가 수직 동기 펄스의 리딩엣지와 트레일링 엣지를 규정하고 있으므로 이것을 증감시킵니다.

만약 이미지가 위로 치우쳤다면 (즉, 아랫쪽 경계부가 너무 커서 이미지를 아래로 이동시키고 싶다면) 두 숫자를 낮춥니다. 만약 이미지가 아래로 치우쳤다면(즉, 윗쪽 경계가 너무 커서 위쪽으로 이동시키려 한다면) 두 숫자를 증가시킵니다.

16.3. 이미지가 상하좌우로 너무 큰 경우

카드의 클럭 스피드를 더 높은 것을 택합니다. 클럭 파일에 모드가 여러개 있다면 아마도 당신이 실수로 낮은 스피드의 모드를 선택한 것일 겁니다.

16.4. 이미지가 좌우로 넓은(좁은) 경우

이를 교정하려면 수평 프레임 길이를 증가(감소)시킵니다. 즉, 첫번째 타이밍 섹션의 4 번째 숫자를 변화시킵니다. 이미지가 움직이는 것을 방지하려면 동기 펄스(즉, 두번째와 세번째 숫자)까지도 움직여서 상대적으로 동일한 위치를 유지하도록 그 절반만큼 움직입니다.

16.5. 이미지가 상하로 길(짧을) 경우

이를 교정하려면 수직 프레임 길이를 증가(감소)시킵니다. 즉, 두번째 타이밍 섹션의 4 번째 숫자를 변화시킵니다. 이미지가 움직이는것을 방지하려면 동기 펄스(두번째와 세번째 숫자)를 움직여서 상대적으로 동일한 위치를 유지하도록 그 절반만큼 움직입니다.

이와 같은 테크닉들을 구사해도 왜곡이 없어지지 않는다면 틀림없이 무언가 근본적으로 잘못되었다는 증거입니다. 가령 계산을 잘못했거나 모니터가 받쳐줄수 있는 도트클럭보다 더 빠른 클럭을 사용했거나 등등입니다.

마지막으로, 프레임 길이를 증가시키면 리프레쉬레이트가 감소하고 반대로 프레임길이가 감소하면 리프레쉬 레이트가 증가하는 관계임을 잊지 마십시요.

경우에 따라서는 모니터의 볼륨단자를 조절하여 자잘한 왜곡을 교정할 수도 있습니다. 이 방법의 단점은 그래픽모드의 문제를 해결하려고 공장에서 맞춰진 중앙값보다 과도하게 볼륨을 조절해 놓으면 텍스트모드로 전환했을시 화면이 치우쳐 버릴수 있다는 점입니다. 모드라인을 올바로 설정하는 것이 더 나은 방법입니다.

17. 모니터의 성능을 그려보기

모니터 모드 다이어그램을 그려보려면 우선 gnuplot 패키지(Unix 류 운영체제 용의 공개된 플로팅 언어)와 **modeplot** 이라는 쉘/gnuplot 스크립트를 이용해 모니터의 특성치를 명령어 라인에서 입력해서 그려야 합니다.

다음은 **modeplot** 프로그램입니다.

```
#!/bin/sh
#
# modeplot -- generate X mode plot of available monitor modes
#
# Do ‘modeplot -?’ to see the control options.
#

# Monitor description. Bandwidth in MHz, horizontal frequencies in kHz
# and vertical frequencies in Hz.
TITLE="Viewsonic 21PS"
BANDWIDTH=185
MINHSF=31
MAXHSF=85
```

```
MINVSF=50
MAXVSF=160
ASPECT="4/3"
vesa=72.5 # VESA-recommended minimum refresh rate

while [ "$1" != "" ]
do
case $1 in
-t) TITLE="$2"; shift;;
-b) BANDWIDTH="$2"; shift;;
-h) MINHSF="$2" MAXHSF="$3"; shift; shift;;
-v) MINVSF="$2" MAXVSF="$3"; shift; shift;;
-a) ASPECT="$2"; shift;;
-g) GNUOPTS="$2"; shift;;
-?) cat <<EOF
modeplot control switches:

-t "<description>" name of monitor            defaults to "Viewsonic 21PS"
-b <nn>             bandwidth in MHz           defaults to 185
-h <min> <max>    min & max HSF (kHz)        defaults to 31 85
-v <min> <max>    min & max VSF (Hz)         defaults to 50 160
-a <aspect ratio> aspect ratio               defaults to 4/3
-g "<options>"    pass options to gnuplot

The -b, -h and -v options are required, -a, -t, -g optional.  You can
use -g to pass a device type to gnuplot so that (for example) modeplot's
output can be redirected to a printer.  See gnuplot(1) for  details.

The modeplot tool was created by Eric S. Ray-
mond <esr@thyrsus.com> based on
analysis and scratch code by Martin Lotter-
moser <Martin.Lottermoser@mch.sni.de>

This is modeplot $Revision: 1.2 $
EOF
exit;;
esac
shift
done

gnuplot $GNUOPTS <<EOF
set title "$TITLE Mode Plot"

# Magic numbers.  Unfortunately, the plot is quite sensitive to changes in
# these, and they may fail to represent reality on some monitors.  We need
# to fix values to get even an approximation of the mode dia-
gram.  These come
# from looking at lots of values in the ModeDB database.
F1 = 1.30 # multiplier to convert horizontal resolution to frame width
```

```
F2 = 1.05 # multiplier to convert vertical resolution to frame height

# Function definitions (multiplication by 1.0 forces real-
number arithmetic)
ac = (1.0*$ASPECT)*F1/F2
refresh(hsync, dcf) = ac * (hsync**2)/(1.0*dcf)
dotclock(hsync, rr) = ac * (hsync**2)/(1.0*rr)
resolution(hv, dcf) = dcf * (10**6)/(hv * F1 * F2)

# Put labels on the axes
set xlabel 'DCF (MHz)'
set ylabel 'RR (Hz)' 6 # Put it right over the Y axis

# Generate diagram
set grid
set label "VB" at $BANDWIDTH+1, ($MAXVSF + $MINVSF) / 2 left
set arrow from $BANDWIDTH, $MINVSF to $BANDWIDTH, $MAXVSF nohead
set label "max VSF" at 1, $MAXVSF-1.5
set arrow from 0, $MAXVSF to $BANDWIDTH, $MAXVSF nohead
set label "min VSF" at 1, $MINVSF-1.5
set arrow from 0, $MINVSF to $BANDWIDTH, $MINVSF nohead
set label "min HSF" at dotclock($MINHSF, $MAXVSF+17), $MAXVSF + 17 right
set label "max HSF" at dotclock($MAXHSF, $MAXVSF+17), $MAXVSF + 17 right
set label "VESA $vesa" at 1, $vesa-1.5
set arrow from 0, $vesa to $BANDWIDTH, $vesa nohead # style -1
plot [dcf=0:1.1*$BANDWIDTH] [$MINVSF-10:$MAXVSF+20] \
  refresh($MINHSF, dcf) notitle with lines 1, \
  refresh($MAXHSF, dcf) notitle with lines 1, \
  resolution(640*480,   dcf) title "640x480  " with points 2, \
  resolution(800*600,   dcf) title "800x600  " with points 3, \
  resolution(1024*768,  dcf) title "1024x768 " with points 4, \
  resolution(1280*1024, dcf) title "1280x1024" with points 5, \
  resolution(1600*1280, dcf) title "1600x1200" with points 6

pause 9999
EOF
```

일단 **modeplot** 와 gnuplot 패키지를 제대로 준비했다면 이제 다음 모니터 특성치가 필요합니다.

- 비디오 대역폭 (VB)
- 수평 동기 주파수의 범위 (HSF)
- 수직 동기 주파수의 범위 (VSF)

플롯 프로그램은 단순화시킨 조건하에서 그림을 그리게 됩니다. 이 조건이 항상 타당한 것은 아니므로 결과로 나오는 다이어그램은 대략적인 그림일 뿐입니다. 조건은 다음과 같습니다.

- 모든 해상도는 고정된 화면비(aspect ratio) AR = HR / VR 하나만을 가집니다. 표준 해상도는 AR = 4/3 혹은 AR = 5/4 이다. **modeplot** 프로그램은 디폴트로 4/3 으로 가정하고 있습니다. 이 값을 다르게 설정할 수도 있습니다.
- 고려하고 있는 모드에 있어서, 수평과 수직 프레임 길이는 수평과 수직 해상도의 일정배율의 곱입니다. 각각,

```
HFL = F1 * HR
VFL = F2 * VR
```

대략적인 값으로서 F1= 1.30, F2 = 1.05 정도로 잡습니다(프레임 크기를 계산하기 부분을 보세요).

이제 특정한 동기 주파수 HSF 를 생각해 봅시다. 방금 말한 가정하에서 보면, 클럭 레이트 DCF 의 모든 값은 이미 리프레쉬 레이트 RR 을 결정짓습니다. 즉, HSF 의 모든 값에 대한 함수값 RR(DCF)가 존재하는 것입니다. 이는 다음과 같이 유도됩니다.

리프레쉬 레이트는 프레임 사이즈들의 곱으로 클럭 레이트를 나눈 값과 같습니다.

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

한편, 수평 프레임 길이는 수평 동기주파수로 클럭 레이트를 나눈 값과 같습니다.

```
HFL = DCF / HSF                     (**)
```

VFL 가 HFL 로 귀결될수 있는 것은 위의 두가지 가정 때문입니다.

```
VFL = F2 * VR
    = F2 * (HR / AR)
    = (F2/F1) * HFL / AR            (***)
```

(**) 와 (***) 를 (*) 에 넣으면 다음을 얻습니다.

```
RR = DCF / ((F2/F1) * HFL**2 / AR)
   = (F1/F2) * AR * DCF * (HSF/DCF)**2
   = (F1/F2) * AR * HSF**2 / DCF
```

고정된 HSF, F1, F2 ,AR 에 대해서 다이어그램에는 쌍곡선이 하나 나타납니다. 수평 동기주파수의 최소값과 최대값에 대해 이와같은 커브를 그리면 두개의 경계선을 가지게 되고 그 사이가 바로 가능한 영역입니다.

가능한 영역을 가로지르는 직선들은 특정한 해상도를 나타냅니다. 이것은 (*) 와 두번째 가정에 근거합니다.

```
RR = DCF / (HFL * VFL) = DCF / (F1 * HR * F2 * VR)
```

원하는 모든 해상도에 대해 이런 라인을 그려봄으로써 당신은 즉각 그 모니터에서 가능한 해상도, 클럭 레이트, 리프레쉬레이트간의 관계를 알아챌 수 있습니다. 이들 라인들은 모니터의 특성에 의존하는 것이 아니라 두번째 가정에서 비롯되는 것임에 주의하십시요.

modeplot 프로그램은 이것을 쉽게 보여줍니다. **modeplot -?** 하면 그 옵션을 알 수 있습니다. 대표적인 예는 다음과 같습니다.

```
modeplot -t "Swan SW617" -b 85 -v 50 90 -h 31 58
```

-b 옵션은 비디오 대역폭을 지정합니다; -v 와 -h 는 각각 수평과 수직 동기 주파수의 범위를 설정합니다.

modeplot의 출력을 읽을 때는 항상 이것이 대략적인 묘사에 불과함에 유념하십시요. 예를 들자면, 동기 펄스는 최소한 일정 폭을 필요로 하며, 따라서 이에 비롯한 HFL 값에 제약이 있음을 이 프로그램은 무시하고 있습니다. 또한 그림의 정확도는 가정들의 타당성에 좌우됩니다. 따라서 종합 부분에서 제시한 것과 같은 정밀한 계산(위에서 부린 마술같은 것)을 대신할 수는 없습니다. 하지만 이 프로그램을 통해 무엇이 가능하고 어떤 타협이 가능한지에 대해 감을 잡을수는 있을 것입니다.

18. 도와주신 분들

이 문서의 최초의 관리자는 Chin Fang <fangchin@leland.stanford.edu> 씨 입니다.

Eric S. Raymond <esr@snark.thysus.com> 씨가 다시 재구성했으며 쉽게 이해할 수 있도록 Chin Fang 씨의 원본을 상당부분 다시 썼습니다. 그 과정에서 Bob Crosson <crosson@cam.nist.gov> 씨의 또다른 하우투 문서의 대부분을 덧붙이게 되어습니다.

인터레이스드 모드에 대한 것은 주로 David Kastrup <dak@pool.informatik.rwth-aachen.de> 씨에 의존했습니다.

Nicholas Bodley <nbodley@alumni.princeton.edu> 씨가 디스플레이의 동작원리에 관한 절을 교정, 정돈해 주셨습니다.

Payne Freret <payne@freret.org> 씨가 모니터 디자인과 관련된 기술적인 세부사항들을 바로잡아 주셨습니다.

Martin Lottermoser <Martin.Lottermoser@mch.sni.de> 씨는 gnuplot 을 써서 모드 다이어그램을 그리는 아이디어를 제공했고 **modeplot** 에 필요한 수학적인 분석을 하였습니다. 배포된 **modeplot** 은 Martin 씨가 특정 케이스에 대해 사용했던 원래의 gnuplot 코드를 E.S.Raymond 씨가 일반화시켜서 다시 디자인한 것입니다.