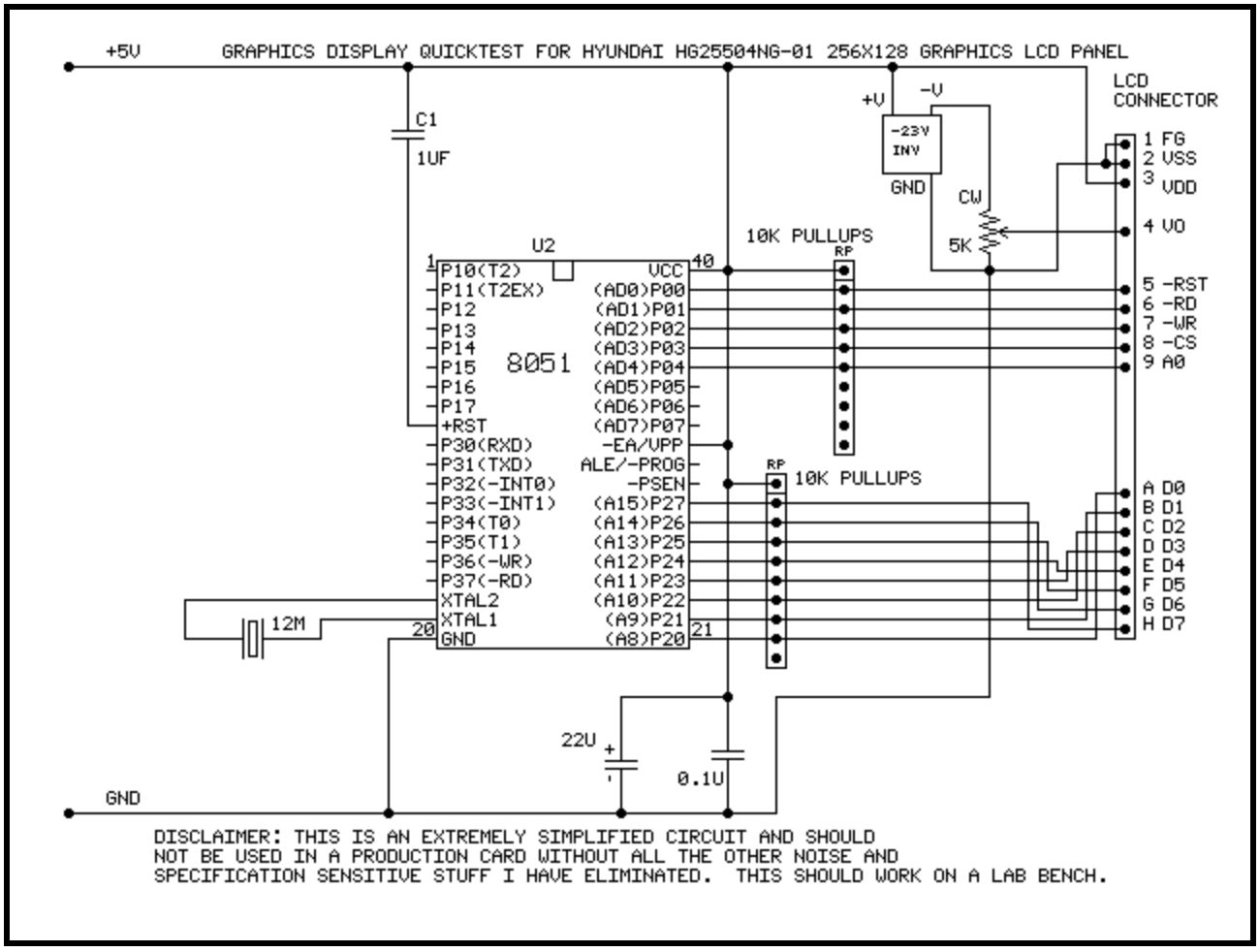
Graphics Display Quick Start Manual for using the Hyundai 256x128 LCD Module from AllElectronics - By Duane Becker Ó 2005, 2006

I get a lot of requests on how to get this module running. Here are some quick setups and answers. This quick setup uses any of the 8051 family microcontrollers. Atmel (flash), Intel 8751 (EPROM), Phillips, etc. are all fair game. I've included a schematic for the quick setup, a schematic for the +5 to -23 Volt inverter I designed for driving these LCD panels, the source code, and some photographs proving that it really does work. If you find this information useful, please send me a simple "Thank you" email to snowleop@sover.net I'd appreciate it.

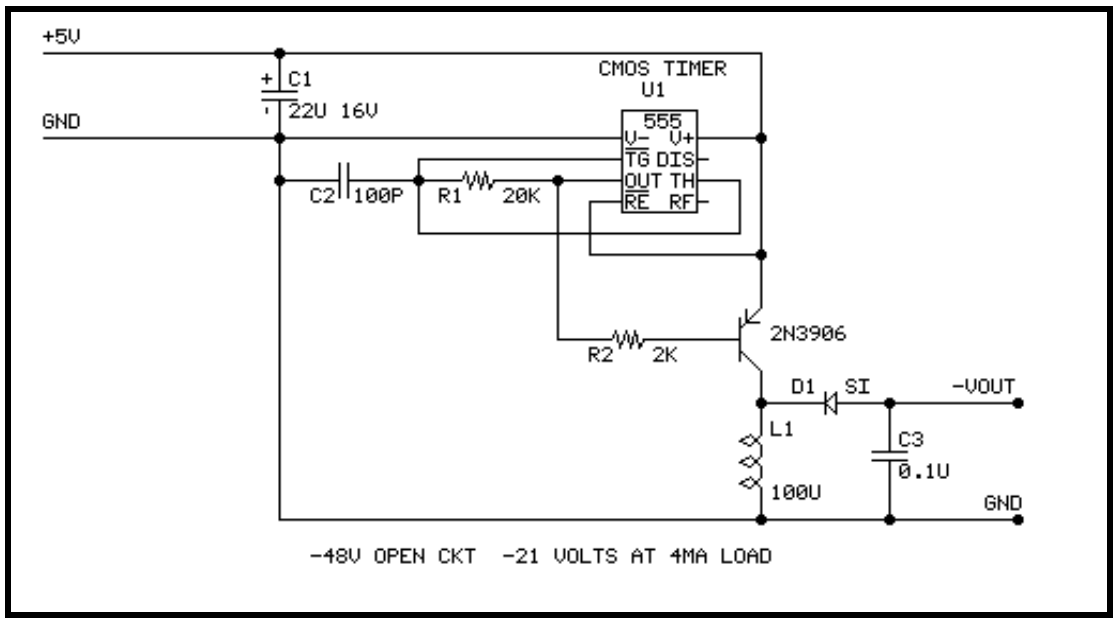
The Quick Setup Schematic



I used an Atmel 89C52 which holds up to 8-Kbytes of code, but the demonstration code is only 160 bytes. The crystal is a 12 MHz parallel resonant crystal, but use whatever suits you. Technically, you shouldn't require the pullups on port 2 driving the LCD data bus, but I found that the internal microcontroller pullups are too weak and the bus doesn't rise from "0" to "1" quickly enough when doing successive writes. Also, you'd normally put 22-pF caps on each leg of the crystal to

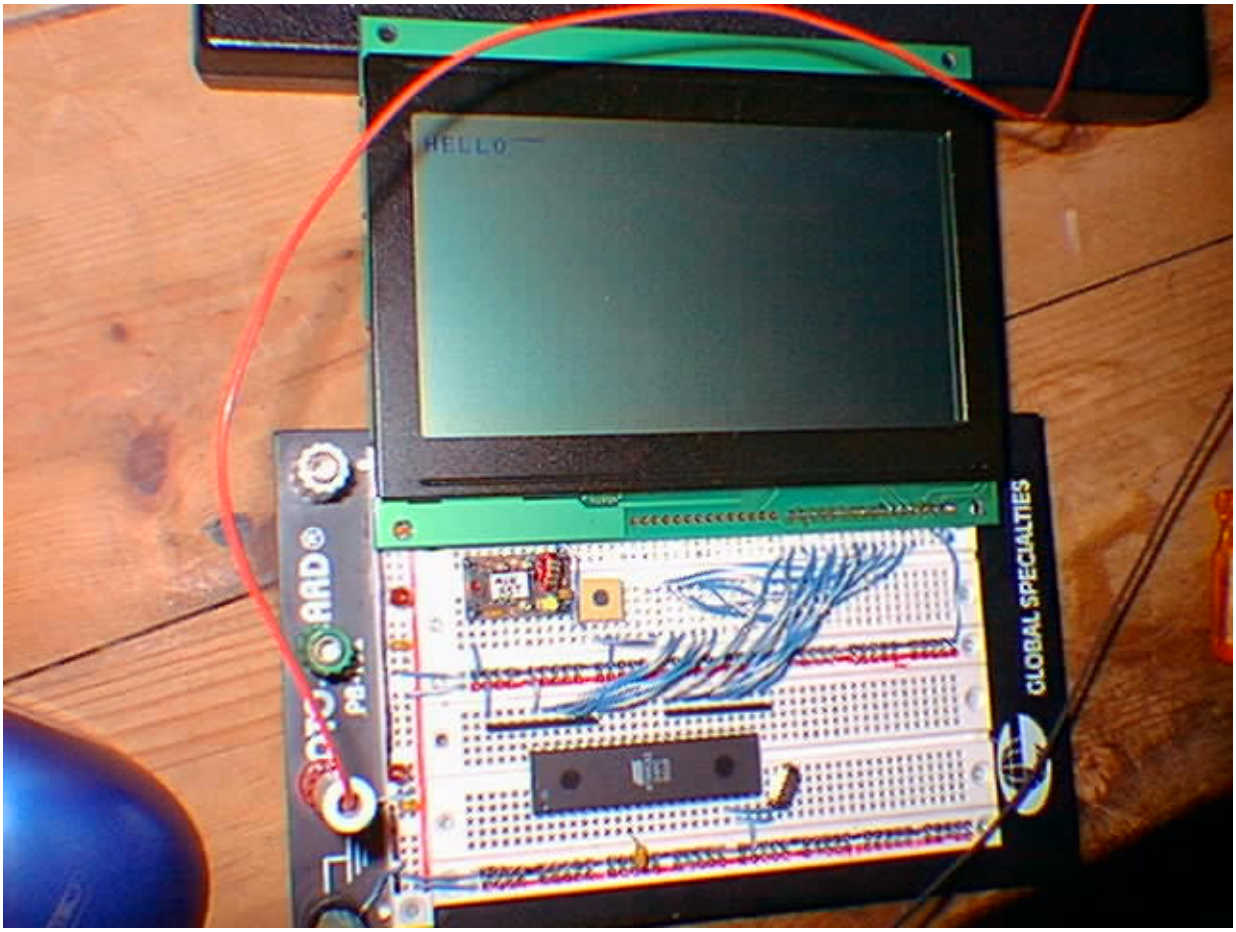
ground. Port 0 on the microcontroller is open-drain and requires pullups if you are driving to TTL input ports. The 1-microFarad cap from VCC to the reset pin is just that, a reset. The other two capacitors are for decoupling and ensuring that the voltage inverter gets all the current it needs when it switches on the inductor.

You may use an adjustable negative power supply for the VO supply to the LCD backplane. If you want to build your own inverter, as shown in the above schematic, here is the schematic for a -23 Volt inverter.

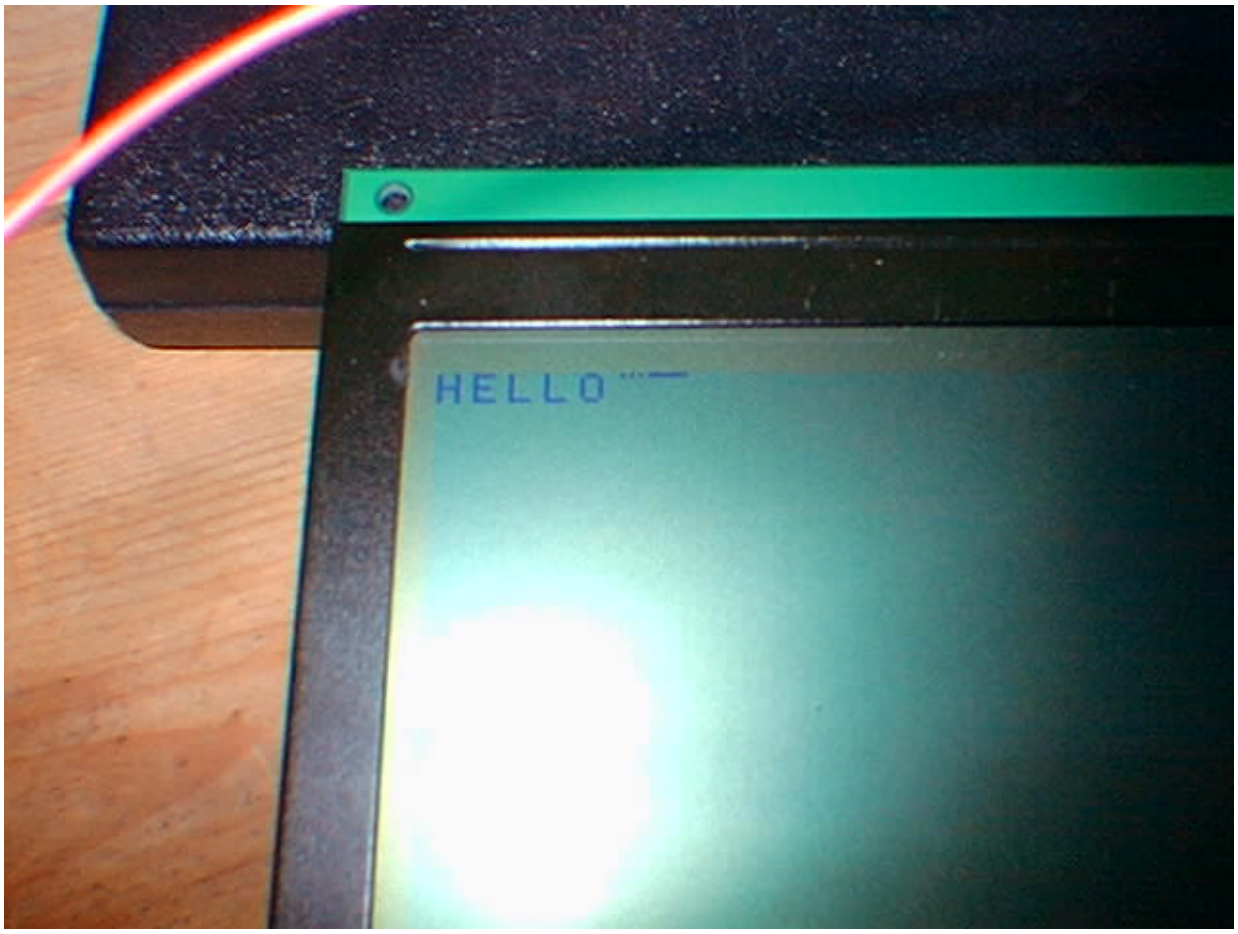


Be sure to use a CMOS version of the 555 timer. The bipolar version just doesn't turn off quickly enough to get a high flyback voltage from the inductor. Note that the current pulled by the VO pin of the LCD changes based upon how many dots you have turned on. In this simple example, the display contrast will change depending on the setting of the contrast potentiometer and how many dots are turned on. A remedy is to put a low-power negative adjustable regulator (like an LM337LZ) after the inverter, or selecting one of the higher priced inverter-regulator chips out there. Maxim makes these. So too, perhaps, do other companies. Another note: CW on the potentiometer means Clockwise, such that turning the potentiometer clockwise increases the negative voltage, and the darkness of the display.

Here is a picture of the finished prototype. Yes, Virginia, it works.



Here is a close up of the resulting test display. Note the pattern of the two graphics bytes (off on off on off on off on on on on on on on on) are as shown in the source code. In this example the text layer address starts at RAM address 0000h and the graphics layer address starts at RAM address 1000h. Each horizontal row displays 32 bytes of contents (32 characters on layer 1 or 32 x 8=256 pixels on layer 2). The top graphics row is displayed from addresses 1000h to 101Fh, the most-significant-bit of each byte being shown on the left of the respective byte column. The second graphics row is displayed from addresses 1020h to 103Fh. In this example, I only put two nonzero bytes to addresses 1005h and 1006h.



Source Code for the Quick Test Software:

```
; Graphics Display Quick Test
; Initialize the display, Put HELLO in the upper left corner,
; and after HELLO, on the top line, draw two graphics bytes.

; The timings used during the hardware reset section are based on
; using a 12 MHz crystal, giving a cycle time of 1 microsecond.
; The DJNZ instruction in the 8051 family takes 2 cycles to complete.


;  These are symbolic names for the ports used.

P0       EQU       80H       ; PORT 0
MINUSRST EQUB      P0.0      ; negative active reset pin of lcd
Minusrd  EQUB      P0.1      ; negative active read pin of lcd
minuswr  EQUB      P0.2      ; negative active write pin of lcd
minuscs  EQUB      P0.3      ; negative active chip select pin of lcd
address  EQUB      P0.4      ; address pin of lcd
DATABUS  EQU       0A0h      ; Data bus on port 2


         ORG 0000H           ; A hard reset starts at 0000h on an 8751.

;------- Initialize the LCD module -----------

           clr minusrst      ; drop -reset to display

           mov r0,#6         ; delay about 3 milliseconds
           mov r1,#0
pordelay1  djnz r1,pordelay1
           djnz r0,pordelay1

           setb minusrst     ; raise -reset

           mov r0,#20        ; delay 10 milliseconds after reset is gone
           mov r1,#0
pordelay2  djnz r1,pordelay2
           djnz r0,pordelay2

           mov a,#58h          ; DISPLAY OFF command so it's blank
           lcall writec

           mov a,#40h          ; System set command
           lcall writec
           Mov a,#30h          ; Int CG, 32chr CGRAM, 8lines/char, single pane
                               ; No invert, lcd, normal shift clock
           lcall writed

           mov a,#87h          ; 8 pixel-wide characters, 2 frame AC Drive
           lcall writed

           mov a,#07h          ; Verfical char size=8 pixels
           lcall writed

           mov a,#1Fh          ; 32 display bytes per line
           lcall writed
```

```
          mov a,#23h          ; Tot. addr range per line (4 extra for horz blk)
          lcall writed

          mov a,#7Fh          ; 128 display lines
          lcall writed

          mov a,#20h          ; low byte of virtual screen width
          lcall writed

          mov a,#00h          ; high byte of virtual screen width
          lcall writed
;------

         mov a,#44h          ; send scroll command
         lcall writec
         clr a               ; set layer 1 home address to 0000
         lcall writed
         clr a
         lcall writed
         mov a,#07fh         ; 128 lines for layer 1
         lcall writed

         clr a               ; Set layer2 (grapics) home address to 1000h
         lcall writed
         mov a,#10h
         lcall writed
         mov a,#07fh         ; 128 lines for layer 2
         lcall writed
                             ; since layers 3 and 4 are not used, stop sending
                             ; scroll parameters.
;------
         mov a,#5Ah          ; set horizontal scroll command
         lcall writec
         mov a,#00h          ; no horz scroll adjustment
         lcall writed
;------
         mov a,#5Bh          ; set overlay selections command
         lcall writec
         mov a,#00h          ; OR layers, Text block 1, 2 layer
         lcall writed
                             ; NOTE: layer 2 can ONLY be graphics (datasheet).
;------
         mov a,#4Ch          ; Auto cursor increment +1 Command
         lcall writec

;---------- clear memory
         mov a,#46h          ; Cursor Write Command
         lcall writec
         clr a               ; 0000h address
         lcall writed
         clr a
         lcall writed

         mov a,#42h          ; send mwrite command
         lcall writec
         mov r0,#0           ; send 256 loops of 4 bytes each
```

```
clr400lp mov a,#20h          ; for a total of 1024 ascii spaces (20h)
         lcall writed
         lcall writed
         lcall writed
         lcall writed
         djnz r0,clr400lp

         mov r1,#28          ; send 7168 zeros of data to clear out CGRAM area
                             ; (28 x 256 = 7168)
         mov r0,#0           ; from 0400h to 0FFFh and the graphics area from
         clr a               ; 1000h to 1FFFh.
clrzeros lcall writed
         djnz r0,clrzeros
         djnz r1,clrzeros

;------
         mov a,#5Dh           ; Although I don't display it, set cursor format
         lcall writec
         mov a,#04h          ; five pels wide
         lcall writed
         mov a,#86h          ; Vertical cursor size=7 pixels block cursor
         lcall writed
;------
         mov a,#59h          ; display ON command
         lcall writec
         mov a,#00010100b    ; no layer3, show layer2, show layer1, no cursor
         lcall writed
;------
         mov a,#5ch          ; Set cgram address command
         lcall writec
         mov a,#00h          ; CGRAM address = 0400h
         lcall writed
         mov a,#04h
         lcall writed
;--------------------
; AT THIS POINT THE DISPLAY IS CLEARED AND ON, CURSOR IS OFF, AND EVERYTHING
; SHOULD BE CLEARED OFF.


                         ; PUT HELLO AT THE TOP OF THE SCREEN

         mov a,#46h          ; set cursor address to top of text block (address 0000h)
         lcall writec
         mov a,#00h          ; Remember that when sending addresses, you send the
                             ; less significant byte first, and the more significant
                             ; byte last.
         lcall writed
         mov a,#00h
         lcall writed

         mov a,#42h          ; send mwrite command
         lcall writec
         mov A,#48h          ; "H"
         lcall writed
         mov A,#45h          ; "E"
         lcall writed
         mov A,#4Ch          ; "L"
```

```
          lcall writed
          mov A,#4Ch          ; "L"
          lcall writed
          mov A,#4Fh          ; "O"
          lcall writed


                              ; Set cursor address (address pointer) to top
                              ; row, in byte column 5 (byte column just after
                              ; the "O" in HELLO, but on the top line) at address
                              ; 1005h.
          mov a,#46h
          lcall writec
          mov a,#05h
          lcall writed
          mov a,#10h
          lcall writed

          mov a,#42h          ; send mwrite command
          lcall writec
                              ; Draw off on off on off on off dots on top
                              ; row right after HELLO.
          mov A,#01010101b    ;
          lcall writed
          mov a,#0ffh         ; DRAW ALL ON dots right after that.  Recall that we have
                              ; set the address pointer to autoincrement after each read
                              ; or write.
          lcall writed

freeze    sjmp freeze         ; halt the firmware right here.


;----- COMMAND AND DATA WRITE SUBROUTINES FOLLOW ----
; -------writec-------------------------------------
writec      equ $             ; write accumulator to command register (address 1)
            setb address      ; select address = 1
            mov databus,a     ; send accumulator data to databus
            setb minusrd      ; raise the read, just in case
            clr minuscs       ; drop -CS
            clr minuswr       ; drop -wr bit low
            setb minuswr      ; raise -wr line
            setb minuscs      ; raise -cs
            orl databus,#0ffh ; tristate the bus
            ret



; -------writed-------------------------------------------
writed      equ $             ; write accumulator to data register (address 0)
                              ; DOES NOT AFFECT ACCUMULATOR VALUE.
            clr address       ; select address = 0
            mov databus,a     ; Put accumulator to data bus
            setb minusrd      ; Ensure the read line is inactive high
            clr minuscs       ; drop -CS
            clr minuswr       ; drop -wr bit low
            setb minuswr      ; raise -wr line
            setb minuscs      ; raise -cs
```

```
            orl databus,#0ffh ; tristate the bus
            ret

;-----------------------------

            END
```

Please refrain from sending me your code version that doesn't work, unless you are willing to pay for engineering consulting work. I have not included algorithms for line-draws, or ellipses, or other such features. I've done this work on my GDISP3 product already, and don't freely give the code or information away. You may license the code from me for a negotiated fee, or buy a completed GDISP3 daughter card from me.