VIDEO-DVM

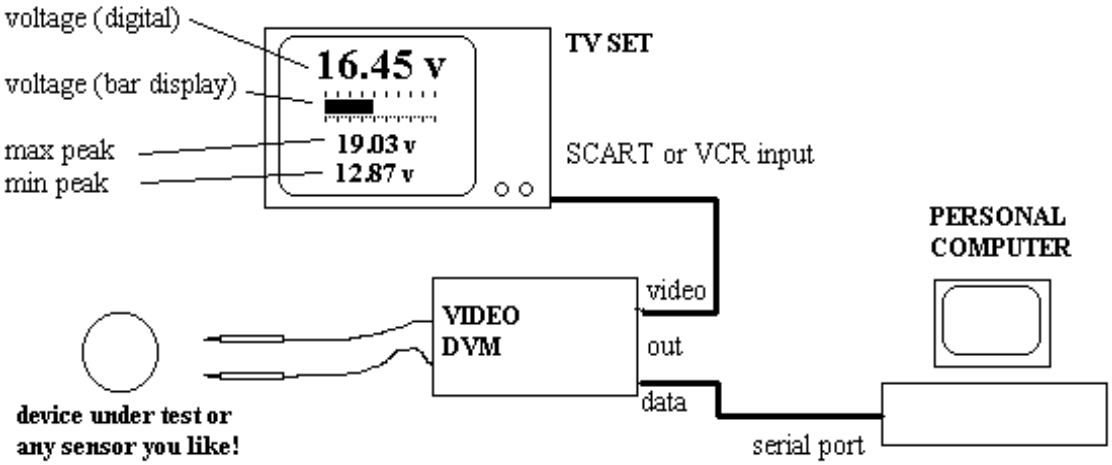
10-LINES DESCRIPTION

a.riccibitti@ra.nettuno.it
www.geocities.com/CapeCanaveral/Launchpad/3632

Video-DVM is a very cheap DVM that shows how an output as complex as a videocomposite signal can be generated entirely in software: two I/O pins and three resistors are all the hardware required. Connected to any TV set it displays voltages, included max and min peaks, using both giant digits and an analog bar-display . A serial data output for computer data logging is provided, too.

The micro is the Atmel's AT90S1200, ideally suited for hobbysts thanks to its 512 words flash memory, public programming protocols, free assembler and simulator available at www.atmel.com.

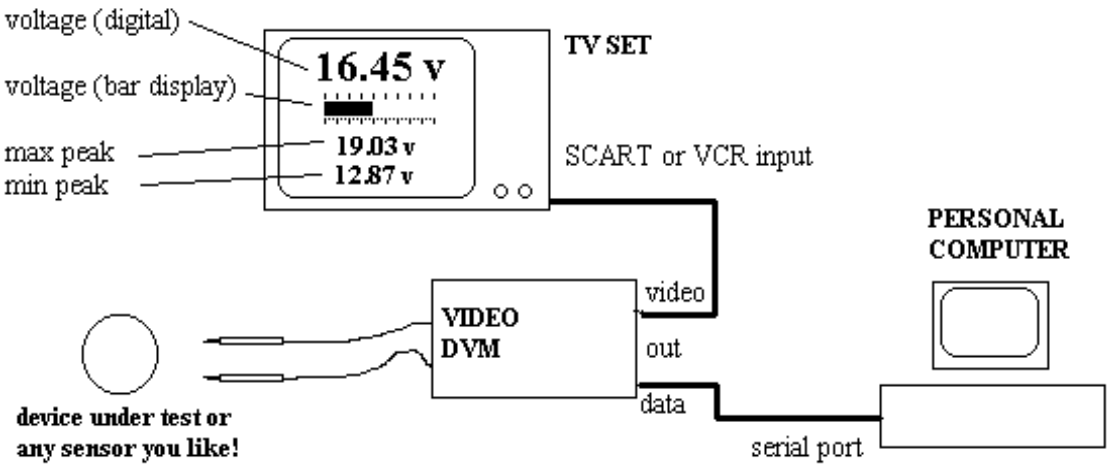
The circuit is not only a working project but also a guideline for any application using a tv set as giant display: all the hard work (interrupt driven, time balanced display software) is ready made, letting even novices to modify the code to build anything ranging from a multimeter to a frequency meter to a game scoreboard to a watering timer to a video pattern generator to a weight scale to....



Video-DVM

a.riccibitti@ra.nettuno.it
www.geocities.com/CapeCanaveral/Launchpad/3632

"building the Video-DVM can be very instructive, a major characteristic in these days of hyper-specialized ICs that leave almost nothing to fantasy and exploration..."



Name a few of the most useful characteristics of microcontrollers and you will name measure, control, display and connectivity. Add latest RISC technology with a lightening fast instruction cycle time (that allows even a video signal to be generated in real time) and the capability to be flash programmed without costly equipments and -*et voilà*- you have the Video-DVM, a cheap, funny, giant-size display volt meter with built in serial interface!

The Video-DVM displays voltages on your TV screen, both with giant digits and analog-bar; records maximum and minimum peak; it also sends measured data to a personal computer through a serial interface.
It sports :

- *An analog input capable to measure voltages in the range 0..4 V;*
- *A video display output capable to direct drive any ordinary TV set via the SCART (VCR) input;*
- *A serial data output, to log data on a personal computer;*
- *Needs only a single +5V, 15 mA typ. supply.*

With his big on-screen display, the Video-DVM is great to show to a large audience practically anythig that can be converted to volts, from temperature in a serious phisics experiment in the classroom to a *love-meter* for a party.

But the Video-DVM is a lot more than a funny toy:

It is really cheap and easy to modify to suit your needs, letting everyone to experiment with video signals and data display. The programming itself meets the severe constraints of real time direct video synthesis, digital conversion, serial data timing, and reduced code size. The analog to digital converter can be easily connected to a variety of sensors, or can be replaced with new code to display time or count pulses. Last but not least, thanks to its serial datastream the Video-DVM can also be used as a simple data logger.

All the hard work of interrupt-driven raster generation, serial data routines, ADC converter driving is ready made, letting anyone to modify the code in little steps to suit any specific need or brilliant idea.
In other words, building the Video-DVM can be very instructive, a major characteristic in these days of hyper-specialized ICs that leave almost nothing to fantasy and exploration.

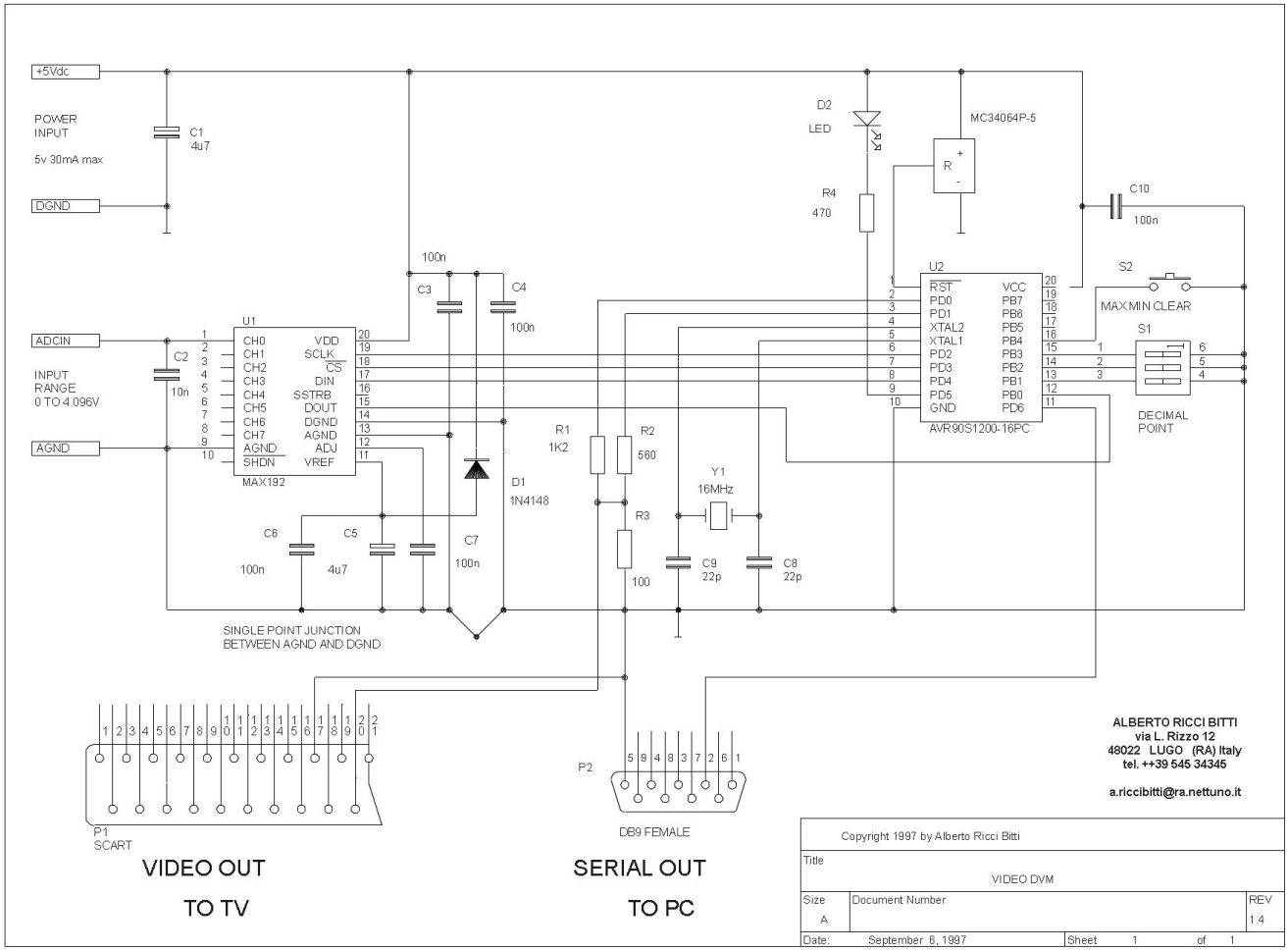
The hardware circuit

The circuit is built around the new Atmel's AVR 90S1200 microcontroller and the Maxim's MAX192 analog to digital converter.
Despite to its 16 MHz clock and Flash program memory and internal EEPROM, the AVR is very cheap, and is ideally suited for hobbysts: all the software needed (included a powerful simulator) is available for free at the Atmel's web site (www.atmel.com). The chip can be reprogrammed at least 1000 times in a flash, using one of the many programmers already available (some are advertised on Elektor's pages). The detailed flash programming protocol is available on the web site, too.
The chip has 32 bytes of RAM, 64 byte of EEPROM, 512 words of program memory, and an 8 bit timer. The instruction set is coincise and very well balanced, and thanks to an Harward RISC architecture even a complex task as the one described here is accomplished using only about 400 instructions.
The instruction cycle time is very short, letting software video synthesis possible.

A simple two bits, asymmetrical DAC built around three resistors feeds the composite video signal at a standard level of 1 Vpp suitable to be input to any TV set with a SCART connector or an AV (VCR) input.
Even the serial datastream (ASCII data at 1200 baud, no parity, 8 data bits, 1 stop bit) is generated by software. The level is TTL: most personal computers work with TTL levels as well as with standard RS232 levels, provided that the connecting cable is not too long, so I found a voltage translator like the MAX232 not necessary. If you want, you can add it externally; in that case you must invert the polarity inverting the "set bit" and "clear bit" instructions in code (the position is clearly annotated on the listing).
All input pins have internal pullups and high current sink, so the “max-min clear” button and the three decimal point selection dipswitch or jumpers are connected directly between the pins and ground.
A led, flashing each time a measurement is completed, is connected to an output pin.

The voltage is read using a 10 bit serial ADC converter. It has its own voltage reference set a 4.096 V. Four I/O lines are needed for data tranfers (Data input, Data output, Chip enable, Serial clock). Even if it is specified for 10 bits, the ADC supplies two more "sub LSB" bits: this means that 12 bits are effectively read from the ADC, altough only 10 bits have the guaranteed precision. Nonetheless, I found these two extra

bits very precise. All 12 bits are shown on the display, covering the span from 0.000 to 4.095 volts with direct reading of millivolts. Only one of the eight inputs available on the MAX192 is used, but the software driver is already capable to read any input you want. With minor changes, you can set up the Video-DVM to measure four voltages, three with digital display and one as an analog bar.
The circuit is operated at 5 volts, stabilized; a reset generator (MC34064-5V) is connected to the reset pin to protect the chip from brown-out during power on and power off.



Direct Video Synthesis.

From the hardware point of view, the hardware involved in video synthesis is exceptionally simple: two output pins and three resistors, allowing for four signal levels to be generated (sync, black, white, light grey). From the software point of view, video synthesis requires a very fast instruction cycle (here 62.5 nS) and a carefully timed, instruction-balanced, hand tuned, optimized code.
In order to achieve proper generation of the complex video signal, a robust timing system is absolutely necessary. Even a single 16 MHz clock cycle (62.5 nS) delay is clearly visible on the display, so you have to think twice before putting down even a single instruction.

First of all, we must choose a suitable time base: choosing the raster line duration (64 uS) as time base let us to build the entire frame a line after another (the non-interlaced frame consists of 312 lines), as well as placing easily the horizontal sync pulse (a few microseconds at start of each video line) and the vertical pulse (a few lines at start of each frame).
The only timer avalable on the micro is capable to generate repetitive interrupts every 16 uS without the need of reloading; that is, four interrupts every raster line.

At each fourth interrupt we make a new video line: to do this, each time the interrupt routine is executed, a counter is incremented and every four interrupts we are (almost) at the start of a new line.
"Almost" because we must take care about the time needed to service the interrupt routine upon interrupt request. This time is not constant, depending on the instruction being executed at the time of the interrupt generation. Some instructions have a longer execution time than others, so the interrupt service time can vary in an unpredictable way, distorting the display.
The best workaround I found for that problem is to go in sleep mode just before the fourth, critical interruption happens: the following interrupt will then wake up the micro with constant, known timing.
I have named the two output pins CsyncBit (composite sync) and VideoBit.
Putting both Csync and VideoBit to logic level zero, the video signal is at 0 volts (sync level); with only CsyncBit high, we get the black level; with both CsyncBit and VideoBit high, we get a white display.

Not surprising, all the video generation works around the timer interrupt routine. Every fourth interrupt a new line is started with the sync pulse; then the repetitive housekeeping (counting of lines, serial communications) routines are executed. All this work is done so quickly that we have to add a delay loop to wait for the start of the visible portion of the video line.
At this point, a multiple jump structure is executed to determine what kind of line we are on.
The jumps are based on line number (vertical position), so that we can decide if the current line will show current voltage (in that case we call the character display routine), or the analog bar (bar routine or bar ruler routines), or small numbers showing the peak values (character display routines with reduced size parameter). Of course, there is also a "void" line routine for blank lines in between.

For blank lines, the interrupt routine simply ends, bringing control to the main program; for display lines, the control is not released until the entire line is drawn with the appropriate graphics; for vertical retrace lines, the display is blanked and the sync polarity inverted.
At the end of display lines, or at most shortly after the third interrupt, the micro is freezed using the sleep instruction, waiting for the fourth, time-critical interrupt.

During the "housekeeping" interval at start of each video line the serial data is output; with no more hardware timers available, the serial port is made in software. Every 13 horizontal lines a new bit is transmitted, achieving a baud rate of 1200 bps: that is perfectly adequate to the small amount of data we have.
Data is sent as an ASCII string, terminated by a CR (ASCII 13) character. You can use a QBasic program, or any terminal program (as the Hyperterminal supplied with Windows 95 or the Terminal supplied with Windows 3.11) to gather the data.

All the video generation routines are carefully hand-tuned. Do not remove the NOPs from code, they are here to balance execution times! When executing branches, execution times associated with every possible path must be

carefully considered. Fortunately for you, all the code needed to handle the video signal is ready made and works by himself under interrupt.
Feel free to modify or add code to the main program to suit your fantasy. Even the multiple jump structure that select the display contents is not too difficult to understand, so you can easy build hundreds of different displays, with charaters of any size or with multiple analog bars.
All the routines and variables have long names, hopefully self-explanatory. The code is intensely commented. Please remember that only one subroutine call is allowed at a time.

Custom Fonts

You can redefine fonts to any shape you like; this is particularly useful for the measurement unit. You can display instead of the volt (V) any other letter or shape that fits in a 8x5 pixel matrix. Font shapes are stored in EEPROM according to the definition file FONTS.INC. The file is automatically included during assembly and results are compiled to VIDEODVM.EEP; don't forget to separately program the EEPROM once you programmed the FLASH.
Fonts are stored rotated 90 degrees clockwise, into 5 adjacent bytes.
In order to make the font shapes graphically readable, the file VIS_BYTE.H is also included: it redefines all single byte values to strings of underscores and "o": look at the file FONTS.INC to see as effective this simple trick can be.

Files

To program the chip you need the following two files:

VIDEODVM.EEP	EEPROM content in Intel hex format
VIDEODVM.HEX	FLASH content in Intel hex format

If you want to see the inner workings, or if you want to customize code the following are the source files:

VIDEODVM.ASM	Main Assembly source
1200DEF.INC	Atmel register definitions for AT90S1200
VIS_BYTE.H	Visually readable byte values redefinitions
FONTS.INC	Font shapes (EEPROM)
Graphic files:	
ORCAD	directory content Schematic + libraries for ORCAD
SCHEMATI.GIF, SCHEMATI.PCX, SCHEMATI.DXF, SCHEMATI.EPS, SCHEMATI.TIF	The same schematic diagram in five graphic formats...
PCB_01.GBX, PCB_02.GBX	Printed circuit board in Gerber format
Text files:	
VIDEODVM.DOC	This text in MS-WORD format

Building the circuit.

All the components used are of the consumer class, so they are cheap and easy to find. The MC34064 can be replaced with any three pin, 5V power reset generator; even the MAX192 has equivalents with the same data protocol with different resolutions.
If your ordinary distributor does not have the Atmel AT90S1200-16PC yet, due to the fact that this component is new, it can be requested to the Elektor’s advertisers that sell also the programmers for the Atmel flash micros.

To build the circuit is an easy task; the circuit is simplicity itself and if you don't like to wait it can be built even without a ready made PCB, if you only take care that the ground connections of the ADC be as short as possible and leave the digital ground (pin 14 of the MAX192) separate from the analog ground (pins 9, 13 of the MAX192) joining them on only one point near the power supply.
Likely you will have your preferred sensorware connected to the inputs: be sure that the range never exceed 4.096 V or goes below 0 V, placing a series resistor (about 1k) and crowbar diodes if that range can be exceeded.

Once powered up, the flashing led will tell you that the microcontroller is working just fine. The LED flashes once per conversion.
Now is time to connect the Video-DVM to the TV set: for TV equipped with SCART connectors the pins for video input are 20 (video) and 17 (ground). The TV must be disconnected from the mains while you insert the SCART plug.
Set the TV in AV input mode using the remote control: older TV sets may not have this capability, in that case the AV input can be forced pulling up the SCART pin 8 placing a 1k series resistor from the pin to +5...12V.
You can now fit the jumpers for the selection of the decimal point position, play around with various voltages to the ADC, verify the effect of the max-min clear button, or simple contemplate your new electronic toy for a while.

Last step is to connect the personal computer. Start the Terminal program in the Accessories folder (Windows 3.11) or the Hyperterminal program (Windows 95). Set up the connection to 1200 baud, no parity, 8 data bits, one stop bit (1200,n,8,1).

Connect the ground and the serial datastream to the RX pin as follows:

9 pin connector: 2=RX 5=GND

25 pin connector: 3=RX 7=GND

If you are using other terminal programs or Basic, you may have to solder some jumpers to the serial connector in order to make the serial port available:

9 pin connector:
7=RTS with 8=CTS;
1=DCD with 6=DSR with 4=DTR

25 pin connector:
4=RTS with 5=CTS;
8=DCD with 6=DSR with =DTR

Each time the LED flashes, you must see a new row of ASCII data on the screen.

Expansions (to be continued!)
I hope that this simple circuit serves as a starting point for personalized applications.
I tried to make the code highly structured (routines have only one entry and exit points), uniform (all interrupt routines use parameters Arg1-Arg4, and all Main program routines use Main1-Main4), self explanatory, deeply commented and modular, so you should be able to modify it to suit your needs.

The technique illustrated here is very powerful, and thanks to both measurement and display capablities available on-board the expansions are limitless, and they can evolve as your knowledge of programming increases.
Precision is good enough to transform it in a complete multimeter; or you can display up to eight voltages at once in a digital or analog display; or you can count pulses, laps, time, frequencies and so on. Implement a watt meter or symple count how long the fridge or the heating central have been on today.
Using the software given as a skeleton, you can even generate simple video patterns (as a

checkerboard) for TV repair; or make a simple video generator with your logo for a personalized VHS erasing signal. You can even use it as a countdown counter for New Year's Eve; or implement a random number generator and have a big, highly visible bingo numbers generator; or just add some pushbuttons and have a cheap game scoreboard, or a quiz machine. Counting pulses at each line start you can generate tones at audible frequencies. Adding a relay you can set up a threshold switch as a thermostat, or a flower watering control. Adding a serial EEPROM you can store lots of data to load in your computer o view on screen....