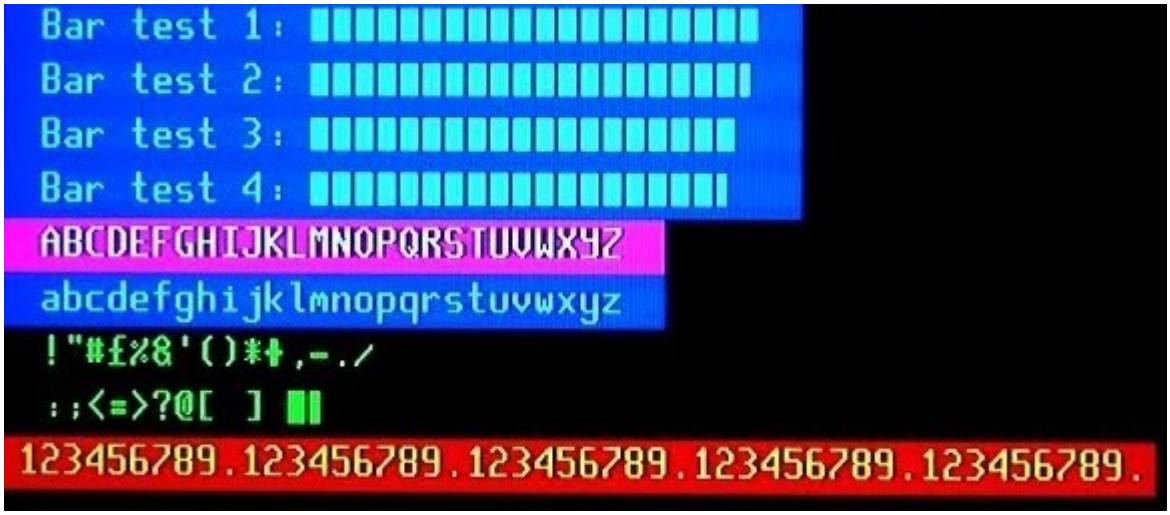
BASCOM-TV

With this software module you can generate output directly to a TV - via an RGB SCART connection - from BASCOM (AVR), using a just few resistors and a 20 MHz crystal.

Write your program with the powerful but easy to use BASCOM compiler for AVR, and display whatever you need on any TV which has a SCART socket. Useful for debugging as well as for the final product.



This is a photo of the TV display function working on a flat panel LCD TV set
The actual display is perfectly straight, some distortion is seen here caused by the camera optics.

Full development kit now available for commercial use. Includes built & tested board, pre-programmed sample IC, TV generation software module including all pixel data which can be edited, plus full support from the developer.

For non-commercial use simply use a "Kite" system which already has the TV code programmed into the IC and can hold up to six separate 4K programs selectable by jumpers or switches.

TV Code Features:

* Generates a 55 column colour TV character display from an AVR MPU without any extra ICs
* Connects via a standard SCART socket giving a sharp RGB output signal (not composite video)
* Flexible RAM use - display RAM can be as large or small as needed
* Completely interrupt driven software - transparent to user
* Character pixel data can be edited or replaced by the user to allow custom characters

The TV software can be customised to special user requirements (e.g. differing character sizes) but is time critical machine code - please use the contact page for advice on different display formats.

ICs supported:

This code is for the ATMega 48/88/168/328 ICs. A PCB is available for testing and / or production. The code can be ported to other ATMega AVR ICs which have 512 bytes or more RAM, and a clock of 16-20 MHz. See the contact details page to discuss your requirements.

BASCOM versions:

When using the company licence (which allows as many copies as you need) you will also require a registered copy of BASCOM to allow sufficient Flash program memory for most projects, as the binary include file for the TV output code is 2.5K, leaving only 1.5K of available space for your program out of the 4K maximum space allowed in the demo version of BASCOM.

How to use BASCOM with the TV software module

The TV display is very simple to use from a programming point of view. As it runs entirely within the ATMega chip, it uses internal RAM to hold the display data. This means that all you have to do to write to the display is write bytes to the RAM which is allocated to the TV screen area.

To start with you need to include the following 7 lines of code in your BASCOM program:

```
On Oc2a Tvinterrupt Nosave
Goto Main
!.org $100
$inc Tvinterrupt , Nosize , "tvinc.bin"
Return
Main:
$include "tv.inc.bas"
```

You don't need to worry about these statements, you can just cut & paste this into your program and it will work.

After including these lines of code you can make the TV display work simply by moving bytes to the screen area in the internal RAM.

The amount of RAM used by the display is very flexible. It can be any number of bytes up to the maximum possible which is 12 lines of around 56 bytes per line, maximum 672 bytes.

The minimum number of bytes which can be used is just one! This byte would be the "End of Screen" code which has to be the very last byte of any screen. In fact, if the interrupt is disabled, then no RAM is needed at all, and you will also have full CPU usage until you enable the interrupt again. To disable the TV software all you need to do is:

```
DISABLE OC2A
```

this will halt the TV code and allow all RAM to be used by your application, then

```
ENABLE OC2A
```

to start the TV code again. You will need to make sure the screen RAM area contains sensible display data before enabling the interrupts.

You can use all of the on-chip RAM to do calculations and for temporary storage, you only need to free enough RAM as you need for the screen while it is actually displaying.

For further examples of how to use the display RAM, see the following BASCOM programs:

- ADCTest6.bas

This program reads the 6 ADC inputs and displays the values as six bar graphs on screen in real time.

- BarTest6.bas

If the ADCs cannot be connected to anything interesting, this program displays six random bar

graphs to simulate the ADCs.

- SerialIn1.bas

This program reads characters from the serial port and displays them on the TV screen. If the serial port is connected via infra red or radio this provides a wireless TV display.

Bascom TV FAQs

Q: How can I add TV output to my BASCOM application?

A: If you are using an ATMega48/88/168/328 everything is already set up for you to use. If you want to use another ATMega IC, you need to refer to the technical information and make sure that the IC you are using has enough hardware resources (CPU speed, SPI port, RAM etc). You can't just add TV output to any AVR chip - ATTiny ICs are not supported as the TV code uses the hardware multiply instruction, and only ATMega ICs have enough RAM.

Q: What about low power applications?

A: Whilst TV output is enabled, power consumption will be around the maximum given in the data sheet for the IC at the speed and voltage used. TV output should be disabled when not needed, then the IC can benefit from all the low power and sleep modes available. This would be relevant to any device which is normally in low power mode, but can have a TV attached to display data only when required.

Q: What about CPU intensive applications?

A: As explained in the previous Q/A about low power, the TV output can be switched off (by disabling the relevant interrupt) so the full CPU power is available, however most applications can easily run in the spare time (approx 20%) of the CPU when running at 16 or 20 MHz.

Bascom TV Software Module - TECHNICAL INFORMATION

You do not have to read this information - BASCOM and the TV code will automatically set up the hardware as required. If you use the supplied PCB this ensures the TV output will work without any knowledge of the module. These technical details are for reference.

The TV code has various fixed hardware requirements as follows.

Clock: The clock MUST use an external crystal to produce a stable screen.
Normally 20 MHz but 16 MHz can be used (20% less columns).

SPI port: The SPI hardware is used by the TV code and cannot be used for other purposes while the TV code is running.

RAM use:
* Amount of RAM used can be very small - EndScreen code marks end of RAM used
* Lines are variable length so only visible characters [excepting space] use RAM

RAM Addresses:
* The address of RAM used by the TV code is fixed at $100 (start of RAM in ATMega 48/88/168/328 ICs)
* The first six bytes of RAM are used to store variables for the interrupt code
* The first byte of RAM used for the screen area is at address $106

Maximum RAM use:
* For a full screen of 55 characters by 12 lines, RAM used = 660 bytes
* In an ATMega48, used screen RAM will need to be kept below around 450 bytes for use with BASCOM

GPIO register:
In ATMega 48/88/168/328 ICs there is a "GPIO" register at $1E. Bit 0 of this is used by the code. The other 7 bits are unused and can be changed by the user software.

Pixel data:
* The pixel data used for the characters shown on screen starts at a fixed Flash ROM word address
* All pixel data can be edited or replaced by the user to allow custom characters

Timers:
* Timer 2, an 8 bit timer, is reserved for use by this code
* Timer 2 causes a 64µS interrupt and can be used for a system "tick"
* Timers 0 and 1 are unused

Reset and interrupt vectors:
* The interrupt vectors for Timer 2 compare match A and B are both used
* Timer/Counter2 Compare Match B vector points to the TV interrupt code

Control characters:
* End of line code = EndLineCode = $0D
* End of screen code = EndScreenCode = $0C
* Set colour to yellow chars on a red background = SetYellowCode = $10
* Set colour to green chars on a black background = SetGreenCode = $11
* Set colour to cyan chars on a blue background = SetBlueCode = $12

* Set colour to white chars on a magenta background = SetWhiteCode = $16

All of Port B is reserved:
* PB0 Sync
* PB1 Blue
* PB2 Red
* PB3 Green
* PB4 & PB5 [2] allocated SPI pins
* PB6 & PB7 [2] Used for XTAL

No pins on Ports C or D are used

Interrupt Code:
* Triggered by Timer 2 interrupt
* Runs every 64 uS
* Consumes up to 80% of CPU time [worst case]

Fuses:
Only the low fuse needs to be changed:
CKDIV8 must be high (unprogrammed) so clock will be at full freq (default 0 = programmed)
1 CKOUT Clock output (default 1 = unprogrammed) i.e. clock output off
1 SUT1 Select start-up time (default 1 = unprogrammed)
1 SUT2 Select start-up time (default 0 = programmed)
SUT1,2 = 11 selects Crystal Oscillator, slowly rising power (in case of PSU problems)
0111 in CKSEL 3210 selects full swing oscillator, slowly rising power so lfuse = $F7