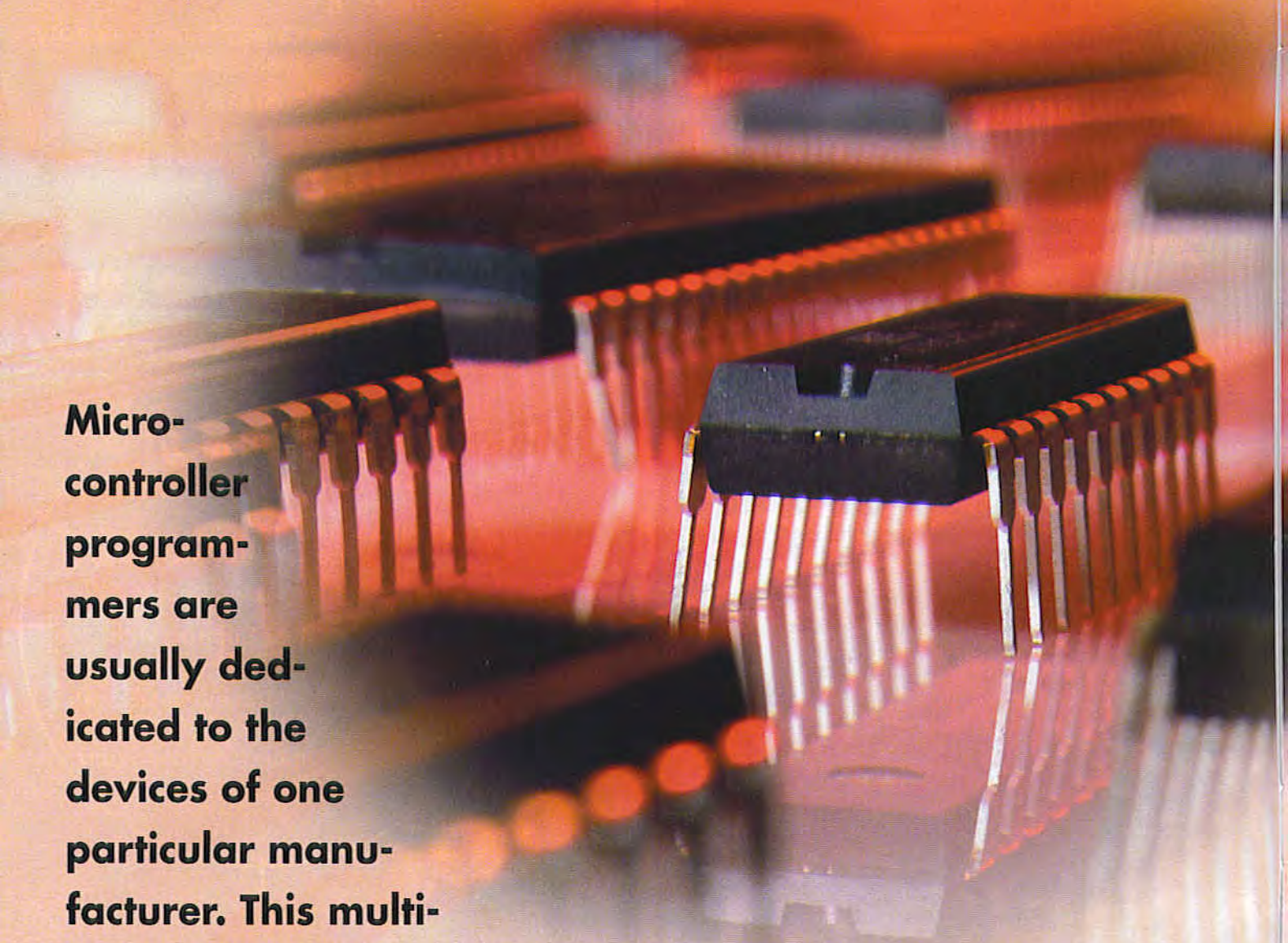
Multi Programmer

one size fits all

Andreas Oyrer



Microcontroller programmers are usually dedicated to the devices of one particular manufacturer. This multiprogrammer, however, can not only program the most important mid-range microcontrollers made by Atmel (and soon also Microchip), but also serial EEPROMs. Thanks to its USB interface, programming is simple, flexible and quick.

This multi-programmer is tailored to the requirements of the semi-professional user. It is not designed for the entire range of microcontrollers from one particular manufacturer, but rather for general use with standard 8-bit microcontrollers which have limited memory capacity. The hardware is capable of programming microcontrollers from more than one manufacturer (currently Atmel and Microchip) as well as serial EEPROMs. Since these use different programming algorithms and voltages, this is something of an unusual feature.

'Semi-professional' also means that the programmer is a development tool, and so must be suitable for use while debugging. The programmer must be fast, so that the job of getting software to work correctly does not become a chore.

The programmer must also be controllable, which is achieved by using a USB interface. HID (Human Interface Device) compatibility means that the device will work straightforwardly with versions of Windows from 98SE onwards. All that is needed to operate the multi-programmer is a spare USB port on the PC. The device takes its power from the USB port (it is 'bus-powered'), and so no mains supply is needed.

The microcontroller used does not have its own program memory, and so the firmware is downloaded directly from the PC over the USB cable when it is plugged in. Updating the firmware simply requires changing a file on the PC.

It is also possible to store the firmware in an EEPROM on the programmer board, accessed by the microcontroller on power-up. In this case the USB must be used to upload new firmware versions into the EEPROM.

A special feature of this device is that the microcontroller to be programmed does not have to be removed from the target circuit and put in the programming socket. The programmer has two ISP (in-system programming) interfaces available, one for Microchip microcontrollers, one for Atmel devices.

USB Microcontroller

At the heart of the hardware is the TUSB3210 (IC1) from Texas Instruments. This is an 8052-compatible microcontroller with a full speed (12 Mbit/s) USB interface, offering four I/O ports each with 8 port pins, a UART, a watchdog timer and an I^2C interface. The TUSB3210 does not have its own flash memory, and so the firmware must be reloaded every time power is applied. Software is loaded into the 8k-by-8 (8 kbyte) RAM memory by a built-in boot loader: this can be over the USB interface, or, alternatively, the software can be stored in serial EEPROM IC5 (a 24LC64). The EEPROM is connected to the I^2C interface pins SDA and SCL of the TUSB3210: its contents are read whenever the device is reset and copied to the TUSB3210's RAM. If the USB option is used a driver is required on the PC to send the software to the TUSB3210. Whether from the EEPROM or from the PC, once all the firmware has been copied into RAM the boot loader software disconnects from the USB. The program stored in the RAM is run and the device is then reinitialised over the USB.

Programming voltages

IC1 controls all the programming signals and voltages over its 32 I/O pins. In order to generate the programming voltages required by various microcontrollers the 5 V supply from the USB is converted to approximately 13 V using a step-up regulator. Normally the output voltage of the switching regulator would be a constant 12 V, but a diode in the feedback path of IC3 raises the output voltage by the forward voltage drop of the diode: this higher voltage allows PIC microcontrollers to be programmed.

The programming voltages are switched as required using p-channel and n-channel FETs. A voltage of 0 V, 5 V or 12 V can be made to appear on pin 1 or pin 31 of the programming socket. A voltage of 13 V is available for the MCLR signal on ISP connector K3, which is used for programming PICs. Diodes D5 and D7 reduce this voltage back down to 12 V: this lower voltage is used when programming Atmel microcontrollers.

TTL gates with open-collector outputs (type 74LS07) are used to drive the FETs. This allows a voltage of 0 V to appear between gate and source, ensuring that the transistors can be fully turned off. If used directly, the voltage on the I/O port pins could only rise to about 3.3 V, giving a gate-source voltage of 10 V: the transistors would then still conduct.

Some microcontrollers require a programming voltage on the reset or crystal inputs. In the case of the 90S1200, for example, a minimum voltage of $0.85 \cdot V_{CC}$ = 4.25 V (assuming a 5 V supply) is required on the reset input. Since the TUSB3210 runs from a 3.3 V supply it can only deliver a logic high level of 3.3 V; the remaining gates in IC4 are used to produce a high level of over 4 V.

Programming socket

Most devices can be programmed directly in socket IC5. Crystal X2 provides a clock source for Atmel 89Cxx and 89Sxx type microcontrollers. Because of the limited number of I/O pins offered by the TUSB3210 only a

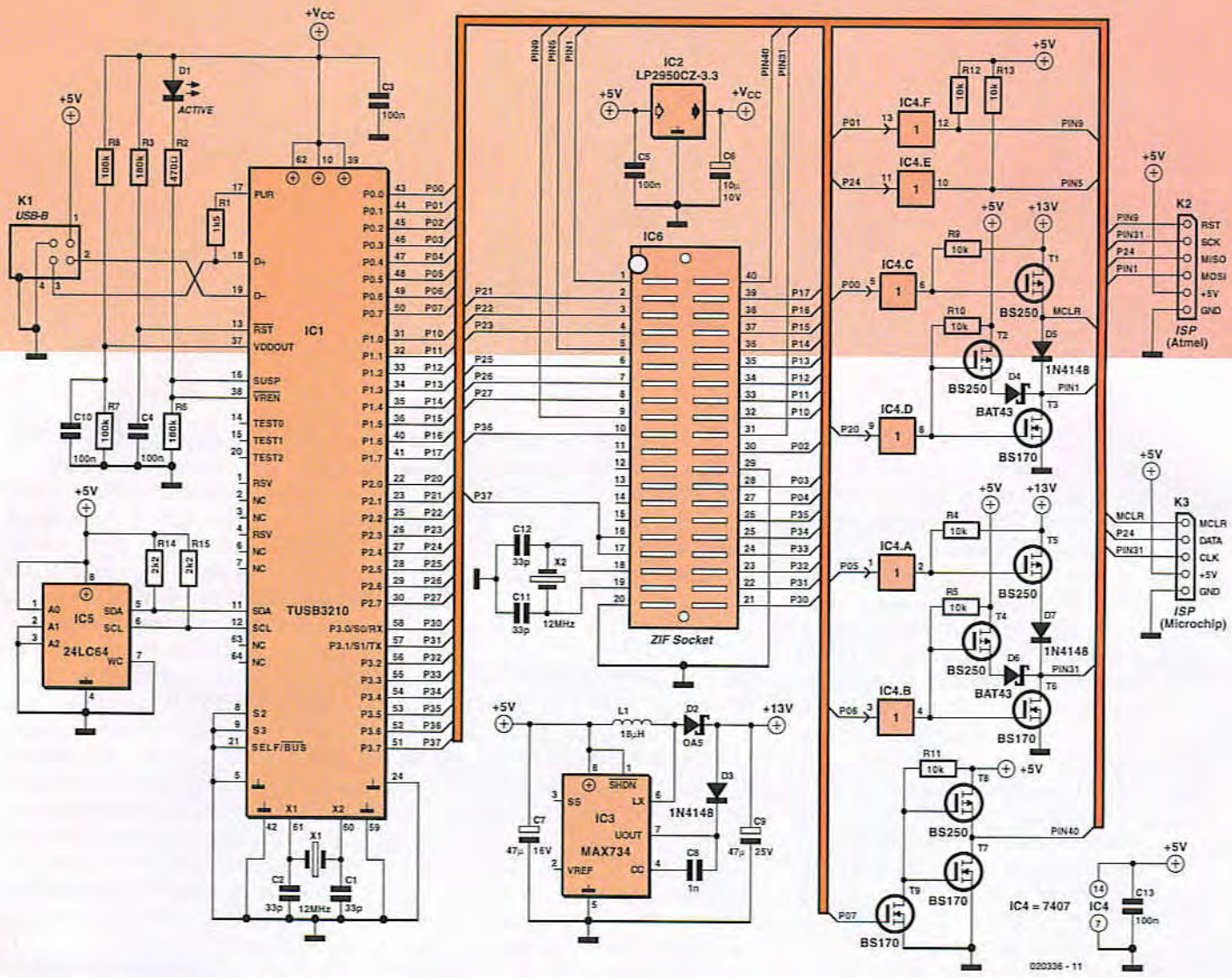


Figure 1. The TUSB3210 includes I²C and USB interfaces, and so the only additional hardware needed is a step-up converter and a few MOSFETs.

certain range of microcontrollers can be programmed. Other (larger) microcontrollers can, however, be programmed using the ISP connectors.

On closer inspection, you will notice that there is no ground connection on pin 10, as would be required, for example, by an 89Cx051. Since the supply current is not particularly high during programming, it is sufficient to provide the ground via port pin P3.6 of IC1. The voltage does indeed rise a little above ground, but remains within reasonable limits.

PC software

The software running on the PC is written in Delphi 7. The menu language can be set (under Setup, as shown in **Figure 2**) to English, French or German. This setting, along with all other settings, is stored in the registry and automatically recalled when the program is next run.

Also under Setup are options to control whether signature bytes are read, and whether the device memory contents are verified after programming.

The device type is set under the Device menu item (see **Figure 3**). There are two sub-menus available here: Socket (i.e., IC5) and ISP Connector (i.e., using connectors K2 and K3). Currently only the Socket option is available. The next choice is between Atmel MCU and serial EEPROM. Under Atmel MCU the available ranges of microcontrollers are 89Cx051, 89C5x, 89Sx and the two AVR microcontrollers, 90S1200 and 90S2313. Of course, functions such as lock bit and fuse bit programming are supported.

On 89C5x microcontrollers only lock bits 1 and 2 can be programmed, since there are not enough port pins available on the TUSB3210 to allow programming of lock bit 3. None of the lock bits can be programmed on 89Sx-type microcontrollers. If a type 90S1200/90S2313 microcontroller is selected, then when configuring the second fuse bit two variants (the RCEN fuse bit and the FSTRT fuse bit) are displayed. If the microcontroller type is detected, the text changes to show the name of the fuse bit supported by the device in question.

If a serial EEPROM device is selected, there is in some cases more than one device type ending in the same digits (the digits correspond to the size of the memory). Devices in the 24AAxx and 24CxxC series with the same capacity differ, however, in their page size: this is the number of bytes that form one 'row' in the memory that can be programmed in one cycle (approximately 2 ms). The bigger the page size, the quicker the

overall programming operation.
All programming functions such as program, verify, erase, read, program EEPROM, read EEPROM, read fuse or lock bits and detect device are directly available either using buttons or under the Action menu (see **Figure 4**). If the microcontroller has been selected as autodetect, then it can be tested using Detect Device. This causes the signature bytes to be read from the device. The bytes are analysed and information including the memory capacity, programming voltage and exact part number are displayed in the upper right-hand corner of the window under Device.
When an action is selected, the signature bytes are first automatically read out of the microcontroller before the action is carried out. The signature byte test can be disabled by deactivating the Read signature bytes option in the Setup menu. This might be necessary if a fault in the microcontroller makes it impossible to read out the signature bytes.
The Read action reads the entire memory contents out of the device. The number of bytes to be read is determined from the information in the signature bytes or by the digits at the end of the part number in the case of a serial EEPROM. If, in the case of a microcontroller, the signature byte has not been read, then the maximum possible memory size for a device in the selected series is used. For example, if the 89Cx051 series is selected, then 4 kbyte will be used, since this is the memory capacity of the biggest device in this series, the 89C4051.
Under the Buffer menu you can choose whether the data in the buffer can be altered using the hex editor (Buffer editable). You can also select whether data in the buffer is synchronised with the data in the previously-opened file before any write or verify action is started (Update buffer from file).

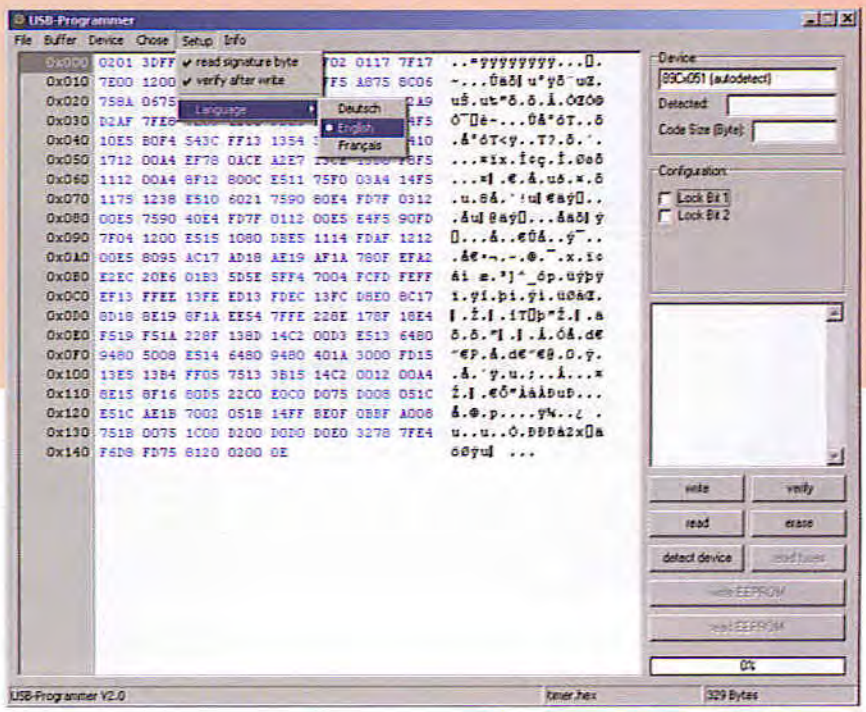


Figure 2. The setup menu

Protocol

A protocol is of course required to ensure that data communications between the PC and the programmer are kept in step. The first byte from the PC to the programmer contains information on the selected microcontroller or memory: the value 1 specifies the 89Cx051 series, the value 2 the 89C5x and 89Sx series. The second byte gives the selected action: 1 to read the signature bytes, 2 to erase and so on. Succeeding bytes contain further information, for example the programming voltage for an 89C5x microcontroller or the page size for a serial EEPROM. When programming, an extra byte is

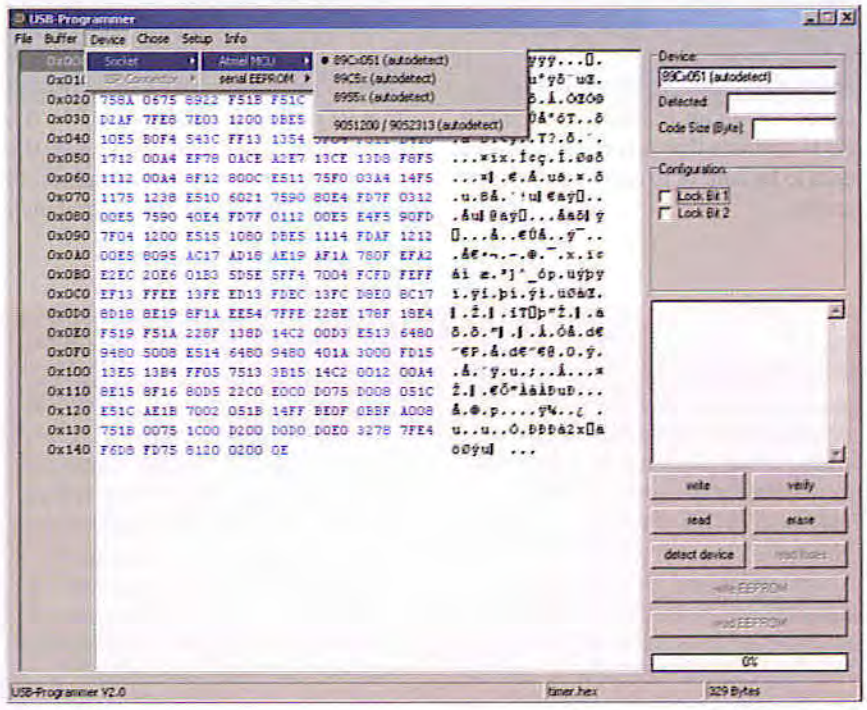


Figure 3. Device selection

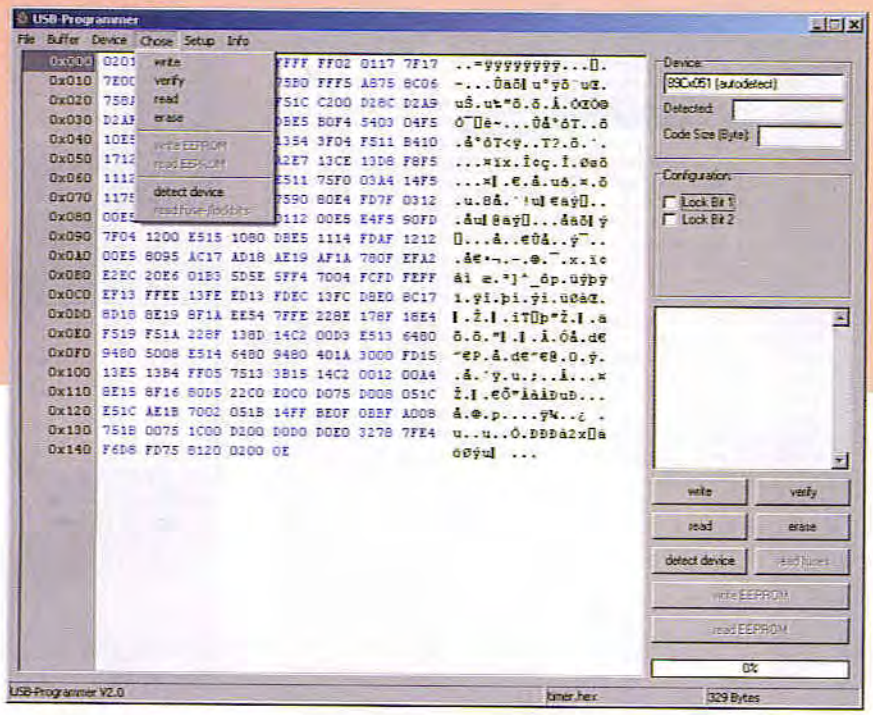


Figure 4. Using 'Detect Device' the signature bytes and other data can be displayed in the upper right-hand corner of the window.

used which gives the number of bytes in each packet. A further byte indicates whether the data packet sent is the last one (byte is zero) or whether further packets follow (byte is 1). When reading, a count of the number of bytes to be read is sent to the programmer. Generally the size is specified in kilobytes or kilobits. The software in the programmer can deduce how to interpret the value from the first bytes that were sent.

After each action the programmer sends a number of bytes back to the PC to indicate that it is ready for more data to be sent or to carry out the next action.

Programmer software

The software in the programmer was written using the Keil µVision2 C compiler. The main routine first disables the watchdog timer and then takes all pins on the programming socket to 0 V using the function *ResetProgrammer()*. Next the USB registers are set up.

When the boot loader software in IC1 completes, it disconnects from the USB by taking output pin PUR low so that R1 no longer pulls up to 3.3 V. The downloaded software must set the SDW bit in the MCNFG register to reactivate this output: the programmer will then reappear on the bus.

The PC host then sends a number of SETUP tokens to identify the device and configure its USB interface. These tokens are processed by endpoint 0. The data transferred includes the unique address for the device, which is subsequently used to communicate with the programmer. Various descriptors are also transmitted to the host during the setup phase, providing information about the device and its functions and characteristics. These include the report descriptor, which in this case identifies the programmer as an HID-compatible device.

Once all the descriptors have been sent to the PC, initialisation of the USB interface on the device is complete and it is ready to operate. The direction of data transfer is specified by a token. If the TUSB3210 detects an IN token, then data, such as status information or a data packet, is to be sent from the programmer to the host. If an OUT token is received, then the data packet is unpacked by the programmer and the payload programmed into the device (assuming that that is the selected action).

HID

The advantage of initialising as an HID-compatible device is that no special Windows driver is required to communicate data between the PC and the programmer. Windows versions from 98SE onwards support this standard.

Under the HID standard, data is exchanged in so-called reports. During USB enumeration the PC provides a number of descriptors. The device descriptor includes information such as the Vendor ID (VID), Product ID (PID), and the USB version supported by the connected device.

The configuration descriptor includes information on the current consumption of the hardware and the number of available endpoints. The report descriptor gives the size and number of reports to be exchanged between the PC and the programmer. It specifies how many bytes are to be sent or received and the function of the attached device (mouse, keyboard, joystick, memory stick etc.). More detailed information on USB and HID can be found on the USB homepage at www.usb.org/home.

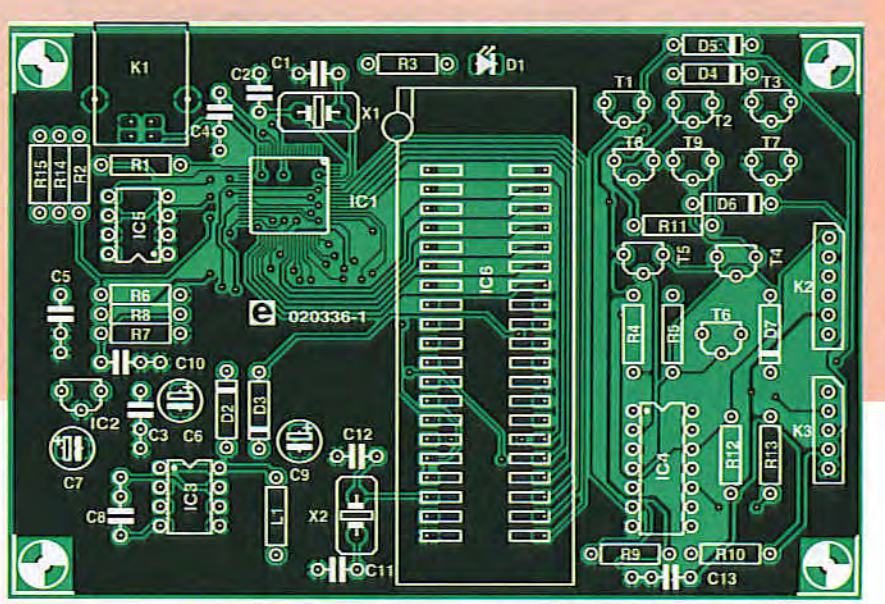


Figure 5. Component mounting plan for the double-sided printed circuit board.

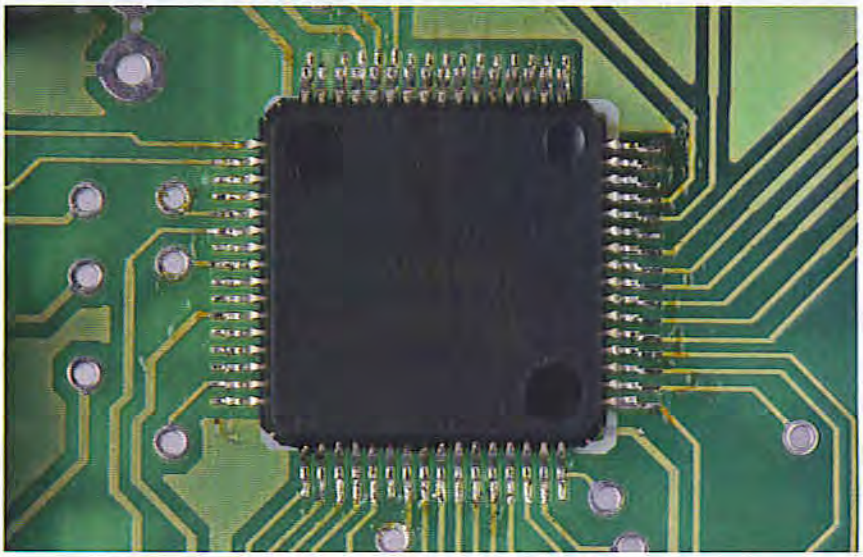


Figure 6. Tiny and tricky to solder: the USB microcontroller comes in a 64-lead SPFP package.

Programming routines

The data received is decoded by the routine *DecodeProgrammerData()* in the file Prog.c. The first byte of the 64-byte report contains the code for the selected microcontroller, while the second gives the desired action. These values are used to call one of a number of different programming algorithms for different devices, as given in the microcontroller data sheets.

Each device series has its own power-up routine which applies power to the correct pins and sets the programming signals used to defined levels. Once an action has been successfully completed, the routine to reset the programmer is called, which sets all the signals on the programming socket back to 0 V. Since in general it is desired to program more than the 64 bytes contained in one report, the PC must send a further data packet to the programmer as soon as the previous one has been processed. The programmer sends a defined message to the software running on the PC to notify it that the next packet can be sent.

The PC then prepares the next report packet and sends it to the hardware. The last packet to the USB programmer contains a zero byte. When the device is read, data is also transferred in reports of 64 bytes each, where the first byte gives the number of valid bytes in the packet.

The above description of the programmer software can give only a broad overview of its operation. More detail can be found in the thoroughly-commented and clearly-structured software itself.

Construction and operation

Populating the printed circuit board would be child's play were it not for the tiny TUSB3210 in an S-PFP-G64 package with many fine leads. Soldering such SMD ICs requires not just a steady hand but also nerves of steel and a good deal of confidence. After fixing the IC in place with a drop of glue, you must take the soldering iron and do what you would normally try to avoid at all costs: rather than soldering the IC's leads to their corresponding pads, solder all the leads together. This should be done as quickly as possible, so the device does not get too hot. When this big short-circuit and the IC have cooled down, lay some unused desoldering wick across the pins and use it (and not a solder pump!) to suck away the excess solder. Again, take care not to get the IC too hot. Finally, check with a magnifying glass under a good light, and using a multimeter, that all the pins are well soldered and no longer shorted to their neighbours. Once the TUSB3210 has been correctly soldered to the printed circuit board, the remaining construction is relatively straightforward. All the ICs (except for the tiny voltage regulator) are provided with sockets. Even the zero insertion force socket should be fitted in a

Devices currently supported

The programmer firmware can easily be updated to the latest version at any time: you simply need to write the new firmware into the EEPROM and, if necessary, replace the software on the PC.

Currently the firmware is capable of programming the microcontrollers and EEPROMs listed below. An update is expected shortly that will support PIC microcontrollers and ATmega devices.

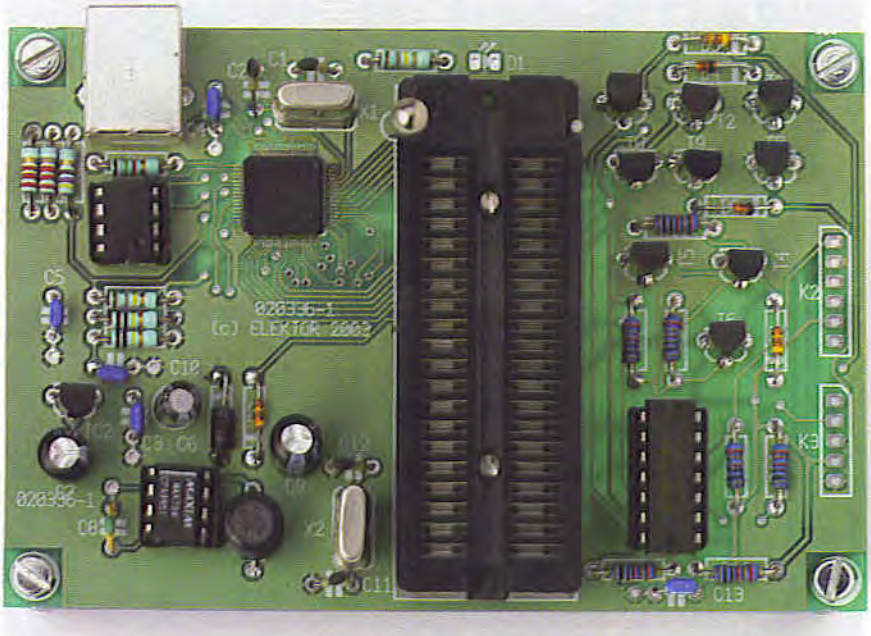
This, as well as all future updates, will be included with the PC software available for free download from the Elektor Electronics website under product number 020336-11, see month of publication.

Atmel microcontrollers:

89C1051, 89C2051, 89C4051, 90S1200, 90S2313, 89C51, 89C52, 89C55, 89LV51, 89LV52, 89LV55, 89S53, 89S8252

EEPROMs:

24xx00, 24xx01, 24xx02, 24xx04, 24xx08, 24xx16, 24xx32, 24xx64, 24xx128, 24xx256, 24xx512



socket rather than being directly soldered to the board.

When construction is complete and you have inspected the board, you can carry out the first test. If a ready-programmed EEPROM is available, no Windows driver will be required. If the programmer is now connected to the USB, it should appear in the Device Manager as an HID-compatible device. You are now ready to program your first microcontroller.

If no serial EEPROM is fitted, the firmware must be downloaded over the USB. The TUSB3210 boot loader registers itself, Windows recognises the new device and will now need the Texas Instruments device driver. This driver (called the *TI Apploader Driver*) is not provided as part of the *Elektor Electronics* disk or download, but can be obtained for free from the TI website at www.ti.com. Select the directory with the file TUSB3210.inf and install the driver and then the file Aploader.sys from the same directory.

Finally you will be asked for the directory containing the firmware: enter the path to the file TUSB3210.bin. This will automatically be copied into the directory /System32/drivers, along with the file Aploader.sys. If the programmer is now reconnected, the driver will send the firmware from the file /System32/drivers/TUSB3210.bin; after a brief delay the code will start executing on the programmer. The programmer will now be re-enumerated as an HID device.

(020336-1)

COMPONENTS LIST

Resistors:
R1 = 1kΩ5
R2 = 470Ω
R3,R6 = 180kΩ
R4,R5,R9-R13 = 10kΩ
R7,R8 = 100kΩ
R14,R15 = 2kΩ2

Capacitors:
C1,C2,C12,C13 = 33pF
C3,C4,C5,C10,C13 = 100nF
C6 = 10µF 16V radial
C7,C9 = 47µF 16V radial
C8 = 1nF

Semiconductors:
D1 = LED, red
D2 = 0A5 or 1N5817 (Farnell # 573-097)
D3,D5,D7 = 1N4148
D4,D6 = BAT43
IC1 = TUSB3210PM
IC2 = LP2950CZ-3.3 or LE33CZ (Farnell # 302-4568)
IC3 = MAX734CP
IC4 = 74LS04
IC5 = 24LC64
IC6 = 40-way ZIF socket with wide slots
T1,T2,T4,T5,T8 = BS250
T3,T6,T7,T9 = BS170

Miscellaneous:
K1 = USB 'B' connector, angled, PCB mount
K2 = 6-way pinheader
K3 = 5-way pinheader
L1 = 18µH miniature choke
X1,X2 = 12MHz quartz crystal
PCB, available from **The PCBShop**
Disk, PC software and firmware / source code files, order code **020336-11** or Free Download