Version

2.9

FUJITSU MICROELECTRONICS EUROPE

Development tools for 8L, 16LX and FR Families

Flash-Kit Serial Programmer

UNIS, spol. s r. o.,
Jundrovská 33, 624 00 Brno

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH restricts its warranties and its liability for the **FlashKit and all its deliverables** (eg. software, application examples, target boards, evaluation boards, etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, the FlashKit and all its deliverables are intended and must only be used in an evaluation laboratory environment**.

1. Fujitsu Mikroelektronik GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days form the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Mikroelektronik GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.

2. Should a Product turn out to be defect, Fujitsu Mikroelektronik GmbH´s entire liability and the customer´s exclusive remedy shall be, at Fujitsu Mikroelektronik GmbH´s sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Mikroelektronik GmbH in original packing and without further defects resulting from the customer´s use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Mikroelektronik GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Mikroelektronik GmbH.

3. To the maximum extent permitted by applicable law Fujitsu Mikroelektronik GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.

4. To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH´s and its suppliers´ liability is restricted to intention and gross negligence.

   NO LIABILITY FOR CONSEQUENTIAL DAMAGES

   To the maximum extent permitted by applicable law, in no event shall Fujitsu Microelectronics Europe GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect.

Contents

Chapter

1

Introduction

The Flash-kit serial programmer solution allows to program the Fujitsu CPUs in both synchronous and asynchronous programming modes.

Asynchronous programming mode

In the asynchronous programming mode, the CPU is able to communicate with PC through standard RS232 interface. So, if the RS232 line interface is provided on the user target board and the target board CPU supports the asynchronous programming mode, no additional HW is needed for programming the CPU – the only thing you have to do is to connect your board to the PC. The only disadvantage of this mode is the limited programming speed – with most of the 16LX family CPUs, the communication is running on 38400 Bd (with 16 MHz crystal – for details, please check Chapter 7).

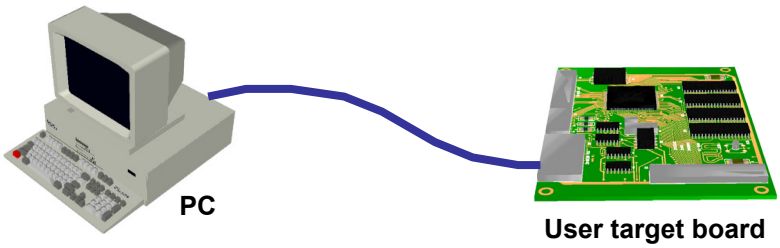


Figure 1: Asynchronous programming mode configuration

Note: If you want to use the asynchronous programming mode with your board, please include the „Serial interface logic recommendation“ schematic, which can be found in the Appendix of this manual, into your board schematics. This recommendation ensures compatibility of your board with the serial programming software.

Synchronous programming mode

For the synchronous programming mode, the serial programmer box must be used. Basically, this box converts the data from RS232 port or parallel port to synchronous data stream that is fed into the target system. No additional HW must be included on the user target board – the serial programmer box connects directly to the dedicated CPU pins. The biggest advantage of this mode is the high-speed communication – when using the PC parallel port, the effective communication speed is as high as 340 kbit/s.

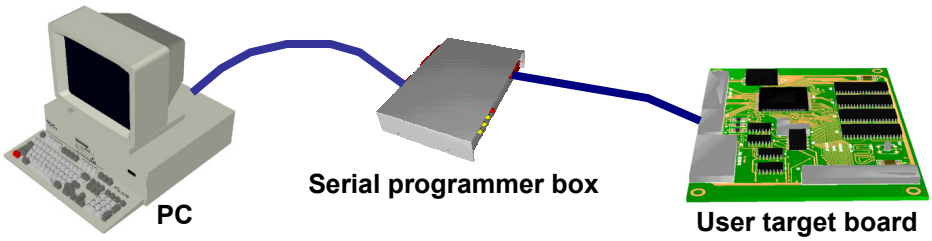
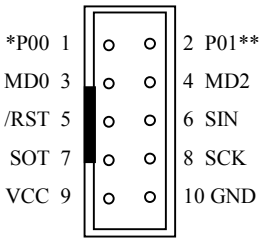


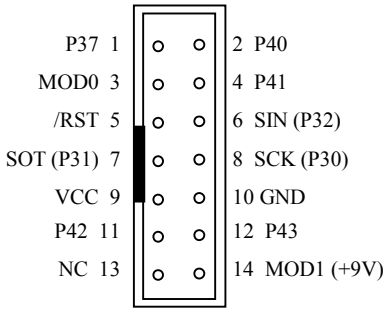
Figure 2: Synchronous programming mode configuration

Note: If you want to use the synchronous serial programming mode with your board, please include one of the following connectors into your board schematics (the one on the left side is for a 16LX family CPU, the one on the right side is for the target board with a MB89P935 CPU). More details about the recommended schematics can be found in Appendix B.
Warning: Before starting the design of your target board, please read the paragraph "General design rules for user target boards" in Appendix B carefully!



16LX CPU

*P80 for MB90F47x
**P81 for MB90F47x
SIN, SOT, SCK used corresponding to chapter 7



MB89P935 CPU

The programming cable connection for MB89P935 can be found in chapter 4, page 25

Serial programmer Features

- Supported CPUs: 8L family, 16LX family and FR families
- The serial programmer SW supports both synchronous and asynchronous programming
- The synchronous programming is supported by programmer box
- The box can be connected to the PC using both serial or parallel port
- The box to target communication speed is 500kbit/s
- Batch mode programming (interactive or non-interactive) is provided by the serial programmer SW
- The box firmware can be updated easily using the serial port
- Support for both 5V and 3V CPUs is provided by the box
- Box can provide supply to the target system

Technical Specification of the box

- Dimension: 126mm x 90mm x 22 mm
- Weight: 0.3 kg
- Operational temperature range: 0-55°C, storage temperature range: 0-70°C
- Power supply voltage: **9V**

  The polarity of the power supply is arbitrary.
  The input voltage is restricted to 9VDC. Higher input voltages can permanently damage the box, or the power supply and lead to malfunction.

- Power supply current (with no target board connected): **100 mA**
- Power supply current (with target board connected): **200 mA max**

We recommend to connect a dedicated power supply to the target board.
Only this way you can prevent the programmer box power supply circuitry from being overloaded incidentally. However, if you decide to use the separate power supply for both the programmer box and your target board, **uncheck the "Supply target" option** in the main window of the serial programmer.

Chapter

2

Installation

The installation of the Serial Programmer software is easy. Just run the SETUP.EXE program from the installation CD and you will be guided through the rest of the installation process automatically.

Important note: In Windows NT/2000, the Flashkit serial programmer must be installed by user who is logged into the system as an administrator. After instalation, the administrator also have to run the Flashkit serial programmer for the first time. This ensures the correct installation of the parallel port drivers into the system.

The "printer driver on LPT port" issue

In Windows NT/2000, it is not possible to have a printer driver set for the same LPT port, which is used by Flashkit SW.
If a printer driver is set for printing on the same LPT port, which is selected for synchronous communication via LPT in Flash-kit, the "Error opening device (Reason:error while opening port/device)" message will appear everytime when "Connect" function is invoked.

Solution:

After installing the Flashkit programmer SW, go to the Windows Start menu and select "Settings | Printers". For every printer installed on your computer, do the following steps:

1. right-click on the printer and select "Properties"
2. in the "Properties" dialog, select tab "Ports"
3. if the printer uses the same LPT port that you want to use in Flash-kit programmer, select another LPT port for the printer. If your computer has only one LPT port, you must select the port "FILE:"; all your printer outputs will be then redirected to a file.

Minimum requirements to PC

- Windows 95/98/NT 4.0/2000
- 32MB RAM (64MB recommended)
- one standard serial port for communication with the target system or the programmer box
- one standard parallel port for high-speed communication with the target system via programmer box
- Screen resolution 800x600
- ComCtl32.dll version 4.70 or higher

Known problems and limitations

If you have problems with the parallel port communication functionality, please check the following things:

1. Make sure the parallel port type is set to standard "SPP" in BIOS (equivalent names for this are "AT" or "Output only" on some machines)
2. On some machines, even if you select "SPP" parallel port type, Windows keep using the ECP port driver. In this case, you must change the driver to standard "Printer port". The port driver can be changed by the following steps: In the Windows "Start menu", select "Settings | Control Panel". In Control panel, select "System". In the "System Properties" Windows, select tab "Device Manager". Expand the item "Ports", double click on the "ECP port" item to invoke the "ECP Port Properties" Window. Here, select the "Driver" tab. Click on the "Update Driver" button and in the "Update Device Driver" Wizard, select the "Display a list of all the drivers in a specific location..." In the follwing dialog, select the "Show all hardware" option and from the "(Standard port types)" category, select the "Printer Port" item. Ignore the warning messages and just finish the driver update - for this, you may need the Windows Installation CD-ROM (but often, the necessary driver file can be found in the Windows SYSTEM directory).
   Note: These steps can be done only in Win9x/2000. NT have a different procedure for changing the printer port driver. However, we have never experienced problems with the ECP driver in WinNT
3. On some machines, Windows 9x sometimes set the parallel port name to "LPT2", even if you have only one parallel port. Make sure that in the "System properties" (Start menu | Settings | Control panel | System), the printer port is displayed as "Printer Port (LPT1)". If you have "Printer Port (LPT2)" there, you must set the Flashkit parallel port setting to "LPT2" as well.
4. In WinNT/2000, make sure there is no printer driver set for printing on the LPT port that you want to use with Flashkit (see installation notes in this file)
5. In WinNT/2000, make sure the Flashkit was installed properly with the Administrator rights (see installation notes)

If you fail to find a solution for parallel port communication problems, set the "Generate Report File" option in the Flashkit SW to "yes". Then select the Synchronous communication via LPT option and press "Connect". Then, send the generated "Log file" to the Flashkit support at microcontroller_info@fme.fujitsu.com.

Flashkit serial programmer directory overview

Installed Flash-kit contains the following files and directories:

Directories:

- Kernels - includes configuration file and kernel binary files.

Files:

- SerProg.exe : The Flash-kit executable
- SerProg.ini : The Flash-kit initialization file
- SerProg.pdf : The Flash-kit manual
- RegistrationForm.rtf : An editable copy of the customer registration form, which can be found in this manual (chapter "Customer registration").

- Firmware_v17.mhx : Current firmware for the programmer box (for information about firmware updates, check the "Firmware update" chapter of the manual)
- myoptions.abc : command line option file, common for all example *.bat files
- Example1.bat : Read FLASH memory from MB90F543 from addr FF0000 to FF0FFF (length 1000) and save it into file test.bin. The CPU is connected via serial cable (asynchronous communication) to COM1, the CPUfrequency is 4MHz.
- Example2.bat : The task is the same as in Example 1 - Read FLASH 1. We have only one CPU (MB90F543) with crystal 4MHz connected to PC with serial cable to COM1, so we can put this values into the options file MyOptions.abc and use this configuration many times.
- Example3.bat : Now we have got the (for us important) content of the FLASH (from previous examples). And now we want to copy this into another CPU. We replace the CPU board and run this batch file:
- Example4.bat : We want only clear the FLASH in the CPU.
- Example5.bat : We try to clear sectors 0 and 2.
- Example6.bat : We have an application compiled in *.mhx, *.ehx or *.ihx file, e.g. TEST.MHX. We want to clear used sectors (by the program), burn it into the FLASH and verify if the FLASH really contain our code. The compiled file is stored in directory C:\TEST.
- Example7.bat : We have an application compiled in *.mhx, *.ehx or *.ihx file, e.g. TEST.MHX. The FLASH is cleared, we want to burn code into the FLASH and verify if the FLASH really contain our code. The compiled file is stored in directory C:\TEST.
- Example8.bat : We want to be only sure that FLASH contain our last code. We don’t need to programm FLASH again, we can compare user program and FLASH. We will compare the file TEST.MHX stored in directory C:\TEST with the FLASH.
- Multiflash.bat : Example of batch file for programming multiple CPUs with retrieving status
- Test.bin : Binary file used in Example3.bat
- UninstIO.exe:
- UninstMem.exe:
- Uninst.isu : These files are needed for proper uninstallation of Serial programmer Windows paraller port driver

Uninstallation

From "Control Panel" (Start | Settings | Control Panel) run "Add/Remove programs". Select "Flashkit serial programmer" from the list and click on the button "Add/Remove". Flashkit will be automatically removed from your computer.

Default option settings

After the installation, the programmer is ready for use with the MB90F543 in the asynchronous mode. The other options are set to the following values:

- Frequency = 4MHz (this is the frequency of crystal used on the target board)
- COM port = COM1, LPT port = LPT1
- Supply target CPU = ON

The options (Main menu | Settings) are set this way:

1. Tab „CPU reset & mode“
   - Reset timing = 600 ms
   - Reset generated by COM port = ON
     - CPU reset – COM port option =RTS line for reset, DTR for mode select
     - Reset line – logic level = low
     - CPU mode line – logic level = low
2. Tab „Synchronous mode“
   - Comparing – synchronous mode = Full compare
   - CPU Operating Mode after Auto Program = Programming mode
   - Trigger pulses = don’t generate
3. Tab „Common“
   - Connect included in Auto Program command = OFF
   - Auto Program – Erase only used sectors = ON
   - Blank check after command „Connect“ = ON
   - Check user file for forbidden addresses = ON
   - Generate report file = OFF
   - Report file name = {installation path}\Report.log
   - Number of trials for connect to target CPU = 5
   - Path to binary files = {installation path}\Kernels\

If the „Connect“ function fails although your target board is connected properly to the PC and to the power supply, please check the „Reset generated by COM port“ option. In the asynchronous mode, it must be ON. Also check the „CPU reset – COM port option“, „Reset line“ and „CPU mode line“ settings – they must correspond to the hardware of your board.

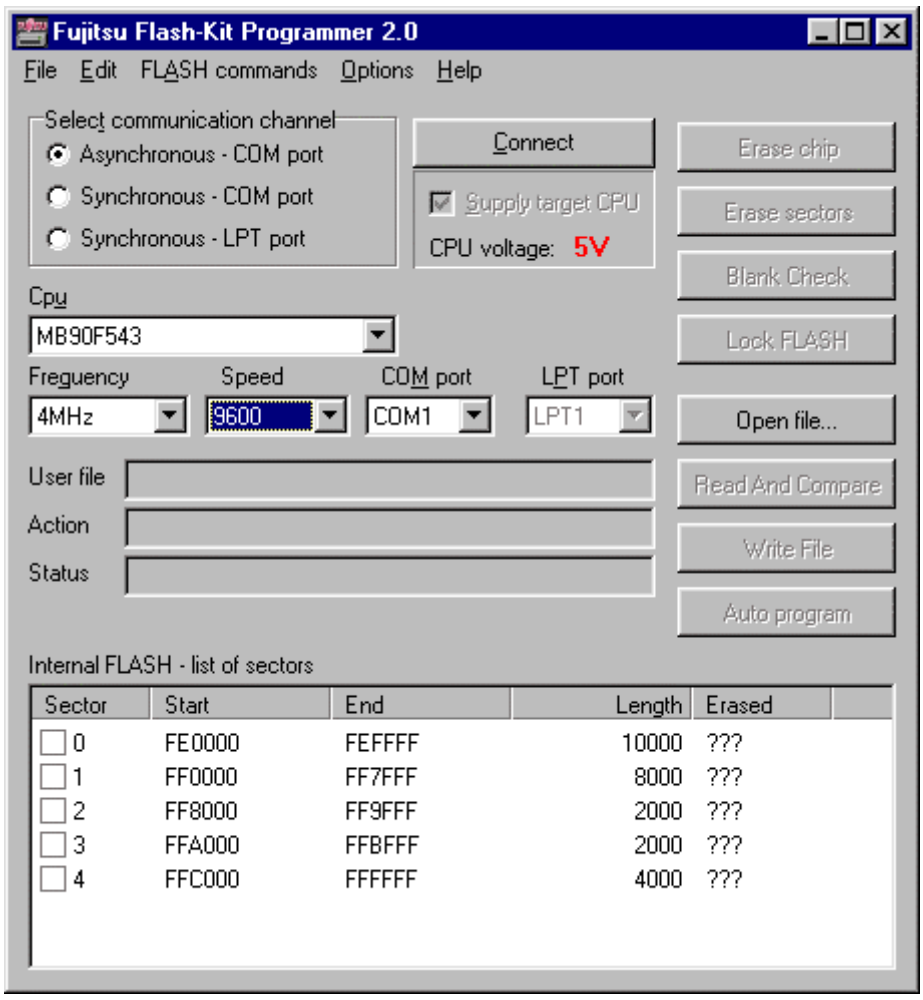
Chapter

3

Serial programmer PC software

The Serial Programmer software has two parts: the PC front-end and the communication & FLASH programming kernel, which is downloaded into the user target board CPU internal RAM everytime the „Connect“ function is invoked in the PC-frontend. In this chapter, only the PC front-end software is described.

After starting the serial programmer, a window similar to the following will appear:



Description of controls

- **Select communication channel** – here you can select, which port will be used for communication with the user board or the programmer box.
- **CPU** – select the target CPU
- **Frequency** – Select the actual external clock frequency of the CPU. The relationship between this frequency and the communication baud rates used for the programmer operations is shown in the tables with UART interface specifications in Chapter 6 – Supported CPUs.
- **Speed** – the communication speed (baud rate), which will be used when the "Asynchronous - COM port" communication channel is selected. If the selected CPU supports changing the PLL setting in the asynchronous mode, the "Speed" list will offer several baud rates.
- **COM Port** – Select the COM port where you have a serial cable for communication
- **LPT Port** – Select the LPT port where you have a parallel cable for communication
  Note: when choosing the crystal frequency, remember that the maximum used external frequency must not exceed the specifications in the corresponding microcontroller datasheet.
- **Supply target CPU** – if the serial programmer box is used, this option can be checked if the user wishes to supply his target board from the programmer box.

Meaning of buttons and status lines

- **Status lines**
  - **User file –** Displays the currently selected user file
  - **Action** – Displays current action
  - **Status** - Displays status of the last action

- **Buttons** – these actions are also in **Main menu** (see **below**)
  - **Open file** – opens file in Motorola S-format or Intel format
  - **Connect** - Downloads the communication & FLASH programming kernel into RAM. **Checkbox appears depending on "Reset the CPU" settings.**
  - **Read and Compare** - Compares data from the currently opened user file with the data in the FLASH memory. If a difference is found, user is informed about it.
  - **Write file** – Writes the user program into the FLASH memory.
  - **Auto program** – This button invokes the „Erase blocks"- „Write file" and „Read & compare" functions. It will erase only those blocks of FLASH memory, which aren't blank and which are used by the user file. If the "Connect" function wasn't invoked previously, it is invoked too. After the FLASH is written and verified, the target CPU mode is changed to single chip mode and the CPU is reset, so the user code starts running.
  - **Erase chip** – Erases whole FLASH
  - **Erase blocks** – erases only selected sectors.
  - **Blank check** – checks if the sectors are blank (erased).

- **Lock FLASH** – This function is enabled only if the selected CPU has the code security feature. Locking the FLASH disables reading from the FLASH – the value 0FFH is read from all FLASH addresses. If the FLASH is locked, it can be unlocked only by the "Erase whole" function.
  Note: The FLASH is not really locked until the next CPU reset! So if you lock the FLASH, you will be still able to work with it normally, until you reset the CPU.

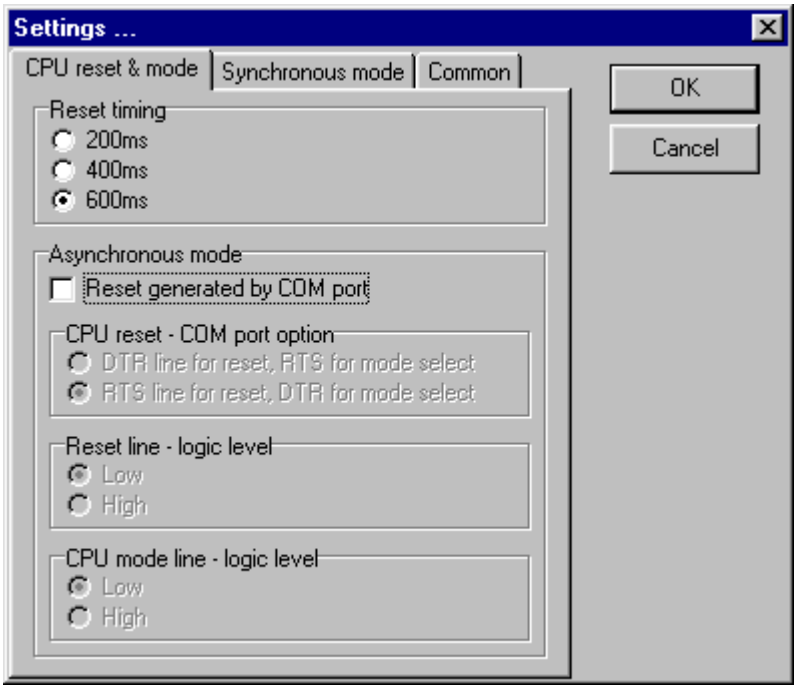
Main menu

- **File**
  - **Open file** – opens file in Motorola S-format or Intel format
  - **Open binary file** – opens any binary file. You are prompted for specifying the destination FLASH memory address where the binary file will be burned to.
  - **Save memory** – displays dialog where you specify the address and the length of the memory, which will be saved into the disk in binary format. Then you must select the file name and its location.
  - **Read and Compare** - Compares data in Motorola S-Record/Intel hex file and data in FLASH memory. If a difference is found, user is informed about it.
  - **Write file** – Writes the user program into the FLASH memory.
  - **Auto program** - It encapsulates the buttons „Erase blocks", „Write file" and „Read & compare". It will erase only those blocks of FLASH memory, which aren't blank and which are used. If the "Connect" function wasn't invoked previously, it is invoked too. After the FLASH is written and verified, the target CPU mode is changed to single chip mode and the CPU is reset, so the user code starts running.
  - **Exit** – closes the Serial programmer
- **Edit**
  - **Edit memory** – displays dialog where you specify the address and the length of the memory, which will be edited. For more detailed description of this function see the „Editing memory" paragraph.
  - **Show file information** – displays window where the information about used FLASH memory is displayed.
- **FLASH commands**
  - **Erase chip** – Erases whole FLASH
  - **Erase blocks** – erases only selected sectors.
  - **Blank check** – checks if the sectors are blank (erased).
  - **Lock FLASH** – Locks the FLASH – access to the FLASH is restricted (enabled only if the selected CPU supports this feature).
- **Options**
  - **Settings –** displays dialog box with settings of Serial Programmer. See chapter Options & Settings.
- **Help**
  - **Contents** – Displays help file SerProg.pdf, which should be stored in the same directory as the Serial Programmer.
  - **About** – displays the About box
  - **Firmware version** – if the box is connected to the PC, this function displays the version number of the programmer box firmware. If the box is not connected to the PC or is not working properly, the firmware version is displayed as "unknown".

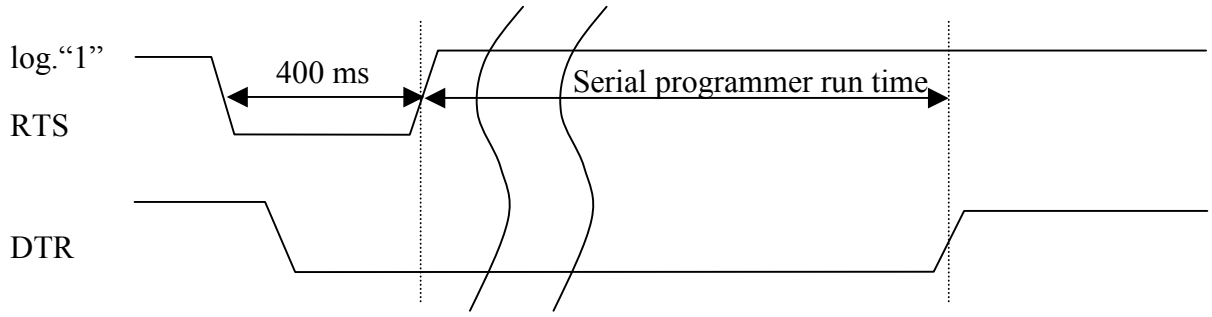
Options and settings

- **Page CPU reset & mode**
  For the asynchronous serial programming mode, the RS232 lines DTR and RTS can optionally be used to reset the MCU and to switch the operation mode of the MCU automatically. The "CPU reset & mode" page is used to configure these lines for the desired mode of operation:

  - **Reset timing** – duration of the reset impulse, generated on the DTR line for resetting target CPU
  - **Reset generated by COM port** – for asynchronous programming if the target CPU is reset by DTR or RTS line (checked) or manually (unchecked)
  - **CPU reset – COM port option** – select one of two options of target CPU reseting in the asynchronous mode. One option is DTR line for reset and RTS for CPU mode select (default) and the second one is RTS line for reset and DTR for CPU mode select. Select the right option for your hardware.
  - **Reset line – logic level –** the logic level of the reset line (DTR or RTS) when the CPU **is not** in reset
  - **CPU mode line - logic level –** the logic level of CPU mode line (RTS or DTR) when the target CPU is in programming mode.



With the settings above, the timing diagram of the RTS line (used for resetting the target board or programmer box) and DTR line (in asynchronous mode, used for choosing the mode of the target CPU) is shown on the following figure:



When the “Connect” button is pressed, the target CPU is reset by the RTS line. In this case, the length of the reset pulse is 400 ms (this value was chosen in the CPU reset & mode tab of the Option settings dialog). The DTR line is set to log. “0”, so the CPU mode is set to “Asynchronous serial programming” mode. The DTR stays low until you exit the serial programmer SW (note, that log. “0” corresponds to positive voltage on the RS232 line, while the log “1” to negative voltage).

If you are interested about the detailed timing of the serial programming pins, check the Apendix A, where the timing diagrams for the CPUs supported by the Serial programmer SW is included.

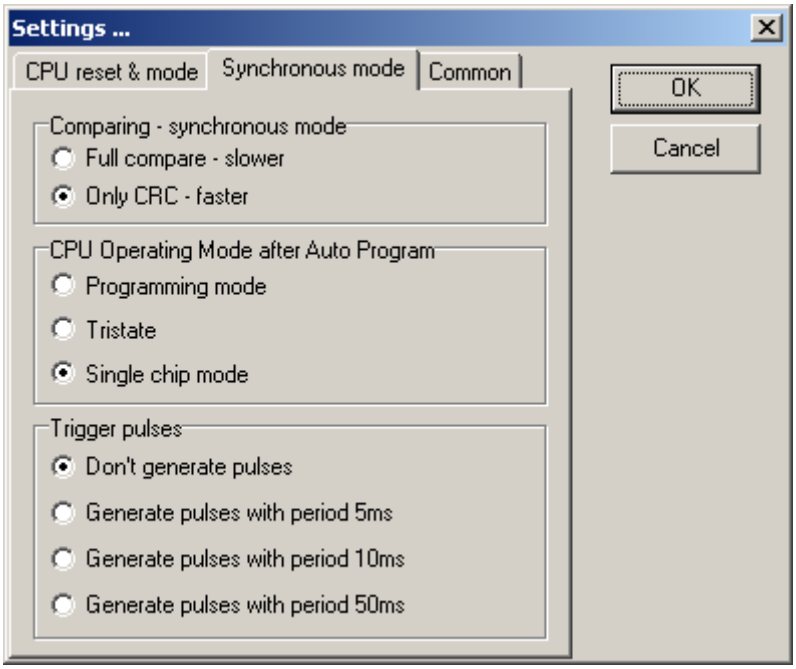
Note: For proper function of the reset line function and CPU mode selection line function, the target board must have the appropriate HW support of these functions. Therefore, when designing the target board, include the „Recommended circuit for asynchronous mode“ or “Target serial interface” schematic, that can be found in the appendix of this manual, to your board schematics.

Recommendation: Although it is possible to use both the RTS and DTR lines for the reset line or mode selection line functions, for timing reasons it is better to use the RTS line for reset and DTR line for mode change. However, if you need the RTS line for implementing the HW flowcontrol (handshaking) on your board, it is more convenient to use the DTR line for resetting the board.

Note: Some Fujitsu tools (Devkit16, Flash-CAN-100) support currently just the “Reset via DTR” function. For details, please check the corresponding manuals.

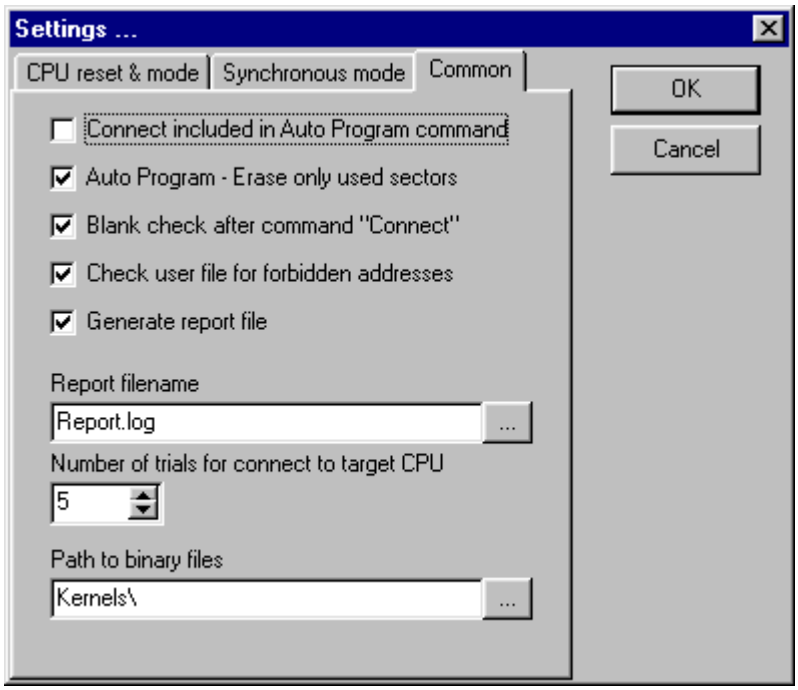
Page Synchronous mode

- **Comparing – synchronous mode –** when programming in synchronous mode, comparing (verifying) of the user file (data) can be done by:
  1. Full compare – data from target CPU are transferred into the PC and there compared with user file
  2. Only CRC – special kernel is downloaded into the target CPU and only CRCs of FLASH memory (where the user file is located) are transferred into the CPU. Here they are compared with locally computed CRC. This method is faster for comparing big applications or whole FLASH. It can be slower for small applications.



- **CPU Operating Mode after Auto Program –** these radio buttons can be used for setting the state of the CPU mode pins after the "Auto Program" operation is finished.
  1. Programming mode – after "Auto Program" operation, the states of the CPU mode pins are not changed. Thus, it is possible to invoke other operations without the need for the "Connect" operation (which downloads the kernel into the CPU)
  2. Tristate - after "Auto Program", the CPU mode pins are put into the third state (they are "released"). Thus, the resulting mode depends on the default levels, which are set by the target board pullups/pulldowns.
  3. Single chip mode – the CPU is set to the single chip mode after "Auto Program" is finished, then it is reset, so the burned application is run.
- **Trigger pulses –** these radio buttons can be used for switching on the generation of pulses on the box programming connector PIO1 pin. This feature can be utilized by 16LX target boards that are have an external watchdog, which must be triggered during the programming not to cause unwanted CPU resets. Note – for 8L family, this feature is not supported.
  1. Don't generate pulses – the pulses are not generated. This is the default value. If you don't need generating the pulses, please leave this option set this way, because generation of the pulses slows all the programming operations down a little bit
  2. Generate pulses with period ... ms – the pulses are generated with the chosen period

- **Page Common**
  - **Connect included in Auto Program command** – every Auto Program causes connection process. Useful when programming more CPUs.
  - **Auto Program - Erase only used sectors** – erases only used sectors by user program. Other sectors are unchanged.
  - **Blank check after command "Connect" –** every command „Connect“ causes command „Blank check“ of the FLASH.
  - **Check user file for forbidden addresses -** Some CPUs have an area in the FLASH memory whose rewriting may cause preventing from further reading/programming of the FLASH. Recommended value is checked. If it is enabled and the user program contains data for these areas, user can choose if the data will be filtered or not.



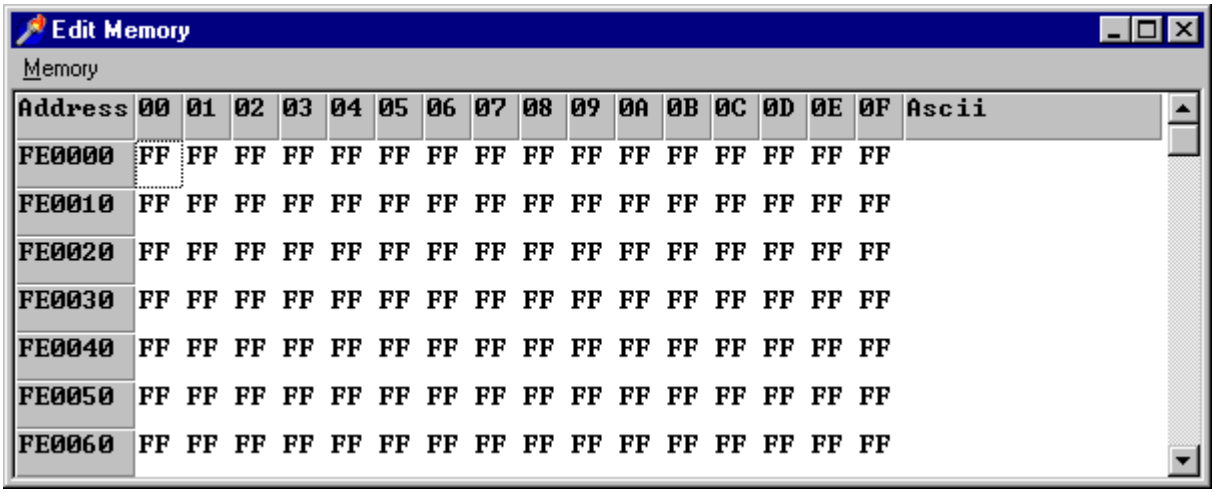
  - **Generate report file –** creates file **Report.log** with every step of programming. Useful when there are some problem with Serial Programmer. It can help developers for finding the problems out. File is stored in the directory where the Serial Programmer is installed.
  - **Report filename** – filename include the absolute path where the report file will be saved.
  - **Number of trials for connects to target CPU -** set the number of times the programmer SW tries to connect with the target CPU. Minimal value is 2, recommended value is between 5 and 10.
  - **Path to binary files** – path where the binary files (ie. kernels) are stored.

Remark: All settings in dialog Options are stored in the windows system directory. The name of the file is **SerProg.INI**.

Editing the memory

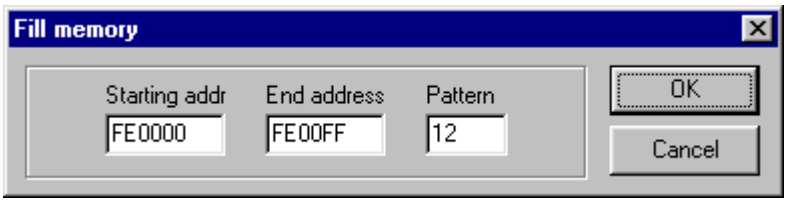
The memory edit function serves for manual changes of the FLASH memory content. You can modify both erased cells (cells with value 0FFH) or already programmed memory cells. If you change the erased cells only during your editing, these cells are programmed immediately. But if you change a cell which has been programmed already (its value is different from FF), then the modified FLASH sector must be loaded into the PC memory, erased and filled with modified content. For this to be done, you have to invoke the „apply changes" function (by **Save changes** in menu **Memory**).

Edit window:



Local menu:

- **Memory**
  - **Save changes** – when you are finished with modifying FLASH memory, apply changes using this command.
  - **Fill… -** here is the possibility of filling edited memory with certain pattern for selected part of edited memory. The content of the memory is changed only in PC, to save the modified memory use command **Save changes**.



Starting addr – Starting address of filled memory
End addr – Ending address of filled memory
Pattern – With this value the selected range of memory is filled
OK – confirms the the fill
Cancel – no fill is applied

Command line parameters

To be able to use the programmer in a non-interactive mode (e.g., from a batchfile), all the programmer commands and options can be invoked directly from the command line. The description of all these command line parameters can be found in the following paragraphs.
Note: all the parameters are case sensitive!

Programming mode
-pm A|SC|SL
- A – Asychronous – via COM port
- SC – Synchronous – via COM port
- SL – Synchronous – via LPT port

Select device
-dev *name*
where *name* = name of the CPU (the name specified in configuration file)

External clock frequency
-c *clock*
where *clock* = 4, 8 or 16 (MHz)

Speed – communication speed (baud rate) in asynchronous programming mode
-speed *speed*
where *speed* = 9600 (e.g.) or any other value supported by the selected CPU

Select PC communication port
-p *port*
where *port* = COM1, COM2, COM3, COM4, LPT1, LPT2, LPT3

DTR line reset function and RTS line mode setting function configuration
-reset
In the asynchronous programming mode, this parameter is used for automatic generation of reset pulse on the DTR line. When this parameter is used, also the **-modelevel** and –**resetlevel** parameters are processed (if present). If the **-modelevel** or –**resetlevel** parameters are not specified on the command line, the values are taken from the controls "**Reset line - logic level**" or "**CPU mode select – logic level**" in the **Options** dialog box.
If the **–reset** parameter is not used in the asynchronous mode, the user must reset the CPU manually before running the Serial programmer.
Note: the Serial programmer does not issue any prompt for resetting the CPU !
In the synchronous mode, this parameter is ignored.

- modelevel *level*
CPU mode select logic level – RTS/DTR (see command resetbyRTS) line is used for setting programming mode – set logic level of CPU mode line when the target CPU is in programming mode. The *level* can be 0 or 1.
This parameter is used only when the **–reset** command is present and the asynchronous programming mode is chosen.

- resetlevel *level*
Reset default level - the logic level of the DTR/RTS (see command resetbyRTS) line when the CPU is in reset. The *level* can be 0 or 1.
This parameter is used only when the **–reset** command is present and the asynchronous programming mode is chosen.

-resetbyRTS
when using this command the RTS line is used for reset and the DTR line for CPU mode select. Otherwise (default) RTS is used for CPU mode select and DTR line for reset.
This parameter is used only when the **–reset** command is present and the asynchronous programming mode is chosen.

-resettiming *value*
where *value* is a decimal number, e.g. 150, which specifies the duration of the reset pulse in milliseconds.
For asynchronous mode, this parameter is used only when the **–reset** command is present.
In synchronous mode, this parameter is used all the time.
If this command is not present, the default value from the "**Options"** dialog box is used.

CPU Operating Mode after Auto Program
-cpumode *P|T|S*
where *meaning is*:
- **P -** Programming mode (default)
- **T -** Tristate
- **S -** Single chip mode

This parameter is used only in synchronous mode. If this command is not present, the default value is **prog**. In asynchronous mode is ignored. For more details see chapter **Options and settings, Page CPU reset & mode**

Watchdog trigger pulses generation
-wd *period*
where *period* = 5, or 10 or 50 (the number represents the watchdog trigger pulses period in ms)

Select path for user data
-d *directory*
where *directory* = path to the directory containing the user files (files, that the user wants to burn into the FLASH)

Select path for binary files used internally by the programmer SW

-b *binpath*

where *binpath* = Full path to directory where binary files are (kernels, etc.)

Erase chip

-erase

Note: this command has no parameters

Erase sector

-esec *nr*

where *nr* = the number of the sector to be erased. *nr* can be in the range (0, *N*), where *N* is dependent on the CPU type –e.g., with the MB90F543, the *N*=4

Command

-comm V, W, WV, A, R

- V – verify
- W – write
- WV – write & verify
- A – auto program (erase, write, verify)
- R – read memory and write the content into the binary file or Motorola-S file (only if the data file has **.mhx** extension).

Source data or target file

-data *file*

where *file* = name of the file in binary or Motorola S-format. The file must be stored in directory specified by parameter **–d datapath**. If the file is binary file, the parameters **binary and starting address** must be present.

Binary file

-binary

Defines source data as pure binary file (it is not Motorola S-format nor Intel HEX format). The serial programmer expects parameter Starting address (see the description of "**-addr"**)

Starting address

-addr *address*

where *address* is hexadecimal number, e.g. FE0000 – This parameter specifies the starting address for command read memory (see the description of "**-comm** R")

Data length

-len *length*

where *length* is hexadecimal number, e.g. A0 or 100000. This parameter specifies the length of the read memory for command "read memory" (**-comm** R)

Options file

-o *filename*

where *filename* is full path to **options file**. This file can be used for storing all the previous parameters, so the user does not have to write them all again when running the programmer from the command line for several times.

Supply target
-supplytarget *value*
where the *value* can be 0 or 1. If the *value* is 1 then the target CPU is supplied by programmer, otherwise the target CPU must have own supply.

Generate report file
-report
Specifies to generate a Report.log file, which is recording every step of programming. Useful when there are some problem with Serial Programmer. It can help developers to fix problems. File is stored in the directory where the Serial Programmer is installed.

Set the report filename
-reportname *filename*
The *filename* is the absolute path where the report will be stored (if the command **–report** is present)

Disable filter for dangerous/forbidden memory areas
-disablefilter
Some CPUs have an area in the FLASH memory whose rewriting may prevent further reading/programming of the FLASH (e.g., the security vector on the MB91F361 – if value different from FFFFFFFFH is written to it, the CPU will not enter the serial programming mode anymore). For protecting these areas from unintended writing, the Flashkit programmer SW filters out all the user data designated to these areas. The filtering can be switched off by the *-disablefilter* switch

Note: For every CPU, the memory areas protected by this filter are configured in the SERIAL.CFG file (section "Forbidden Count"). For details about the SERIAL.CFG file, see the "Configuration file for serial programmer" part of this chapter.

Batch files examples:

Example 1 – Read FLASH 1
Read FLASH memory from MB90F543 from addr FF0000 to FF0FFF (length 1000) and save it into file test.bin. The CPU is connected via serial cable (asynchronous communication) to COM1, the CPU frequency is 4MHz.

The batch file should contain:

```
SerProg.exe -dev MB90F543 -pm A -c 4 -p COM1 -comm R -data test.bin
-addr FF0000 -len 1000
```

Example 2 - Read FLASH 2
The task is the same as in **Example 1 – Read FLASH 1**.
We have only one CPU (MB90F543) with 4MHz crystal connected to PC via serial cable to COM1. The values can be put into the options file *MyOptions.abc* and use this configuration many times.
The Options file *MyOptions.abc* should contain:

```
-dev MB90F543 -pm A -c 4 -p COM1 –reset –resetlevel 1 –modelevel 0
```

The "-reset –resetlevel 1 –modelevel 0" commands are used for configuration of the automatically generated reset impulse.
The batch file should contain:

```
SerProg.exe -o myoptions.abc -comm R -data test.bin -addr FF0000
–len 1000
```

Example 3 – Auto program binary file into the FLASH
Now we have got the (for us important) content of the FLASH (from previous examples). And now we want to copy this into another CPU. We replace the CPU board and run this batch file:

```
SerProg.exe -o myoptions.abc -comm A -data test.bin -binary
-addr FF0000
```

Example 4 – Erase whole FLASH
We want only clear the FLASH in the CPU.
The batch file should contain:

```
SerProg.exe -o myoptions.abc -erase
```

Example 5 – Erase specified sectors
We try to clear sectors 0 and 2.
The batch file should contain:

```
SerProg.exe -o myoptions.abc –esec 0 –esec 2
```

Example 6 – Auto program user code
We have an application compiled in *.mhx, *.ehx or *.ihx file, e.g. TEST.MHX. We want to clear used sectors (by the program) ,burn the program into FLASH and verify if the FLASH really contains the code. The compiled file is stored in directory C:\TEST.
The batch file should contain:

```
SerProg.exe –o myoptions.abc -comm A -data TEST.mhx –d C:\TEST
```

Example 7 – Write user code and verify
We have an application compiled in *.mhx, *.ehx or *.ihx file, e.g. TEST.MHX. The FLASH is cleared, we want to burn code into the FLASH and verify if the FLASH really contain our code. The compiled file is stored in directory C:\TEST.
The batch file should contain:

```
SerProg.exe –o myoptions.abc -comm WV -data TEST.mhx –d C:\TEST
```

Example 8 – Verify user code with the FLASH
We want to be only sure that FLASH contain our last code. We don't need to programm FLASH again, we can compare user program and FLASH. We will compare the file TEST.MHX stored in directory C:\TEST with the FLASH.
The batch file should contain:

```
SerProg.exe –o myoptions.abc -comm V -data TEST.mhx –d C:\TEST
```

Using the programmer in batch files

Serial programmer batch mode is provided for cases when user interaction is not necessary or should be minimized, e.g. during final production of the user application. For this purpose, all the serial programmer functions are accessible by command line switches.

When the Serial programmer execution completes in the command line mode, it returns the status of the last action to the operating system. The list of all the status values follows:

Value	Comment
0	no error
1	error in option file
2	wrong input parameter
3	cannot open file with kernel
4	cannot open user file
5	cannot open communication device
6	cannot connect to target CPU
7	communication with kernel failed
8	cannot erase FLASH
9	error during writing into the FLASH
10	error during reading from the FLASH
11	error during verifying data
12	unknown error
13	disk I/O error

Example of a batch file testing the return values. This batch file is used for programming of multiple CPUs.

```
@echo off
:start
choice start next programming
if errorlevel 2 goto end
echo programming...
start /w SerProg.exe -o myoptions.abc -esec 0 -esec 2
:: Remark: do the ERRORLEVEL checks in descending order
if errorlevel 13 goto error13
if errorlevel 12 goto error12
if errorlevel 11 goto error11
if errorlevel 10 goto error10
if errorlevel  9 goto error9
if errorlevel  8 goto error8
if errorlevel  7 goto error7
if errorlevel  6 goto error6
if errorlevel  5 goto error5
if errorlevel  4 goto error4
if errorlevel  3 goto error3
if errorlevel  2 goto error2
if errorlevel  1 goto error1
if errorlevel  0 goto error0

:: actions according to error-code
:: continue if no error occurred, otherwise display error
```

```
echo undef
exit

:error0
echo successful.
goto start

:error1
echo Error : error in option file
goto end

:error2
echo Error : wrong input parameter
goto end

:error3
echo Error : cannot open file with kernel
goto end

:error4
echo Error : cannot open user file
goto end

:error5
echo Error : cannot open communication device
goto end

:error6
echo Error : canot connect to target CPU
goto end

:error7
echo Error : communication with kernel failed
goto end

:error8
echo Error : cannot erase FLASH
goto end

:error9
echo Error : error during writing into the FLASH
goto end

:error10
echo Error : error during reading from the FLASH
goto end

:error11
echo Error : error during verifying data
goto end

:error12
echo Error : unknown error
goto end

:error13
```

```
echo Error : Disk IO error
goto end

:end
echo end
```

Configuration file for serial programmer

In the configuration file **Serial.cfg**, the description of all the currently supported CPUs is stored. When the Serial programmer starts, it automatically loads and analyzes this file, which allows adding new CPUs to the programmer just by modifying this configuration file– it is just a matter of appending a new [CPU] record to the file and providing a communication kernel for that CPU.
Warning: Do not modify this file unless you really know what you are doing!

An example of this configuration file is given here:

```
;comments start with the ';'
;all commands are case sensitive

[CPU]
Name= cpu_name             //name of the CPU

AsynchroKernel=filename    //relative path to kernel for asynchr. programming
AsynchroSpeed=speed        //communication speed with asynchro kernel
SynchroKernel=filename     //relative path to kernel for synchronous programming
CRCKernel=filename     //kernel name for CRC computing - for synchronous programming
HardWired=Yes/No    //reset vector - Hardwired yes/no

CPUMode=value       //bits : 0- MD0, 1-MD2, 2-RESET (when the cpu is not in reset),
3-P00, 4-P01
3Volts=Yes/No       //Yes = the CPU is supplied with 3 Volts, No = with 5 Vols

KernelFilterLow=value (HEX)      //memory range where the kernel will be in the RAM,
KernelFilterHigh=value (HEX)     //other adresses will be removed


CPUFreqCount=number //number of possible CPU frequences - decimal number
                                 //repeat it "number"time:
CPUFreq=freq                     //Frequency [MHz] decimal (e.g. 4, 8, 16)
Rate=Speed                       //comunication speed (4800, 9600, ...)  for curent
Frequency

FlashBaseAddr=FE0000             //Start address of the FLASH - HEX value
FlashBlocks=number               //number of blocks in FLASH - decimal value
                                 //number-times:
From=000000                      //offset of block from beginning of the FLASH (HEX)
Length=10000                     //size of the block (HEX)

                                 //Burn-in-ROM commands
BIROMType=value                  //BIROMType=8, 16, 17 or 32
DownloadToRAM=Command_nr         //command number for Download to RAM (HEX)
Run= Command_nr                  //command number for Run programm from RAM (HEX)
BIROMResponds=yes/no             //if the BiROM responds or not. Default yes

SecurityFeature=yes/no           //if the CPU has the security feature
                                 //next 2 lines only if the SecurityFeature=yes
SecurityAddr=address             //address (HEX) where to write Value
SecurityValue=value              //Value - one byte (HEX) which locks the FLASH

                                 //Now, here goes the list of areas of memory in the
                                 //FLASH whose programming is forbidden, since their
                                 //programming would prevent further
                                 //reading/programming of the FLASH. Any user data
                                 //designated to these areas will be "filtered out".
                                 //(Note: for disabling the filtering, the
                                 //commandline switch "-disablefilter" can be used

ForbiddenCount=value             //number of dangerous memory blocks (DEC)
```

```
                                //repeat these two lines "ForbiddenCount" times
StartAddr=value                 //starting address of the memory area(HEX)
EndAddr=value                   //ending address of the memory area(HEX)
```

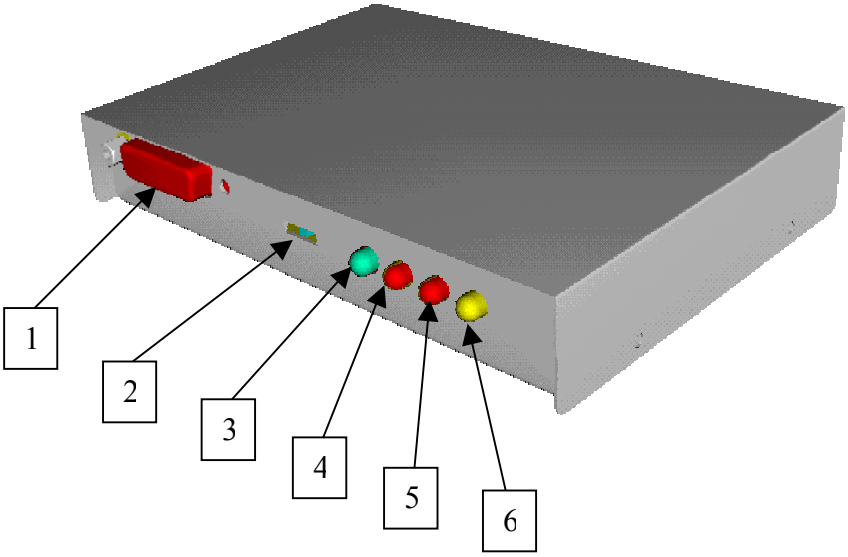
Chapter

4

Serial Programmer box

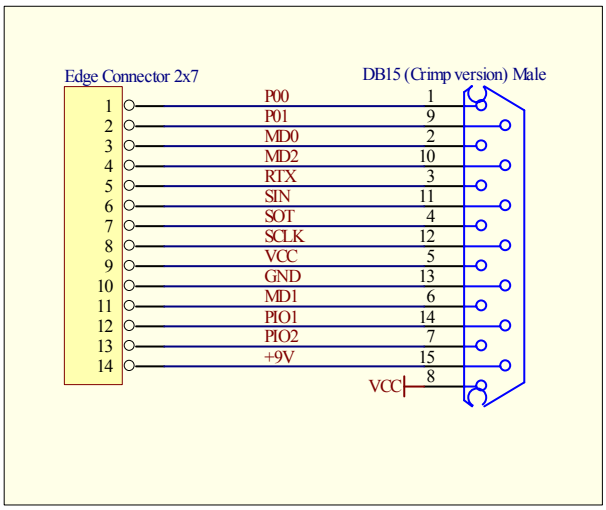
The serial programmer box must be used for the synchronous programming mode. In this chapter, the basic description of the box can be found.

Front panel



1. Programming connector

This connector is used for connecting the target system. The pinout of this connector is designed in a way that it allows using a flat ribbon cable with crimp connectors on both ends – see the following figure for the connection schema of the cable:



The states of the connector pins during and before/after programming of a 16LX device can be found in the following table:

DB15 Con. pin nr.	Edge con. pin nr.	Pin Name	State before/after programming	State during programming	State after** "Auto Program" command
1	1	P00	Hi-Z (with pull up)	OC (level 0)	Hi-Z (with pull-up)
2	3	MD0	Hi-Z (with pull up)	OC (level 0)	OC (level 1 via pull-up)
3	5	RTX	Hi-Z (with pull up)	OC (level 1 via pull-up)	OC (level 1 via pull-up)
4	7	SOT	Input (with pull-up)	Input (with pull-up)	Input (with pull-up)
5	9	TVcc	0V / 3V or 5V (*)	3V or 5V (*)	3V or 5V (*)
6	11	MD1	Hi-Z (with pull-up)	OC (level 1 via pull-up)	OC (level 1 via pull-up)
7	13	PIO2	Input (with pull-up)	Input / Output	OC (level 0)
8		Vcc	+5V	+5V	+5V
9	2	P01	Input (with pull-up)	Input / Output	Input (with pull-up)
10	4	MD2	Hi-Z (with pull up)	OC (level 1 via pull-up)	OC (level 0)
11	6	SIN	Output (level 0)	Output	Output (level 0)
12	8	SCLK	Output (level 0)	Output	Output (level 1)
13	10	GND	GND	GND	GND
14	12	PIO1	Input (with pull-up)	Output	Input (with pull-up)
15	14	Vpgm	9V	9V	9V

(*) Output voltage depends on selected device type. Zero voltage level is available on this pin after reset of the serial programmer box only.
(**) The state after Auto Program depends on the value of the „CPU Operating Mode after Auto Program“ setting in the options. The values displayed in this table hold for „Single chip mode“ value of this option.

The **Hi-Z** state stands for an tristate-output pin in the high impedance state. All the Hi-Z pins have the 10k pullups on them, so the logic level is defined.
The **OC** means „open collector“ – the pin is either in the log. „0“ state, or in the weak log. „1“ (high-impedance output with the pullup 10k).
The **input** pins have a 10k pullup resistor attached to them, so the logic level is defined even if no device is attached to the programmer.
The **Power** pins are used for powering the target system (or powering the programmer from the target system). If the target system is powered from the programmer, the voltage on the TVCC is dependent on the CPU that is being programmed – if a 3V CPU is chosen, the TVCC pin is switched to 3V after „Connect“ is pressed. Also, all the other programming interface output signals are switched to the 3V logic levels.
The **Vpgm** pin provides a +9V programming voltage needed for programming the 8-bit MCUs (e.g., the MB89P935). The +9V programming voltage is present always on this pin.

Note: tables with programming connector pin states for other CPU families can be found in Chapter 7 – Supported CPUs.

2. Mode switch

The programmer can operate in two modes:

- Standard

- SW update – in this mode, the firmware of the programmer can be updated. For the updating the SW, the asynchronous serial programming mode of the serial programmer can be used – for details, check the instructions that will come with the new version of the SW.

These modes can be selected by the mode switch. In the leftmost position, the "SW update" mode is selected. In the rightmost position, the "Standard" mode is selected.

For avoiding an accidental mode switching, the switch is sunk in the box so you must use a screwdriver when changing the mode.

Note: in fact, the switch has 3 positions. However, the Standard mode is selected when the switch is in the middle or the third (rightmost) position.

3. Ready/Busy LED

This 2-color LED indicates, whether the programmer is executing some operation, is ready for operation, or in some other state (e.g., is in the „SW update" mode):

- LED is green: the programmer is ready for operation
- LED is red: the programmer is busy (is transmitting/receiving data)
- LED is off: the programmer is in a third state – this can be caused by the „SW update" mode chosen, improper version of the firmware SW or some HW fault inside the box

4. 3V voltage indication

If the user selects a 3V CPU in the serial programmer software, this diode goes ON when the „Connect" button is pressed.

Note: The 3V operating voltage is selected by default (after the programmer is switched ON), so if a 3V board is connected to the programmer, it is not damaged by supply voltage provided on the box programming connector.

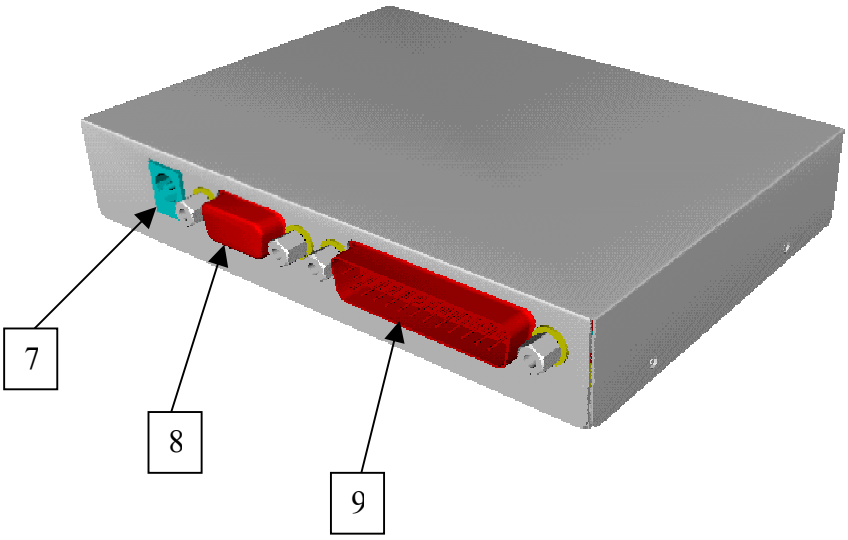
5. 5V voltage indication

If the user selects a 5V CPU in the serial programmer software, this diode goes ON when the „Connect" button is pressed.

6. „Power ON" LED

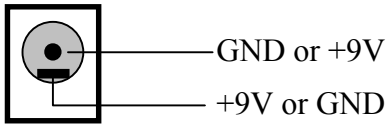
This LED goes ON when the power voltage is applied to the programmer box.
Remember that the programmer can be powered either by the input power voltage connector, or from the target board.

Rear panel



7. Power supply connector

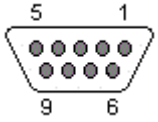
The power supply can be plugged into this connector. The programmer power supply circuitry allows using power supply connectors of both polarities (with GND on the shield or in the center). The voltage of the power supply should be set to 9V.

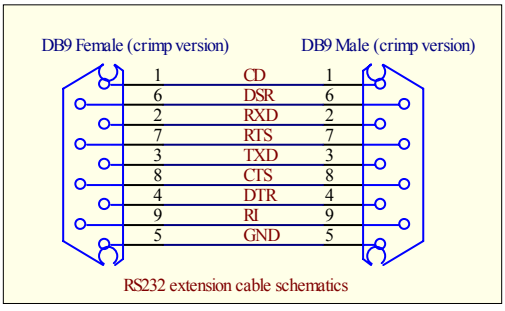


8. RS232 interface connector

This connector is used for connecting the programmer box to the PC serial port. The pinout of the connector is following:

1: NC
2: RX
3: TX
4: DTR (Reset)
5: GND
6: DSR
7: RTS
8: CTS
9: NC



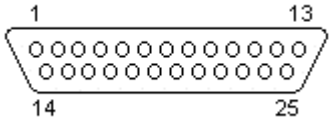


A standard extension cable for 9-pin COM port can be used for connecting the programmer box to the PC. The cable schematics is on the picture on the right. From this cable, the serial programmer box requires these signals: RXD, TXD, RTS, CTS, DTR and GND.

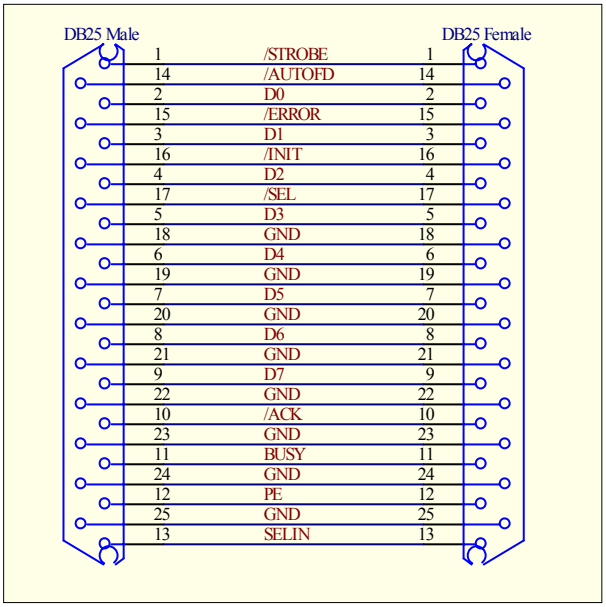
9. Parallel port connector

This connector is used for connecting the programmer box to the PC parallel port. Pinout of the connector is following:

1: /STROBE
2: PD0
3: PD1
4: PD2
5: PD3
6: PD4
7: PD5
8: PD6
9: PD7
10: /ACK
11: BUSY
12: PEO
13: SELIN
14: /AFEED
15: /ERROR
16: /INIT
17: /SEL
18-25: GND



A standard parallel port extension cable can be used for connecting the programmer to the PC. Schematics of the cable is on the following picture:



The serial programmer box requires all the signals from the parallel interface to work properly.

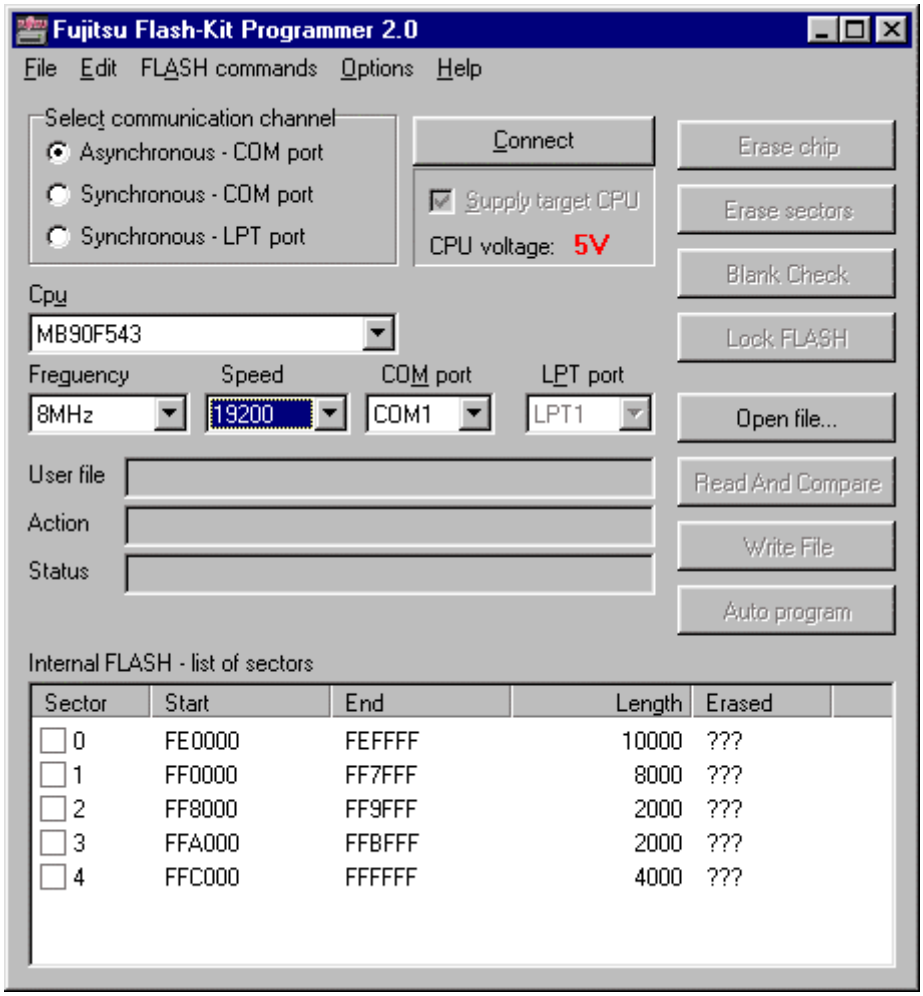
Chapter

5

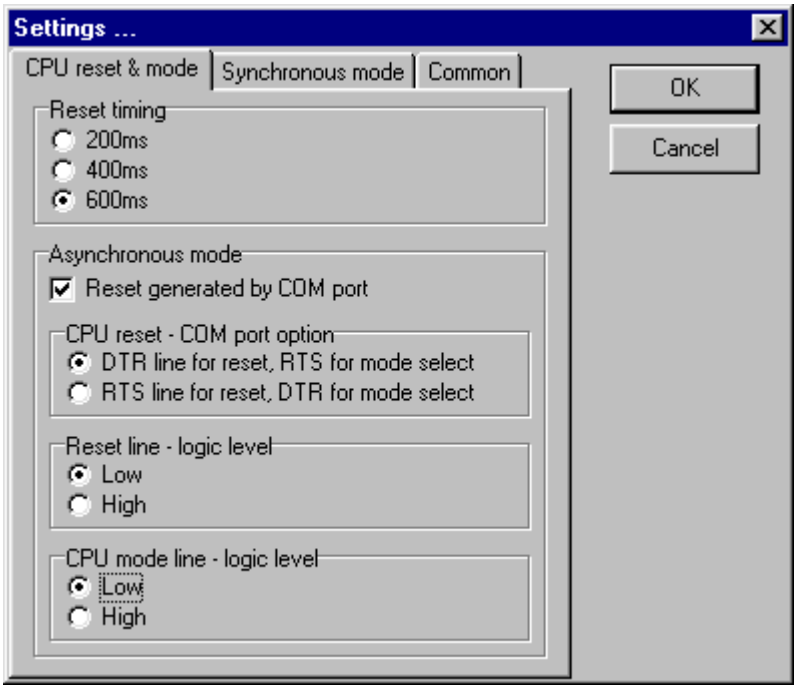
Firmware update

The firmware of the programmer box can be easily updated by the Serial Programmer SW. If you download the MHX file with the firmware update, you have to do the following steps:

1. Connect the serial programmer box to the PC COM port using serial cable
2. Set the „Mode switch“ to „SW update“ mode.
3. Run the Serial Programmer SW
4. In the programmer main window, set the „Select communication channel“ option to „Asynchronous – COM port“, „CPU“ to MB90F543 and „Frequency“ to 8MHz – see the settings on the following picture:



5. In the main menu, select “Options | Setting”. Dialog box with various setting appears. The settings must be as on the following picture (“Reset timing”: 600 ms, “Reset generated by COM port”: checked, “CPU reset – COM port option”: DTR line for reset, RTS for mode select, “Reset line – logic level”: Low, “CPU mode line – logic level”: Low).



6. In the programmer main window, press the „Open file“ button and select the MHX file containing the firmware update. Note: the current version of the firmware can be found in the file "Firmware_Vx.MHX", where „x“ stands for the current firmware version. This file resides in the directory, where the Flashkit software was installed
7. Press the „Auto program“ button
8. After programming is successfully finished, set the „Mode switch“ back to the „Normal“ mode. Now, your programmer box is ready to use!

Chapter

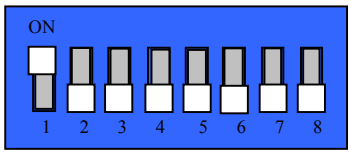
6

Programming the Devkit16

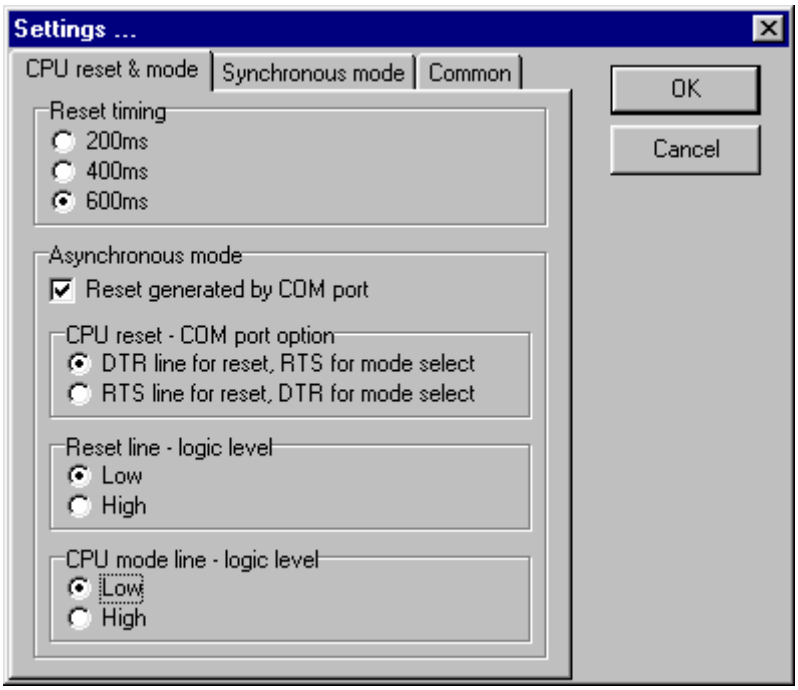
In this chapter, you will find information about how to interface the Flashkit serial programmer with the Devkit16. For informations about programming the other Fujitsu boards, please check the user's manuals of these boards.

Programming Devkit16 in asynchronous mode

1. Connect the PC serial (RS232) cable to the USER UART of the Devkit16 Mainboard
2. For setting for CPU FLASH programming mode, change the "Mainboard system configuration DIP switch" SW1 to the following configuration:

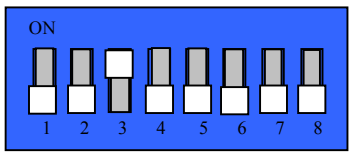


3. Set the J1 jumper to the 2-3 position (this is for the "Reset line-logic level"=Low setting)
4. In the programmer main window, set the „Select communication channel" option to „Asynchronous – COM port". Then select the „CPU" and "Frequency" options (in accordance with the CPU and crystal used on your CPU board), e.g.: MB90F543, 4MHz.
5. In the main menu, select "Options | Setting". Dialog box with various setting appears. The settings must be as on the following picture ("Reset timing": 600 ms, "Reset generated by



COM port": checked, "CPU reset – COM port option": DTR line for reset, RTS for mode select, "Reset line – logic level": Low, "CPU mode line – logic level": Low).

6. Open the file with your application (Motorola/Intel HEX or binary)
7. Press the „Connect" button, then the „Autoprogram" button
8. Set the "Mainboard system configuration DIP switch" SW1 to the following configuration to switch the CPU to the single chip mode:



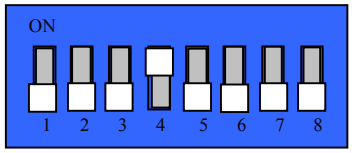
9. After resetting the board, your application should be running

Programming Devkit16 in synchronous mode

To be able to program the CPU of the Devkit16, you have to disconnect the Devkit16 CPU board from the mainboard first. This is because the FPGA on the Devkit16 mainboard controls the MD0-MD2 and P00-P01 lines of the CPU.

After disconnecting the CPU board from the Mainboard, do the following steps:

1. Connect the programmer box by the programming cable to the K7 connector of the Devkit16 CPU board.
2. Change the CPU board DIP switches SW3 to the following configuration:



This will connect the /RST line of the target programming connector to the CPU /RST pin.

3. Check the jumpers J7, J8 and J9. These jumpers select, which serial port (UART0 or UART1) is used for the synchronous programming of the CPU. E.g., with CPU MB90F543, these jumpers should be set to the "1-2" position.
4. In the programmer main window, set the „Select communication channel" option to „Synchronous – COM port" (or "Synchronous – LPT port" in case you want to use the high-speed parallel port communication). Then select the „CPU" and "Frequency" options (in accordance with the CPU and crystal used on your CPU board), e.g.: MB90F543, 4MHz. If you wish to use a separate power supply for your CPU board, uncheck the "Supply target CPU" option.
5. Open the file with your application (Motorola/Intel HEX or binary)
6. Press the „Connect" button, then the „Autoprogram"
7. After the programming is finished, the target CPU mode is changed automatically to the single chip mode and the CPU is reset, so your application should start running.

Chapter

7

Supported CPUs

F²MC-16LX family

In the following table, you can find the list of all the supported CPUs from this family and the pins they use for setting the serial programming mode:

CPU	Pin name	Pin No.	Logic level	Pin name	Pin No.	Logic level	Pin name	Pin No.	Logic level
MB90F428PF	MD 2,1,0	51,50,49	1,1,0	P00	85	0	P01	86	0*, 1**
MB90F497PFM	MD 2,1,0	21,20,18	1,1,0	P00	25	0	P01	26	0*, 1**
MB90F497GPFM	MD 2,1,0	21,20,18	1,1,0	P00	25	0	P01	26	0*, 1**
MB90F523PFV	MD 2,1,0	87,88,89	1,1,0	P00	95	0	P01	96	0*, 1**
MB90F543GPF	MD 2,1,0	51,50,49	1,1,0	P00	85	0	P01	86	0*, 1**
MB90F543PF	MD 2,1,0	51,50,49	1,1,0	P00	85	0	P01	86	0*, 1**
MB90F546GPF	MD 2,1,0	51,50,49	1,1,0	P00	85	0	P01	86	0*, 1**
MB90F548PF	MD 2,1,0	51,50,49	1,1,0	P00	85	0	P01	86	0*, 1**
MB90F548GPF	MD 2,1,0	51,50,49	1,1,0	P00	85	0	P01	86	0*, 1**
MB90F549PF	MD 2,1,0	51,50,49	1,1,0	P00	85	0	P01	86	0*, 1**
MB90F553PF	MD 2,1,0	51,50,49	1,1,0	P00	85	0	P01	86	0*, 1**
MB90F562PFM	MD 2,1,0	21,20,18	1,1,0	P00	25	0	P01	26	0*, 1**
MB90F562BPFM	MD 2,1,0	21,20,18	1,1,0	P00	25	0	P01	26	0*, 1**
MB90F568PFM	MD 2,1,0	21,20,18	1,1,0	P00	25	0	P01	26	0*, 1**
MB90F574PFV	MD 2,1,0	87,88,89	1,1,0	P00	95	0	P01	96	0*, 1**
MB90F583PF	MD 2,1,0	51,50,49	1,1,0	P00	85	0	P01	86	0*, 1**
MB90F591PF	MD 2,1,0	51,50,49	1,1,0	P00	85	0	P01	86	0*, 1**
MB90F594APF	MD 2,1,0	51,50,49	1,1,0	P00	85	0	P01	86	0*, 1**
MB90F598PF	MD 2,1,0	51,50,49	1,1,0	P00	85	0	P01	86	0*, 1**

* .. asynchronous programming
** .. synchronous programming

The UART interfaces used for the serial **asynchronous** programming mode and the CPU pin numbers are shown in the following table:

CPU type	**UART used**	**SIN pin**	**SOT**	**P01**	**P00**	**Baud rates*** [Bd]	
						Kernel download	Programming
MB90F428PF	UART1	88	89	86	85	4800	9600
MB90F497GPFM	UART1	60	62	26	25	4800	9600, 19200, 38400
MB90F497PFM	UART1	60	62	26	25	4800	9600, 19200, 38400
MB90F523PFV	UART1	X	X	96	95	Asynchro mode not supported	
MB90F543PF	UART1	21	24	86	85	4800	9600
MB90F543GPF	UART1	21	24	86	85	4800	9600, 19200, 38400
MB90F546GPF	UART1	21	24	86	85	4800	9600, 19200, 38400
MB90F548PF	UART1	21	24	86	85	4800	9600
MB90F548GPF	UART1	21	24	86	85	4800	9600, 19200, 38400
MB90F549PF	UART1	21	24	86	85	4800	9600
MB90F553PF	UART0	20	19	86	85	4800	9600
MB90F562BPFM	UART1	14	15	26	25	4800	9600
MB90F562PFM	UART1	14	15	26	25	4800	9600
MB90F568PFM	UART1	14	15	26	25	4800	9600
MB90F574PFV	UART0	9	10	96	95	4800	9600
MB90F583PF	UART0	18	19	86	85	4800	9600
MB90F591PF	UART0	16	14	86	85	4800	9600
MB90F594APF	UART0	16	14	86	85	4800	9600
MB90F598PF	UART1	21	24	86	85	4800	9600

*The baud rates are specified for 4 MHz external clock (crystal) frequency. For other clock frequencies (8 MHz or 16 MHz), the baud rates must be multiplied by factor of 2 or 4.

The UART interfaces used for the serial **synchronous** programming mode and the CPU pin numbers are shown in the following table:

CPU type	UART used	SIN pin	SOT	SCK	P01	P00	Note
MB90F428PF	UART1	88	89	90	86	85	
MB90F497PFM	UART1	60	62	61	26	25	
MB90F497GPFM	UART1	60	62	61	26	25	
MB90F523PFV	UART1	11	12	13	96	95	Asynchro mode not supported
MB90F543PF	UART1	21	24	22	86	85	
MB90F543GPF	UART1	21	24	22	86	85	
MB90F546GPF	UART1	21	24	22	86	85	
MB90F548PF	UART1	21	24	22	86	85	
MB90F548GPF	UART1	21	24	22	86	85	
MB90F549PF	UART1	21	24	22	86	85	
MB90F553PF	UART0	20	19	18	86	85	
MB90F562PFM	UART1	14	15	60	26	25	
MB90F562BPFM	UART1	14	15	60	26	25	
MB90F568PFM	UART1	14	15	60	26	25	
MB90F574PFV	UART0	9	10	11	96	95	
MB90F583PF	UART0	18	19	20	86	85	
MB90F591PF	Serial I/O	24	26	25	86	85	
MB90F594APF	Serial I/O	24	26	25	86	85	
MB90F598PF	UART1	21	24	22	86	85	

Warning: the P01 pin is used as an output of the handshake signal, which ensures that the high-speed synchronous communication between the box and target CPU is possible. Therefore, there should not be any hard-wired logic value on the P01 CPU pin in the synchronous mode. If you have any DIP switch or jumper connected to the P01 pin on your board, make sure these are OFF in the serial synchronous programming mode !

NOTE: The recommended schematics for the user target board serial interface can be found in Appendix B.

FR family – MB91F109

In the following table, you can find the list of all the supported CPUs from this family and the pins they use for setting the serial programming mode:

CPU	Pin name	Pin No.	Logic level	Pin name	Pin No.	Logic level	Pin name	Pin No.	Logic level
MB91F109PF	MD 2,1,0	21,20,19	1,1,0	P20	28	0	P21	29	0*, 1**

* .. asynchronous programming
** .. synchronous programming

The UART interfaces used for the serial **asynchronous** programming mode and the CPU pin numbers are shown in the following table:

CPU type	UART used	SI pin	SO Pin	P20 Pin	P21 Pin	Baud rates* [Bd]	
						Kernel download	Programming
MB91F109PF	UART0	79	80	28	29	19200	19200

*The baud rates are specified for 12.5 MHz external clock (crystal) frequency. For the clock frequency 25 MHz, the baud rates must be multiplied by factor of 2.

The UART interfaces used for the serial **synchronous** programming mode and the CPU pin numbers are shown in the following table:

CPU type	UART used	SI pin	SO Pin	SC Pin	P20 Pin	P21 Pin	Note
MB91F109PF	UART0	79	80	81	28	29	

Warning: the P21 pin is used as an output of the handshake signal, which ensures that the high-speed synchronous communication between the box and target CPU is possible. Therefore, there should not be any hard-wired logic value on the P21 CPU pin in the synchronous mode. If you have any DIP switch or jumper connected to the P21 pin on your board, make sure these are OFF in the serial synchronous programming mode !

FR family – MB91F361

In the following table, you can find the list of all the supported CPUs from this family and the pins they use for setting the serial programming mode:

CPU	Pin name	Pin No.	Logic level	BOOT pin	BOOT pin No.	BOOT pin logic level
MB91F361	MD 2,1,0	113,112,111	0,0,0	P93	46	1

The UART interfaces used for the serial **asynchronous** programming mode and the CPU pin numbers are shown in the following table:

CPU type	UART used	SIN pin	SOT Pin	**Baud rates*** [Bd]	
				Kernel download	Programming
MB91F361	UART0	152	153	9600	115200

*The baud rates are specified for 4 MHz external clock (crystal) frequency. Other crystal frequencies are not supported

Note: Synchronous mode is not supported with this CPU

F²MC-8L family

In the following table, you can find the list of all the supported CPUs from this family and the pins they use for setting the serial programming mode:

CPU	Pin name	Pin No.	Logic level	Pin name	Pin No.	Logic level	Pin name	Pin No.	Logic level
MB89P935	MOD 1,0	6,5	1,1	P40	22	1	P37	11	0*, 1**
				P41	23	1			
				P42	24	0			
				P43	25	1			

* .. asynchronous programming
** .. synchronous programming

Note: The MOD1 pin must be set to +9V to enable writing to ROM. Otherwise, the ROM can be only read.

The UART interfaces used for the serial **asynchronous** programming mode and the CPU pin numbers are shown in the following table:

CPU type	UART used	SI pin	SO Pin	Baud rates* [Bd]	
				Kernel download	Programming
MB89P935	UART0	17	18	2400	2400

*The baud rates are specified for 2 MHz external clock (crystal) frequency. For the clock frequencies of 4 or 8 MHz, the baud rates must be multiplied by factors of 2 or 4, respectively.

The UART interfaces used for the serial **synchronous** programming mode and the CPU pin numbers are shown in the following table:

CPU type	UART used	SI pin	SO Pin	SC Pin	Note
MB89P935	UART0	17	18	19	

Warning: the P40 pin is used as an output of the handshake signal, which ensures that the high-speed synchronous communication between the box and target CPU is possible. Therefore, there should not be any hard-wired logic value on the P40 CPU pin in the synchronous mode. If you have any DIP switch or jumper connected to the P40 pin on your board, make sure these are OFF in the serial synchronous programming mode !

NOTE: The recommended schematics for the user target board serial interface can be found in Appendix B.

Chapter

8

Customer registration

To register, please fill this form and send it by email to the following address: microcontroller_info@fme.fujitsu.com.

Note:
1. **Boldface** indicates required fields.
2. An editable copy of this form can be found in the file "RegistrationForm.rtf", which resides in the Flashkit installation directory.

FLASHKIT serial programmer registration form

Fujitsu Microelectronics Europe GmbH
Marketing Microcomponents
Am Siebenstein 6-10
63303 Dreieich-Buchschlag, Germany
Tel.: (++49) (0)6103-690-0
Fax: (++49) (0)6103-690-122

This registration form can be used to register your FlashKit software. After your registration you will be informed about new versions of the FlashKit software. Please take care that your personal information given here is correct and complete.

First name:
Surname:
Company:
Department:
Address:
City:
State/Province:
ZIP/Postal code:
Country:
Phone:
Fax:
Email address:
Local Fujitsu or Distribution Sales Office (where Flashkit was bought)**:**

Serial number of FlashKit:
Current Version of FlashKit PC Software:
Current Version of FlashKit Firmware:

Chapter

9

Trouble shooting

Error	What happened and what to do
Kernel was not downloaded - check the settings and cables	The CPU doesn't respond. Make sure you have selected correct CPU. Check the communication cable.
Error in communication – bad CRC	During transferring data via communication cable data were repeatedly damaged. Check if the cable is properly connected to PC and CPU
Error while opening port/device	The specified COM/LPT port cannot be opened. It can be used by another application – close that application. If the LPT port can't be opened, check if: ▪ the Flash-kit SW was installed with Administrator rights. If unsure, please reinstall the Flash-kit with Administrator rights ▪ the LPT port type in BIOS is set to "SPP" ▪ there are no conflicting Windows drivers on the LPT port
Error while closing port/device	An error occurred when the port has been closed
Not supported	The kernel doesn't support some command. You probably have the different versions of Serial Programmer and Kernels
Error during transfering	During data transfer via communication cable data were repeatedly damaged. Check if the cable is properly connected to PC and CPU
Time out	The kernel doesn't respond. Make sure you have selected correct CPU. Check the communication cable.
Error during programming the FLASH - FLASH is not erased or FLASH may be damaged	Data cannot be written into the FLASH. FLASH is not erased or FLASH may be damaged. This error appears also when the FLASH is locked!
Error during erasing the FLASH - FLASH is NOT erased (FLASH may be damaged)	Error during erasing the FLASH – FLASH is NOT erased (FLASH may be damaged)
Error on communication line. Kernel returned unexpected command	During transferring data via communication cable data were damaged. Check if the cable is properly connected to PC and CPU

Chapter

10

Revisions and changes

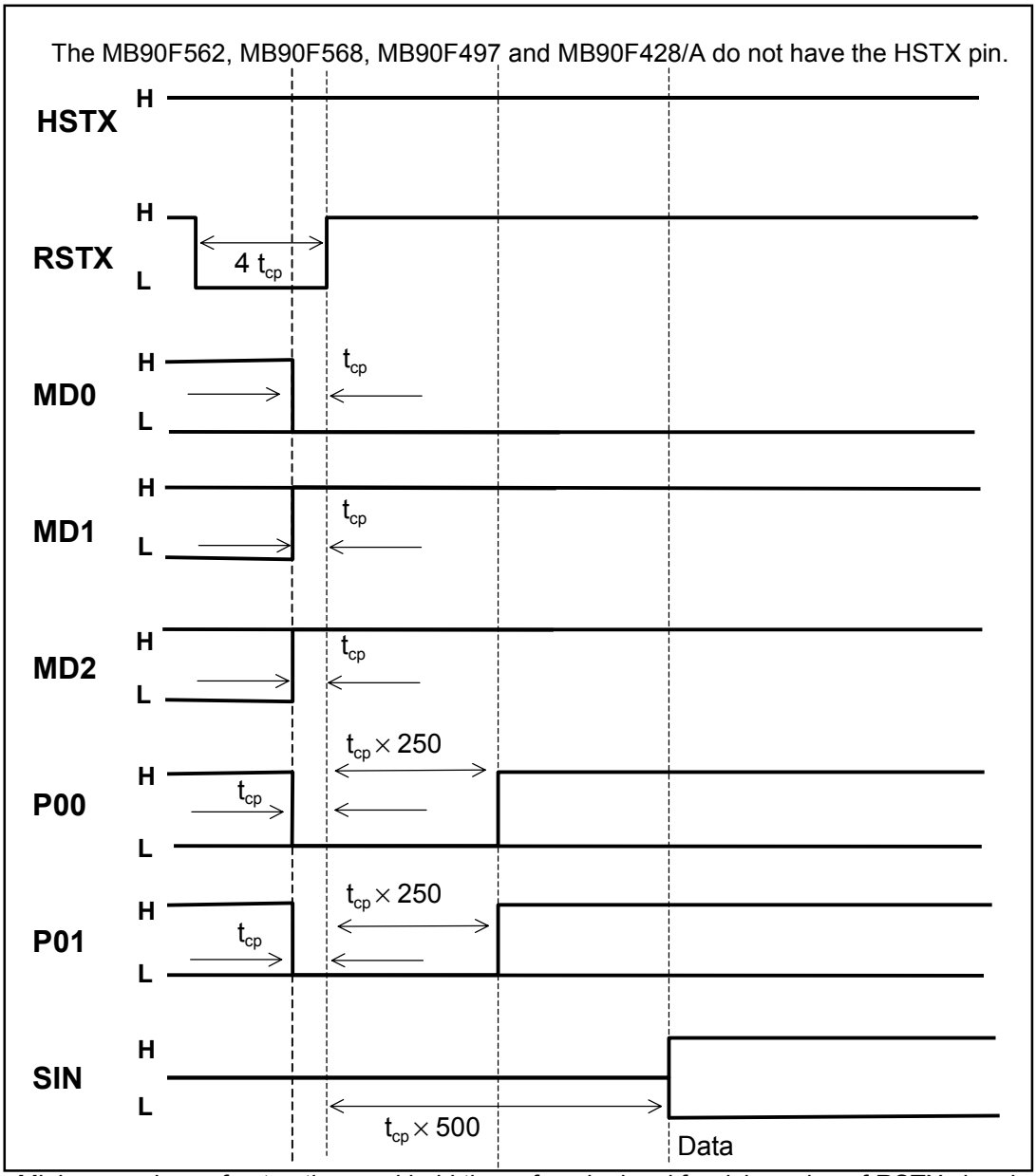
Date	Revision – Errors	Revised version
15.01.2001	This version of the manual is distributed with the Flashkit programmer 1.4	V2.0
23.04.2001	In Appendix B, the following changes were made: ▪ Paragraph "Recommended circuit for asynchronous mode", page 63 – the text was changed in order to be more comprehensive ▪ paragraph "Schematics for MB89P935 programming", page 67 – the figure of the target programming connector had no description on pin 14	V2.1
24.04.2001	In Appendix B, the following changes were made: ▪ added paragraph "General design rules for user target boards" ▪ All the schematics revised to comply with these design rules	V2.2
18.06.2001	The following changes were made: ▪ the description of the new features in Flashkit SW ver. 2.0 was added into the manual (see chapter 3) ▪ new CPUs, supported by the Flashkit version 2.0c, added into chapter 7 – "Supported CPUs" ▪ Chapter 2 "Installation" was extended to contain all the known installation problems ▪ on page 25, the programming cable schematics was corrected. On page 2, an explanation note was added to the MB89P935 connector	V2.3
9.07.2001	The following changes were made: ▪ in Appendix B, the chapter "Workaround solution for Pulldown on PIO2 issue" was added. ▪ In chapter 4, the name "Serial interface connector" was changed to "Programming connector" to keep consistency with the rest of the manual, where this name was used for this connector, while the name "Serial interface connector" was used only in Chapter 4.] ▪ the "Warranty and Disclaimer" was updated	V2.4
31.07.2001	Several minor errors corrected from V2.4.	V2.5
02.01.2002	Several minor errors corrected from V2.5.	V2.6
13.01.2003	In Command line parameters: new command line parameter CPU Operating Mode after Auto Program (-cpumode)	V2.7

29/01/2003	The following changes were made: ▪ In chapter Timing diagrams note added (timing diagram is for asynchronous mode) ▪ Changed all notices about the latest firmware (file Firmware_v17.mhx) ▪ In Appendix B: Schematics "The target serial interface recommendation – version with the transistors" changed – fixed generating MD2 signal (added resistor R9 to the base of the transistor T6) ▪ In chapter F^2MC-16LX family table for asynchronous programming updated (kernel communication speeds for MB497PFM).	V2.8
09/03/2004	In chapter Command line parameters incorrect case for parameter **Watchdog trigger pulses generation** fixed (correct form is -wd). In chapter Using the programmer in batch files added one return error code („13 – disk I/O error") and updated batch example.	V2.9

Appendix A

Timing diagrams

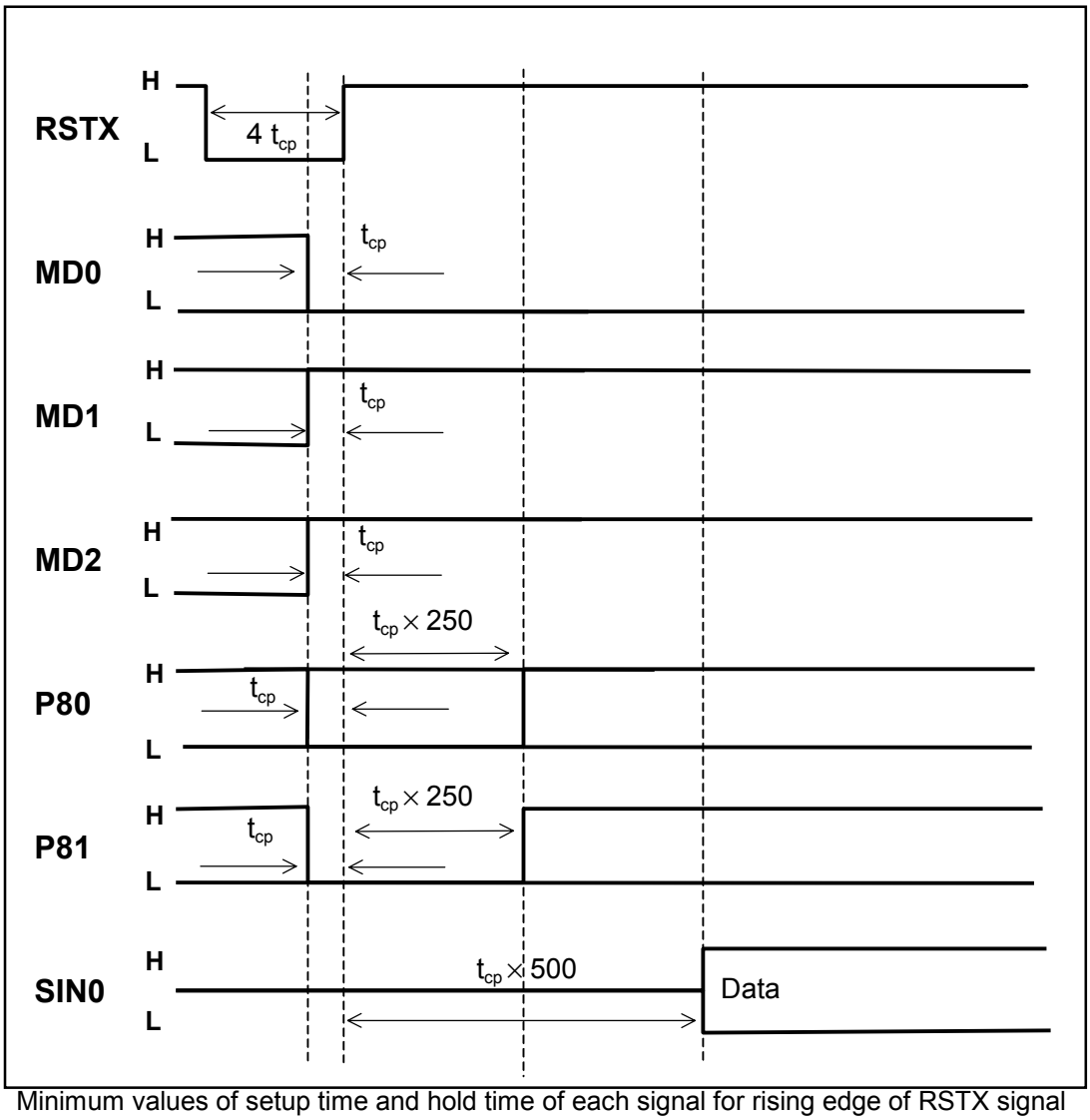
Timing Chart for each pin of microcontrollers other than MB90F474 and MB90F476:



Minimum values of setup time and hold time of each signal for rising edge of RSTX signal

Note: This timing diagram is for asynchronous programing, for synchronous programming the level of pin P01 must be high during reset (for more details see table for F²MC-16LX family in the Chapter 7 Supported CPUs)

Timing Chart for Each Pin of MB90F474 and MB90F476:



Minimum values of setup time and hold time of each signal for rising edge of RSTX signal

Communication protocols (16LX family)

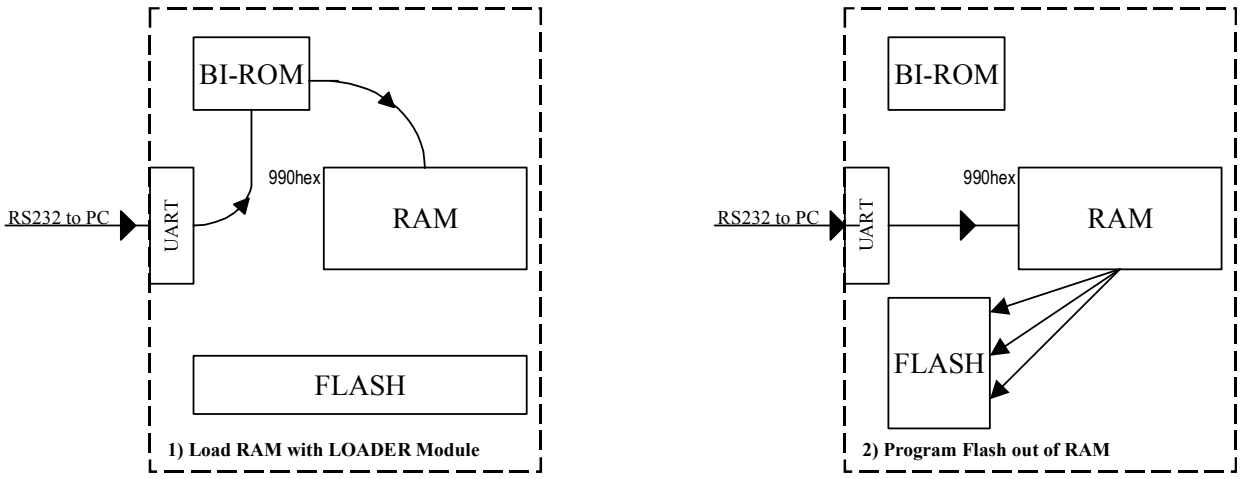
In the following text, the communication protocols used by the Serial Programmer software within the 16LX family are described.

For basic understanding of how the FLASH programming is done within the 16LX family, it is important to know that the 16LX CPUs have an in-built ROM memory called BiROM, which is activated only in the Serial programming mode. In this memory, there is a short code, that can be used for downloading and executing another program in the CPU internal RAM. Since the Bi-ROM code has no ability for programming the FLASH, it can be used only for dowloading of a more sophisticated code, that can be then used for FLASH programming. In case of the Flashkit programmer, we call this „more sophisticated code“ as „kernel“. In the following paragraphs, the communication protocols used by the Bi-ROM code and our kernel are described.

Bi-ROM protocol

The F²MC-16LX Flash MCU contains a burn-in ROM (BiROM) program that supports a proprietary protocol to allow downloading of a user program to on-chip RAM memory (step 1). The user program is then able to manipulate on-chip Flash memory as required (step 2).
Two basic serial modes are supported, synchronous serial and asynchronous serial. It is not important to the protocol which serial mode is in use.

The below diagram illustrates the context.



The following text describes the commands, which are supported by the BiROM of the 16LX Flash MCUs in order to generate an own programming environment.

As already mentioned, two basic serial modes are supported, synchronous serial and asynchronous serial. After reset of the MCU, the mode pins and two port pins select the programming mode respectively. It is not important to the protocol which serial mode is in use. However, communications settings obviously vary:

Synchronous	8 data bits, external clock (500kbs max)
Asynchronous	8 data bits, 1 stop bit, no parity, baud rate: (mcu clk / 4) / (8 x 13 x 2) (4800 @ 4MHz, 9600 @ 8MHz, 19200 @ 16MHz)

Follow the sequences in the examples to download and execute the user program. Once the user program is running, the BiROM is no longer active and all further communication is user defined. To allow compatibility with all devices, it is important that the user program uses the minimum of resources. Therefore, we recommend your program uses the following memory map:

Memory Map

0100 – 016F	Variables
0170 – 017F	Stack
0180 – 018F	Registers (bank 0)
0190 – end of RAM	User program code and write buffer (512B max)

Common Pin Settings

Pin Name	Logic Level	Description	QFP100	QFP120
MD2,1,0	110	Programming mode	51,50,49	87,88,89
P00	0	Programming mode	85	95 (J19/21)*)
P01	0 1	Async, 4800, 8bit, 1 stop, no parity Clk Sync, Ext clock (500kbs max)	86	96 (J19/20) *)
Vss	-	Power supply	81	91 (J19/25) *)
Vcc	-	Power supply	84	94 (J19/22) *)

*) Pin numbers in brackets refer to the QFP120 Flash-Test-Board (FLASH-EVA2-120P-M13)

Commands

Byte 0								Byte 1								Byte 2								Byte 3								Byte 4								Byte n							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Command 7-0								Address 15-8								Address 7-0								Count 15-8								Count 7-0								Data/Checksum 7-0							

Command	Various actions, see table below.
Address	Start address of RAM download code
Count	Number of bytes to transfer. 1 = 1 byte.
Data	Data bytes sent
Checksum	Cumulative sum

Commands									Description	Comment
0	0	0	1	1	-	-	-	18	Communication check	General communications check
0	0	0	0	0	-	-	-	00	Download	User program is downloaded to RAM
0	1	0	0	0	-	-	-	4x	Execute	User program is executed. Address and count ignored (address fixed to 0990h (0190h MB90560))

Command Responses

Byte 0							
7	6	5	4	3	2	1	0
Command 7-4				Resp 3-0			

Resp Status response from MCU (bits 7-4 return bits 7-4 of command byte)

Response								Description	Comment
-	-	-	-	0	0	0	1	x1	OK	
-	-	-	-	0	0	1	0	x2	Command Error	

EXAMPLES

General Communications Check

This command is in fact used for initailizing the communication with the BI-ROM. The PC sends the byte 18H, and if the BI-ROM is ready, it responses with the byte 11H.

PC	18	
MCU		11

Download (00h)

	command / address			count		data		chk	resp
PC	00	09	90	00	02	01	02	9E	
MCU									01

This example downloads 2 data bytes, 01_{hex} and 02_{hex} onto RAM location 990_{hex}. See also the cumulated checksum $9E_{hex}$ and response from the MCU.

Execute (40h)

	command / address			count	
PC	40	xx	Xx	00	00
MCU					

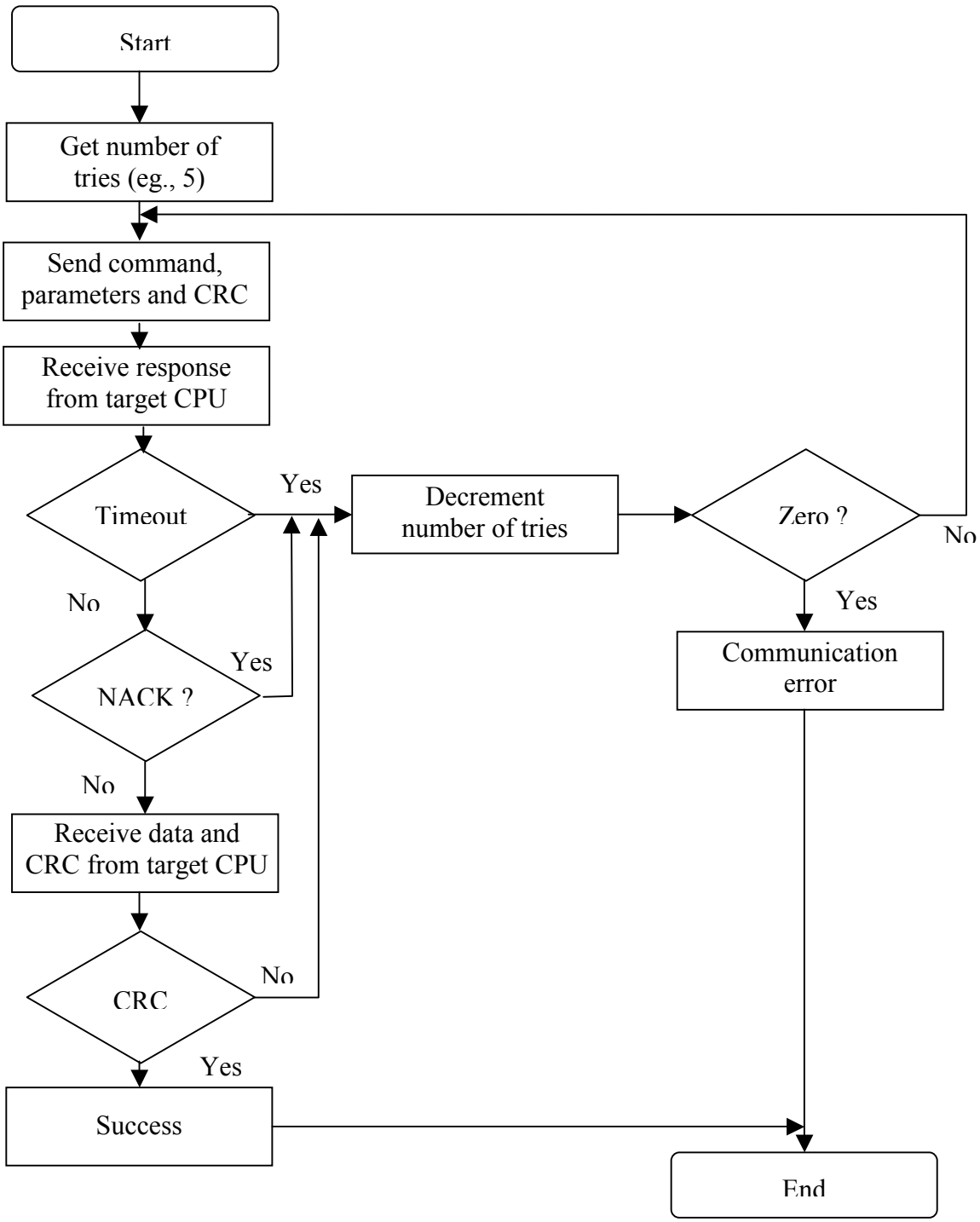
no response, jump is immediate

Note

When you select the Burn-IN ROM mode for the CPU, and you try to program the upper Flash memory area with code executed in RAM the situation is as follows:
In Burn-IN ROM mode the Burn-IN ROM is always visible at FF0000-FFFFFF. So you cannot program the page FF directly. Therefore Bit 3 of the FMCS register is used. Bit 3 of the FMCS register is used as a upper memory enable. To program the page FF, you have to set this bit first. After this the page FF will be mapped to page FE.

Communication with kernel – asynchronous mode

If the asynchronous communication is chosen in the serial programmer SW (the „Asynchronous – COM port“ option is selected in the „Select communication channel“ radiobox), the communication protocol looks like the one on the following picture:



What does this diagram mean ?

The PC frontend communicates with the kernel, which was downloaded to the CPU RAM after the „Connect“ function was invoked, using various commands. A general command can

optionally contain parameters and is always appended with CRC checksum of the command and parameters.

The command, its parameters and the CRC are sent as one „packet“ to the target CPU. After the last byte of this „packet“ is sent, a response from target CPU is expected. If the target does not response for a certain period of time, the timeout is caused and the whole operation is repeated. The number of repetitions (5), which is in the diagram, is just for illustration purposes, its value was „hardwired“ in the software after some experiments with reliability/speed of error detection.

If the target CPU responds, it can send NACK to indicate a CRC check error, or a valid response. The valid response forms also a „packet“. First byte of this „packet“ is always the same command that the target CPU replies to. The other bytes are dependent on the type of command issued, but the last bytes are always the CRC checksum.

If the PC SW receives a „NACK“ response, it repeats the whole operation of sending the command „packet“. The same thing happens in case the PC receives valid response, but its CRC check fails. If the CRC check is OK, the processing of the command successfully finishes.

The formats of „packets“ used in the asynchronous communication is described in the following paragraphs. Note that there are 2 kinds of CRC checksum in the asynchronous communication, denoted as „CRC24“ and „CRC16x“. The algorithms for computing these CRCs are described in the part „**CRC checksum algorithms**“ of this chapter.

General command for target CPU

Byte 0	Byte 1..n	Byte n+1	Byte n+2	Byte n+3
Command nr.	Parameters	CRC24 7-0	CRC24 15-8	CRC24 23-16

Remark: CRC24 is computed from Command and all the parameters

General response from target CPU

When the transmission is successful:

Byte 0	Byte 1..n	Byte n+1	Byte n+2
Command nr.	Returned data	CRC16x 7-0	CRC16x 15-8

In case of corrupted data on the transmission line (the target CPU receives the command packet, but its CRC check fails):

Byte 0
NACK - $F9

Here goes the detailed description of data sent in the various command packets:

Blank check – Command nr. = $EC

Checks if the specified block of memory is erased (filled with $FF):

Parameters:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Start addr 7-0	Start addr 15-8	Start addr 23-16	End addr 7-0	End addr 15-8	End addr 23-16

Returned data:

Byte 1
STATUS

STATUS $FF – specified memory range is erased
otherwise - specified memory range is NOT erased

Erase chip – Command $EF

Erases whole chip.

Parameters:

Byte 1	Byte 2	Byte 3
Addr 7-0	Addr 15-8	Addr 23-16

Where “Addr” is the base address of the FLASH memory.

Returned data:

Byte 1
STATUS

STATUS 0 – chip is erased
otherwise - chip cannot be erased

Erase sector – Command $ EE

Erases only one sector of the FLASH memory

Parameters:

Byte 1	Byte 2	Byte 3
Addr 7-0	Addr 15-8	Addr 23-16

Where “Addr” is the starting address of the FLASH sector.

Returned data:

Byte 1
STATUS

STATUS 0 – sector is erased
otherwise - sector cannot be erased

Write FLASH – Command $ED

Parameters:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6..Length-1
Start addr 7-0	Start addr 15-8	Start addr 23-16	Length 7-0	Length 15-8	Data

Remark:
Size of the packet cannot be greater than input buffer of the target CPU (see command **Kernel buffer length)**

Returned data:

Byte 1
STATUS

STATUS 0 – FLASH memory has been programmed
otherwise – error during programming FLASH memory

Read FLASH – Command $F2

Read data from FLASH memory. Address range is specified by **start address** and **length** of the block.

Parameters:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Start addr 7-0	Start addr 15-8	Start addr 23-16	Length 7-0	Length 15-8

Returned data:

Bytes 1..Length
Data

Kernel buffer length – Command $FE

Tests if the kernel is running and returns the size of internal buffer for incomming commands.

No parameters

Returned data:

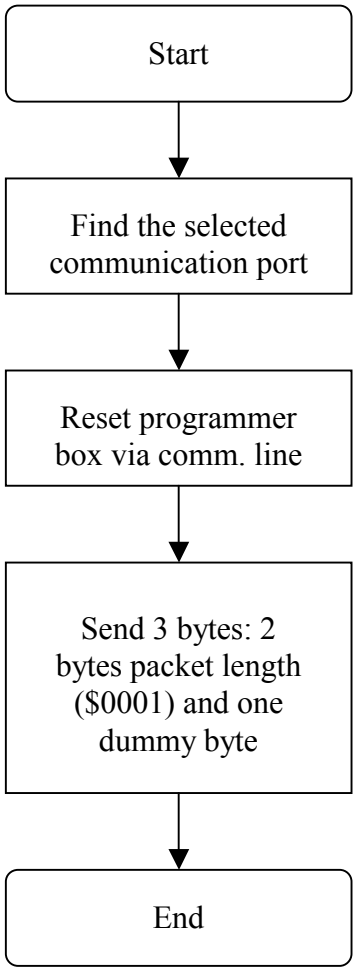
Byte 1
Length

Length – Length of the input buffer for current target CPU.

Communication with kernel - synchronous mode

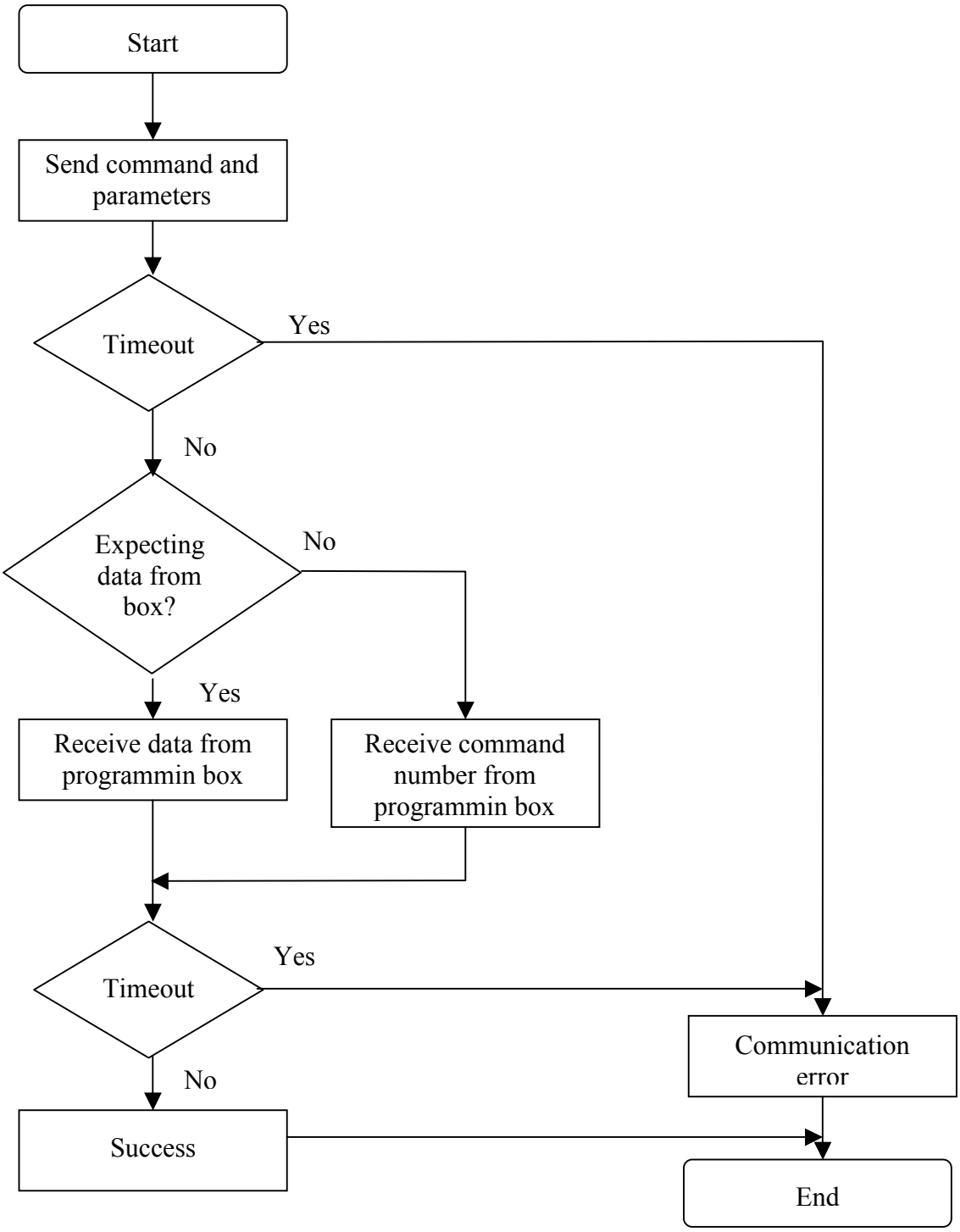
When the „Synchronous – COM port“ or Synchronous – LPT port“ option is selected in the „Select communication channel“ radiobox, the synchronous communication is chosen. Here, the situation is a little bit more complicated, because there is the programmer box in the data path between the PC and target CPU. The programmer box is used as a bi-direction re-transmitter in most case, but there are few dedicated commands that can be used for controlling the box functions (e.g., for setting the target system voltage levels or lighting-up the LED diodes).

In the synchronous mode, when the „Connect“ button is pressed, the following initialization sequence is done by the PC software:



Basically, after the „Connect“ button is pressed, the box is reset by the reset line of the selected communication port. Then the PC SW sends a „dummy“ packet to the selected port. The box (if present) is waiting for this packet on both the parallel and serial ports to find which port will be used for further communication. Note that no response is submitted by the box. If the box is not connected, the timeout error will be issued during the following communication (after the initialization sequence, commands for setting the LED lights & target CPU voltage levels are issued).

The commands in the synchronous mode are issued in the following sequence:



Note that no repeating is used in case of error and no CRC check is used when communicating with the box. This solution was chosen to achieve to maximum programming speed – when communicating by the parallel port, there all the CPU time is used for the communication a FLASH programming purposes. Without the CRC, the effective programming speed while programming large block of data can be as high as 320 kbit/s. If the CRC computation, which is quite time consuming should take place during FLASH programming, the programming speed would be cut to half.

The format of the command and response packets used in the synchronous communication is described in the following paragraphs:

General command for the programmer box

Byte 0	Byte 1	Byte2	Byte 3..n
Packet length 7-0	Packet length 15-8	Box command	Parameters

General response from programmer box

Byte 0	Byte 1	Byte 2.. Packet length+1
Packet length 7-0	Packet length 15-8	Returned Data

Remark: if the programmer box has nothing to return, then Returned data are 1 byte long and the content is "Box command" (the same command that the box replies to).

Commands for programmer box:

Prepare for executing kernel for target CPU – Command $F0

Before the kernel in the target CPU is executed, this command must be called just before the kernel is started.

No parameters

Returned data:

Byte 2
$F0

Forward data to target CPU – Command $F7

Serves for sending (in synchronous mode) data into the target system. Also returns the response from the target CPU.

Parameters:

Byte 1	Byte 2	Byte 3..n
Response length 7-0	Response length 15-8	Data for target CPU

Response length – number of bytes returned by the target CPU.

Returned data:
If response length is 0, returned data are:

Byte 2
$F7

Otherwise:

Byte 2..Response length+1
Data from target CPU

Forward data from target CPU to PC – Command $F6

Reads data from target CPU.

Parameters:

Byte 1	Byte 2
Response length 7-0	Response length 15-8

Response length – number of bytes which the target CPU will return

Returned data:

Byte 2..Response length+1
Data from target CPU

Set target CPU and supply mode – Command $F5

Sets the pins for serial programming mode, supply voltage and supply source.

Parameters:

Byte 1	Byte 2
Mode1	Mode2

Mode1:

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Pin MD0	Pin MD2	Reset line	Pin P00	Pin P01	Voltage 3/5V	Supply target	Power supply info

- Pin MD0: state of this pin
- Pin MD2: state of this pin
- Reset line: controls reset line for the target CPU
- Pin P00: state of this pin
- Pin P01: state of this pin
- Voltage: 1 = 5V, 0 = 3V
- Supply target: 1 = do not supply the target CPU from programmer box, 0 = supply the target CPU
- Power supply info: the mode contains the supply information

Mode2:

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Pin MD1							

Returned data:

Byte 2	Byte 1
$F5	Power good

Power good:
If Supply target is 1 (do NOT supply target CPU) then Power good is ignored
Otherwise if the Power good is >0 then the power is not good (target CPU cannot be supplied)

Remark: To reset the target CPU this command must be called twice with right value of bit 2. "Power supply info" should be only in the first of both command.

Set the Busy LED to ON/OFF – Command $EF

Lights green/red busy LED for displaying of the activity of the programmer box.

Parameters:

Byte 1
State

State: 1 = busy, 0 = not busy

Returned data:

Byte 2
$EF

Commands for target CPU

These commands are the same as for the asynchronous programming, but they are inserted into the command **Forward data to target CPU** as the parameter. Responses from target CPU can be read by directly by command **Forward data to target CPU** or **Forward data from target CPU**. The better is the first one.

Special command for target CPU (synchronous mode only, special kernel)

Computing CRC of a block of data in target system – Command $F0

Kernel in the target CPU computes the CRC of specified block of memory and PC software compares with own precomputed CRC. It is used for comparing data in the FLASH memory and in the user program. This is useful for large user data.

Parameters:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Start addr 7-0	Start addr 15-8	Start addr 23-16	Length 7-0	Length 15-8

Returned data:

Byte 0	Byte 1
CRC16 7-0	CRC16 15-8

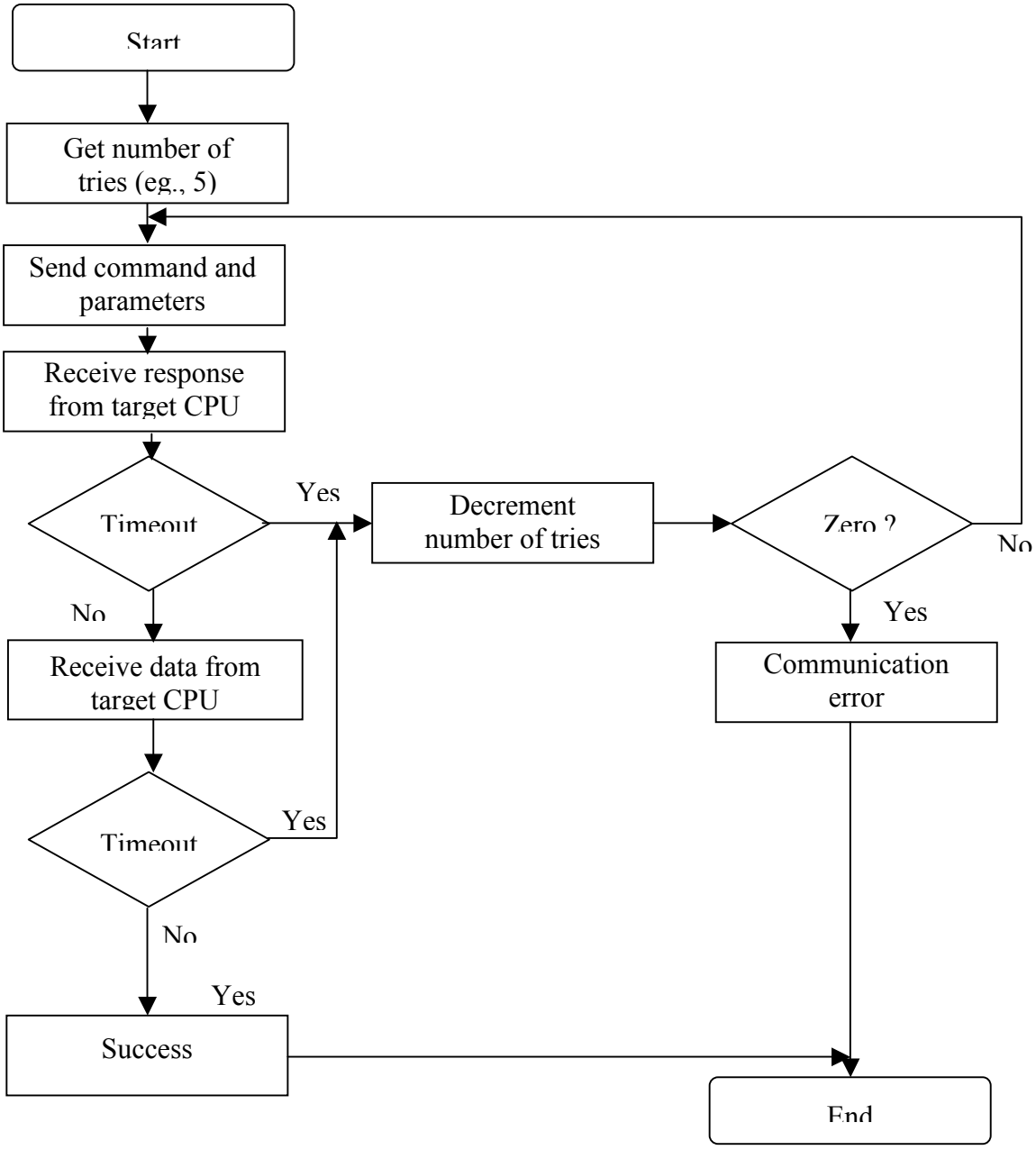
Remark: The CRC16 checksum is described in following part of this chapter.

Communication protocols (8L CPUs)

In the following text, the communication protocols used by kernels of the Flashkit programmer within the 8L family are described (for description of protocols used by the CPU boot-ROM, please check the documentation of the relevant CPU).

Communication with kernel – asynchronous mode

If the „Asynchronous – COM port“ option is selected in the „Select communication channel“ radiobox, the communication protocol looks like the one on the following picture:



This diagram is similar to the one, which is presented in the 16LX family communication description part, except for

- CRC checksum, which is not computed nor transmitted due to the limitations of the memory space and execution speed of the 8L family CPUs.
- Address is only 16-bits number
- Data length is 8-bits number

The formats of „packets“ used in the asynchronous communication is described in the following paragraphs.

General command for target CPU

Byte 0	Byte 1..n
Command nr.	Parameters

Remark: CRC24 is computed from Command and all the parameters

General response from target CPU

When the transmission is successful:

Byte 0	Byte 1..n
Command nr.	Returned data

In case of corrupted data on the transmission line (the target CPU receives the command packet, but its CRC check fails):

Byte 0
NACK - $F9

Here goes the detailed description of data sent in the various command packets:

Blank check – Command nr. = $EC

Checks if the specified block of memory is erased (filled with $FF):

Parameters:

Byte 1	Byte 2	Byte 4	Byte 5
Start addr 7-0	Start addr 15-8	End addr 7-0	End addr 15-8

Returned data:

Byte 1
STATUS

STATUS $FF – specified memory range is erased
otherwise - specified memory range is NOT erased

Erase chip – Command $EF

Erases whole chip.

Parameters:

Byte 1	Byte 2
Addr 7-0	Addr 15-8

Where "Addr" is the base address of the FLASH memory.

Returned data:

Byte 1
STATUS

STATUS 0 – chip is erased
otherwise - chip cannot be erased

Erase sector – Command $ EE

Erases only one sector of the FLASH memory

Parameters:

Byte 1	Byte 2
Addr 7-0	Addr 15-8

Where "Addr" is the starting address of the FLASH sector.

Returned data:

Byte 1
STATUS

STATUS 0 – sector is erased
otherwise - sector cannot be erased

Write FLASH – Command $ED

Parameters:

Byte 1	Byte 2	Byte 4	Byte 5..Length-1
Start addr 7-0	Start addr 15-8	Length 7-0	Data

Remark:
Size of the packet cannot be greater than input buffer of the target CPU (see command **Kernel buffer length)**

Returned data:

Byte 1
STATUS

STATUS 0 – FLASH memory has been programmed
otherwise – error during programming FLASH memory

Read FLASH – Command $F2

Read data from FLASH memory. Address range is specified by **start address** and **length** of the block.

Parameters:

Byte 1	Byte 2	Byte 4
Start addr 7-0	Start addr 15-8	Length 7-0

Returned data:

Bytes 1..Length
Data

Kernel buffer length – Command $FE

Tests if the kernel is running and returns the size of internal buffer for incomming commands.

No parameters

Returned data:

Byte 1
Length

Length – Length of the input buffer for current target CPU.

Communication with kernel – synchronous mode

When the „Synchronous – COM port“ or Synchronous – LPT port“ option is selected, synchronous communication is used.
The communication between PC and the programmer box is the same as with the 16LX family. The only difference is in the commands for the target CPU.

Commands for target CPU

These commands are the same as those for the asynchronous programming, but they are inserted into the “synchronous” command **Forward data to target CPU** as the parameter. Responses from target CPU can be read by directly by command **Forward data to target CPU** or **Forward data from target CPU**. The better is the first one.

Note: in comparison with the 16LX family, the 8-bit CPUs have no special kernel for CRC verification of the burned memory

Communication protocols (FR CPUs)

In the following text, the communication protocols used by kernels of the Flashkit programmer within the FR family are described (for description of protocols used by the CPU boot-ROM, please check the documentation of the relevant CPU).

Communication with kernel – asynchronous mode

The asynchronous communication is exactly the same as the one presented in the 16LX family communication description part, except for

- Addresses are 32-bits number

Communication with kernel – synchronous mode

The synchronous communication is exactly the same as the one presented in the 16LX family (except that addresses in the commands for the target CPU, which are 32-bits long)

CRC checksum algorithms

CRC16x

This 16-bit CRC is returned from the target CPU in the asynchronous programming mode. The algorithm for computing it (written in Object Pascal for better readability) is following:

```
(*************************************************************)
(*xor16calc
   calculates CRC from block of data
   Parameters:
     initCRC      - initial value for CRC
     buf          - starting address of data block
     len          - length of the data block
*)
(*************************************************************)
function.xor16calc(initCRC :word; buf :PCHAR; len :word) :word;
begin
{$R-}
  while (len>0) do begin
    initCRC:=((initCRC shl 5) or (initCRC shr 11)) xor BYTE(buf^);
    dec(len);
    inc(buf);
  end;
  result:=initCRC;
{$R+}
end;
```

CRC24

The 24-bit CRC is sent to the target CPU in the asynchronous programming mode.
Remark: CRC for target CPU is first of all computed by function `xor16calc` and the result is passed to this function `xor16compl3`.

```
(*************************************************************)
(*xor16compl3
   calculates advanced CRC from basic CRC (xor16Calc) - easy for
checking for target system
   Parameters:
     x16          - CRC
*)

(*************************************************************)
function.xor16compl3(x16:word):longword;
var
  b1,b2,b3 : byte;
  res : longword;
begin
x16 := ((x16 shl 5) OR (x16 shr 11));
  b1  := x16 AND $FF;
  x16 := x16 XOR b1;
  res := b1;

  x16 := ((x16 shl 5) OR (x16 shr 11));
  b2  := x16 AND $FF;
  x16 := x16 XOR b2;
  res := res OR ( longword(b2) shl 8 );

  x16 := ((x16 shl 5) OR (x16 shr 11));
  b3  := x16 AND $FF;
  res := res OR ( longword(b3) shl 16 );

  Result := res;
end;
```

CRC16

This CRC is used in the synchronous mode by the special CRC kernel for comparing data in the FLASH with the user data file.

It is standard 16-bit CRC (generating polynom is $x^{16}+ x^{12} +x^5 + 1$). The following source code is taken from the document "GZIP file format specification version 4.3", copyright © 1996 L. Peter Deutsch. (Note: permission is granted to copy and distribute this document for any purpose and without charge, including translations into other languages and incorporation into compilations, provided that the copyright notice and this notice are preserved, and that any substantive changes or deletions from the original are clearly marked.)
A pointer to the latest version of this and related documentation in HTML format can be found at the URL ftp://ftp.uu.net/graphics/png/documents/zlib/zdoc-index.html.

```
unsigned long crc_table[256];    /* Table of CRCs of all 8-bit messages. */
int crc_table_computed = 0; /*Flag: has the table been computed? Initially false.*/

void make_crc_table(void)        /* Make the table for a fast CRC. */
{
  unsigned long c;
  int n, k;

  for (n = 0; n < 256; n++) {
    c = (unsigned long) n;
    for (k = 0; k < 8; k++) {
      if (c & 1) {
        c = 0xedb88320L ^ (c >> 1);
      } else {
        c = c >> 1;
      }
    }
    crc_table[n] = c;
  }
  crc_table_computed = 1;
}
/*  Update a running crc with the bytes buf[0..len-1] and return the updated crc.
The crc should be initialized to zero. Pre- and  post-conditioning (one's omplement)
is performed within this function so it shouldn't be done by the caller. Usage
example:
   unsigned long crc = 0L;
   while (read_buffer(buffer, length) != EOF) {
     crc = update_crc(crc, buffer, length);
   }
   if (crc != original_crc) error();
*/

unsigned long update_crc(unsigned long crc, unsigned char *buf, int len)
{
  unsigned long c = crc ^ 0xffffffffL;
  int n;

  if (!crc_table_computed)
    make_crc_table();
  for (n = 0; n < len; n++) {
    c = crc_table[(c ^ buf[n]) & 0xff] ^ (c >> 8);
  }
  return c ^ 0xffffffffL;
}

/* Return the CRC of the bytes buf[0..len-1]. */
unsigned long crc(unsigned char *buf, int len) {return update_crc(0L, buf, len); }
```

Appendix B

Schematics

General design rules for user target boards

The following rules must be kept when designing the user target board in order to allow the board interoperability with Flashkit serial programmer.

Target board with asychronous programming mode only

1. The asynchronous programming mode requires an RS232 interface to be included on the user target board. There are no special requirements for the RS232 interface design, but in order to utilize the „automatic reset & mode setting“ function of the Flashkit programmer SW, we recommend to design the RS232 interface according to the schematic, which can be found in the paragraph „ **Recommended circuit for asynchronous mode**“
2. If an external watchdog is used on the user target board, there must be a way to disable it (by a jumper or a switch) during serial programming of the CPU.

Target board with synchronous programming mode

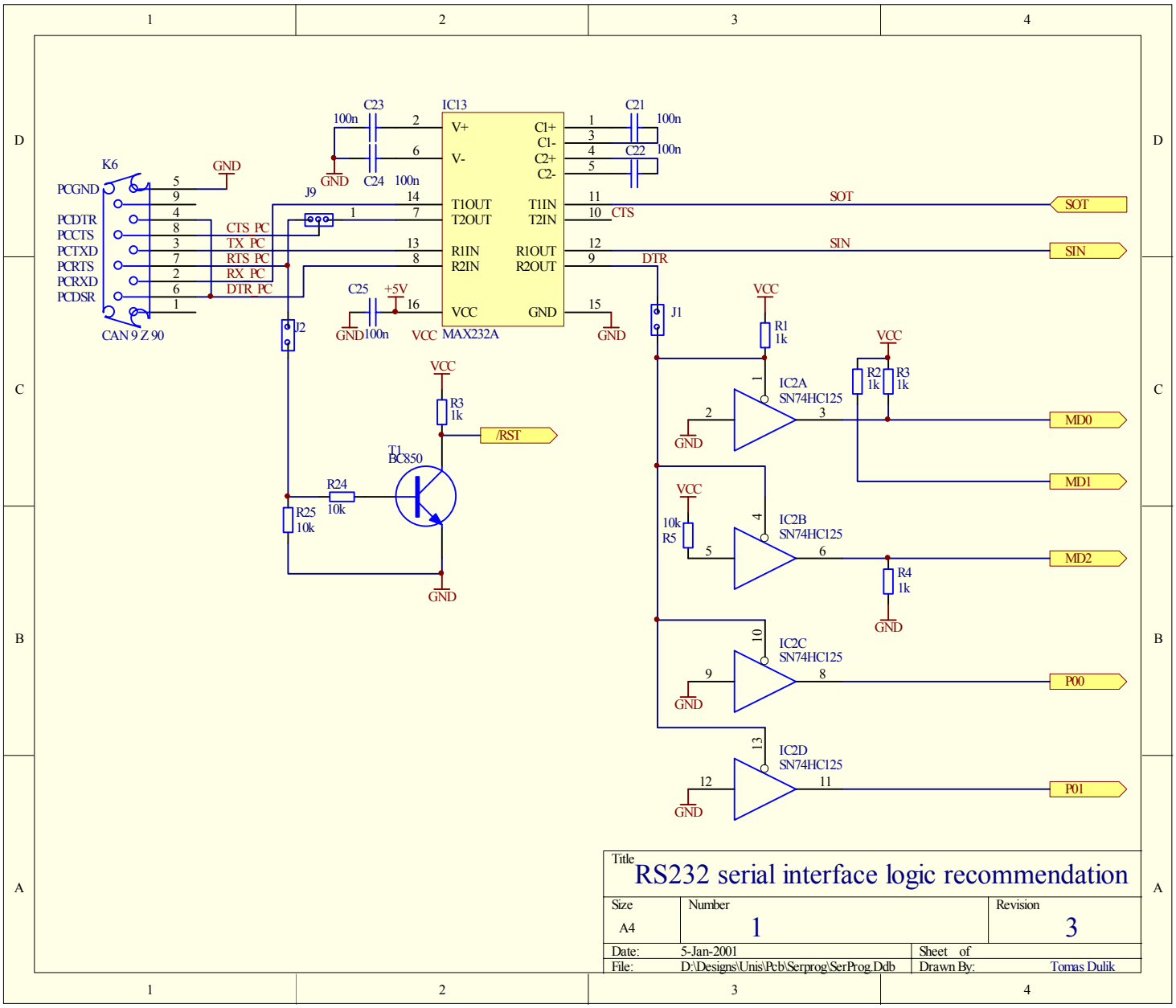
1. The most important rule to keep in mind is that the following pins of the programmer box programming connector work in the „open collector“ mode: P00 (pin 1), MD0 (pin 2), MD1 (pin6), MD2 (pin 10) and RTX (pin3). Since the „open collector“ drivers inside the serial programmer box use 10k pullup resistors, the signals connected to these pins on your target board MUST NOT be pulled down by any jumper, switch or resistor smaller than 50k during the serial programming.
   Warning: Take care, a pull down resistor of 50k or higher might not be sufficient to guarantee the correct low level specified in the Datasheet of the corresponding microcontroller for normal operation mode! So it is recommended to assign a jumper, to disconnect the pull-down resistor in case of serial synchronous programming.
   Note: For target boards that already have pulldown resistor < 50k on the MD2 pin (16LX CPU) or MOD0-1 pins (8L CPU), the Flashkit programmer software version 2.0 can be used. This version implements a workaround solution, which utilizes a modified programming cable – for further details, see the chapter „Workaround solution for Pulldown on MD2 issue“.
2. It is possible to design the user target board in a way that it supports both the synchronous and asynchronous programming mode – the recommended schematics for serial interface of such a board can be found in the paragraph „**Target serial interface schematics for 16LX and FR30**“
3. If an external watchdog is used on the user target board, there must be a way to disable it (by a jumper or a switch) during serial programming of the CPU. If this is not possible, then the box programming connector pin PIO1 can be used to trigger the watchdog during the programming. The generation of the watchdog trigger pulses can be activated in the Options menu, tab „Synchronous mode“.
   Warning: the watchdog trigger pulses generation function is not supported for the 8L family!

Recommended circuit for asynchronous mode

This schematics is recommended as the RS232 serial interface for user target boards. It can be considered as an option, which makes the usage of our Serial programmer SW even more comfortable in the case the user wants to use the asynchronous mode programming only.

In this case the additional hardware can be used, in order to switch to serial programming mode automatically, using the RTS and DTR handshake lines of the RS232 interface, which are controlled by the Flashkit software. The following schematics shows such an additional hardware, where:

- the RTS line of the RS232 interface is used for the board reset,
- the DTR line of the RS232 interface is used for the mode selection



Target serial interface schematics for 16LX and FR30

The following schematics contain the circuitry that supports both the asynchronous and synchronous programming modes withing the 16LX and FR CPU families.

The **asynchronous programming interface** is composed from the connector K1, RS232 interface IC1, the gates IC2A-IC2D and the transistor T1.

The transistor T1 is used for resetting the target board by the RTS_PC signal of the RS232 interface. The jumper J5 can be used for enabling or disabling this function. The jumper J1 is used for cases where the PC front-end software, which is used with the user target board application, requires a loopback on the RTS/CTS signals.

The gates IC2A-IC2D can be optionally used to switch the mode from the single chip to the serial programming mode (and back) automatically by the Serial Programmer software. For this automatic mode switching, the DTR_PC signal is used. The jumper J4 can be used for enabling or disabling this function.

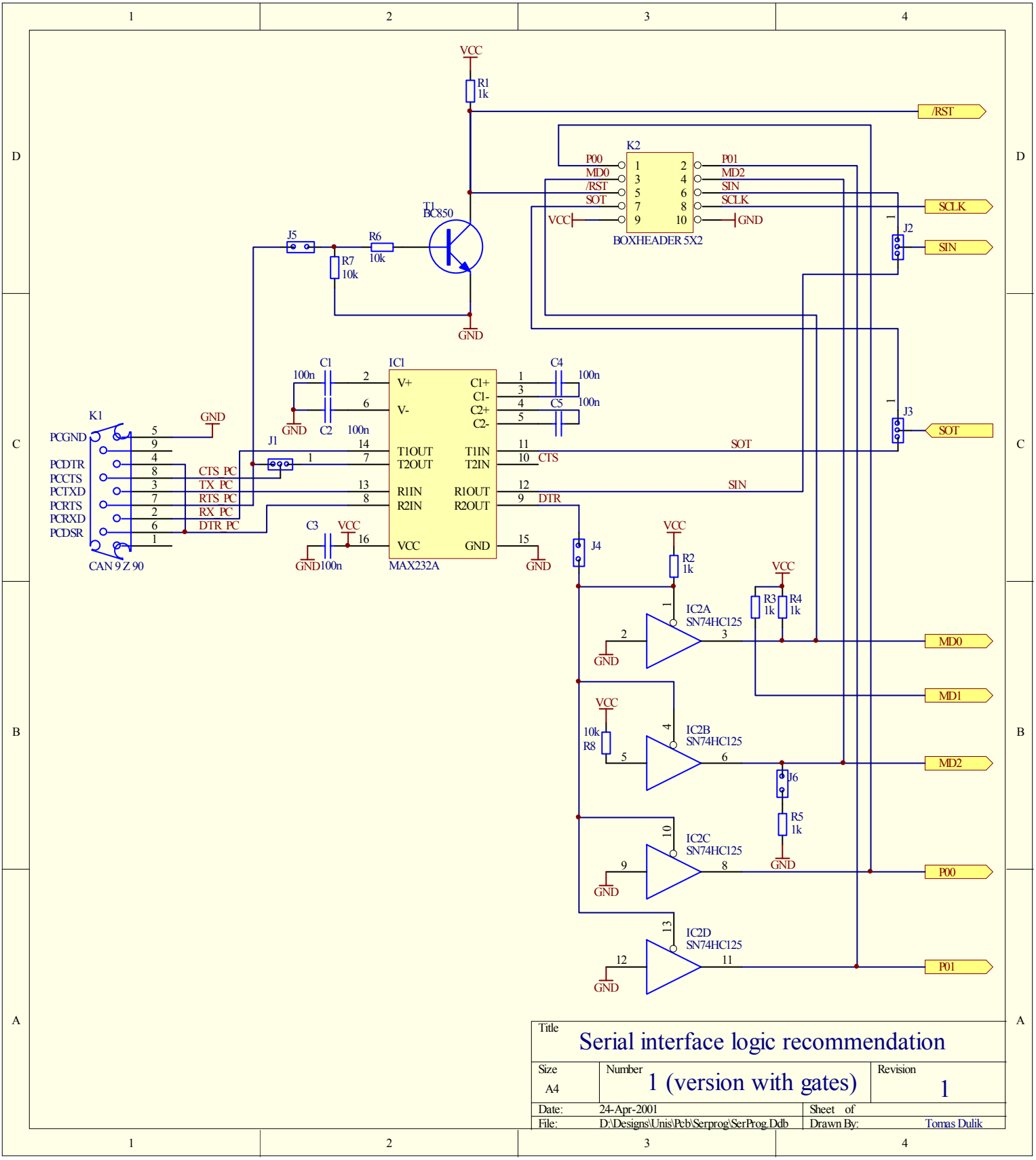
The **synchronous programming interface** is composed just from the connector K2. The signals from this connector are connected directly to the CPU pins.

The jumpers J2, J3 are used for selecting which of the two interfaces will be used – in the 1-2 position, the synchronous interface is used, otherwise the asynchronous one is selected.

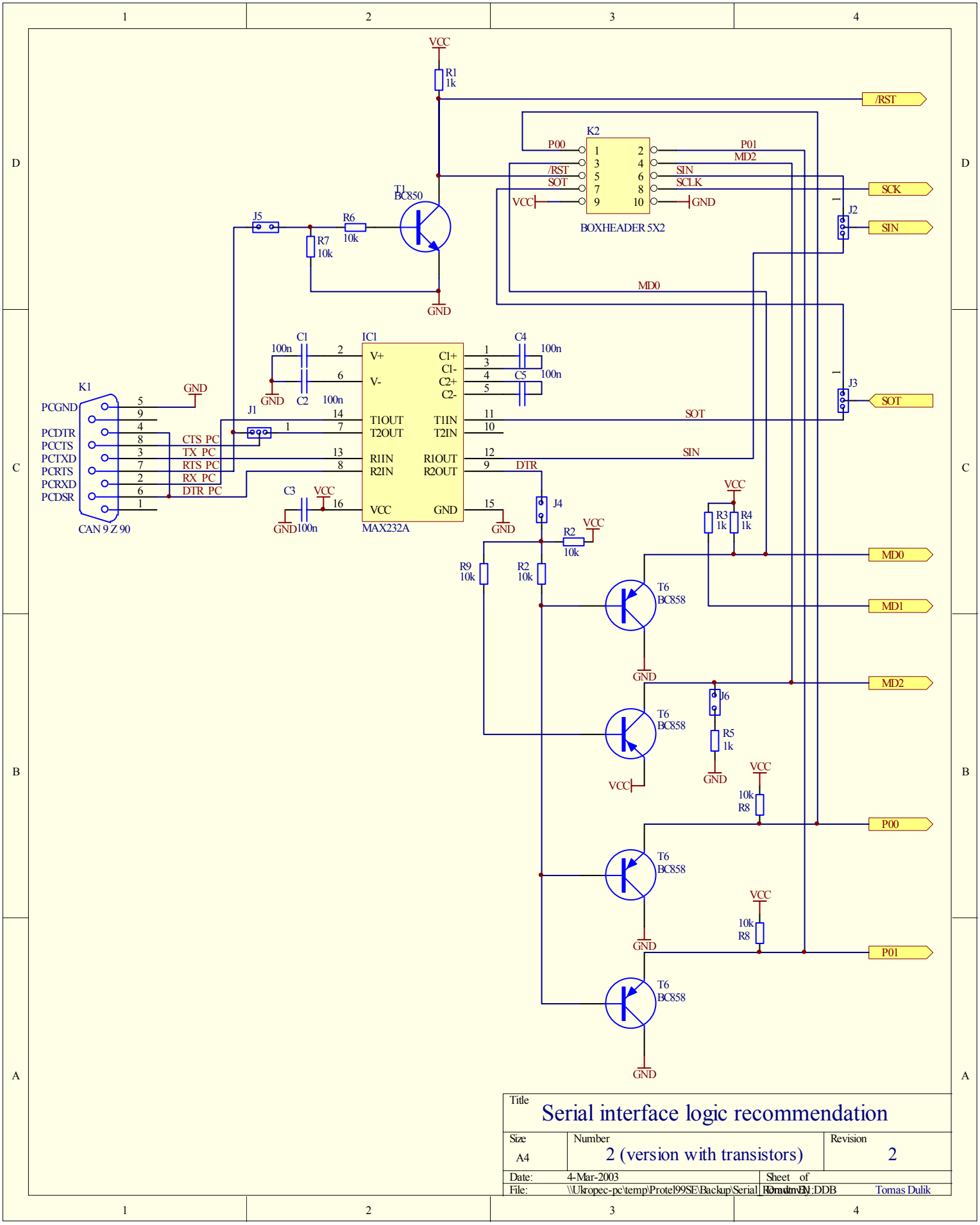
There are two versions of the schematics – the first one uses the gates as decribed above, the second one uses transistors instead of the gates IC2A-IC2D. This can save some space on the target board PCB, especially when the double transistors in a single SMD package are used.

> The jumper J6 must be OFF in the synchronous programming mode for the programmer box to be able to pull the MD2 signal high. For details see the paragraph „General design rules for user target board desing“.

Note: with Flashkit programmer version 2.0 and modified programming cable, the jumper J6 can stay ON.



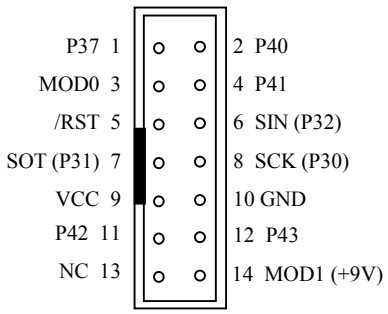
The target serial interface recommendation – version with the gates



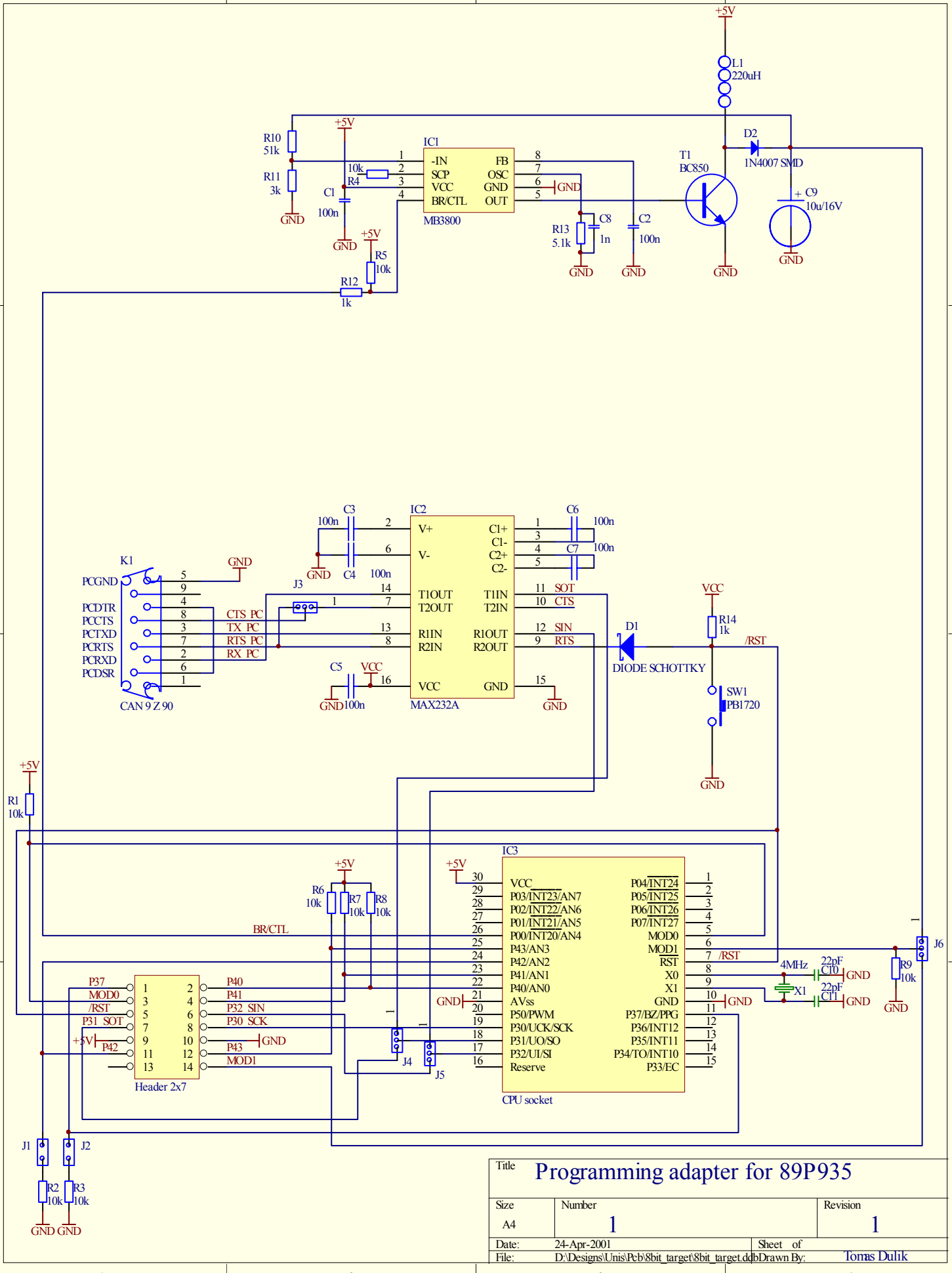
The target serial interface recommendation – version with the transistors

Schematics for MB89P935 programming

Synchronous programming mode
Using the serial programmer box, the only thing that must be added to the user target board is the target programming connector, which is shown on the following picture:



Asynchronous programming mode
If the user needs to program the MB89P935 using the RS-232 interface, some more parts must be added. Since the MB89P935 is an OTP product, it does not make any sense to include the asynchronous programming support on the user target board. For this purpose, it is better to build a programming adapter or to get the Fujitsu programming adapter.
The schematics of such an adapter is shown on the next page.
The IC1 (MB3800) is used for generating the +9V voltage, which is necessary for burning the CPU ROM.
The asynchronous interface logic (IC2) is the same as with the 16LX family.
Jumpers J4, J5, J6 are used for selecting whether the CPU should be programmed from the Flashkit programmer box or from the asynchronous interface.
Jumpers J1, J2 should be ON in the asynchronous programming mode and OFF in the synchronous mode.



Title	Programming adapter for 89P935	
Size: A4	Number: 1	Revision: 1
Date: 24-Apr-2001	Sheet of	
File: D:\Designs\Unis\Pcb\8bit_target\8bit_target.ddb	Drawn By: Tomas Dulik	

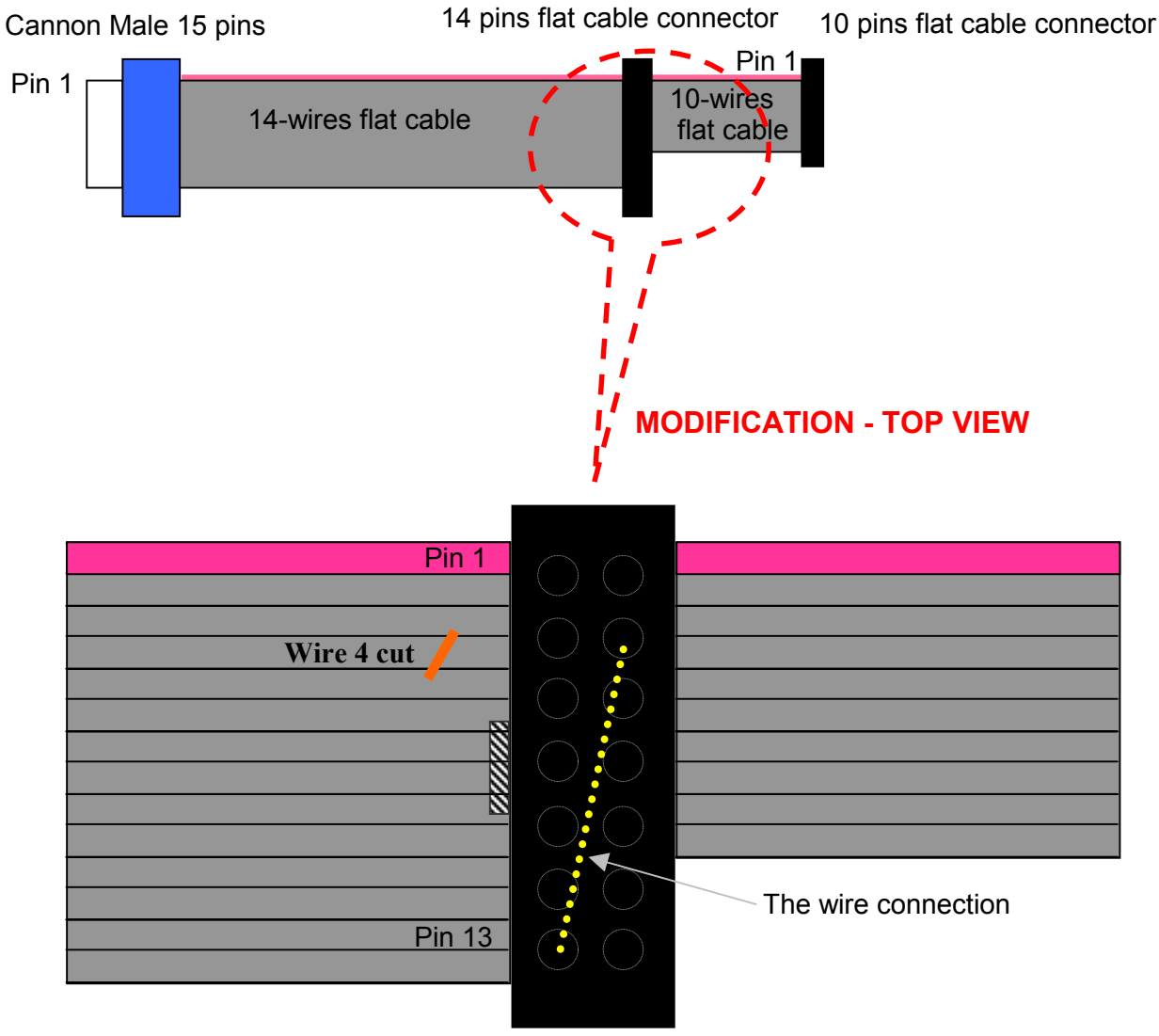
Workaround solution for “Pulldown on MD2” issue

In the Flashkit programmer SW version 2.0, a workaround solution is implemented for the „Pulldown on MD2“ issue.

The "Pulldown on MD2" issue is mentioned in the chapter „General design rules for user target boards“ in this Appendix B. It concerns user target boards (with 16LX family CPU) that were designed with a hardwired pulldown resistor < 50kΩ connected to the MD2 CPU line.

With such user target boards, it was impossible to use the synchronous programming mode with the older version of Flashkit programmer, because the serial programmer box open-collector driver of the MD2 line could not set the voltage of the MD2 line to the "high" level, which is required for setting the CPU to serial programming mode. With the new (2.0) version of the Flashkit serial programmer SW, it is now possible to program also boards with a pulldown resistor hardwired to the MD2 CPU line.

The workaround solution uses a modified programming cable. The cable modification is shown on the following picture:



The cable modification in fact connects the programmer box connector pin PIO2 to the MD2 pin of the target board programming connector. Since the PIO2 pin of the box programming connector is driven by a standard CMOS output driver (it is not an open collector), it can set the level of the MD2 to any value even when a pullup or pulldown resistors are connected to the MD2 pin.

Warning: this workaround solution **requires** to update the programmer box firmware to version 1.7. The new version of the firmware can be found in the file "FIRMWARE_V17.MHX", which resides in the directory, where the Flashkit software was installed. For instructions how to upgrade the programmer box firmware, please see the chapter „Firmware update“ in this manual.