TOPAS900 *Flash*
TOshiba Program development and Application System
Users Manual

HWU Elektronik GmbH
TOSHIBA Electronics Europe GmbH

Preface

Thank you for placing your trust in this HWU Elektronik product.

In choosing this starter kit you have decided for the powerful microcontroller family TLCS-900 from Toshiba. The starter kit TOPAS900 *Flash* will help you to get familiar with this MCU-family and will open up some of the opportunities of TMP95FY64F to you.

About this manual
This manual provides all the information you will need to get the best out of TOPAS900 *Flash*. The accompanying utilities are described, and you will find detailed information about them.

Changes to this manual
HWU Elektronik products are subject to continual further development. It is therefore possible that the printed documentation does not always correspond to the latest release. However, information about the latest changes is always to be found in the README files on the installation CD-ROM. Furthermore, check our Web-Site for latest versions, patches or updated software.

Contents

Figures

Definitions and Abbreviations

Application Board	Separate available board for TLCS-900 with application components as LCD, Keys, EEPROM, LEDs etc.
Chip	Means an integrated circuit, a high-integrated semiconductor
C-Spy	Debugger from IAR Systems designed for Windows™
CE	**E**uropean **C**onformity
CPU	**C**entral **P**rocessing **U**nit
Embedded Workbench	Integrated Development Environment of IAR Systems for Windows™
EMC	**E**lectro**m**agnetic **C**ompatibility
IAR	Short form for "IAR Systems"
MCU	**M**icro **C**ontroller **U**nit
Microcontroller	CPU with On-Chip peripherals for embedded systems
RAM	**R**andom-**A**ccess **M**emory
ROM	**R**ead-**O**nly **M**emory
RTE	**R**eal-**T**ime-**E**mulator : high end system with many real-time debugging functions, e.g. time-measurement, access breakpoints, events, trace buffer etc.
ROM-Monitor Program	Communicates with a debugger (TMPro / C-Spy) and provides debugging capabilities for TLCS-900 based MCU boards. It is a low cost version with a subset of debugging facilities of an RTE
PCB	**P**rinted **C**ircuit **B**oard
TMPro Debugger	**Toshiba**'s debugger for **TLCS-900** series
TOPAS900 *CAN*	Name of a starter kit for **TMP95PS54** MCU
TOPAS900 *Flash II*	Name of a starter kit for **TMP95FY64** MCU
TOPAS900 *Standard*	Name of a starter kit for **TMP93CS41** MCU
Windows™	Windows™ is a registered trademark of Microsoft Corporation. In this manual "Windows" stands for Windows95/98/NT.

1. Introduction

Because TOPAS900 *Flash* is a technical product using high-end electronic components it is worth to read the whole manual to get most out of this product and to avoid possible damage in case of unintended misuse.

The following chapters describe how to set-up the hardware and how to use the software. All the information you need to set-up and work with TOPAS900 *Flash* is provided on the following pages. Please read them attentively.

1.1. What is TOPAS900 Flash?

TOPAS900 *Flash* is a bundled package of hardware and software components to give a quick introduction to the main features of the TMP95FY64F microcontroller. Furthermore there are valuable features for software development and debugging. Sample source code, basic environment set-ups and this manual will minimize the time needed for learning about the key features for the TLCS-900 family.

1.2. Key Features of TOPAS900 Flash Package

- Flash MCU (TMP95FY64) with 256 kb on-chip Flash ROM and 8 kb on-chip RAM
- Additional external 512 kb Flash ROM and external 128 kb RAM on a compact Carrier Board (51x56mm)
- Carrier Board extendable by pin-connectors to a standard PGA-104 socket (4x26 pins)
- Extensive Programming and Debugging Facilities
- In-Circuit Debugging
- In-Circuit Programming
- Windows based Application for Programming and Debugging
- Two independent Environments: Toshiba / IAR
- Limited Versions of original Toshiba Tools like C-Compiler, Assembler, Linker, Converter etc.
- Toshiba's TMPro Debugger with ROM-Monitor
- Demo Version of IRA's Embedded Workbench incl. Compiler, Assembler etc.
- Demo Version of IRA's C-Spy Debugger with ROM-Monitor
- Software samples for both environments

2. Let's start with TOPAS900 *Flash*

Now it is time to start with TOPAS900 *Flash*. The first step is to unpack everything and to connect the power supply and the serial connection to the PC.

2.1. What is in the box?

After opening the box you should check whether all of the components listed below are present:

- One TOPAS900 *Flash* PCB
- Programming and Debugging cable (10 cm, 10-pin connectors)
- Standard serial null-modem connection cable (9-pin D-sub connectors)
- Net-plug for power-supply
- TOPAS900 *Flash* installation CD-ROM
- Quick Start print

2.2. Unpacking

Please unpack everything delivered in the box carefully.
IMPORTANT NOTE: Avoid touching any electronic components due to possible static discharge.

2.3.   A Glance at the TOPAS900 Flash Board

After removing the plastic cover from the PCB you should see the following (for the picture below the cable connection between the Programming-and Debugging Board and the Flash Carrier Board has been removed to avoid covering components, this cable must be plugged in by default):
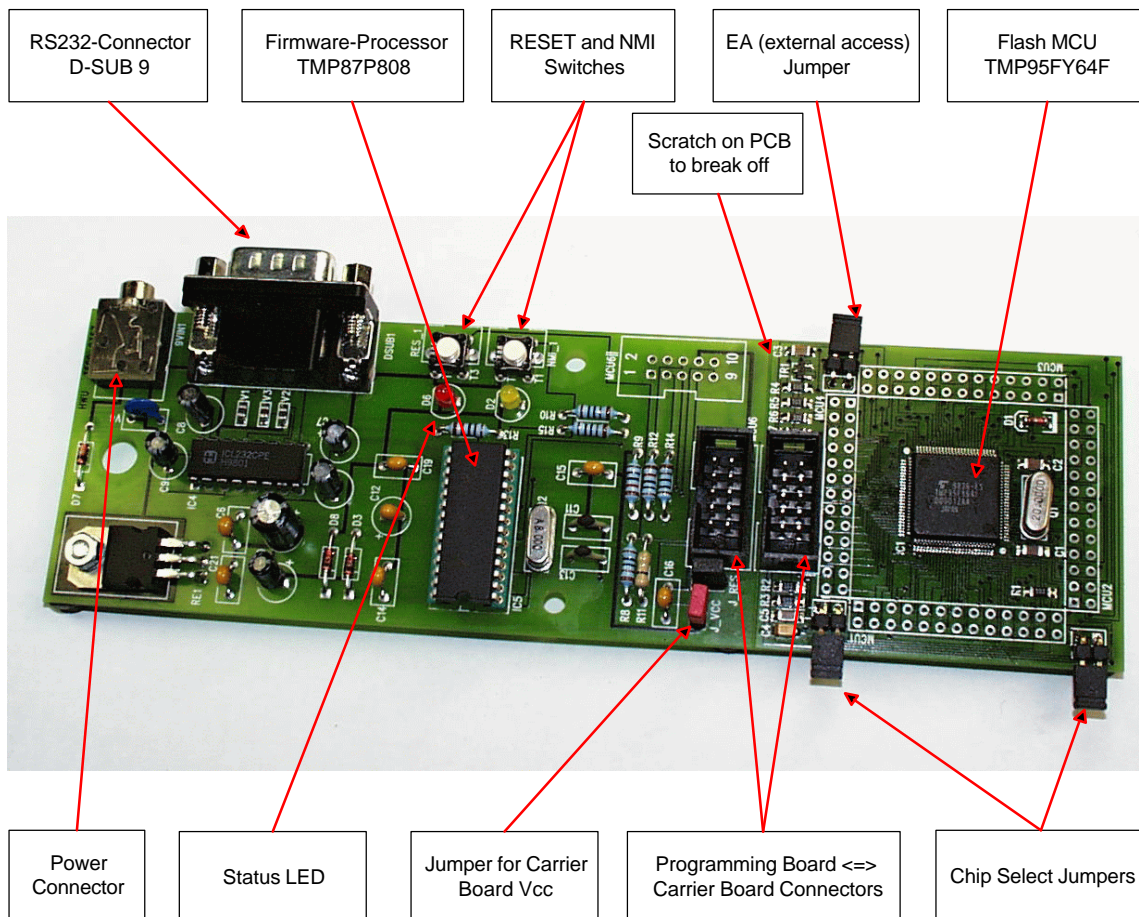
Figure 1 : The TOPAS900 *Flash* Board (Top View)

As shown in the picture above the TOPAS900 Flash PCB can be divided into two functional groups: On the left side the "Programming- and Debugging- Board" is located. On the right there is the "Flash Carrier Board", surrounded by four 26-pin connectors (not soldered). The PCB is delivered in one piece.

The Flash Carrier Board can also be used independently from the Programming and Debugging Board. For independent usage it must be separated from the main board by breaking it off.

To break the Flash Carrier Board off please use this technique: Remove the connection cable between the two parts when plugged in. Put the TOPAS900 *Flash* board on a flat and stable surface. Keep the scratch between "Programming-and-Debugging Part" and

"Flash Carrier Part" exactly onto the edge of the surface (see also fig. 2). Push down both parts carefully until the board breaks into two parts. By breaking into two parts the electrical circuitry is not modified in any way. It is just a mechanical separation.
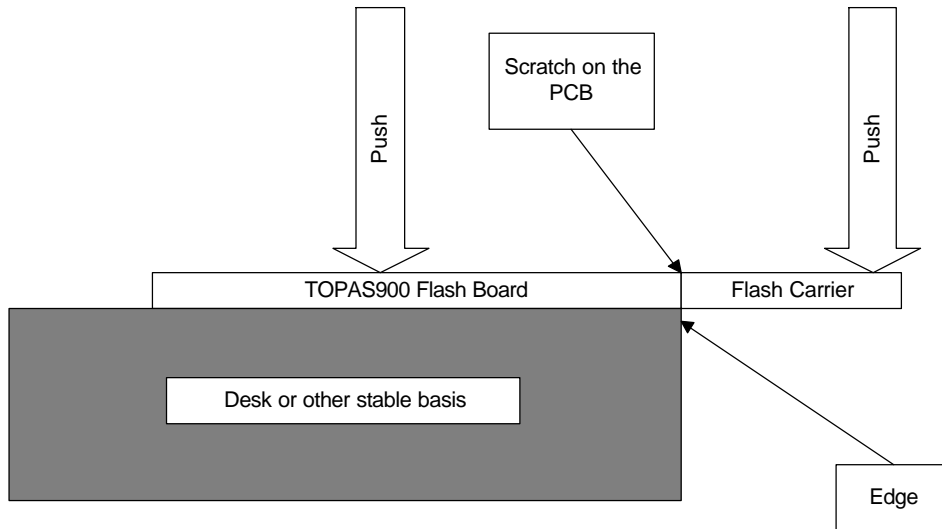
Figure 2 : Breaking off TOPAS900 Flash Board from Flash Carrier Board

2.4. Principe of Connection

The components are to be connected as the scheme below demonstrates:

Figure 3 : Principe of Connection

3.   Hardware Description

TOPAS900 *Flash* consists of two boards: Programming-and-Debugging Board and Flash Carrier Board.

3.1.   Programming and Debugging Board

As the name betrays this board is for programming and debugging purposes. The firmware processor, a TMP87P808P, is listening the serial data communication between a PC and the TOPAS900 *Flash* board. The firmware reacts on special data sequences to switch the target processor to several modes.
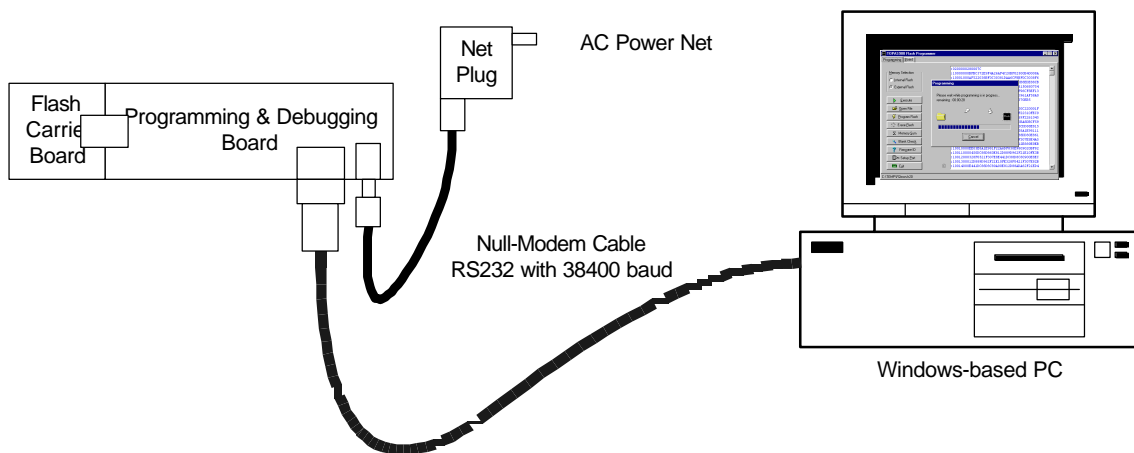
The Programming and Debugging is composed of the following components (to understand the design in detail it is recommended to make a print out of the schematics shown at the end of this manual):

- Firmware Processor
- Power supply
- Status LEDs
- Control Switches (Reset and NMI)
- Serial Communication, Connector and Line Driver
- Connector to Flash Carrier Board

3.1.1. Firmware Processor

A Toshiba TMP87P808 8-bit microcontroller of the TLCS-870 family is used. It is driven by an 8 MHz quartz resonator.

The processor drives control lines of the target system (Flash Carrier Board with TMP95FY64). The lines are /EA, /BOOT and /RESET. With these lines the target processor can de driven into several modes that are selected by the rising edge of the RESET input. Please refer to the manual for further information on **Single Boot Mode** / **Single Chip Mode** and **Multi Chip Mode**. Also read the information given by the memory maps later in this manual.

To control the target system the firmware is listening to the serial communication for certain binary control sequences that are generated by the Windows-based Flash Programming Tool. If a control sequence match is detected, the firmware processor switches the target system into the required mode.

3.1.2. Power Supply

The power supply is mainly build by the net plug and a 5V voltage regulator. Because the board has no on/off switch unplug the power plug from the board when connecting or disconnecting the target or when connecting the Flash Carrier Board to another application. The input power (at the board's power plug) can be up to 12 V DC.

3.1.3. Status LEDs

There are two status LEDs on the board. A red one and a yellow one. The red Led is controlled by the firmware processor and the yellow led is driven by the Flash MCU on the Flash Carrier Board. The yellow Led can be used as a very simple output device to show a internal state. Take a look at the LedDim sample to see how a Led can be dimmed by pulse-width modulation.

3.1.4. Control Switches (Reset and NMI)

On the Programming and Debugging Board two switches (keys) are mounted. The Reset-switch resets the firmware processor. After reset it lets boot the Flash MCU from external flash memory. Normally the external memory contains one of the two ROM monitors (IAR/Toshiba).
The second switch is to give a falling edge to the Flash MCU's NMI (non-maskable interrupt) input. The Toshiba ROM monitor reacts with stopping the user software, if it runs.

3.1.5. Serial Communication, Connector and Line Driver

The serial communication is build by a null-modem cable connected between the PC and the TOPAS900 *Flash* Board. For connection the board has a 9-pin D-sub male connector. To adapt the level between RS-232 and TTL a common RS-232 line driver is used. The serial communication between PC and firmware controller uses the parameter 9600,8,N,1. This baudrate is only used for control commands. The communication speed between Windows-Software and MCU on the Flash Carrier Board is always 38400 baud.
The communication uses the TxD and RxD lines only. These two lines are directly wired with the 10-pin connector to the Flash MCU. The firmware controller is connected in parallel. Both, the firmware processor and the Flash MCU are connected to the RxD and TxD line. To avoid both processors sending data to the same line the firmware processor switches it TxD output to high impedance when it is not used by the firmware itself.

3.1.6. Connector to Flash Carrier Board

To connect the Flash Carrier Board a 10-pin connector is mounted at the edge of the board. The 10-pin cable is to be plugged in on both sides to connect both boards together.

3.2. The Flash Carrier Board

The Flash Carrier Board is composed of the following parts:

- The TMP95FY64F Microcontroller
- 512 kb Flash ROM TC58F400 (90 ns)
- 128 kb RAM TC551001 (70 ns)
- Connector to Programming and Debugging Board
- MCU Terminal Connectors
- Jumpers

3.2.1. The TMP95FY64F Microcontroller

The Flash Carrier Board has the main component of the starter kit soldered on it: the
Toshiba TMP95FY64F microcontroller.

To get an overview of its powerful features lets have a look to its original data sheet
(extract):

...
1. TMP95FY64F Basic Specification
          1.1 Outline and Feature
TMP95FY64 is high-speed advanced 16-bit microcontroller developed for controlling medium to large-
scale equipment. TMP95FY64 has 256K-Byte Flash memory which can be rewritten and erased on board.
TMP95FY64 is housed in QFP-100pin package.

Device characteristics are as follows:
(1) Original High speed 16-bit CPU(900/H CPU)
          TLCS-90/900 instruction mnemonic upward compatible.
          16M-byte linear address space
          General-purpose registers and register bank system
          16-bit multiplication/ division and bit transfer/arithmetic instructions
          Micro DMA :4 channels(640ns/2bytes at 25MHz)
(2) Minimum instruction execution time:160ns at 25MHz
(3) Internal RAM:8Kbyte
          Internal ROM:256Kbyte Flash memory
(4) External memory expansion
          Can be expanded up to 16M byte (for both programs and data)
          AM8/16pin (select the external data bus width)
          Can mix 8- and 16-bit external data buses. .... Dynamic data bus sizing
(5) 8-bit timer:8 channels
          Including event counter function(2 channels)
(6) 16-bit timer/event counter:2 channels
(7) Serial interface:3 channels
(8) 10-bit A/D converter:8 channels
(9) 8-bit D/A converter:2 channels
(10) Watchdog timer
(11) Chip select/wait controller:4 blocks
(12) Interrupt functions:45-Interrupt sources
          9-CPU interrupts ..... SWI instruction, and Illegal instruction
          26-Internal interrupts .....7-level priority can be set.
          10-External interrupts .....7-level priority can be set.
(13) I/O ports : Single chip mode 81 pins
          Multi chip mode 55 pins(at AM8/16="H")
(14) Standby function:4 HALT mode(RUN,IDLE2,IDLE1,STOP)
(15) Operating Voltage : Vcc = 4.5 to 5.5V
(16) Package:100pin QFP(LFFP100-P-1414-0.50C:Thickness 2.4mm)
...

To get an overview on the internal organization of the TMP95FY64 let's have a look to its
schematic block diagram:
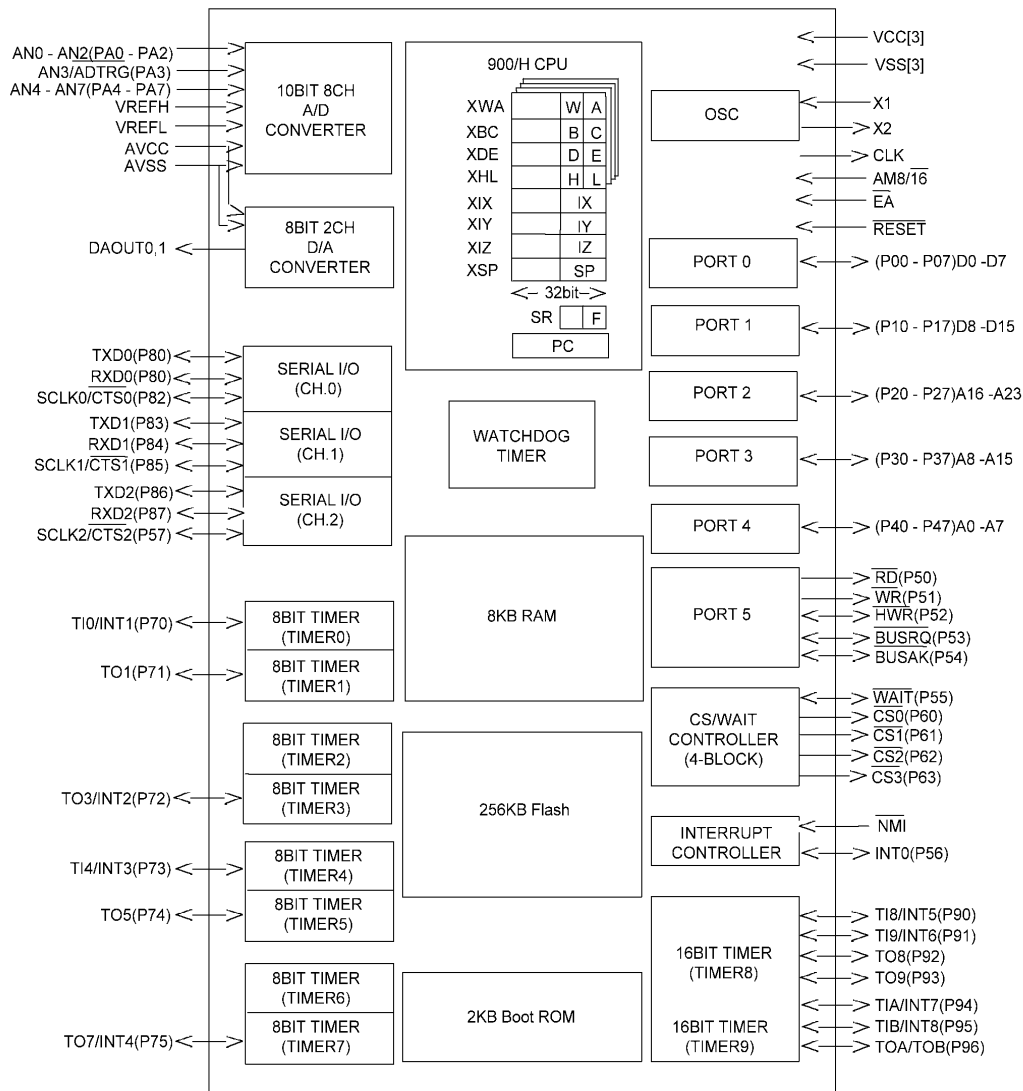
Figure 4 : TMP95FY64 Schematic Block Diagram

3.2.2. 512 kb Flash ROM - Toshiba TC58F400 (90 ns)

This memory component is soldered on the bottom side of the PCB. It has a size of 512 kb. The chip can contain any software and data that fit into 512 kb. On delivery it contains the IAR ROM monitor.

The device can be addressed in 8 bit (byte) or 16 bit (word) organization. In case of the present starter kit it is wired by an 8-bit data bus. The disadvantage of lower operating speed is equalized by the gain of 8 port pins that are free for application purposes. The external flash ROM is addressed by 19 address lines A0..18. The data lines are D0..D7 and the chip is selected by /CS2 pin of the MCU. The type of operation is selected by the /WR and /RD line accordingly. The /BYTE input is fixed to low level so that the byte mode is activated.

Please refer to the memory maps to get information about the address range of the chip in memory.

3.2.3. 128 kb RAM - Toshiba TC551001 (70 ns)

This memory component is soldered on the bottom side of the PCB. It has a size of 128 kb and is organized in 8-bit words. It is addressed by 17 address lines A0..16. The data lines are D0..D7 and the chip is selected by /CS0 pin of the MCU. The type of operation is selected by the /WR and /RD line accordingly.

Please refer to the memory maps to get information about the address range of the chip in memory. The external RAM is used additionally to the internal RAM of 8 kb of the MCU. When the Toshiba ROM monitor is running almost 32 kb are allocated to the ROM monitor. For all monitor operations the RAM is mapped to address 0xFE0000..0xFFFFFF and covers the interrupt vector area that must be writable for debugging purposes.

3.2.4. Connector to Programming and Debugging Board

To connect the Programming and Debugging Board a 10-pin connector is mounted at the edge of the board. The 10-pin cable is to be plugged in on both sides to connect both boards together.

3.2.5. MCU Terminal Connectors

The Flash Carrier Board can be used - when properly programmed - stand-alone in a user's application. For this kind of usage it is necessary to insert and solder four 26-pin connectors into the prepared drill holes. The prepared space for the connectors can easily found by searching the Flash Carrier Board for the printed labels "MCU1", "MCU2", "MCU3" and "MCU4". The connections to the MCU are made as easy as possible. Each of the 100 pins of the MCU are connected directly to MCU connectors. E.g. pin 11 of the MCU is wired with pin11 of "MCU1" connector. Pin 26 of each connector is left out and not connected (the MCU has 100 pins, but 4x26=104). It is recommended to do direct measurement at the terminal connectors (or their metalized drill holes) instead of contacting the very small and sensitive MCU pins.

3.2.6. Jumpers

J_CS0 : This jumper is to connect/disconnect the chip select line CS0 to the on-board RAM device. If an external data/address bus is not required, it is recommended to open this jumper. The CS0 pin can be used as a port pin additionally.

J_CS2 : Same as above. The CS2 pin is used to enable the on-board flash ROM device.

J_EA : Jumper to select between internal (high/open) and external flash memory access (low/closed) after reset. When the Flash Carrier Board is connected to the Programming

and Debugging Board, the jumper can be left open – the line is controlled by the firmware controller.

If the Flash Carrier Board works stand-alone (without Programming-and-Debugging Board), the jumper must be set when external flash software should be executed. If the jumper is left open, the /EA pin is pulled up by a 10k pull-up resistor and the Flash MCU boots from internal flash memory. When the MCU is reset with /EA at high level, the external memory can be accessed additionally. The entire ROM size would be 768 kb and 128 kb of RAM. When the MCU is reset with /EA at low level, the internal flash ROM cannot be accessed additionally.

4. Software Description

The development software of TOPAS900 *Flash* can be chosen from two alternatives

➢ the Toshiba and

➢ the IAR Systems Environment.

Both are powerful tool sets for program development and debugging. The user can decide for either of them but should take into account that for debugging programs on TOPAS900 *Flash* board each environment has its own ROM-Monitor for supervising user programs. **By default the IAR C-Spy ROM-Monitor is installed** and the on-board flash memory has to be reprogrammed by Toshiba TMPro Monitor before debugging.

Furthermore the **IAR environment contains** an outstanding, window-oriented **processor simulator** which can be used for off-line debugging before switching to on-line debugging on the board.

Both environments are described in the following.

4.1. IAR Tools

4.1.1. Embedded Workbench

When having decided for IAR environment user program development can be done by the embedded workbench. Choose "**IAR Embedded Workbench**" program group and click the workbench icon. The upcoming initial desktop should be used to load a project. The workbench is directed by project files which comprise all necessary setting for a user project. To get started, first of all open the demo project by loading the "**DEMO.PRJ**" file through the file menu from "**IAR\EW22DEMO\T900**". From this project all new projects can be derived. The previous desktop is always regenerated. The way to change a project is explained in the respective help. The Demo project opens the directory "**DemE900H**" with a demonstration example. Before the demo is recompiled do the following changes to the project: select **Project** from the main menu and select **Options**. A dialog will pop up. Select the category **C-Spy**. Then choose **Serial Communication** on the right side. Select the COM port that is used for debugging with TOPAS900 *Flash* kit. The baudrate must be set to **38400 baud**. After confirmation by clicking OK press the

debugger button ![icon] of the tool bar on the top of the window. The project will be recompiled and the C-Spy will be launched. The connection to the TOPAS Board will be made and the debugger is ready to work with. The demonstration shows how to use terminal functions. To view the output of the demo open the **Terminal I/O** window.

After having loaded the Demo project the desktop looks like shown in fig. 4.
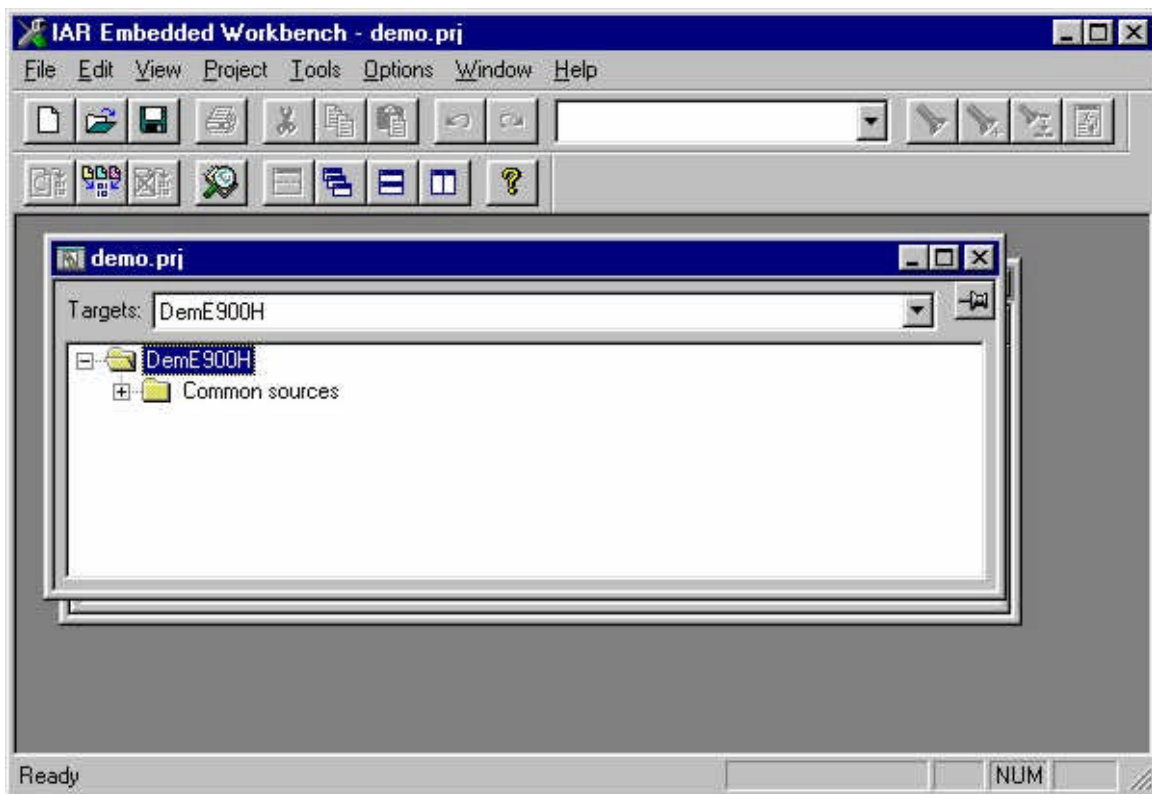
Figure 5 : IAR Embedded Workbench Desktop with Project Window

Editing Sources

The workbench contains a powerful source-code editor. The editor is tailored for C code. The C syntax is highlighted by colors and intensity and on request bookmarks and/or line number can be displayed. Functions like cut & paste, find & replace are included as well as bracket matching on request.

Compiling and Linking

By selecting "**Project**" from the menu bar, compiler and linker are invoked simply by clicking the respective command in the menu like "**Compile**", "**Make**", "**Link**" and "**Build All**". The listfiles are written to the "**...\List**" directory (default) selected by the project file. All directory settings are made by choosing the "**Options...**" command.

4.1.2. C-Spy Debugger/Simulator

There are two ways to invoke the C-Spy debugger.

- First by selecting the icon directly from program group "**IAR Embedded Workbench**" or

- from the Embedded Workbench desktop.

In case of a direct start C-Spy will open a desktop with "**File**"-"**Open**" enabled. An executable file can be browsed then.

When starting C-Spy from the Embedded Workbench the executable file is defined by the project file. Hence, no definite settings have to be done in addition. Clicking the magnifying glass icon in the project bar starts the C-Spy Debugger

The C-Spy Debugger is a powerful tool for program debugging because it comprises a platform for real-time debugging in co-operation with the Monitor on TOPAS900 *Flash* board as well as an off-line, window supported controller simulator.

Consequently the user has to select between both:

- In case of a direct start of C-Spy and trying to load an executable, a window named "**Session Options**" comes up in which "**ST900**" or "**RT900**" can be selected. Choose ST900 for invoking the simulator or RT900 to select the download to Monitor option.

- In case of using the Embedded Workbench, before clicking the magnifying glass icon, in the "**Project**" menu the "**Options...**" have to be selected to adjust the debugger. Choose C-Spy from the "**Category**" and "**Simulator**" or "**ROM-Monitor**" to preset C-Spy. Then click the magnifying glass to invoke C-Spy.

In both modes the following C-Spy desktop is displayed. When coming up only the report window is displayed. All other windows can be opened on request and tiled like shown (Toolbar, <u>W</u>indow).

4.1.3. C-Spy ROM-Monitor

The C-Spy ROM-Monitor is installed by default on TOPAS900 *Flash* board. For re-programming the on-board (external) flash memory with this monitor please refer to section 5.2. The yellow led does **not flash** in opposite to the Toshiba ROM monitor when the ROM monitor is executing. The speed of serial connection is **38400** baud This is very important to know when recompiling projects for the debugging environment. The Intel-hex file containing this monitor is **IAR.h20**.

4.1.4. IAR Tools Limitations

- The Demo version of the ICCT900 compiler does not support the -A and -a options.
- The Demo version of the ICCT900 compiler has a code size limit of 4K compiled code.
- The Demo version of the AT900 assembler has a code size limit of 4K assembled code.
- The Demo version of the EWT900 workbench has a code size limit of 4K linked code.
- The Demo version of the EWT900 workbench does not include the command line versions of the assembler and compiler.
- Use the supplied project file: <installation root>\t900\demo.prj to generate demo files for C-SPY simulator and ROM-monitor.

- The project file assumes that the workbench has the <installation root> directory:
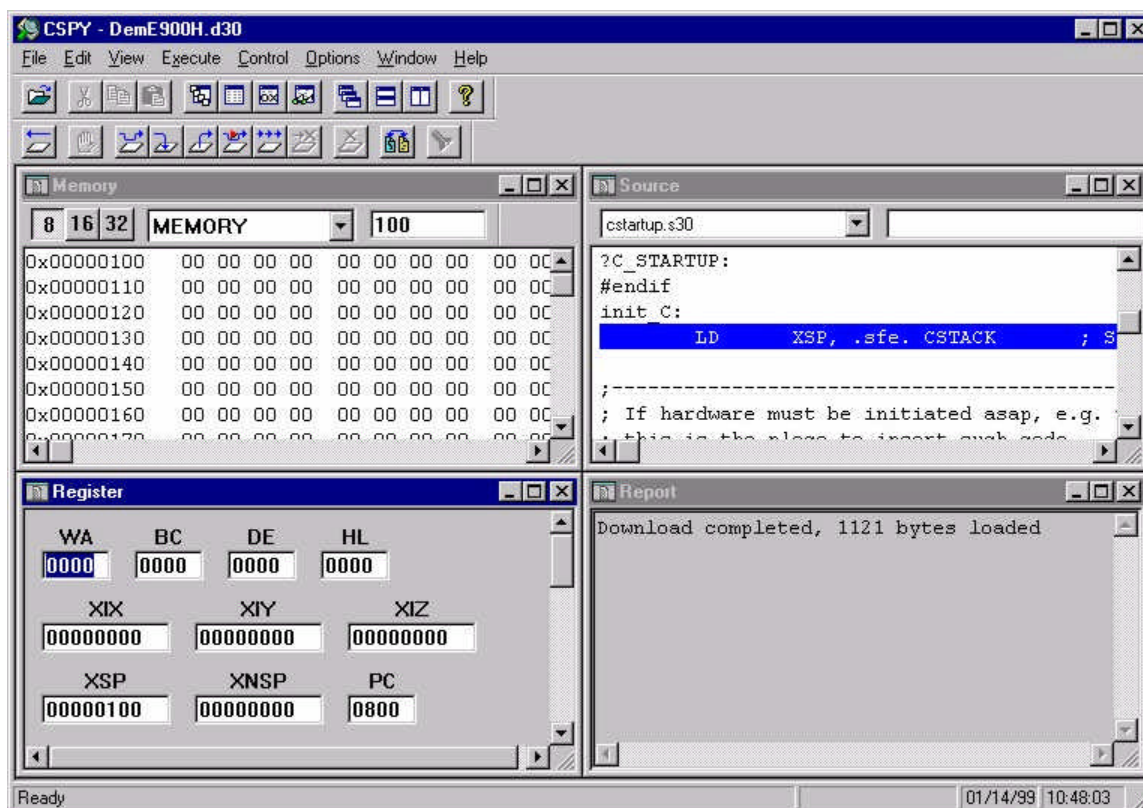 C:\IAR\EW22DEMO

Figure 6 : C-Spy Desktop with Source Code, Memory and Register Window

For all further handling of the C-Spy Simulator or ROM-Monitor please click the "**Help**" in the Toolbar and select the topic of interest.

4.2.   Toshiba Tools

4.2.1. Compiler, Assembler, Linker, Converter

All Toshiba software development tools are running in DOS environment. Hence, under WINDOWS a DOS box has to be opened by selecting the MS-DOS icon or clicking a tailored batch file from the Windows-Explorer. All respective manuals can be accessed by the program group "**TLCS-900 Tools**".

Editing sources
The source code of user C- or TLCS-900 Assembler programs should be plain ASCII text edited by regular text editors like Windows-Editor, WORDPAD or others. The extensions of the filenames should be "**.C**" for C-language programs and "**.ASM**" for assembler programs.

Compile and Link
Preparing programs for execution, i.e. compile and link, can be done by a single DOS
command line invoking a so called driver: for C-programs "**cc900**" driver and for
Assembler programs "**mac900**" driver.
For further information on drivers please refer to the "language tool operation manual".

Please notice that the drivers additionally invoke the linker "**tulink**". For linking a linkage
command file is needed with the extension is needed. The usual filename extension is
"**.LNK**" or "**.LCF**". This file defines the desired mapping of variables and of program
code to memory addresses. Examples of linkage command files to fix user programs into
TOPAS900 memory please refer to the examples in "**\SAMPLE\TEST900**" directory. In
all subdirectories are files named "**MAKE.BAT**". Executing these files will compile and
link the sample projects. Detailed information can be found in the respective manual. The
user always should add the assembler file "**STARTUP.ASM**" to the link command
which embeds user programs into some initializing code and should use the linkage
command file "**SAMPLE.LNK**" which sets the correct target address space. The
STARTUP.ASM program also includes the "**IO900.H**" file in which all implemented
input/output addresses are defined by variables and can be applied by user programs.

Program preparation for debugging
To prepare programs for source level debugging the driver has to be invoked with "-g"
switch.

Program format conversion
Executable programs (extension "**.ABS**") can be converted to Intel-Hex or Motorola-S
format by invoking the format converter "**tuconv**".

Converting for Internal and External Memory
To program software to internal or external flash memory Intel-hex files must be created
by using the "**TUCONV.EXE**" tool. The execution address range is the same
(0xFC000..0xFFFF00) but the programming address space is different. In programming
mode both, the internal and the external flash memory can be programmed, but in
different address ranges.

Sample command lines:

Converting for **external** memory:
-l -Fh20 -ra 0xf80000,0x80000,0x80000,,

Converting for **internal** memory:
-l -Fh20 -ra 0xfc0000,0x040000,0x010000,,

Examples
For examples of program development please refer to subdirectories of
\SAMPLE\TEST900
where some example programs and respective compile and link driver command lines can
be found.

4.2.2. TMPro Debugger

The TMPro Debugger is invoked by selecting the "**Toshiba Debugger**" program group and clicking the TMPro Debugger icon. Program debugging is done by downloading user programs to external RAM on TOPAS900 *Flash* board and running them supervised by a Monitor program in the external flash memory on the board.
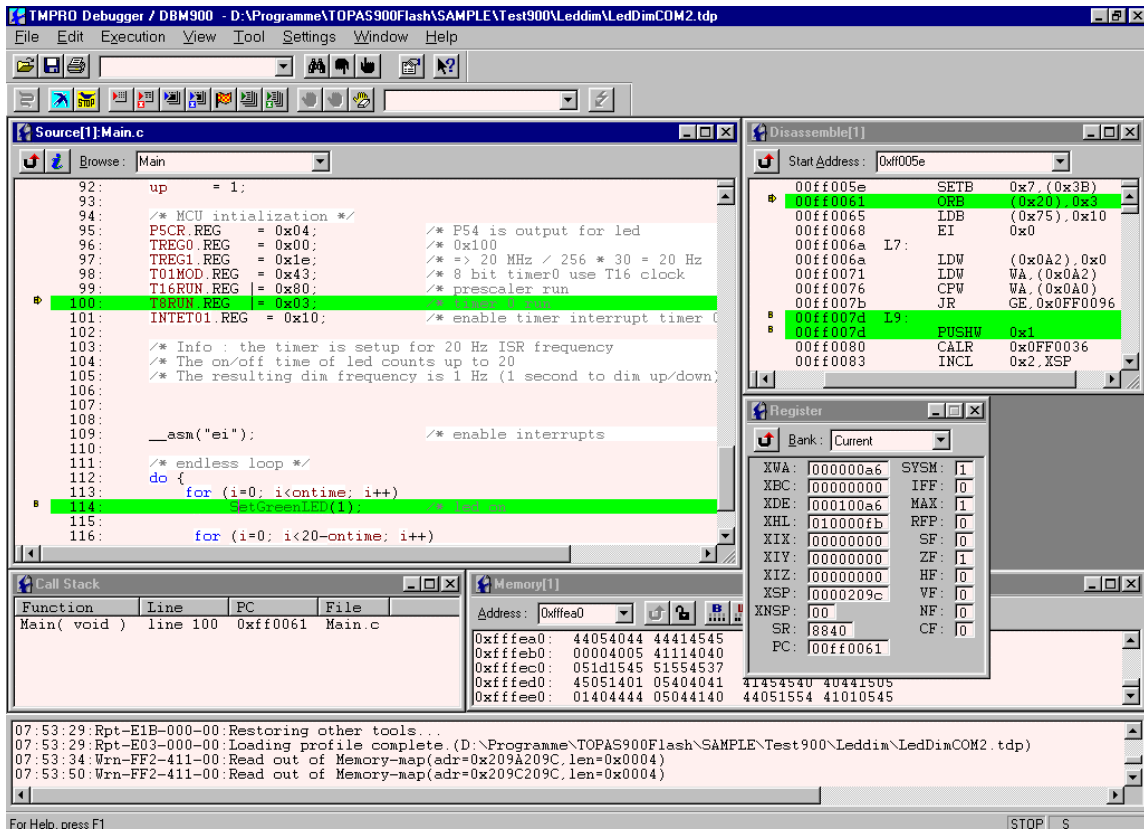

Figure 7 : TMPro Window

When using the Toshiba environment there has to be a connection between the TMPro Debugger on the PC and the respective Monitor (..\monitor\toshiba.h20) being started on TOPAS900 *Flash* board. Hence, in beforehand the TOPAS900 board has to be connected and active which is notified by the quickly flashing yellow LED. Invoking the TMPro Debugger sets up the serial transfer line. **Important** to know when creating new debugging profiles is the fixed communication speed of **38400 baud**. The success in setting this connection can be watched by LED stopping to flash a short while and subsequently the Monitor will change the ratio of flashing to ca. 1:5. Hence, both the Debugger and the Monitor are ready to work together.

Executable user programs are those with extension "**.ABS**". Some small experimental examples can be found in the directory "**\SAMPLE\TEST900**". Select menu "**FILE**

LOAD" to load a program. The TMPro Debugger displays the source code in the source window and automatically downloads the user program to the TOPAS900 *Flash* board.

After downloading a user program to the TOPAS900 *Flash* board RAM it can be started (clicking go button) or traced (clicking one of the step buttons) by TMPro debugger. If a user program is running the red LED on the TOPAS900 *Flash* board is switched on permanently. Running programs can be stopped by TMPro debugger (click stop or finish button), by a breakpoint being defined before (break set at a specific source code line) or by actuating NMI switch on the TOPAS900 *Flash* board. The next statement of the user program to execute is marked in the source code window.
For further details please refer to the help manual in the "**Toshiba Debugger**" program group.

After starting the TMPro Debugger, a profile file which comprises a lot of settings for a user program should be loaded or – if not existing – created.
If TMPro is called the first time after installing the creation of a new debugging profile is necessary. From this profile further individual profiles can be derived. Before starting the TMPro debugger be sure that the Toshiba ROM monitor is installed properly on the TOPAS900 *Flash* board (external flash ROM) and is started (by pressing the reset button). The yellow led must be flashing as an indicator for "Toshiba ROM monitor running".

4.2.2.1. How to create a new debugging profile (*.tdp)

The following steps show how to setup a new profile. Just follow the steps below, which give a short description. Please refer to the TMPro help file for further information (topic: **Starting the Profile Wizard and Wizard Processing**).

- Launch the TMPro Debugger by clicking its icon in the start menu.

- Select **New Profile** from the **File** menu

- A dialog will come up and requires information about the connection. Select the COM-port that is used for the serial connection to the TOPAS900 *Flash* Board. The DTE-Speed must be set to 38400 baud as shown on the left side. Confirm this dialog for the next step. After confirmation the serial connection to the TOPAS900 *Flash* board's ROM monitor is established.
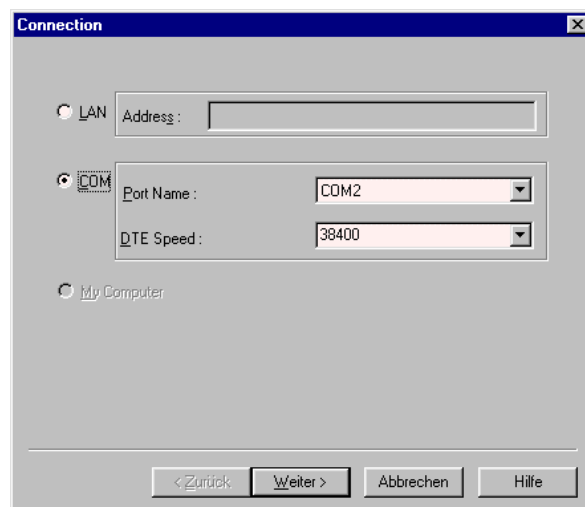
- The fields of this dialog are concerned with the Real-Time-Emulation System (RTE) from TOSHIBA. No changes can be mage here, so just confirm this dialog for the next step.
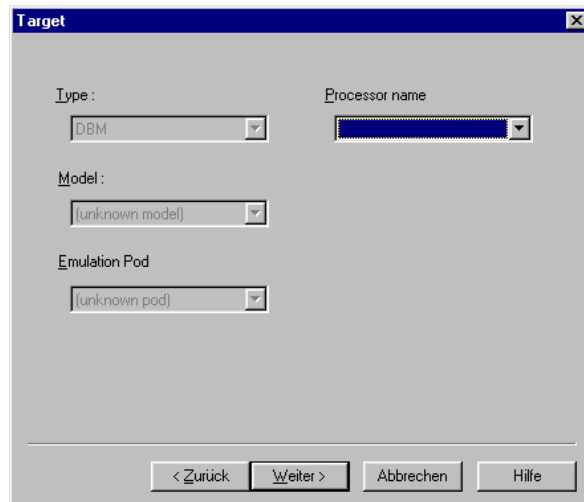
- The fields of this dialog show the memory mapping of the TOPAS900 *Flash* Board. Because of limitations of the ROM-Monitor no changes can be mage here, so just confirm this dialog for the next step.
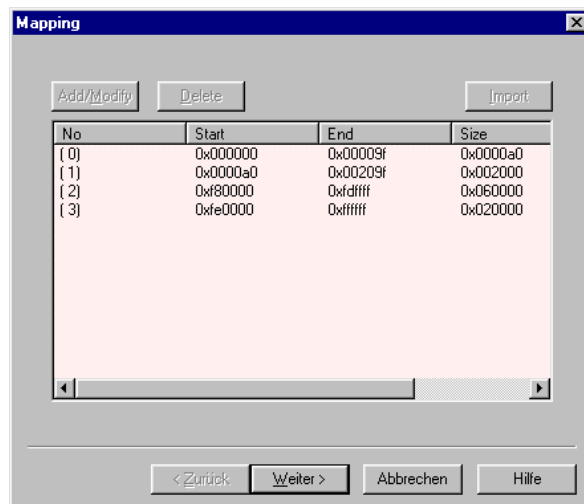
- **Load Object File**: enter the object file of the project that should be load by TMPro Debugger. Click the **Browse** button to select the ABS-file of your project. **Source Search Path**: the path information of the source files of a project are stored in the abs-file absolutely. The source search path gives an alternative path to load the source files, if the files given by the abs-file are not valid. **Work Space** : Specifies the directory
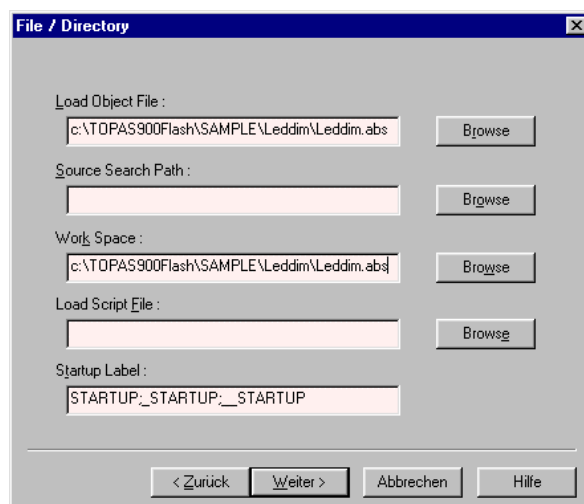
where the Debugger places temporary files. Confirm this dialog for the next step. Script File and Startup Label can be left unchanged. Confirm the dialog to get to the next dialog.

- This dialog has no effect for the ROM monitor operation and should be confirmed unchanged.
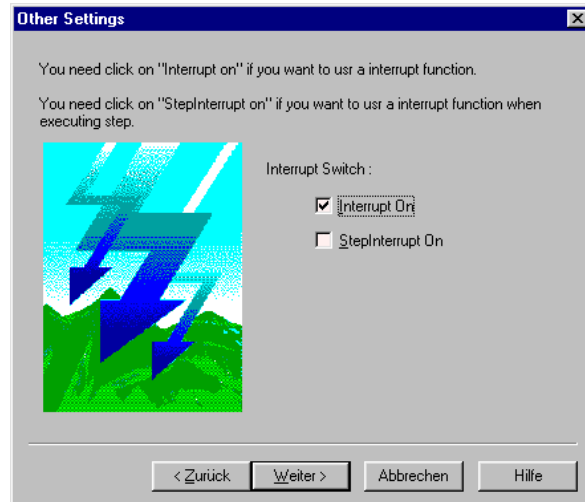
- This dialog asks for a filename for the new profile that has been created in the steps before. The default name is **..\profile1.tdp**. Choose a meaningful name for the project like **leddim.abs**. After confirmation of this last step the profile is created and stored to disk. The abs-file will be loaded. Check the information given at the bottom of the TMPro window about the loading of debug information or warnings or error messages. When the profile is double-clicked in the Windows-Explorer the TMPro Debugger is started automatically with all the setting contained in the profile.
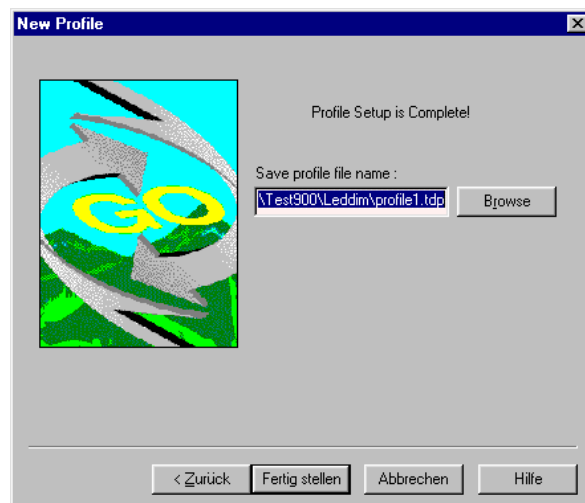
4.2.3. TMPro ROM-Monitor

Using the Toshiba environment makes need of changing the ROM-Monitor in TOPAS900 *Flash* board because the IAR C-SPY Monitor is installed by default. For programming the on-board (external) flash memory with the TMPro Monitor please refer to section 5.2. The file that is to be programmed into external flash ROM is named **Toshiba.h20**.The speed of serial RS-232 link to the PC is always **38400** baud. After starting the monitor (by pressing the reset button) the **yellow led flashes** to show that the monitor is running.

4.2.4. Toshiba Tool Limitations

The CC900 comes in a limited version with TOPAS900 *Flash*.
These restrictions are to be noticed:

- The C-compiler has a limit of 2000 lines per file
- The delivered tools are not supported and no subject of further development or maintenance.

4.3. ROM-Monitor Memory Usage

4.3.1. IAR ROM-Monitor

Memory Usage of ROM-Monitors

Enviroment :   IAR
Target: TOPAS900 Flash II

000000H	Internal I/O (160 bytes)	Not allocated
0000A0H	Internal RAM (8K bytes)	Free
		ROM Monitor — 000100H ... 0007FFH
		Free
0020A0H	external memory	Free
	external memory	
F80000H	External Flash Memory (512 K bytes)	ROM Monitor — F8681CH
		Free
FE0000H	External RAM (128 K bytes)	ROM Monitor — FEFFFFH
		Free
FFFF00H	Vector table (256 bytes)	ROM-Monitor uses: RESET vector (FFFF00) RxD2 vector (FFFF94)
FFFFFFH		

Figure 8 : Memory Usage of IAR ROM-Monitor

4.3.2. Toshiba ROM-Monitor

Memory Usage of ROM-Monitors

Enviroment :   Toshiba
Target: TOPAS900 Flash II

000000H	Internal I/O (160 bytes)	**Not allocated**
0000A0H	Internal RAM (8K bytes)	**Free**
0020A0H	external memory	**Free**
	external memory	
F80000H	External Flash Memory (512 K bytes)	**ROM Monitor**
		Free
FE0000H	External RAM (128 K bytes)	**ROM Monitor**
		Free
FFFF00H	Vector table (256 bytes)	**ROM-Monitor uses:** **RESET vector (FFFF00)** **RxD2 vector (FFFF94)**
FFFFFFH		

F8C961H

FE2A9BH

Figure 9 : Memory Usage of Toshiba ROM-Monitor

4.4. Restrictions of ROM-Monitor Usage

The table below lists the restrictions that are caused by the usage of the ROM-monitor software.

Resource Type	IAR C-Spy ROM-Monitor	Toshiba TMPro ROM-Monitor
ROM Address Space	0xF80000..0xF868FF	0xF80000..0xF8CAFF
RAM Address Space	0x000100..0x0006FF, 0x000700..0x0007FF, 0xFE0000..0xFE01F1, 0xFEFD00..0xFEFFFF	0xFE0000..0xFEFFFF
Interrupts	RESET, SWI7, RxD2	RESET, NMI, SWI7, RxD2
MCU Pins	D0..D7, A0..A18, /RD, /WR, /CS0, /CS2, TxD2, RxD2, /BOOT	D0..D7, A0..A18, /RD, /WR, /CS0, /CS2, TxD2, RxD2, /BOOT
Other	DI forbidden, clock speed fixed @20 MHz	DI forbidden, clock speed fixed @20 MHz

5. Functional Description

5.1. Operating Modes

Besides I/O page, 8 Kbytes internal RAM and the interrupt vector area the TMP95FY64 controller has 256 Kbytes of internal flash memory. TOPAS900 *Flash* board additionally is equipped with 512 Kbytes external flash memory (double size of internal flash) and 128 Kbytes external RAM (half size of internal flash). The flash memories and the external RAM are differently mapped in three different modes of operation.

5.1.1. MCU Internal Memory Map

The memory map of the MCU itself is as follows:
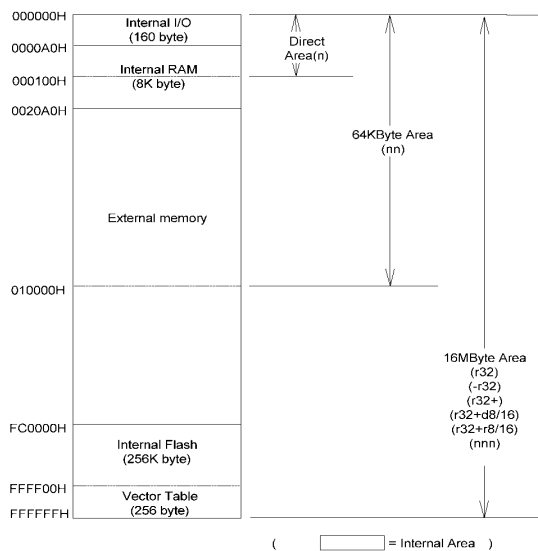

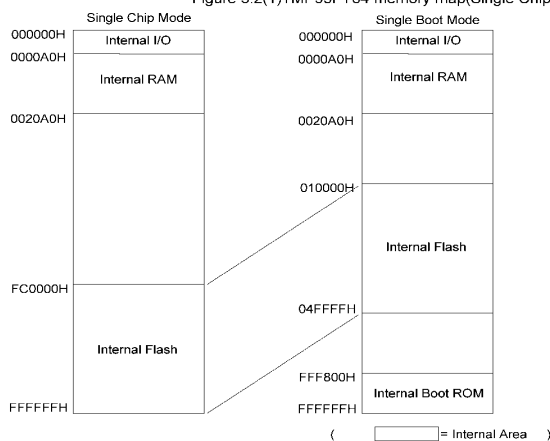
Figure 3.2(1)TMP95FY64 memory map(Single Chip Mode)

Figure 10 : The MCU's Internal Memory Map

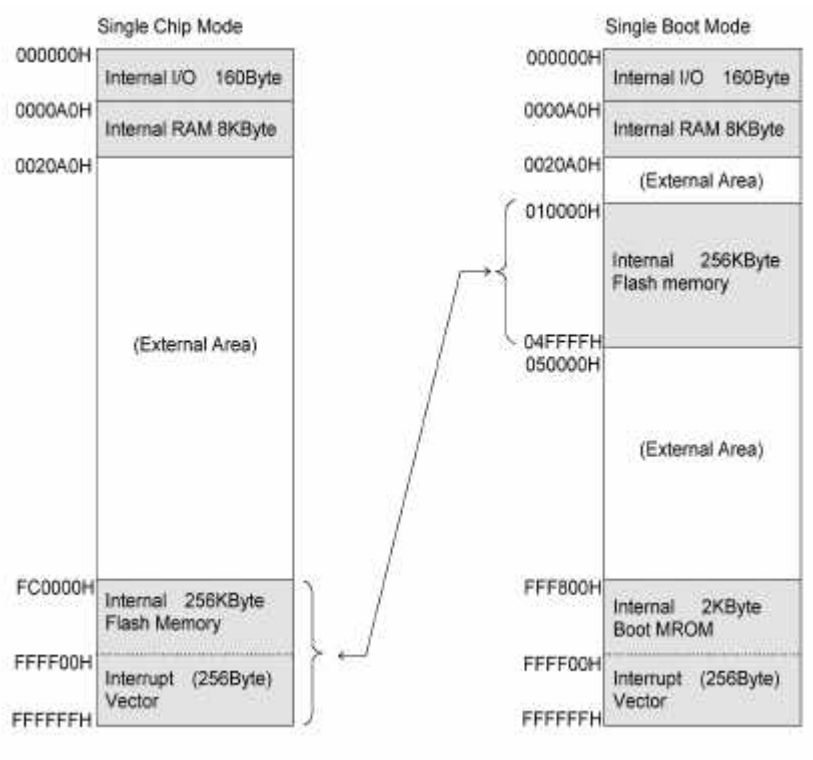### 5.1.2. Internal Mapping in different Modes

Figure 11 : Internal Mapping in Single Boot and Single Chop Mode

5.1.3. Programming in Single Boot Mode

The TMP95FY64 flash controller comprises 256 Kbytes of internal (on-chip) flash memory. For first-time programming of this memory a small portion of internal program located in an internal Boot-ROM area has to be activated. This is done by resetting the controller and restarting it by holding the /BOOT line initially to Low. The controller then runs in SINGLE BOOT mode (see also controller manual) executing a program from Boot-ROM for programming the internal flash from a serial line. For this programming action the internal flash is switched to addresses from off 10000h (see fig. 5/6).

Handling of the /BOOT line and programming from PC is managed by a small additionally implemented 8-bit controller TMP87P808.
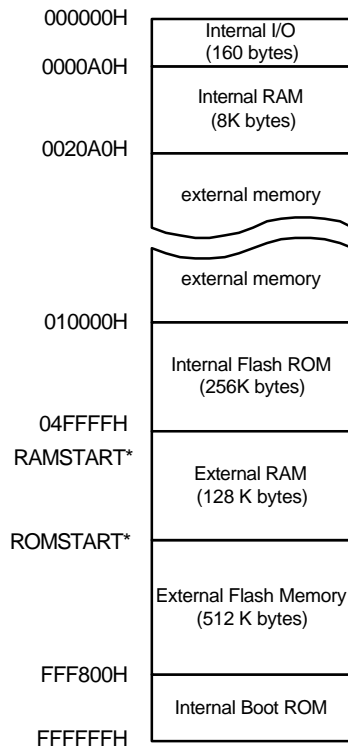
MEMORY MAP
Boot Memory : Boot ROM
Reset-Conditions :
/EA = H
/BOOT = L (single boot mode)

000000H	Internal I/O (160 bytes)
0000A0H	Internal RAM (8K bytes)
0020A0H	external memory
	external memory
010000H	Internal Flash ROM (256K bytes)
04FFFFH RAMSTART*	External RAM (128 K bytes)
ROMSTART*	External Flash Memory (512 K bytes)
FFF800H	Internal Boot ROM
FFFFFFH	

* ROMSTART is programmed by CS2 registers
* RAMSTART is programmed by CS1 registers

Figure 12 : Map for Internal Flash Programming

5.1.4. Normal Operation Mode

For normal operation the /BOOT line is switched to High which disables the Boot-ROM. In this mode there are two possibilities for mapping the internal flash memory determined by the /EA (external address) line. The TOPAS900 *Flash* board has external flash memory for the ROM-Monitors and/or user programs and external RAM mainly for program debugging. The /EA line enables/disables the internal flash and performs the mappings shown in fig.7 and fig. 8.

MEMORY MAP
Boot Memory : internal ROM
Reset-Conditions :
/EA = H
/BOOT = H

000000H	Internal I/O (160 bytes)
0000A0H	Internal RAM (8K bytes)
0020A0H	external memory
	external memory
ROMSTART*	External Flash Memory (512 K bytes)	ext. ROM
RAMSTART*	External RAM (128 K bytes)
FC0000H	Internal Flash ROM (256K bytes)	int. ROM
FFFF00H	Vector table (256 bytes)
FFFFFFH	

MEMORY MAP
Boot Memory : external ROM
Reset-Conditions :
/EA = L
/BOOT = H

000000H	Internal I/O (160 bytes)
0000A0H	Internal RAM (8K bytes)
0020A0H	external memory
	external memory
RAMSTART*	External RAM (128 K bytes)
F80000H	External Flash Memory (512 K bytes)	ext. ROM
FFFF00H	Vector table (256 bytes)
FFFFFFH	

* ROMSTART is programmed by CS2 registers
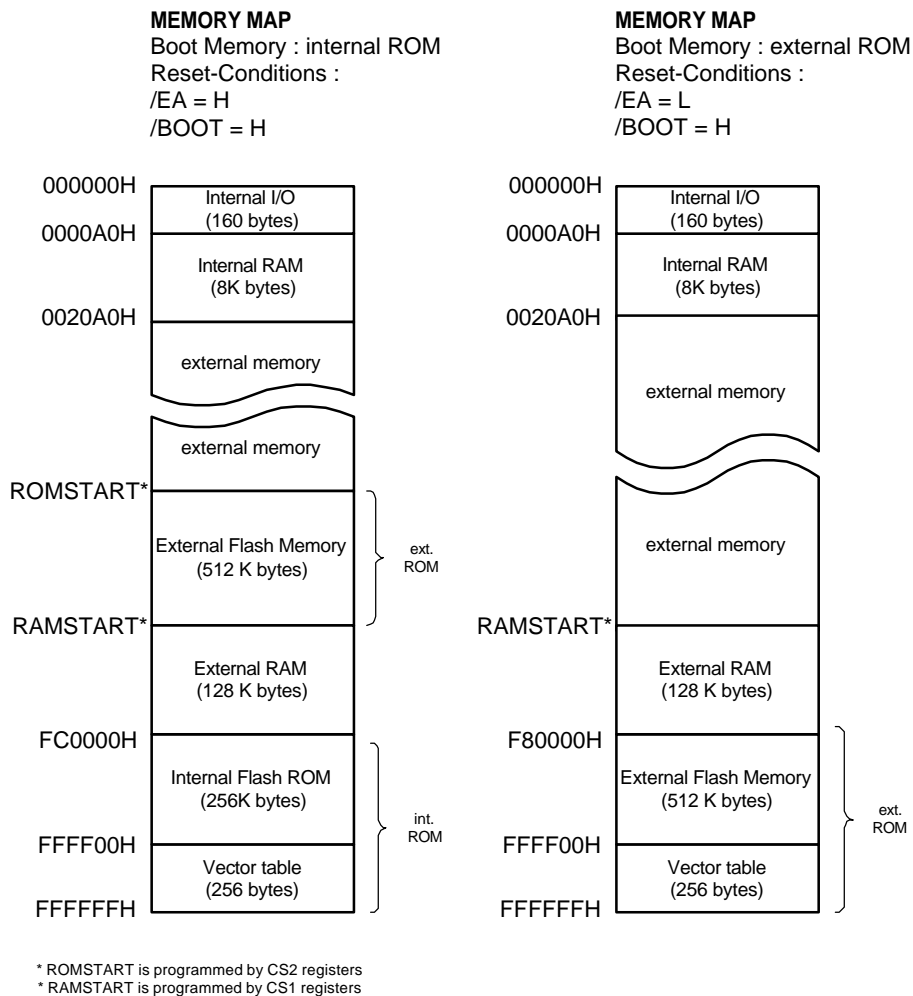* RAMSTART is programmed by CS1 registers

Figure 13 : Memory Map for using external / internal Flash

For programming the external flash the /EA line has to be set High which can be derived by opening the jumper J_EA (see fig. 1 at the right top and in the schematic fig. 10 at the bottom in the middle). The map of fig. 7 is valid and the external flash can be programmed supported by the 8-bit controller.

For normal operation and debugging it is important that the system starts from external flash, i.e. the vector table and the firmware have to be located and to be started from there. Like shown in the map of fig. 8 the external flash is located from off address F80000h to FFFFFFh including the vector area.

For running in normal operation the jumper J_EA can be left open, because the /EA line is controlled by the firmware controller. If the Flash Carrier Board works stand-alone (without Programming-and-Debugging Board), the jumper must be set when external

flash software should be executed. If the jumper is left open, the /EA pin is pulled up by a 10k pull-up resistor and the Flash MCU boots from internal flash memory. When the MCU is reset with /EA at high level, the external memory can be accessed additionally. The entire ROM size would be 768 kb and 128 kb of ram. When the MCU is reset with /EA at low level, the internal flash memory cannot be accessed additionally.

5.2.   Jumper Description

5.2.1. The jumpers of the Programming-Debugging Board

J_VCC : must only be opened when in-circuit debugging is performed with Flash Carrier Board plugged in to a PGA socket of a target application with its own power supply. In all other cases the jumper must be close to connect the power supply to the Flash Carrier Board.

J_RES : Jumper to connect/disconnect the reset line controller by the firmware controller. Should only be opened when in-circuit debugging is performed with a target application with its own reset generation circuit.

5.2.2. The jumpers of the Flash Carrier Board

J_CS0 : This jumper is to connect/disconnect the chip select line CS0 to the on-board RAM device. If an external data/address bus is not required, it is recommended to open this jumper. The CS0 pin can be used as a port pin additionally.

J_CS2 : Same as above. The CS2 pin is used to enable the on-board flash ROM device.

J_EA : Jumper to select between internal (high/open) and external flash memory access (low/closed) after reset. When the Flash Carrier Board is connected to the Programming and Debugging Board, the jumper can be left open – the line is controlled by the firmware controller.

5.3.   Programming of Flash Memory

Internal and external flash memories can be programmed with the respective tool. To start it select the program group "**TOPAS900 *Flash***" and **select "TOPAS900 *Flash* Programming Tool**" from the menu. If there is no board connected a respective error message will be displayed. If there is no physical connection this should be mounted. In a lot of cases the reason is the COM port setting. You can set the COM and the transfer data by clicking the "**Setup Port**" icon and use the setting options. If the connection is set

up the following window occurs (fig. 9) and the board reacts by red LED flashing quickly. **Be sure that the jumper J_EA is opened**.
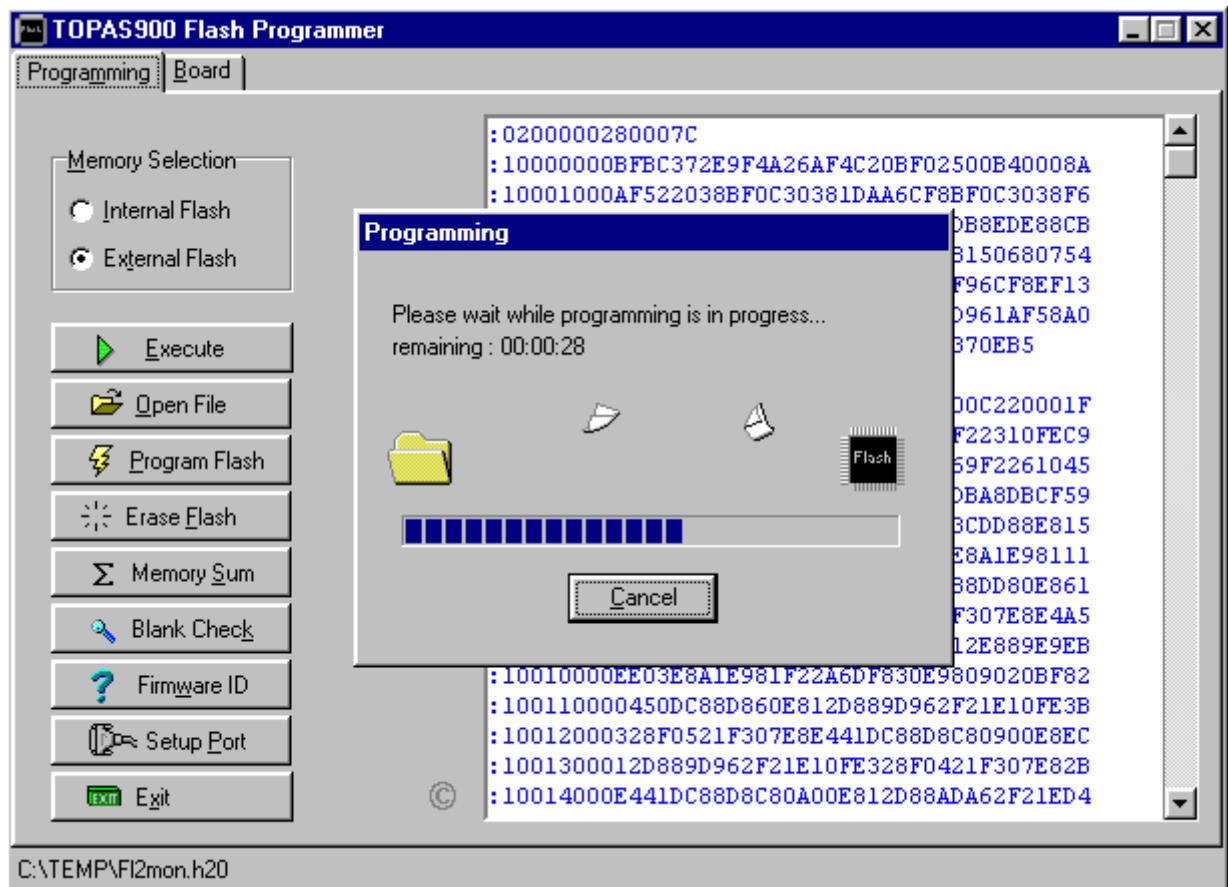

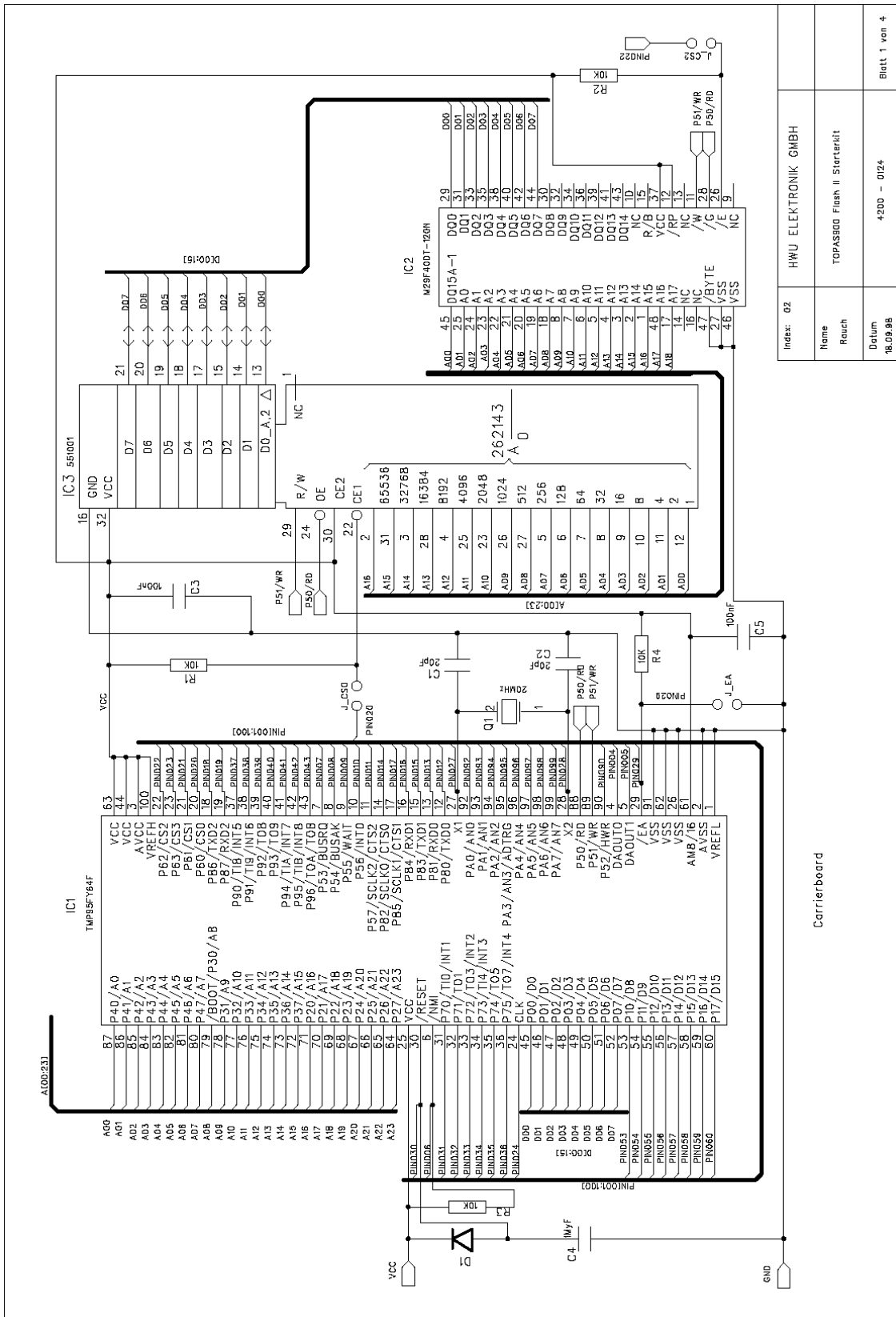Figure 14 : Flash Programmer Window

All bars and icons are self explaining. It should be noticed that either internal or external flash memory can be selected and programmed. Files to be downloaded have to be in HEX16 or HEX20 format. Conversion tools are available in both tool sets.

It is important to mention that the **ROM Monitors** have to be located in the **external flash**. Hence the reprogramming procedure is to choose the access "**External Flash**" then "**Erase Flash**" then open alternatively the files "**iar.h20**" for IAR or "**toshiba.h20**" for Toshiba ROM Monitor and then "**Program Flash**". Both files are to be found in "**Program Files/topas900flash/monitors**".

6. Technical Sheets

6.1. Board Schematics

The schematics of the complete TOPAS900 *Flash* board are given in the following four pages.

Figure 15 : Schematic Page 1 of 4
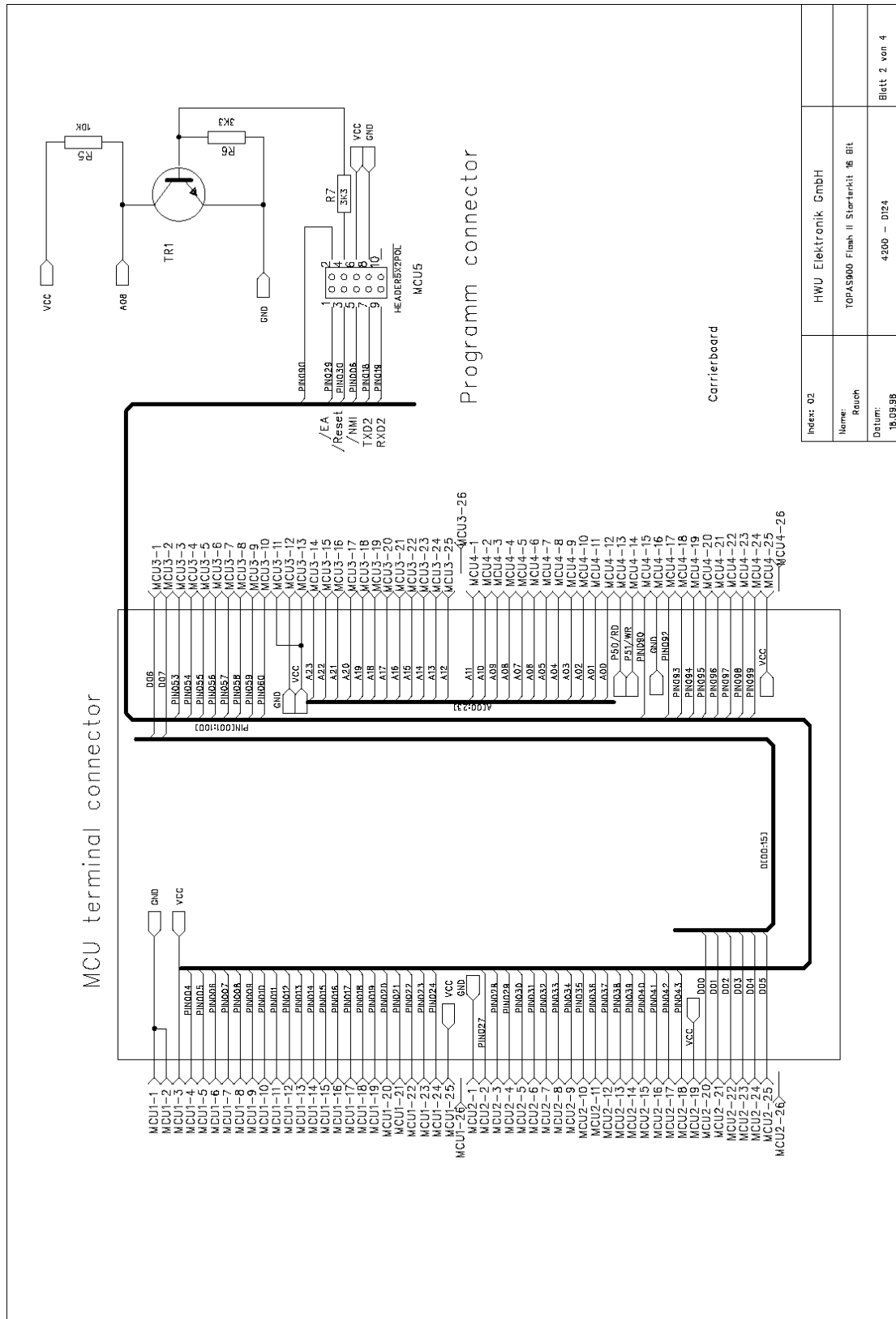
Figure 16 : Schematic Page 2 of 4
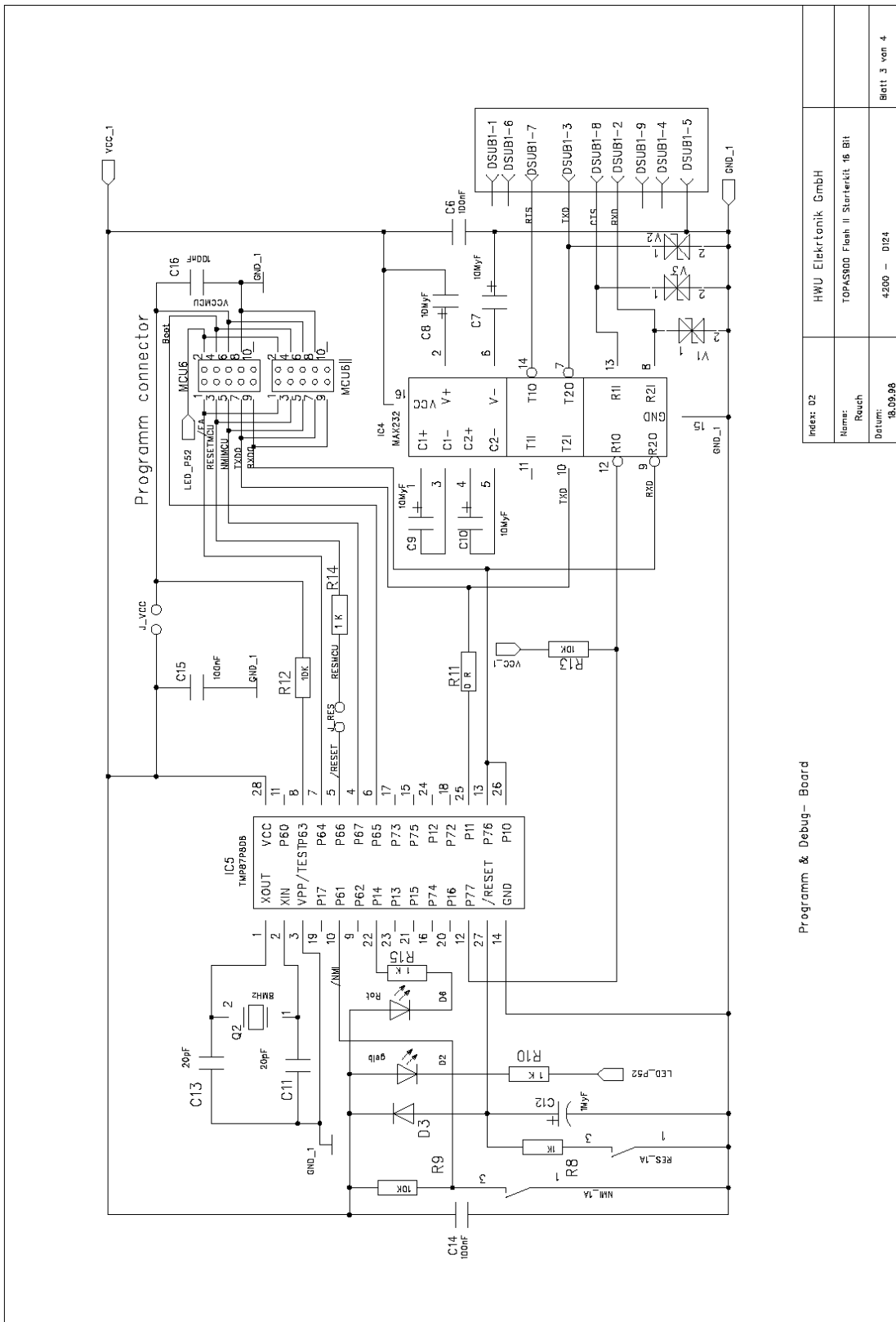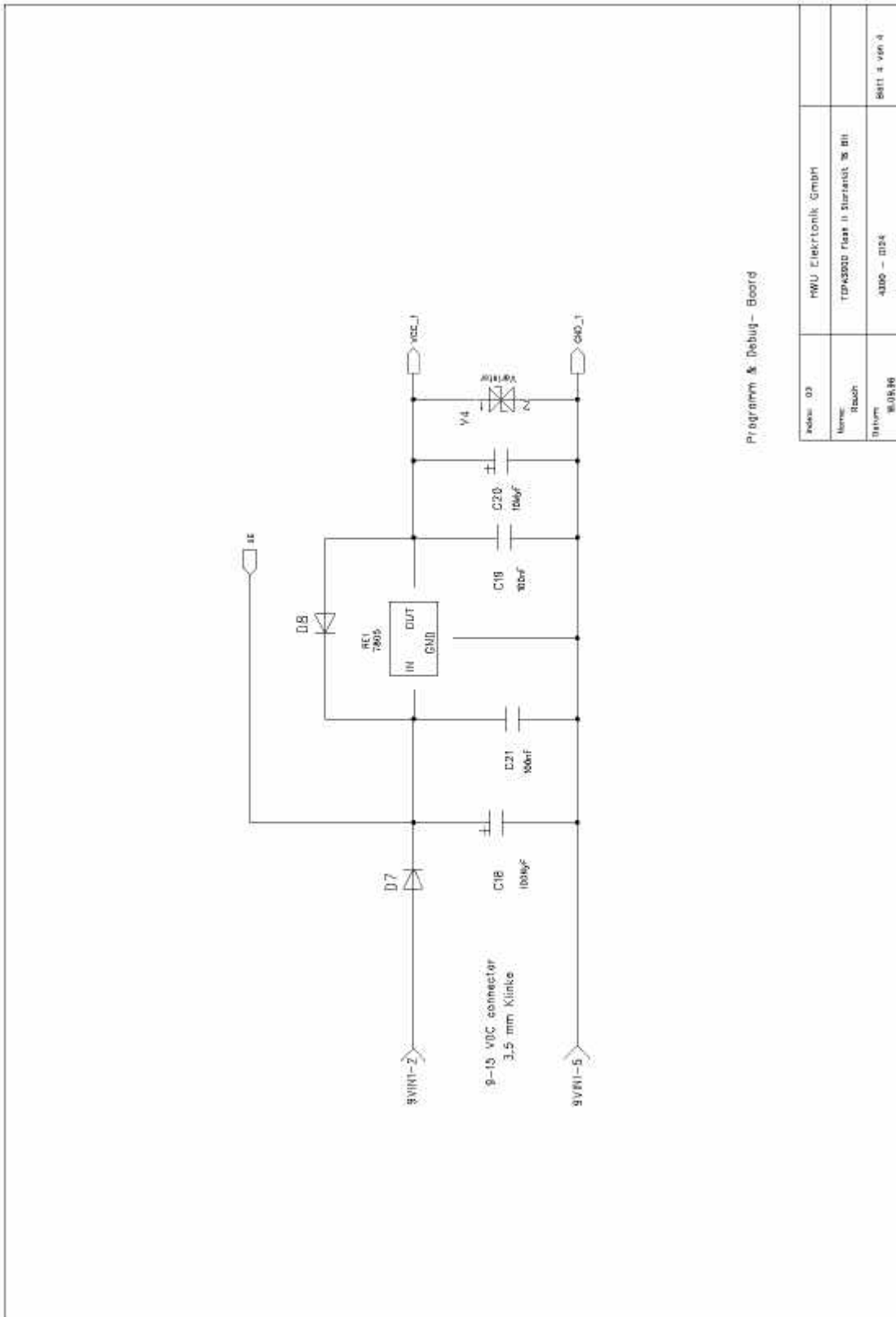
Figure 17 : Schematic Page 3 of 4

Figure 18 : Schematic Page 4 of 4

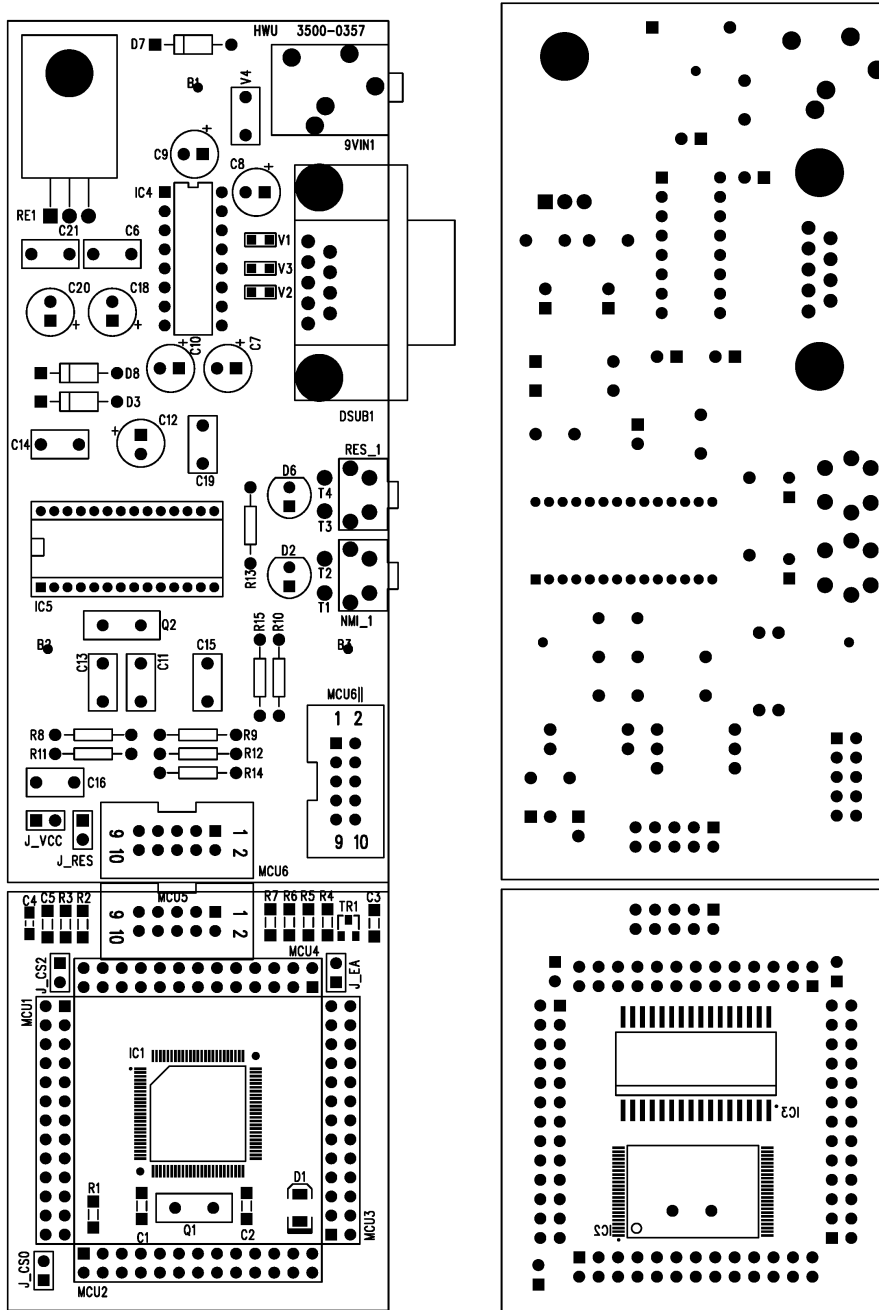### *6.2. Component Print*

Figure 19 : Component Print – Top & Bottom View
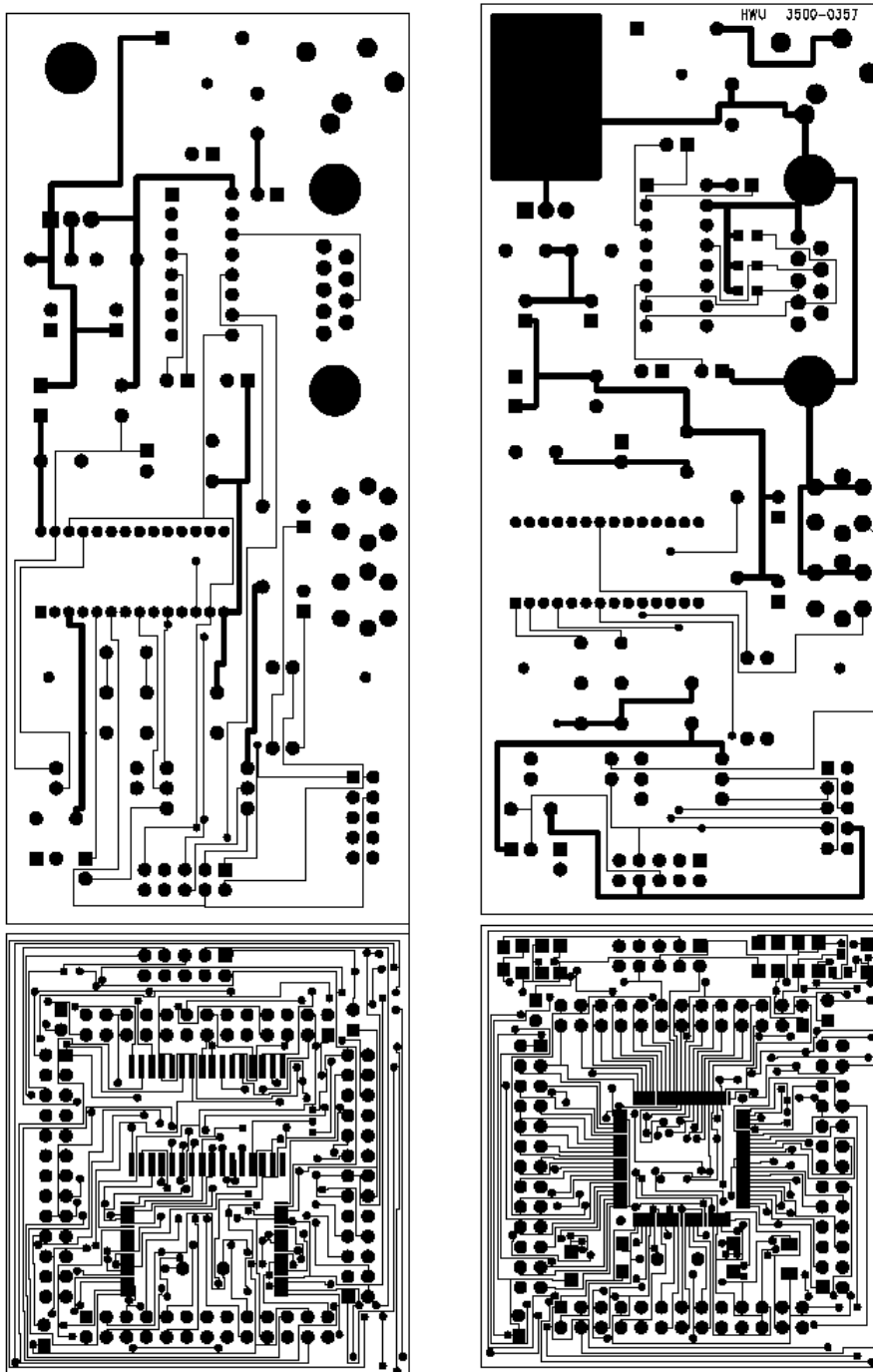
6.3. *PCB Routing*

Figure 20 : PCB Routing – Top & Bottom View

7. Application Board

To demonstrate how useful microcontrollers can be for industrial solutions and in common applications a hardware extension board – the *Application Board* – has been developed. The picture below shows the Application Board.
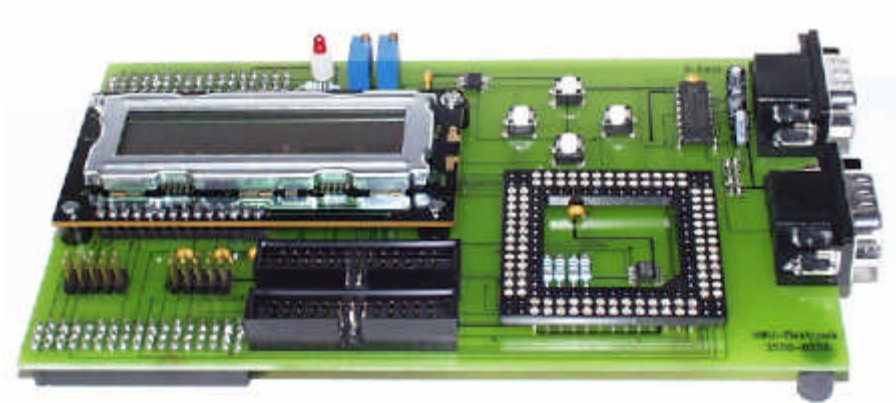

Figure 21 : The Application Board

With its hardware components and their corresponding software drivers a lot of different applications can be developed very quickly - without the common overhead of driver development. The software developer can concentrate on developing algorithms for solving specific problems instead of spending time in things that have been developed many times before.

The Application Board package comes with a set of software modules to ensure short development cycles and to give enough software samples that demonstrate the way of developing embedded systems with Toshiba microcontrollers.

The software modules are:
* 4-bit bi-directional LCD-Panel driver
* I²C driver (I²C master protocol)
* LM 75 temperature driver
* SPI bus driver
* ST95040 serial EEPROM driver
* keyboard matrix input driver
* A/D conversion driver
* serial RS232 line driver

For all software modules the hardware components are available on the board and can be tested directly.
Additionally there are some applications that uses the software modules to show how more complex applications are to be developed. One of these examples is a digital

thermometer that uses I²C bus driver, the LM75 temperature sensor and the LC-display for data output.

The Application Board is compatible with
- TOPAS900 *Standard*
- TOPAS900 *Flash II*
- TOPAS900 *CAN*

The Application Board hardware is a separate product and therefore is has to be ordered separately.

The Application Board Software is part of TOPAS900 *Flash* and will should be installed to **C:\AppBoard.**

For actual software versions please refer to our Web-Site www.hwu.de.

8. Electromagnetic Compatibility

EMC
TEST NRW

Competent Body

EMC Test NRW GmbH
electromagnetic compatibility
Emil-Figge-Straße 76
44227 Dortmund

accredited by the
**Federal Office for Posts & Telecommunications
of the Federal Republic of Germany**

Certificate

**from a competent body (as defined by § 2 no. 8 of the EMC Act) within the meaning of
§ 5 (2) EMC Act or Article 10 (2) of the EMC Directive on compliance with the
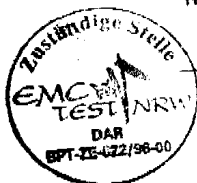protection requirements**

Certificate-No.: **001a/99**

Holder of Certificate:	**HWU-Elektronik GmbH, 46145 Oberhausen (Germany)**
Manufacturer:	**HWU-Elektronik GmbH, 46145 Oberhausen (Germany)**
Technical report, date:	**001a/99 from 1999-01-13**
Object identification:	**TOPAS900 Flash**
Object description:	**MCU Flash Starterkit**
Number of pages of the enclosure:	**6**

This certificate of conformity was issued in accordance with Article 10.2 of the Council Directive 89/336/EEC on the approximation of the laws of the Member States relating to electromagnetic compatibility implemented in the Federal Republic of Germany by § 5 (2) of the Electromagnetic Compatibility Act (EMVG) of 28. September 1998 (Federal Law Gazette I p. 1118) This certificate does not testify to compliance with the EMC protection requirements of other laws implementing Directives of the European Community other than Council Directive 89/336/EEC.

Place, date

44227 Dortmund, 1999-01-13

Head of Competent Body

(i.V. Uwe Rörden)

Version 10/98