ST92F150-EMU2
HDS2V2 Emulator
User Manual

Release 1.1

July 2001

Ref: DOC-ST92F150-EMU2

INSTRUCTIONS FOR USE—WARNING

This product is conform to the 89/336/EEC Directive. It complies with the ITE EN55022 standard for EMC emissions and generic 50082-1 (1992 edition) immunity standards.

This product is an FCC Class-A apparatus. In a residential environment, it may cause radioelectrical disturbances.

In addition, some parts of this emulator are not contained in an outer casing; consequently, it cannot be immune against electrostatic discharges (ESD). It should therefore be handled only in static safe working areas. Please refer to *Appendix A: EMC Conformity and Safety Requirements* on page 33 for relevant safety information

USE IN LIFE SUPPORT DEVICES OR SYSTEMS MUST BE EXPRESSLY AUTHORIZED.

STMicroelectronics PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF STMicroelectronics. As used herein:

1. Life support devices or systems are those which (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided with the product, can be reasonably expected to result in significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can reasonably be expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Table of Contents

Table of Contents

1 INTRODUCTION

Thanks for choosing ST9+! This manual will help you get started with the ST9 HDS2V2 emulator kit.

The emulator will assist you in debugging your application hardware as well as your software. The emulator kit comes with ST9+ V6 Toolchain, including ST9+ Visual Debug—which contains all of the necessary resources to help you design, develop and debug ST9+ application software running in a real environment.

Note: This emulator is also supported by the ST9+ V4 Software Toolchain which also includes a graphical debugger, WGDB9.

Your emulator performs two major functions:

- It reproduces the behavior and functionality of the ST9+ microcontroller in real time, by replacing it on your application board.
- By the means of the ST9+ Visual Debug graphical interface, it provides access to the microcontroller's internal resources (such as registers and memories). The management of breakpoints and application code trace offers you powerful capabilities for debugging your application programs.

The emulator is the hardware system that enables you to control and debug your application program by:

- stopping program execution at particular instructions,
- stopping the program after defined events, such as access to specific addresses, specific registers or a sequence of events,
- tracking the execution of the program in a trace memory whose contents you can filter,
- triggering a peripheral such as an oscilloscope after the occurrence of an event,
- analyzing your code performance.

The emulator takes advantages of an architecture based on two microprocessors. This enables you to keep control of the emulator, even if the ST9+ application hangs.

All of the above emulation functions are accessible through ST9+ Visual Debug. Complete information concerning how to use ST9+ Visual Debug can be found in its on-line help.

1.1 Emulator description

The ST9 HDS2V2 emulator is composed of three main parts (see *Figure 1* on page 6):

- **The ST9 HDS2V2 mainboard casing**, connected at one end to a host PC via a parallel cable. This mainboard contains all emulation resources and communicates with the PC via a master microcontroller.
- **A specific probe** connected to the mainboard casing via three flat cables. This probe contains the emulated ST9+ microcontroller, which runs the application program using its internal peripherals.
- **ST9+ Visual Debug** integrated development environment, running on a host PC.

The probe is connected to your application via the ST9+ microcontroller's socket and will function in its place. A software debugger sends and receives information to and from the emulator and the ST9+ Visual Debug graphic interface displays all available information on your PC screen.

Note: When receiving the emulator kit, please refer to the Delivery Checklist on page 11 to confirm that all of the contents of the package are present.

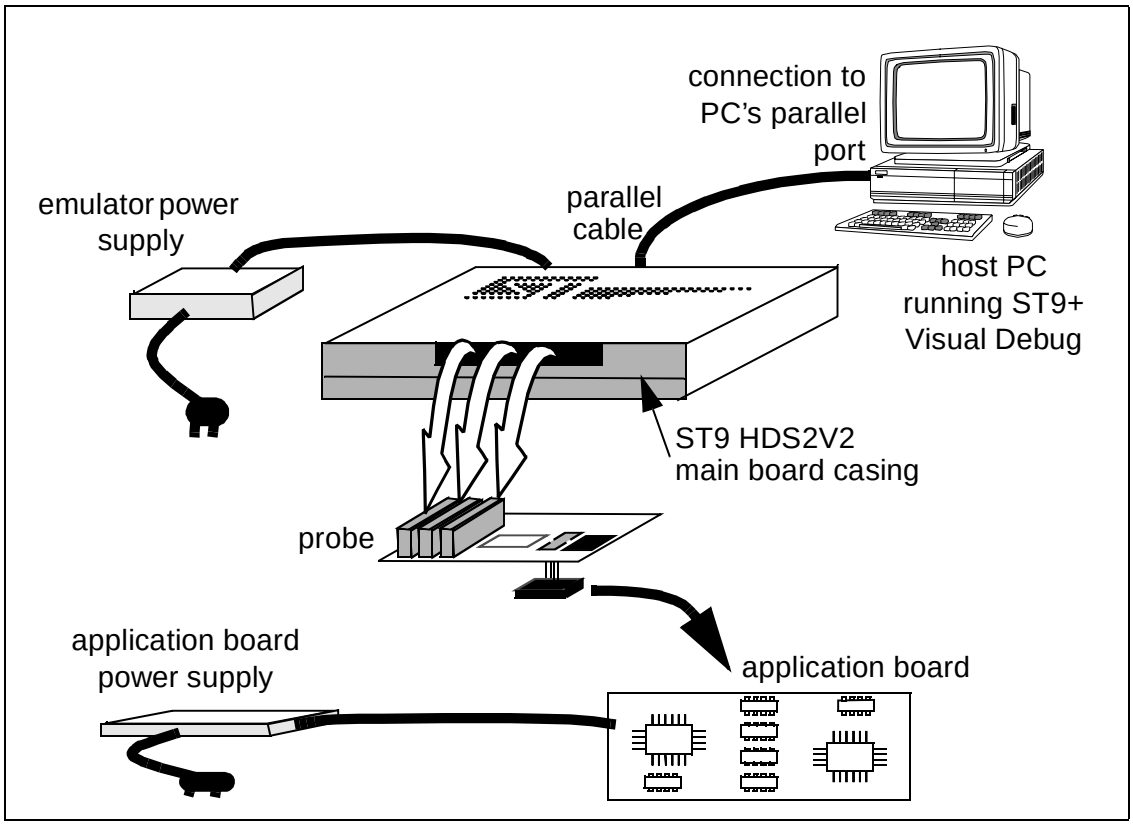


Figure 1: ST9 HDS2V2 General Configuration

1.2 ST9+ Visual Debug

The ST9 HDS2V2 emulator is controlled using **ST9+ Visual Debug**, (also referred to as **STVD9**). ST9+ Visual Debug integrates all phases of an application development in a single, powerful and easy to use environment.

The graphical interface is composed of one main window which consists of an integrated set of windows, menus, toolbars and other elements that allow the user to create, build, debug and refine an application using a single tool. The visual environment can be personalized; the user can change the position of all windows, open or close individual elements, alter which windows are visible at startup, and save the configuration. User-defined toolbars may be created, and command icons added or removed from toolbars. The integrated syntax-highlighting editor allows several source files to be edited in a single place.

Thanks to the powerful features of the debugger, the debugging and tuning of an application is made easier and quicker. Debugging with ST9+ Visual Debug involves the following functional aspects:

- Loading the application.
- Defining the memory mapping for the target MCU.
- Running the application.
- Viewing source and disassembled code.
- Setting instruction breakpoints.
- Setting registers breakpoints.
- Setting advanced breakpoints.
- Viewing variables, memory and registers.
- Viewing history of execution: trace (up to 256k records) and call stack features.
- Analyzing the performance of a piece of code.

ST9+ Visual Debug is run on the host PC which is connected to the HDS2V2 emulator. *Section 2.5* on page 17 explains how to install ST9+ Visual Debug on your PC, and set up the emulator configuration so that you can begin your debugging session.

1.3 Software for the emulator kit

The “MCU on CD” CD-ROM contains the following software:

- ST9+ V6 Toolchain, comprising the following software:
  - The source-level graphic debugger, ST9+ Visual Debug, that operates with HDS2V2 Emulators.
  - An assembler, linker, librarian and gmaker and binutils.
  - A ST9 C compiler is also available on the CD-ROM, but you must purchase a license in order to use it.
- The STVP9 programming software to allow you program your MCU target devices with ST9 EPBs (available separately).

1.4 Related documentation

To get all the essential information about your ST9+ MCU and the software that comes on the “MCU on CD” CD-ROM with the emulator kit, you may also need to refer to these documents—all contained in PDF format on the CD-ROM:

- ST9-Family Data Sheets
- The ST9+ V6 Software Toolchain documentation set, consisting of:
  - ST9+ V6 Software Toolchain User Manual
  - ST9+ V6 Software Toolchain — V4 to V6 Migration Notes
  - ST9+ V6 Software Toolchain — ELF, as9, ld9 and Binutils Reference Manual
  - ST9+ V6 Software Toolchain — gcc9, Libraries and Startup Files Reference Manual
  - ST9+ V6 Software Toolchain — gmake Reference Manual
- ST9 release notes, application notes (with sources), demos, training slides and exercises, this manual (in PDF version), and other useful reference materials.

1.5 About this manual....

This user manual describes the emulator configurations from the hardware point of view. For topics completely related to software part of the tool, refer to the *ST9+ V6 Software Toolchain User Manual* and to the ST9+ Visual Debug on-line help.

Detailed instructions on how to install your emulator hardware and software is described in *Chapter 2: Getting Started* on page 11.

How to configure your emulator’s hardware features is described in *Chapter 3: Configuring the Emulator* on page 19.

The following conventions are used in this manual:

Bold text highlights key terms, phrases and is used when referring to names of dialog boxes, windows and tabs within windows.

Bold italic text denotes menu commands (or sequence of commands), options, buttons or checkboxes which you must click in order to perform an action.

Italicized text highlights document names, variable strings, column names and field names.

`Code font` designates file names, programming commands, path names and any text you must type.

The **>** symbol is used in a sequence of commands to mean “then”. For example, to open an application in Windows, we would write: “Click ***Start>Programs>ST9 Tool Chain>***....”.

1.6 Getting assistance

For more information, application notes, FAQs and software updates on all the ST microcontroller families, check out the CD-ROM or our website:

http://mcu.st.com

For assistance on all ST microcontroller subjects, or if you need help with using your emulator, use the contact list provided in *Contact list* on page 77. We’ll be glad to help you!

2 GETTING STARTED

2.1 Your system requirements

The ST9 HDS2V2 Emulator (both hardware and software components) has been designed to work with PCs meeting the following requirements:

- One of the following operating systems: Microsoft® Windows® 95, 98, 2000 or NT®.
- Intel® Pentium (or compatible) processor with minimum speed of 133 MHz.
- Minimum RAM of 32 MB (64 MB recommended).
- 50 MB of free hard disk space to install all of the ST9 tools.

2.2 Delivery checklist

The emulator is delivered with the following equipment:

- the ST9 HDS2V2 mainboard,
- the specific probe (ref.: DB492-C),
- three Yamaichi QFP sockets to solder in place of the ST92F150 MCU package to be used in your application
  - TQFP64 (14x14) Yamaichi socket (ref.: IC149-064-008-S5)
  - TQFP100 (14x14) Yamaichi socket (ref.: IC149-100-025-S5)
  - TQFP100 (20x14) Yamaichi socket (ref.: IC149-100-014-S5)

Note: Be aware when designing your application board that Yamaichi socket overall dimensions are larger than device ones. For more information see *QFP64/TQFP64 footprint issues on page 37,* *and the Yamaichi WEB site at address:*
http://www.yamaichi.de/Pu/quad_flat_pack/spec/a21-ic149.htm

- three rigid adapters to fit the target package:
  - a TQFP64 (14x14) rigid adapter (ref.: DB503)
  - a TQFP100 (14x14) rigid adapter (ref.: DB502-B)
  - a PQFP100 (20x14) rigid adapter (ref.: DB329-B)
- Three flex cable adapters to fit the target package:
  - a TQFP64 (14x14) flat cable adapter (ref.: DB424)
  - a TQFP100 (14x14) flat cable adapter (ref.: DB501-B)
  - a PQFP100 (20x14) flat cable adapter (ref.: DB471)
- a 5 V power supply for the ST9+ emulator,
- a parallel cable for connection between the ST9+ emulator and the host PC,

- three flat cables with ferrites for connection between the probe and emulator mainboard,
- an analyser probe cable,
- the “MCU on CD” CD-ROM, containing both the ST9+ V4.3x and V6.x Software Toolchains (each software toolchain includes a Graphical Windows Debugger and its on-line help),
- this manual.

2.3 Before you start...

Be careful that neither the emulator nor application board is powered up before the entire installation session has been performed. The emulator should be powered up first, followed by your application board, in order to prevent damage to the probe.

2.4 Installing the emulator hardware

The ST9 HDS2V2 emulator is connected through the parallel port to a PC computer which runs the control software (ST9+ Visual Debug) as explained later. To connect your ST9 HDS2V2 emulator, you will have to follow these general steps:

1 Connect the ST9 HDS2V2 to your PC using the parallel cable provided.

2 Connect the three flat cables of your ST9 HDS2V2 emulator to the emulation probe connectors.

3 Connect the appropriate device adapter to the emulation probe, then connect the device adapter to the matching socket on your application board.

4 Connect the power supply cable between the power supply block and the power connector located on the rear panel of your ST9 HDS2V2 emulator.

5 Power up the emulator and then connect your application power supply.

A connection flow diagram is shown in *Figure 2* on page 13. Each installation step is described in detail in the following sections.

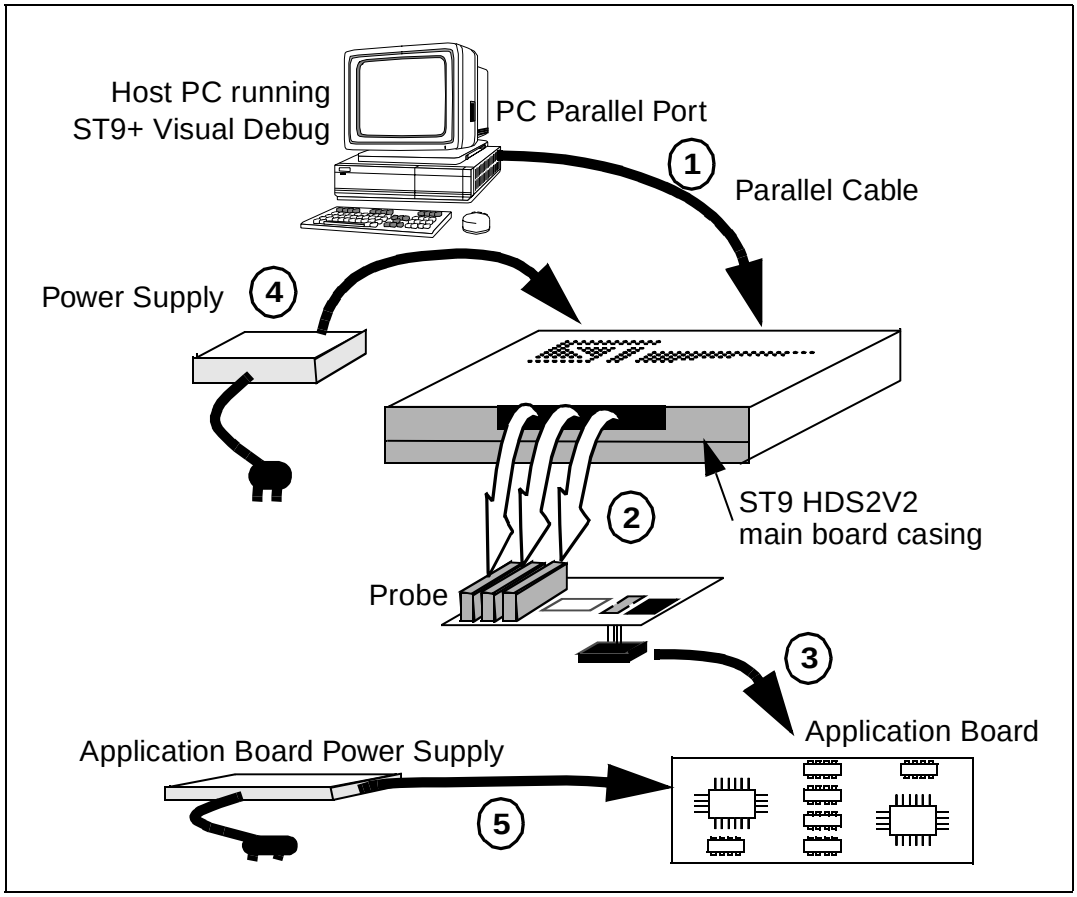


Figure 2: Emulator connection flow diagram

Caution: ***Power OFF before installation! Before making any connections, power OFF the application board and the mainboard switch located on the rear panel.***

2.4.1 Step1: Connecting the mainboard casing to the PC

1 Shutdown and power-off the PC that is to be connected to the emulator.

2 Connect one end of the parallel cable to the mainboard casing's rear panel 25-pin SUB-D connector and the other end to one of the PC's parallel ports (LPT1 to LPT2)—refer to *Figure 3*.

Note: *Centronics (or PC-AT or SPP), ECP and EPP parallel port configurations are supported by the emulator.*
Be sure to use the parallel cable provided with the emulator—using a longer parallel cable may cause emulator malfunctions.

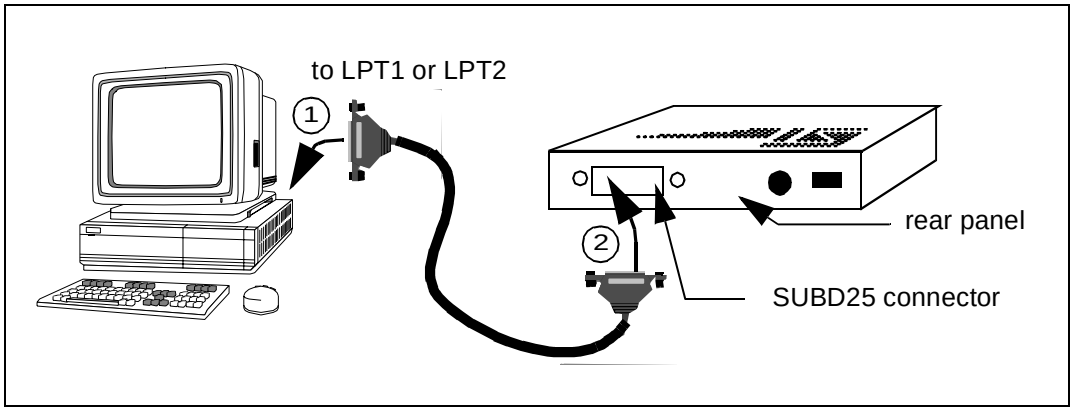


Figure 3: Connecting the emulator to the PC

2.4.2 Step 2: Connecting the mainboard to the probe

The three flat cables must be connected as follows (see *Figure 4* on page 15 for schematic):

1 The upper cable from mainboard to the J1 connector on probe.

2 The middle cable from mainboard to the J2 connector on probe.

3 The lower cable from mainboard to the J3 connector on probe.

4 **EMC conformity** (optional): In order for the ST9 HDS2V2 emulator to meet the EMC requirements of the European guideline 89/336/EEC, you must follow the guidelines laid out in *Appendix A: EMC Conformity and Safety Requirements* on page 33 and place the three ferrites provided as shown in *Figure 5* on page 15.

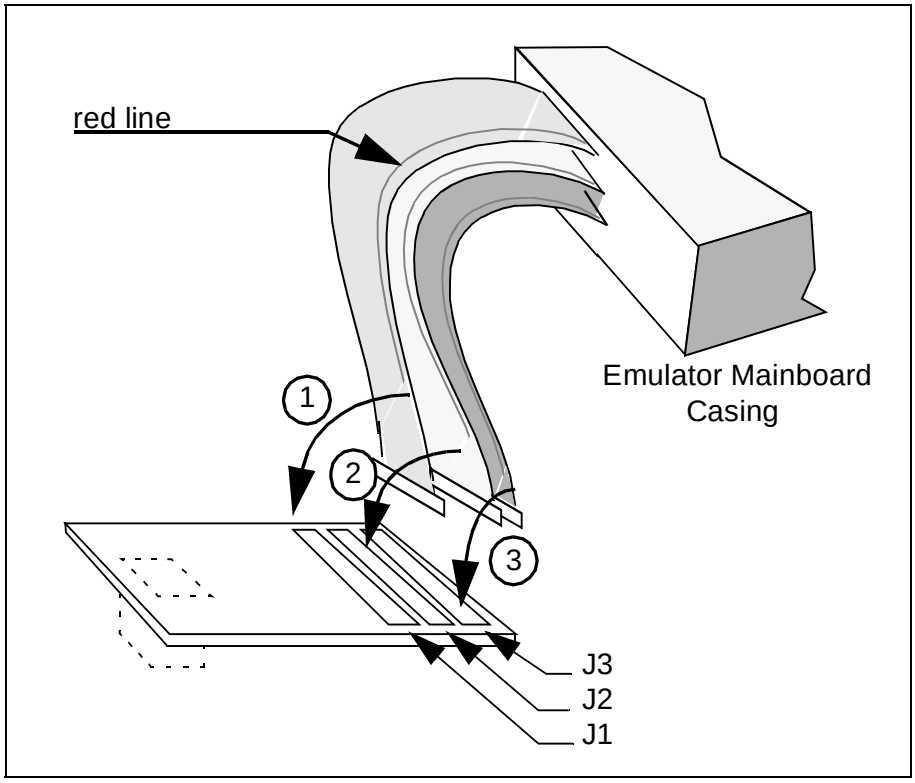


Figure 4: Connecting the mainboard to the probe

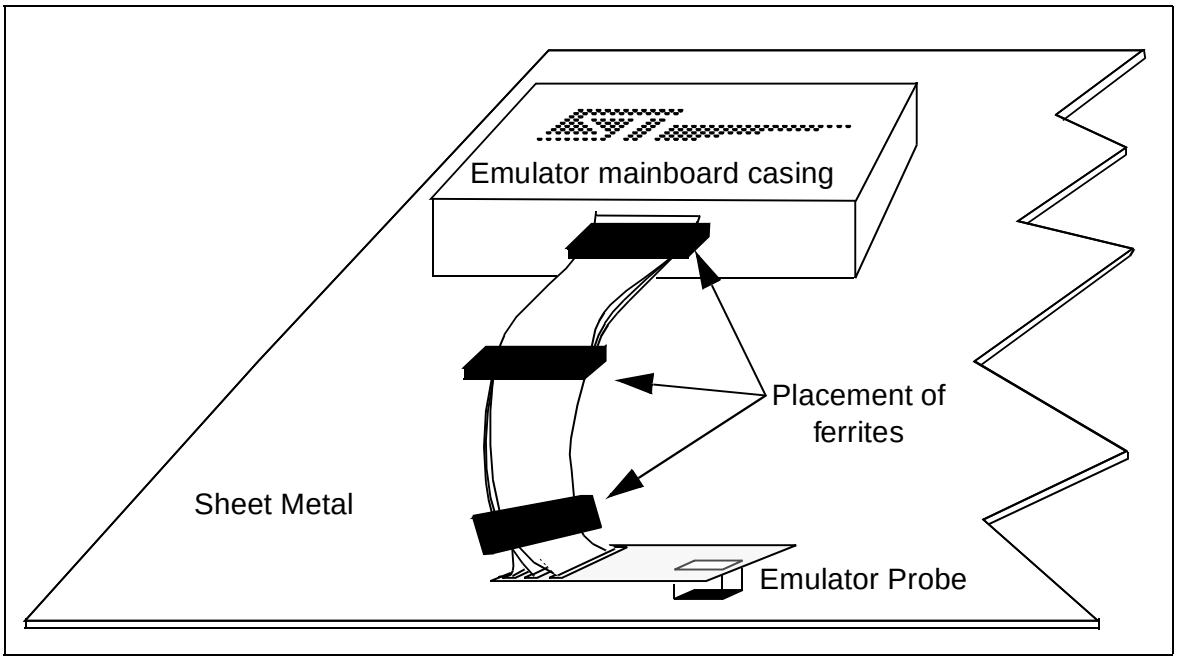


Figure 5: Placement of ferrites for EMC conformity

2.4.3 Step 3: Connecting the emulator to the application board

According to the ST92F150 package you want to emulate (TQFP64, TQFP100 or PQFP100), choose the appropriate socket, rigid adapter or flat cable adapter.

1 Unpack the Yamaichi QFP socket provided with the probe, setting the cover aside and keeping the screws for step 3.

2 Solder the Yamaichi socket in place of ST92F150 microcontroller onto your application board.

3 Screw the rigid adapter or flat cable adapter onto the socket.

4 Connect the ST92F150 probe onto the adapter (rigid or flat cable).

Note: For the rigid adapter, pin "1" is at the cut corner. For the flat cable adapter, pin "1" is labelled on the adapter. For the ST92F150 probe, pin "1" is labelled on the component-side of the probe.

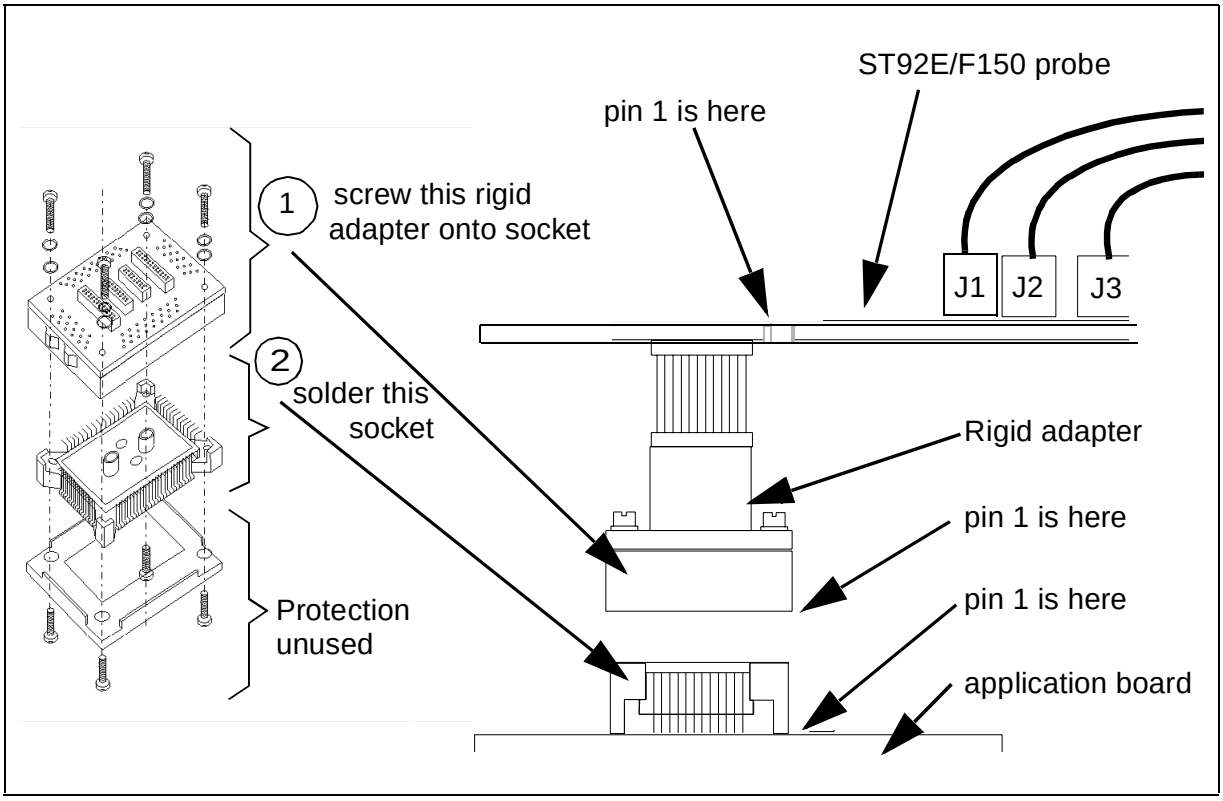


Figure 6: Connecting the probe to your application board

2.4.4 Step 4: Connecting the power supply

Connect the power supply provided with the emulator to the mainboard rear panel.

2.4.5 Step 5: Powering up

1 Power up the ST9 HDS2V2 emulator from the ON/OFF switch located on the rear panel. The three LEDs labelled ***Power***, ***Emulator Reset*** and ***ST9 Reset*** on the front panel should then light up.

2 Power up your application board.

Warning: Always power on your ST9 HDS2V2 emulator first, then power up your application board.
When powering off, always power off your application board first, then power off your emulator.
NEVER have your application board under power while the emulator is powered off—this will cause serious damage to the emulator.

2.5 Installing the ST9+ software toolchain

Your ST9 HDS2V2 emulator comes with the “MCU on CD” CD-ROM which contains ST9+ software toolchain. These tools are compatible with Windows® 95, 98, 2000 and Windows® NT® operating systems.

Note: Windows® 2000 and NT® users must have administrator privileges to install the software toolchain.

To install ST9+ Visual Debug or the entire ST9+ Software Toolchain, follow these steps:

1 Close all other open applications on your Windows desktop.

2 Insert the “MCU on CD” into your CD-ROM drive. The CD-ROM’s autorun feature will open up a welcome screen on your PC. If the autorun feature does not work, use Windows® Explorer to browse to the CD-ROM’ s root folder, and double-click on `Welcome.exe`.

3 Select ***Install Your Development Tools*** from the list of options. A new screen will appear listing the different families of STMicroelectronics MCUs.

4 Use your mouse to place the cursor over the ***ST9 Tools*** option. Choose ***ST Tools*** and then ***ST9 Toolchain V6*** which contains ST9+ Visual Debug, as well as the ST9 Assembler-Linker software and other tools.

Note: You may also choose the ST9+ Toolchain V4 which is an older version of the software toolchain. Installing the V4 toolchain is only recommended if your application makes use of legacy libraries, etc.

5 The install wizard will be launched. Follow the instructions that appear on the screen.

The installation is now complete. A tutorial on how to start using ST9+ Visual Debug is contained in the *ST9+ V6 Software Toolchain User Manual*, available in PDF format on the “MCU on CD” CD-ROM.

3 CONFIGURING THE EMULATOR

3.1 Selecting the clock source for the ST9+ chip

You can select one of the three clock sources for ST92F150 emulation chip:

- A quartz (not provided), to be located on the probe between C4 and C5.
- A 4 MHz oscillator (provided), located on probe at location OSC2.
- The target TTL clock source, located on your application board.

The clock source selection is done thanks to five solder points (G3, G4, G5, G6, G7) and one jumper W4. On the following table “N” means no contact, “Y” means the solder points contacts must be soldered together.

Clock Source	G3	G4	G5	G6	G7	W4
on probe quartz (4 MHz)	N	Y	N	Y	N	x
on probe 4 MHz oscillator (XT1)	Y	N	Y	N	N	P-OSC
external TTL clock source	Y	N	Y	N	N	T-OSC

Table 1: Clock source solder points and jumper settings

As delivered, the emulator clock source selected is the on-probe 4 MHz oscillator. The two contact points of both the G3 and G5 solder points are connected with a thin copper wire, and the W4 jumper is in the P-OSC position.

To remove the connection on the G3 and G5 solder points, simply scratch the thin copper wire between the two contacts of each solder point.

3.2 Selecting the A/D converter analog reference source AV_{DD}

You can supply AV_{DD} either from emulator probe power supply or from your application using AV_{DD} pin of ST92F150.

	G1	G2
AV_{DD} from emulator probe	Y	N
AV_{DD} from application (AVDD pin)	N	Y

Table 2: AV_{DD} supply selection solder points

As delivered, AV_{DD} is supplied from emulator probe—i.e. the G1 solder point contacts have been connected with a thin copper wire. To remove contact on the G1, scratch the thin copper wire between the two contacts of the solder point.

3.3 Configuring the P5.0 pin

You can choose to between configuring the P5.0 pin for use as a port or for use with the WAITn function. *Figure 10* below shows how to set jumpers W2 and W3 for each configuration

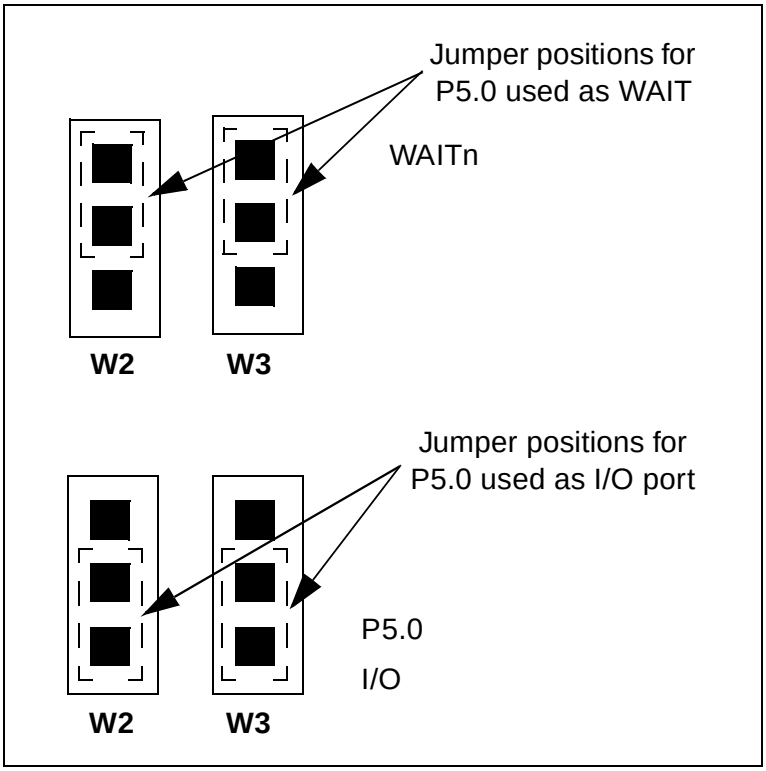


Figure 10: P5.0 pin configurations

3.4 Mapping the memory space

3.4.1 Internal memories

The actual memories located inside an ST9+ chip (such as RAM, EEPROM and FLASH) are called *internal memories*. When we say that we are using internal memory, it means that we are in fact using the memory internal to the target ST9+ device located on the emulation probe. Doing this, however, greatly limits the debugging functions that the emulator can perform, as described in the next paragraph.

3.4.2 System memory

Some of the ST9+ chip's internal memories, such as RAM and FLASH, are, in general, invalidated for emulation purposes. In the emulator, emulation RAM memory replaces some ST9+ internal memory types, and enables you to debug an application program by modifying the code whenever an error is encountered. This emulation memory is located on the emulator's mainboard and is called *system memory*.

3.4.3 FLASH and EEPROM memories

The emulator allows you to define the FLASH and EEPROM memory mapping as either internal or system. However, all FLASH memory must be mapped in the same location (internal or system); EEPROM memory has the same constraint.

When these memories are mapped as internal, the emulator does not have the same debug information available as it would have when using system memories. For this reason, when using internal memories, the values read from them cannot be accurately known when the chip is running. In addition, the use of advanced breakpoints conditional on values read from these memories is forbidden.

Another consequence of using internal memories is that when the Trace window displays values read from these internal memories, they will be inaccurate. To show this in the Trace window, the such values will appear in grey text with a question-mark ("?") at the beginning and at the end of the inaccurate data.

Nevertheless, when the user software execution is stopped, the data in these memories can be dumped correctly to the Memory window.

During the main development stage, you should map the FLASH memory as system, which allows all of the debugging features of the emulator to be used.

However, when you want to test the FLASH programming, map the FLASH memory to internal and program the FLASH memory of the emulation chip on the probe with your application board (using ISP or your boot-program located in RAM). Afterwards, you can display the FLASH memory of the emulation chip in ST9+ Visual Debug's Memory window (accessible by selecting ***View>Memory*** from the main menu bar).

Note: If you program the emulation chip on the probe using ISP, the emulator must be running (executing a loop program).

Usually the EEPROM memory will be mapped to internal. Nevertheless if you want to put a condition on a value read in the EEPROM memory, or trace the values read in it, you can map it as system memory.

3.4.4 User memories

You can map external memories segments in two different ways:

- either you can use the on-emulator memories to emulate those segments. In this case, you define them as system memories,
- or you can use the memories located on your application board. In this case, you define them as user memories.

In both cases all emulation features (such as breakpoints, trace, memory window, memory access control) are available.

Some differences in behavior must however be considered.

When mapping external segments as system memories, the following points must be taken into account:

- When external memory is accessed through port 0 or port 1, the emulator will not detect if the ports are incorrectly configured as alternate functions to access external memories. If this configuration has not been handled correctly, your application program may run correctly with system memories, but it will not run with user memories or with the chip itself.
- The emulator will function at its maximum operation speed without need of wait states for the system memories access whereas the wait states could be needed for the chip to access your application memories.

In order to avoid problems of external memory interface configuration, we advise you to debug your application program in two steps:

1 The first step is to debug your program using the emulation memories.

2 The second step is to debug your program with the application memories to ensure that ports and access configurations are correct.

3.4.5 Memory location

In most cases, configuration is done by memory segment.

All MMU bits are available on **ST92F150 emulation chip**, however none are available as port P2 alternate function on ST92F150 target device. So, with the ST92F150-EMU2 emulator, all 4 MB of memory space can be used as system memory but only 64 KB can be used as user memory. This has been done to support future versions of ST92F150 devices. All ST92F150 device mappings are available, plus an additional mapping—the “ST92F150 emulator” mapping which has the possibilities listed in *Table 3* on page 23.

ST92F150 memory areas	Memory types
segment 00h: @0000h-001Fh	system
segment 00h: @0020h-FFFFh segment 01h: @0000h-FFFFh	system or internal or non-existent
segments 02h, 03h	system or non-existent
segments 04h-1Fh	system or user or non-existent
segment 20h: @0000h-03FFh (16K-RAM)	system or non-existent
segment 21h	reserved
segments 22h from 0000h to 03FFh (1K-eeprom) from 1000h to 101Fh (control reg.) from 4000h to 401Fh (control reg. from 8000h to 8FFFh (4K-F4) from 9000h to 90FFh (256:status) from C000h to CFFFh (4K-F5) from D000h to D0FFh (256:status)	 internal or system or non-existent internal or non-existent internal or non-existent internal or system or non-existent internal internal or system or non-existent internal
segments 23h from 0000h to 1F7Fh (Test-FLASH) from 1F80h to 1FFFh (FLASH-OTP)	 internal or system internal or system
segments 24h-3Fh	system or user or non-existent

Table 3: Choosing the appropriate memory types for your application

3.4.6 Controlling the memory accesses

The memory mapping feature in ST9+ Visual Debug allows you to define access characteristics of the memory: non-existing, only readable, or readable and writable.

When not specified in the map, the memory areas are considered as non-existent, and any access to that memory zone by your application will generate an error message.

3.4.7 Mapping the memory space

Mapping the memory space can be performed by using the **Memory Mapping** window of ST9+ Visual Debug, accessible by selecting ***Emulator>Memory Mapping*** from the main menu bar.

3.4.8 Mapping granularity

You can define the memory mapping with a granularity of 32 bytes.

3.5 Resetting the chip

3.5.1 Performing a software reset

The command **CHIP RESET** of the debugger allows you to send a reset to the ST9+ device and stops the application on the reset vector.

3.5.2 Performing a hardware reset

You can send a reset to the ST9+ device by applying a 0 level on the ST9+ RESETn pin (on the emulator probe adapter).

- if your application program is running, the reset will be taken into account as a normal device reset and the program will keep on running from the reset vector address without any disturbance from the emulator.
- if your application program is not running, the reset is ignored. If maintained when you will run our program, the reset will be taken into account.

3.6 Interrupting the program execution at any time

Loading a program into the emulator allows you to execute it and stop it whenever you need to. You can stop the program execution at any time with the debugger—refer to the ST9+ Visual Debug on-line help.

Note: If you perform a program stop in ST9+ Visual Debug while the ST9+ is in HALT mode, a chip reset is applied, and the ST9+ will stop on the reset vector. The HALT instruction will be recorded in the Trace buffer.
If you perform a program stop in ST9+ Visual Debug while the ST9+ is in STOP mode, you will wake up the ST9+. If you perform a Continue or a Step command afterwards, the ST9+ will return to STOP mode. The only real way to exit STOP mode and continue executing your program is to have the STOP mode exit condition satisfied (i.e. reception of a wake-up input signal).

3.7 Setting breakpoints

To set instruction breakpoints, register breakpoints and advanced breakpoints refer to the ST9+ Visual Debug on-line help.

3.8 Front panel connections

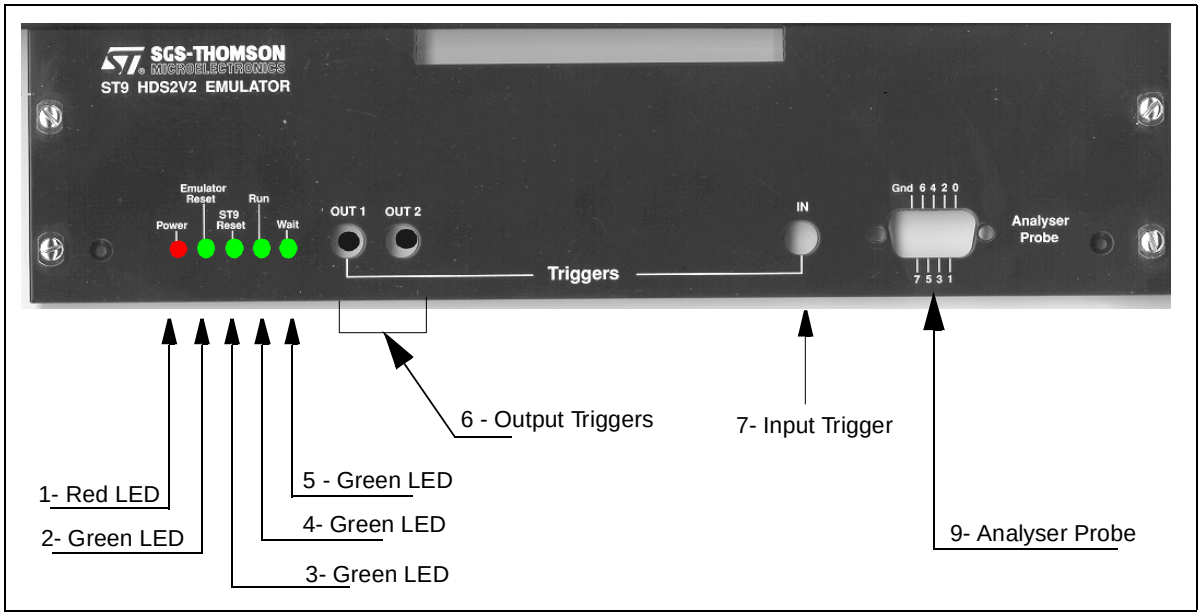


Figure 11: HDS2V2 mainboard front panel

The HDS2V2 front panel connections are shown in *Figure 11* and are described below:

1 Red LED, when ON, indicates that the emulator power supply is on.

2 Green LED, when ON, indicates that the emulator is in RESET state.

3 Green LED, when ON, indicates that the ST9+ microcontroller is in RESET state.

4 Green LED, when ON, indicates that the ST9+ is running the application program.

5 Green LED, when ON, indicates that the ST9+ is in WFI state.

6 SMB Connector for the Output Trigger 1.

7 SMB Connector for the Output Trigger 2.

8 SMB Connector for the Input Trigger.

9 Connector for the Logic Analyser Probe inputs.

Note: For SMB connector references, see Hardware spare parts on page 52.

LED activity / Emulator state	Power	Emulator Reset	ST9 Reset	Run	Wait
Emulator switched off	-	-	-	-	-
Emulator powered on	ON	ON	ON	-	-
Emulator reset	ON	ON	ON	-	-
ST9+ reset	ON	-	ON	-	-
ST9+ running	ON	-	-	ON	-
ST9+ in WFI mode	ON	-	-	-	ON
ST9+ program stopped	ON	-	-	-	-

Table 4: Front panel LEDs and their meanings

3.9 Triggering out a signal

The Advanced Breakpoints feature allows you to trigger out signals when a specified event has occurred. You can:

- trigger out a pulse,
- toggle the trigger output,
- trigger out a high level signal,
- trigger out a low level signal.

Trigger Signal	Signal characteristic	Trigger Activation
pulse	width = 2*T between 2 DSn rising edges	on the 3rd internal Data Strobe rising edge after the event
toggle	invert the level of the trigger on a DSn rising edge	
high level	output a high TTL level	
low level	output a low TTL level	

Table 5: Trigger signal types

Note: T = 1 CPU cycle.

3.10 Analyser probe

The analyser probe, a SUBD9 connector located on the front panel of the emulator main board casing, provides eight external probe inputs that can be used as:

- possible advanced breakpoint events
- bus signals in the trace.

They are synchronized with DSn and respect the 5 V TTL input levels: V_{IL} = 0.8 V, V_{IH} = 2 V.

The analyser probe cable consists of 8 colored wires that correspond to specific pin numbers, as shown below in *Table 6*.

Input Number	Wire Color
0	Brown
1	Red
2	Orange
3	Yellow
4	Green
5	Blue
6	Violet
7	Grey

Table 6: Pin/wire color correspondence on analyser probe

3.11 Configuring the timestamp clock

The Timestamp clock can be configured using ST9+ Visual Debug. Open the Trace window by selecting, from the main menu, ***View>Trace***. With your cursor in the Trace window, right-click to obtain the Trace contextual menu, and then select ***Emulator Commands>Set Timestamp Clock***. The Timestamp Clock dialog box, shown in *Figure 12* on page 28, will appear.

The clock source available for timestamp clock depends upon the ST92F150 clock source selected (see *Section 3.1* on page 19):

- When using the **on-probe quartz**, only use the fixed 20 MHz reference is available for use as the timestamp clock. No configuration is required—the timestamp clock is set by default to **Internal**.
- When using the **on-probe oscillator** or the **external TTL clock source** from

your application, you can select either:

- a fixed 20 MHz reference, which is the **Internal** option in the Timestamp Clock dialog box.
- or the ST9+ OSCIN source—in this case, you must check the **External** radio button of the dialog box shown in *Figure 12* and enter the frequency of the external OSCIN source.

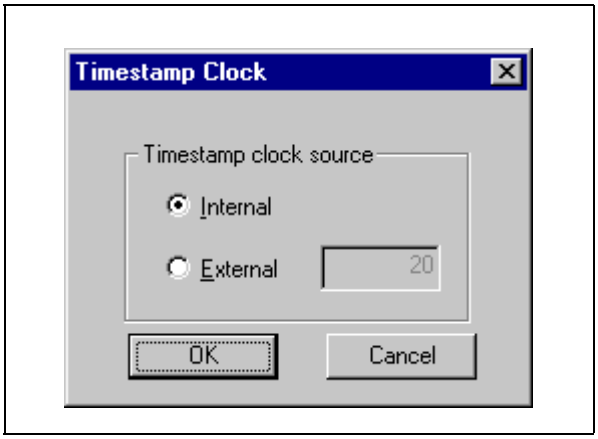


Figure 12: Timestamp clock source dialog box

3.12 Using the performance analysis

For details, please refer to ST9+ Visual Debug's on-line help.

Note, however, that the performance analysis uses both Advanced Breakpoints and the Trace. As a consequence, when analysing the performance of your code you can use neither the Advance Breakpoints feature nor the Trace. When the performance analysis is finished, your Advanced Breakpoint and Trace configurations will be restored.

3.13 Configuring peripherals behavior

Some peripherals can be configured so that they are stopped, or continue to run, when the program stops during a ST9+ Visual Debug session (i.e. when a breakpoint is encountered, etc.).

In ST9+ Visual Debug, configuring peripherals is be done by selecting, from the main menu, ***Emulator>Peripheral Control...*** The **Peripherals Control** window will display how the emulated device's peripherals will behave (i.e. either run or stop) when the program execution is stopped.

By default, the peripherals are configured to Run—so that they will continue to run even if the program execution is stopped.

For the ST92F150, six control bits are available for this functionality, so the following six individual selections have been chosen for configuration:

- **Watchdog Timer** - When this peripheral is configured as standard timer, the Run option allows the timer to continue to run even when the program execution has stopped. The Stop option will freeze the timer when the program is stopped. However, when configured as watchdog, this timer is automatically disabled whenever the program is stopped, regardless of the option chosen.
- **Multifunction Timer 0**- If the Stop option is chosen, timer counting is disabled when the program is stopped.
- **Multifunction Timer 1** - If the Stop option is chosen, timer counting is disabled when the program is stopped.
- **Extended Function Timer 0** - If the Stop option is chosen, timer counting is disabled when the program is stopped.
- **Extended Function Timer 1 / CAN 0** - If the Stop option is chosen, both the timer counting and the CAN 0 is disabled when the program is stopped.
- **J1850 / CAN 1** - If the Stop option is chosen, both the timer counting and the CAN 0 is disabled when the program is stopped.

Note: The A/D conversion is always stopped when the program is stopped by the window debugger. It restarts when the user program runs again.
The Watchdog Timer is always stopped while the program is stopped by the debugger. It continues running when your program runs again.

4 EMULATOR FUNCTIONAL DISCREPANCIES

4.1 Extended function timers

The two Extended Function Timers work correctly on the emulator but the related registers cannot be read using the **Registers Windows**, nor through the **Watch Window**.

It is advised not to display the alternate counter contents using windows debugger. If required, a workaround can be done:

When you need to verify the value of the counter, add in your program instructions that copy the alternate counter contents in two working registers (one for the Alternate Counter High byte, one for the Alternate Counter Low byte) (remember that the MSB must be read first).

Display the working registers window to check the counter values.

4.2 HW0SW1 pin option

A 10 KΩ pull-up resistor has been added to the emulator. So, when the emulator is not connected to your application board, the default option is the software watchdog reset.

On your application board, if you want to force a hardware watchdog reset, we advise you to connect the HW0SW1 pin directly to GND.

4.3 Protection register and emulation chip life

The emulator uses the OTP area in the same way that the target MCU does (*or will do, when finalized*). This means that this register can be written to only once.

Your emulator uses a chip similar to the target MCU. For the FLASH and EEPROM memories, this means that write cycles in these areas are limited. If, after a while, writing to memory is no longer possible for the FLASH or EEPROM areas, the BGA144 emulation chip will have to be changed. (Note that loading application software in system memory does not write to the FLASH area at all).

APPENDIX A: EMC CONFORMITY AND SAFETY REQUIREMENTS

This emulator respects the EMC requirements of the European guideline 89/336/ EEC under the following conditions:

- Any tester, equipment, or tool used at any production step or for any manipulation of semi-conductor devices have its shield connected to ground.
- All ferrites provided with the emulatorkit must be attached as described in the hardware installation instructions of the relevant user manual(s).
- Your emulator must be placed on a conductive table top, made of steel or clean aluminum, grounded through a ground cable.
  All manipulation of finished goods must be made at such a grounded worktable.
- The worktable must be free of all non-antistatic plastic objects.
- It is recommended that you wear an antistatic wrist or ankle strap, connected to the antistatic floor covering or to the grounded equipment.
- If no antistatic wrist or ankle strap is worn, before each manipulation of the powered-on emulator, you must touch the surface of the grounded worktable.
- It is recommended that antistatic gloves or finger coats be worn.
- It is recommended that nylon clothing be avoided while performing any manipulation of parts.

APPENDIX B: TROUBLESHOOTING

B.1 Emulator reset

This command performs a reset of all emulator configurations such as mapping, breakpoints, trace, timestamp and ST9+ device. It should be used only when a problem is suspected with the emulator itself or if the emulator was powered off without closing the debugger. The emulator reset behaves like a hardware reset of the emulator. The application loaded in the debugger is automatically closed.

B.2 ST9+ Visual Debug cannot connect to the emulator

Verify that the whole installation has been performed correctly, that is:

- Connect the parallel cable to the emulator,
- Connect the three flat cables between "ST9 HDS2V2 emulator" and ST92F150 emulator probe,
- Connect the power supply cable to the mainboard,
- Verify that emulator is switched off
- Connect the emulator to your application board,
- Switch on the emulator,
- Switch on the application board,
- Verify that ST9+ selected clock is running.
- If ST9+ Visual Debug still cannot connect to the emulator, check that you selected the appropriate parallel port. If the parallel cable is not connected to LPT1 but to LPT2, change the port name (refer to the ST9+Visual Debug's on-line help).
- Make sure that:
  - You are using the interface cable delivered. Do not use a longer cable.
  - You have not connected the cable to a serial interface of the PC, this could damage the emulator.
  - The cable is connected directly to the DB-25 female connector of the PC parallel port. No additional cables nor switchboxes between the PC and the board should be installed.
  - If the PC parallel port connector is not a DB25-type connector (e.g.: such as miniature plug or receptacle 36-pin ribbon type connector as specified in IEEE1284-1994 standard), use the adapter provided with your PC. Note that the different connector type indicates that your PC parallel port should

be IEEE-1284 compliant: check that proper mode (Centronics Compatible Mode) is installed in your PC's BIOS.

- If a dongle (hardware key required by some PC software) is already connected to the PC parallel port, it should not interfere with the emulator. However, if you notice a malfunction, you should remove the dongle(s) and try to establish the connection again.

B.3 Your emulator stays in reset mode

Your emulator is connected to your application board. When you try to run your program, you observe that the LED ST9 Reset remains switched on. When you stop your program, the program counter is pointing to the reset vector.

- Verify that your application board is powered up. When your emulator is connected to your application board, it can detect whether your application board is powered up or not. If not, the emulator will remains in reset state.
- Verify that the reset line on your application board is not pulled low. When your application board is reset, the ST9+ chip will remain in reset also, when you try to run your program.

B.4 You have problems displaying your application memory in the Memory Window

- You should not display contents of memories such as FIFO, LIFO... Otherwise you could lose the synchronisation of your memory when the display reads its contents.

  The same problems can also occur when the memory area you selected to display in the Memory window overlaps such memories.

  For example, your FIFO is located at address 0x01C000, some RAM is located at address 0x018000 to 0x01BFFF, and you display the memory space from 0x01BE00. Depending on the size of your Memory window, the display can show memory contents up to 0x01C0ff and cause problems in your FIFO.

  In that case, it is advised to take special care with the size of the Memory window.
- If you are asking for a display of external memory mapped as user memory, you have to configure the external memory interface using EMR1, EMR2, and P0Cx, P1Cx, P2Cx registers as your final software should do before asking for the dump.
- When using the Memory window, the ST9+ wait states are configured differently than when you are running your program. Two wait states on Data

Strobe are automatically selected. When running your program, the number of wait states you choose is used.

B.5 WGDB9 debugger did not download your code correctly

You have a code mapped in several segments. When loading the application within WGDB9 debugger (software toolchain V4.x), you observe WGDB9 load only your code in segment 0 but not in the other segments.

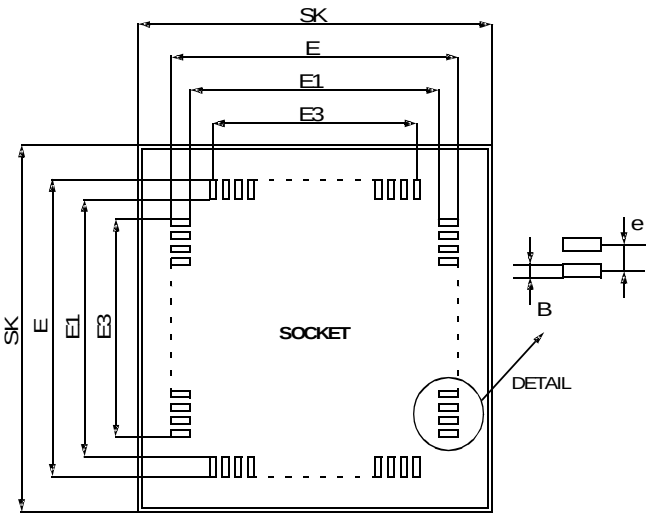
You must create a file `appli.gdb` that contains only the following line:

```
source appli.bl9
```

B.6 QFP64/TQFP64 footprint issues

Some of the emulated devices have a TQFP64 footprint and require you to solder a Yamaichi QFP64 socket onto your application board. However, be aware that you may encounter problems owing to the fact that the Yamaichi QFP64 sockets used to connect the TQFP64 device adapter require footprints that are not compatible with TQFP copper traces.

For this reason, when designing your board, plan to have a double trace compatible with both Yamaichi QFP sockets and TQFP copper traces. Specifications for compatible footprints are provided below. If you use compatible footprints, a single printed circuit board design will serve for both the development stage and the final product. When you are finished the development stage, you can simply replace the development Yamaichi socket by the programmable target device in an actual TQFP64



Dim	mm			inches		
	Min	Typ	Max	Min	Typ	Max
B	0.35	0.45	0.50	0.014	0.018	0.020
E	20.80			0.819		
E1		14.00			0.551	
E3	11.90	12.00	12.10	0.468	0.472	0.476
e	0.75	0.80	0.85	0.029	0.031	0.033
SK*		26			1.023	
	Number of Pins					
N	64 (4x16)					

* SK: Plastic socket overall dimensions.

Figure 13: TQFP64 Device and Emulation Probe Compatible Footprint

APPENDIX C: HARDWARE LAYOUTS

C.1 Probe board schematics

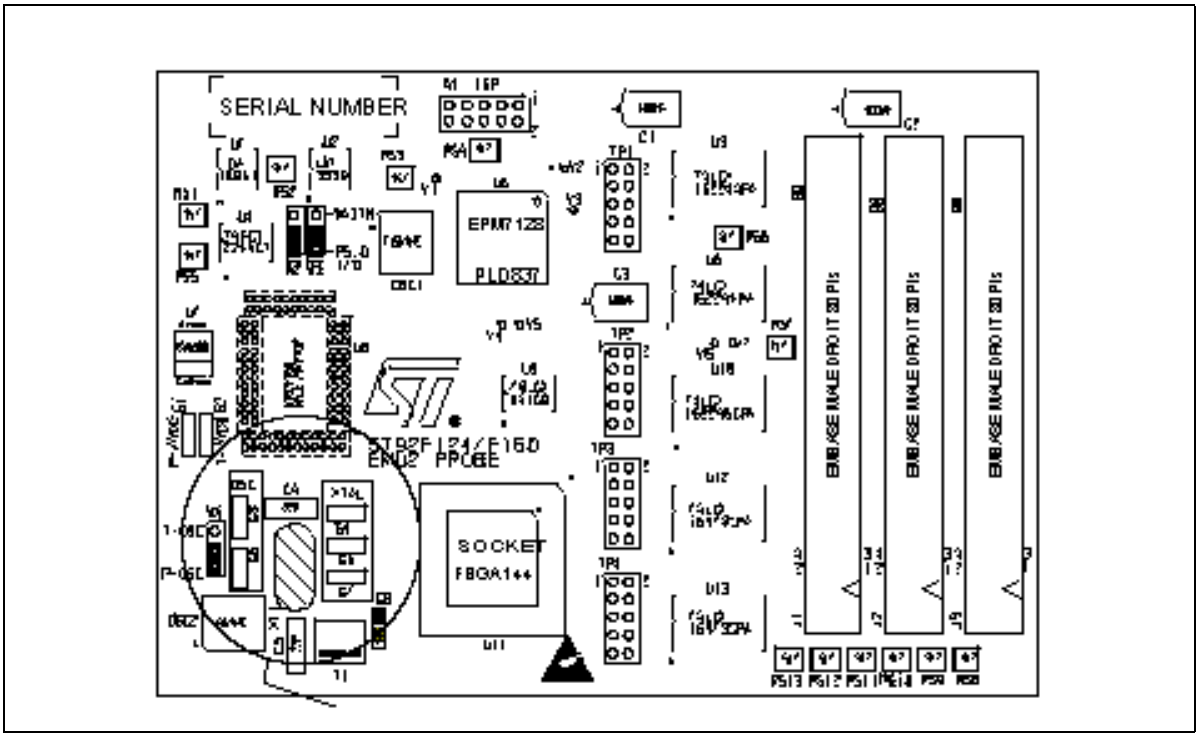


Figure 14: Probe board top-side schematic

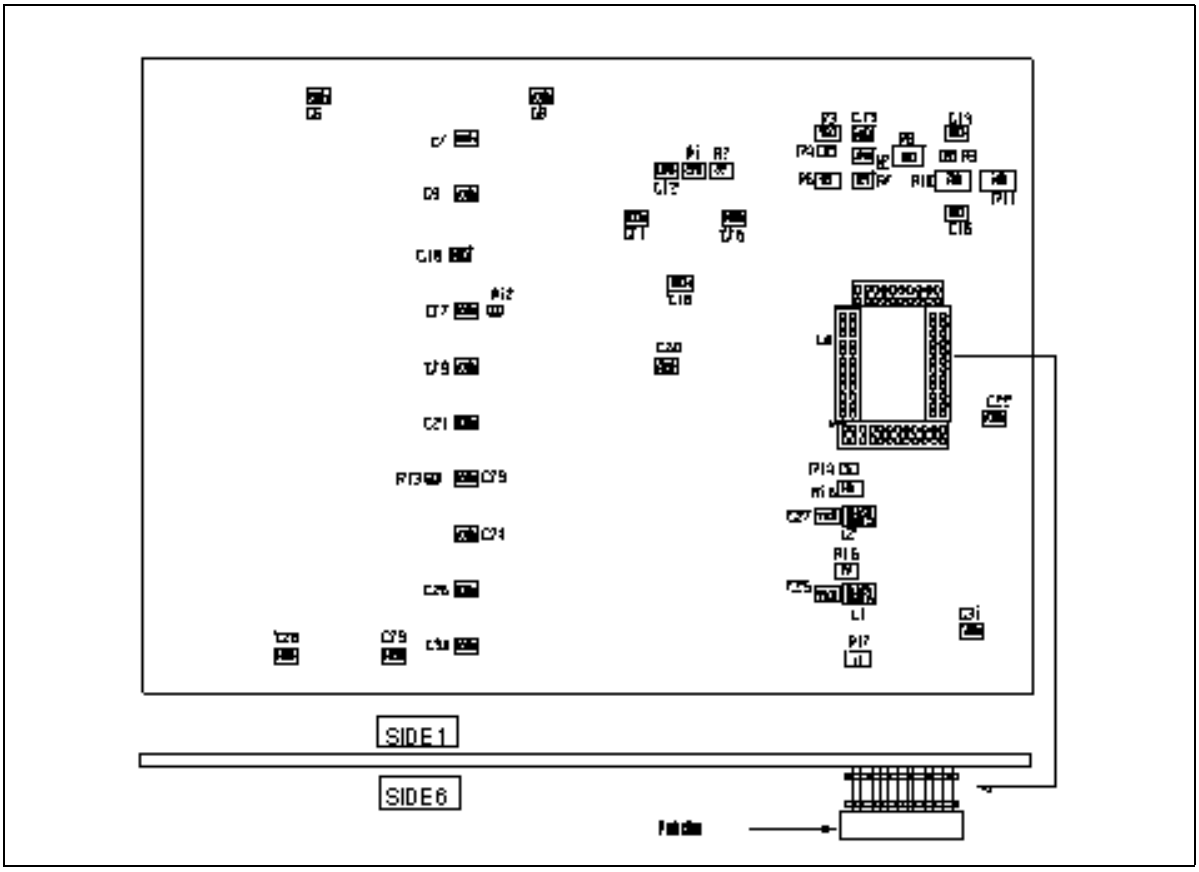


Figure 15: Probe board bottom-side schematic with side-view

C.2 Probe adapter pinout

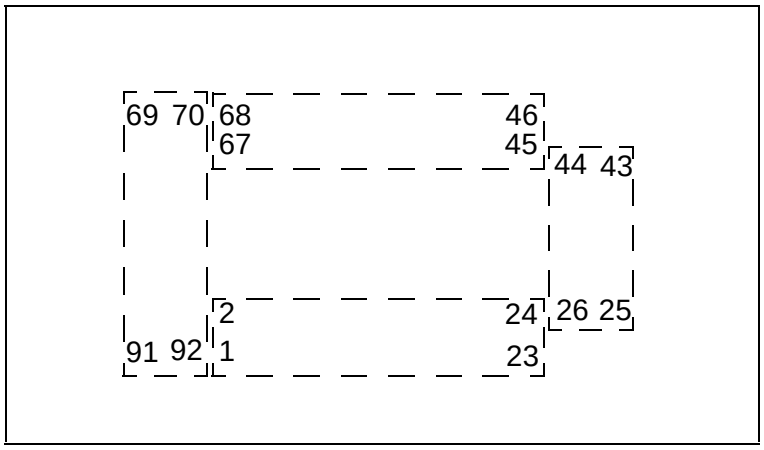


Figure 16: Probe device adapter pin placements

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	P9.3	**32**	P2.2	**63**	P6.3
2	P9.4	**33**	P2.3	**64**	P6.4
3	P9.5	**34**	P2.4	**65**	P6.5
4	P9.6	**35**	P2.5	**66**	VPWO
5	P9.7	**36**	P2.6	**67**	P8.0
6	P5.0	**37**	P2.7	**68**	P8.1
7	P5.1	**38**	VSS	**69**	P8.2
8	P5.2	**39**	VDD(target)	**70**	P8.3
9	P5.3	**40**	P1.0	**71**	P8.4
10	P5.4	**41**	P1.1	**72**	P8.5
11	P5.5	**42**	P1.2	**73**	P8.6
12	P5.6	**43**	RXCAN	**74**	P8.7
13	P5.7	**44**	TXCAN	**75**	AVDD(target)
14	P4.0	**45**	P1.3	**76**	VSS
15	P4.1	**46**	P1.4	**77**	P7.0
16	P4.2	**47**	P1.5	**78**	P7.1
17	P4.3	**48**	P1.6	**79**	P7.2
18	P4.4	**49**	P1.7	**80**	P7.3
19	P4.5	**50**	DSn	**81**	P7.4
20	P4.6	**51**	ASn	**82**	P7.5
21	P4.7	**52**	P0.0	**83**	P7.6
22	P3.1	**53**	P0.1	**84**	P7.7
23	P3.2	**54**	P0.2	**85**	VSS
24	P3.3	**55**	P0.3	**86**	VDD(target)
25	P3.4	**56**	P0.4	**87**	OSCIN(target)
26	P3.5	**57**	P0.5	**88**	RESETn
27	P3.6	**58**	P0.6	**89**	HW0SW1
28	P3.7	**59**	P0.7	**90**	P9.0
29	RWN	**60**	P6.0	**91**	P9.1
30	P2.0	**61**	P6.1	**92**	P9.2
31	P2.1	**62**	P6.2		

Table 7: Probe adapter pinout

C.3 Emulator hardware schematics

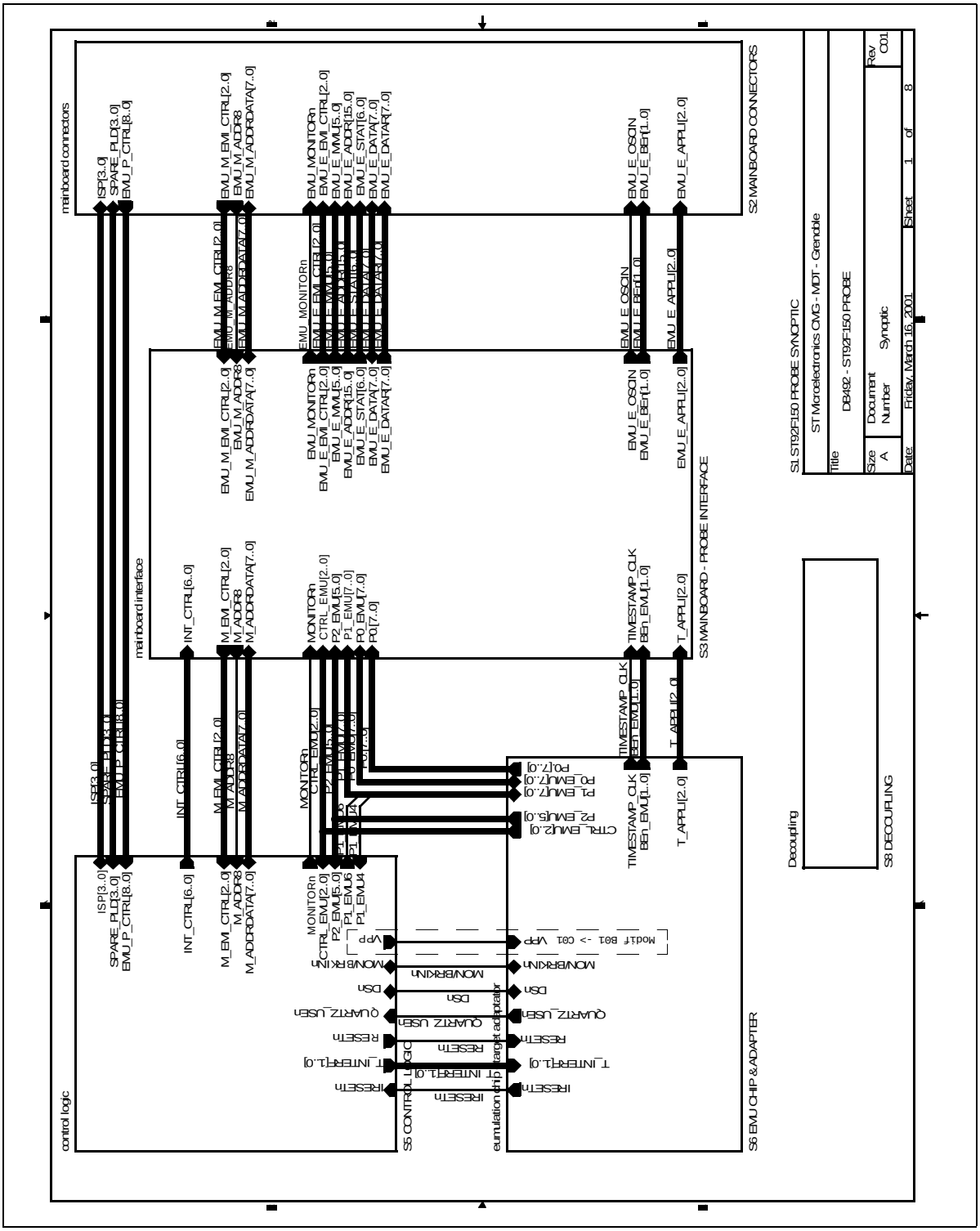


Figure 17: Probe synoptic schematic

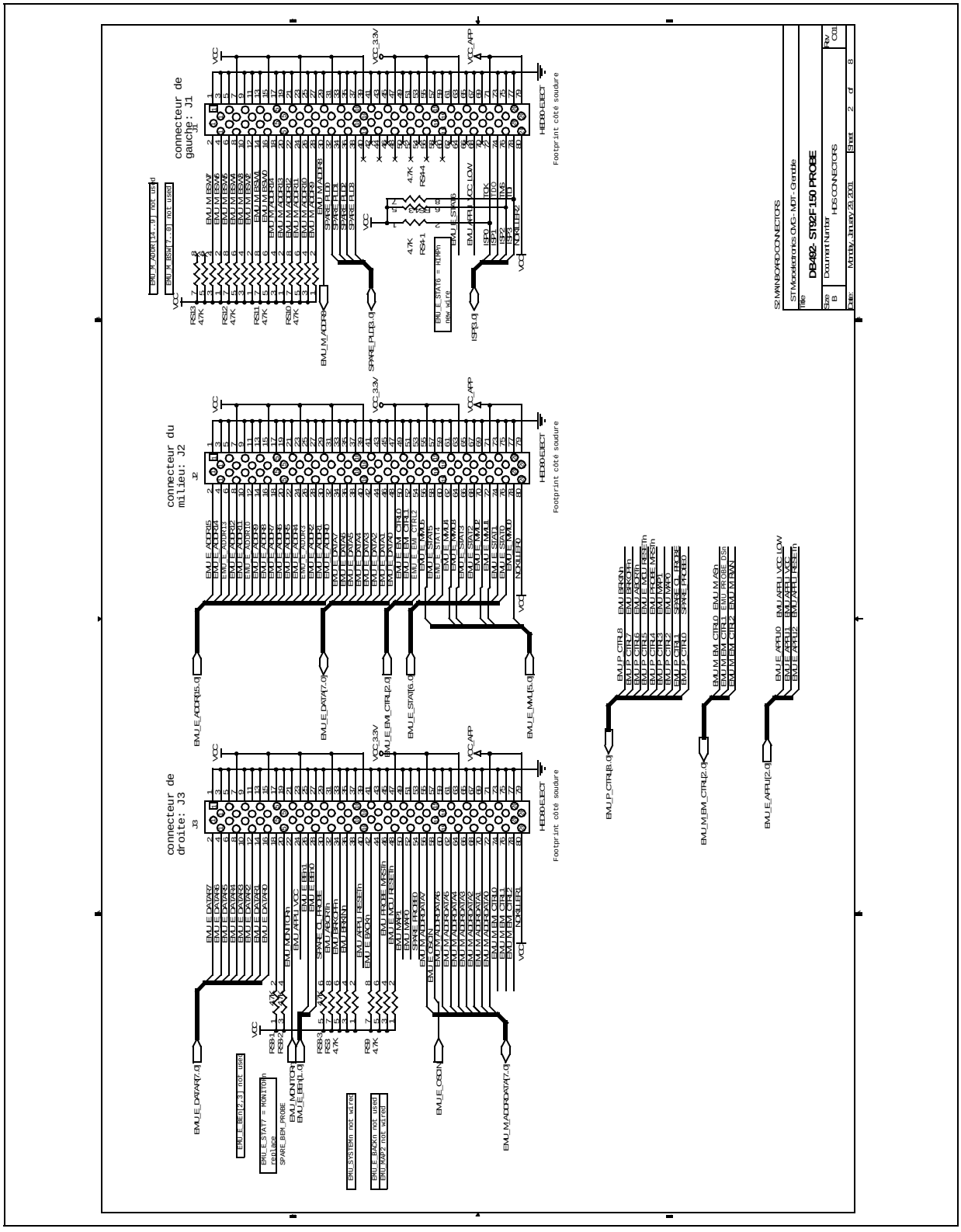


Figure 18: Probe/mainboard connectors

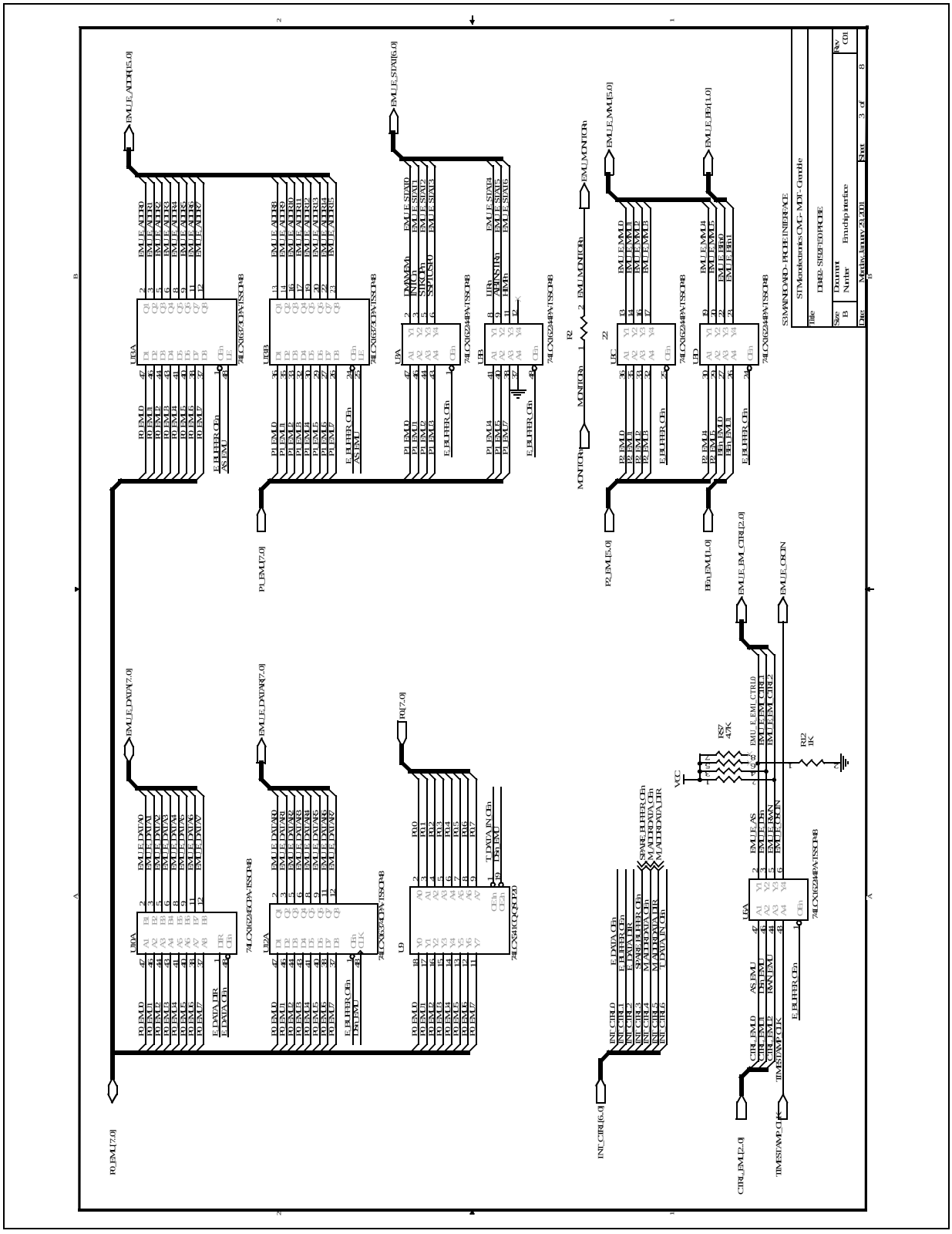


Figure 19: Mainboard/probe interface schematic

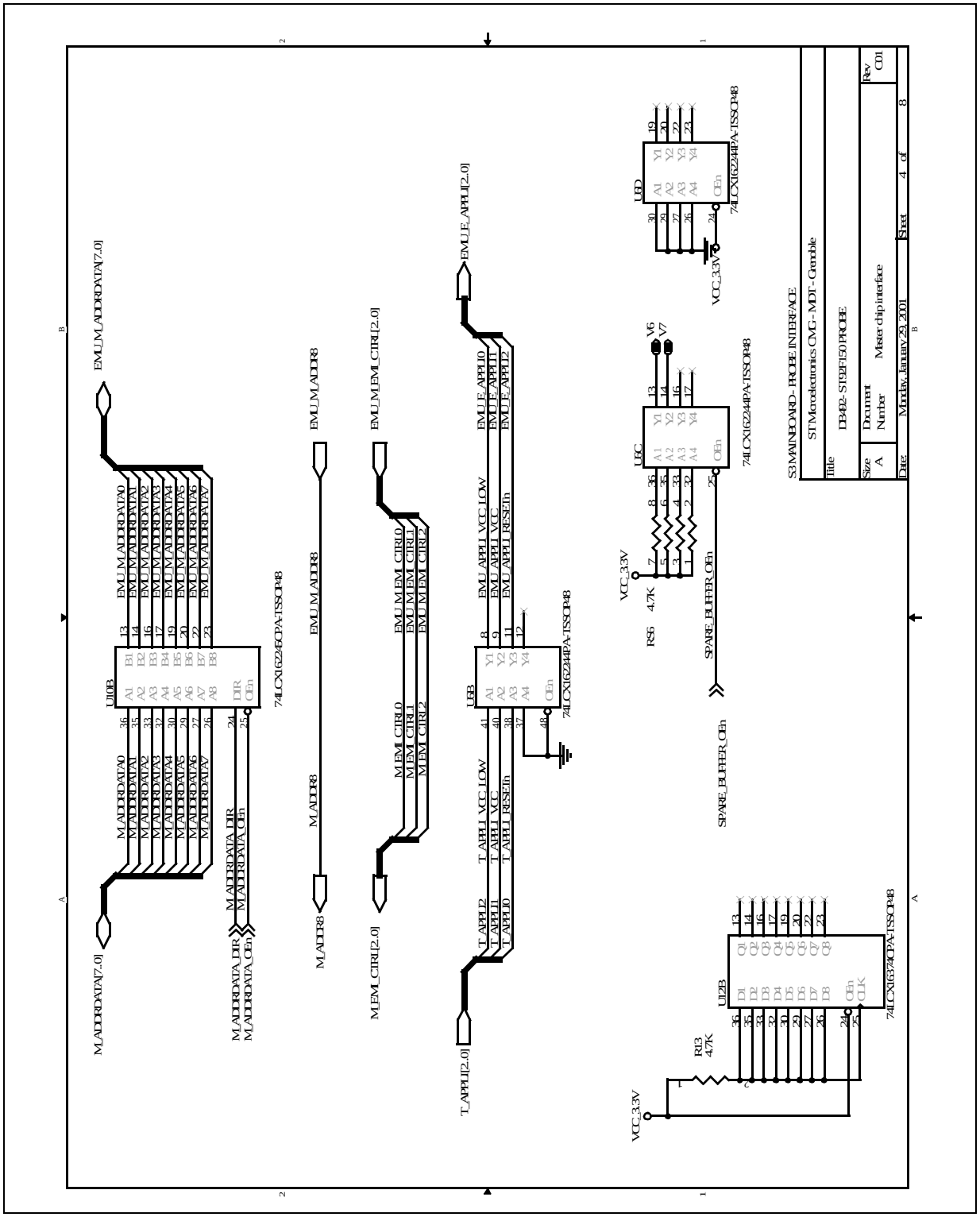


Figure 20: Mainboard/probe interface

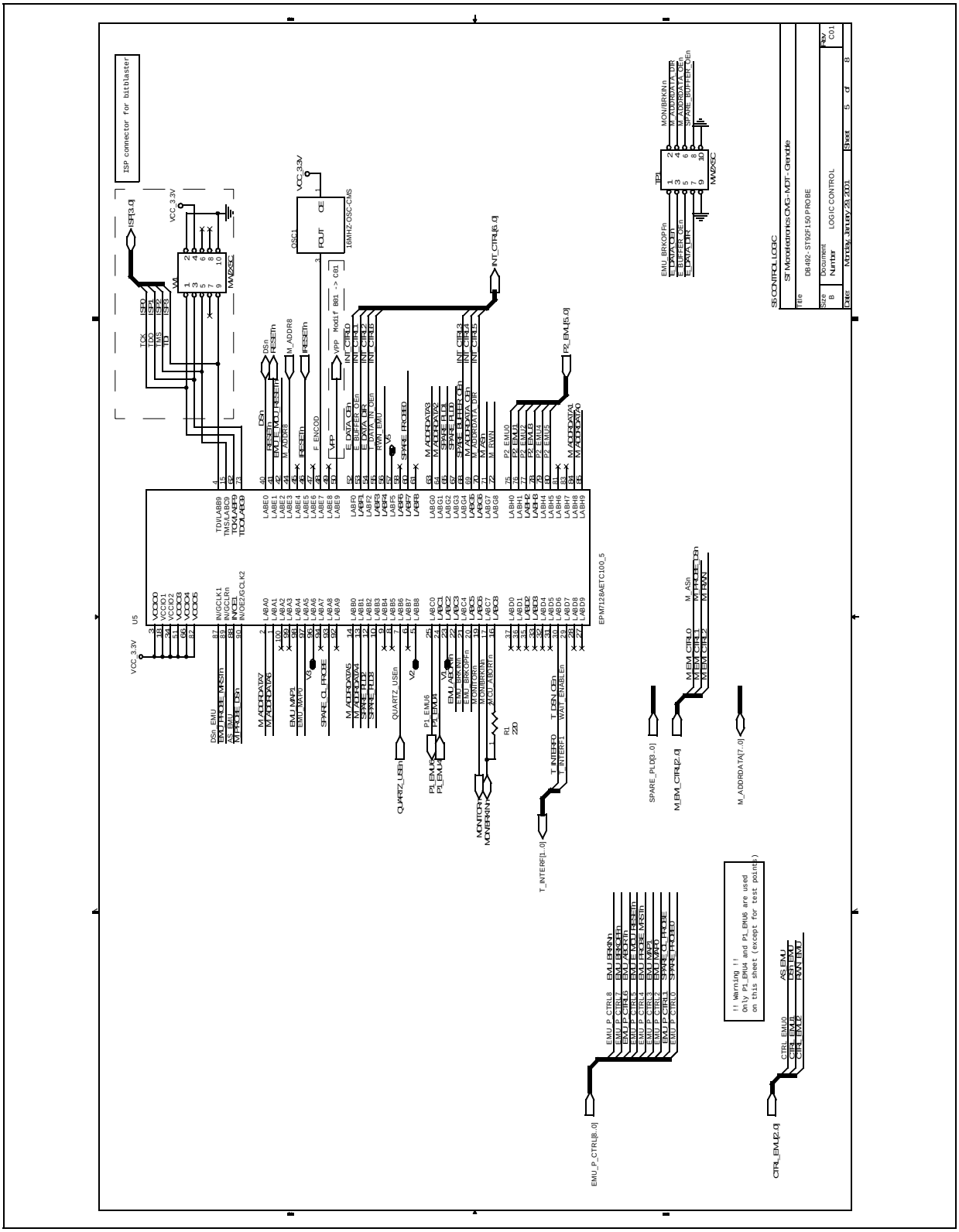


Figure 21: Control logic schematic

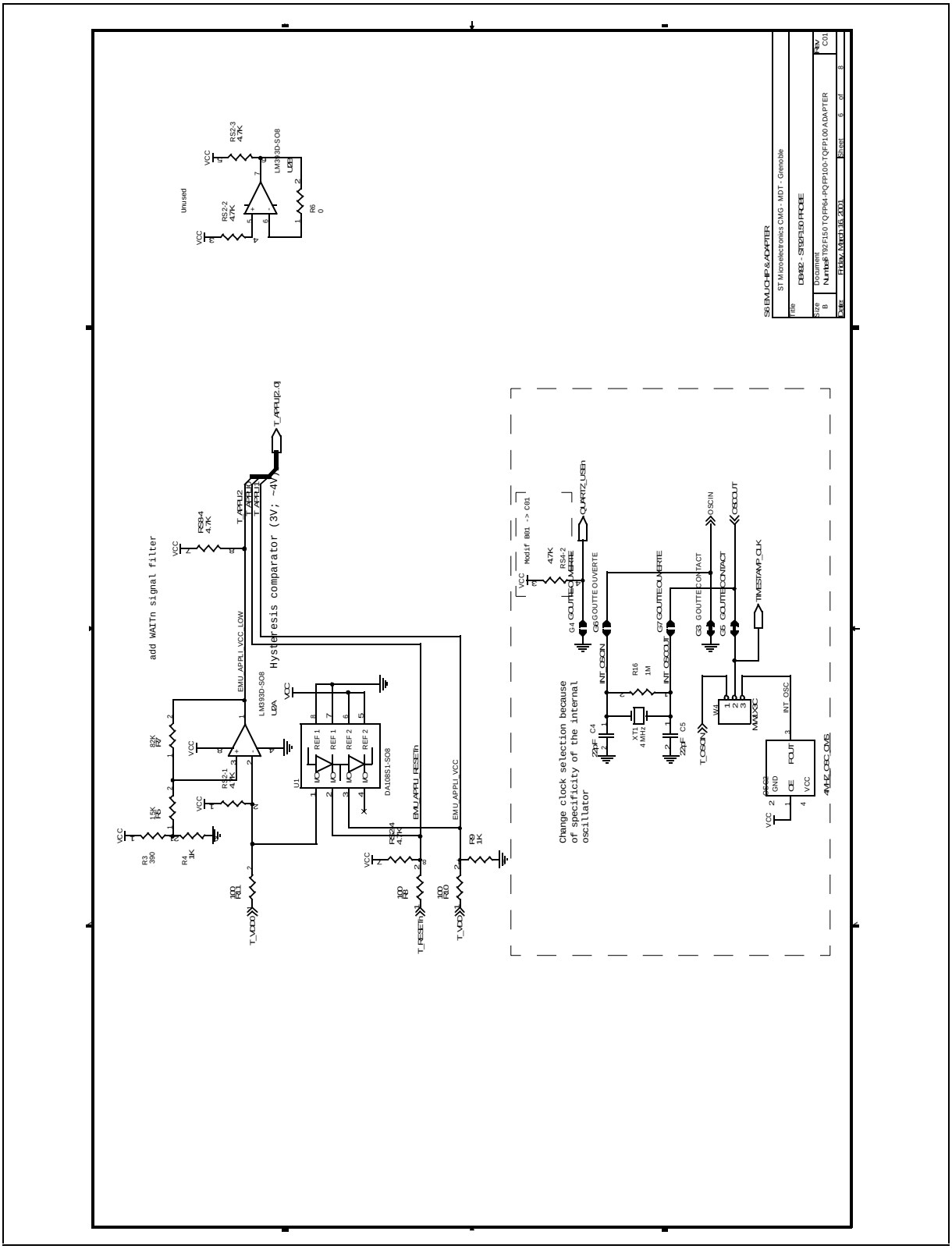


Figure 22: Emu chip and adapter schematic

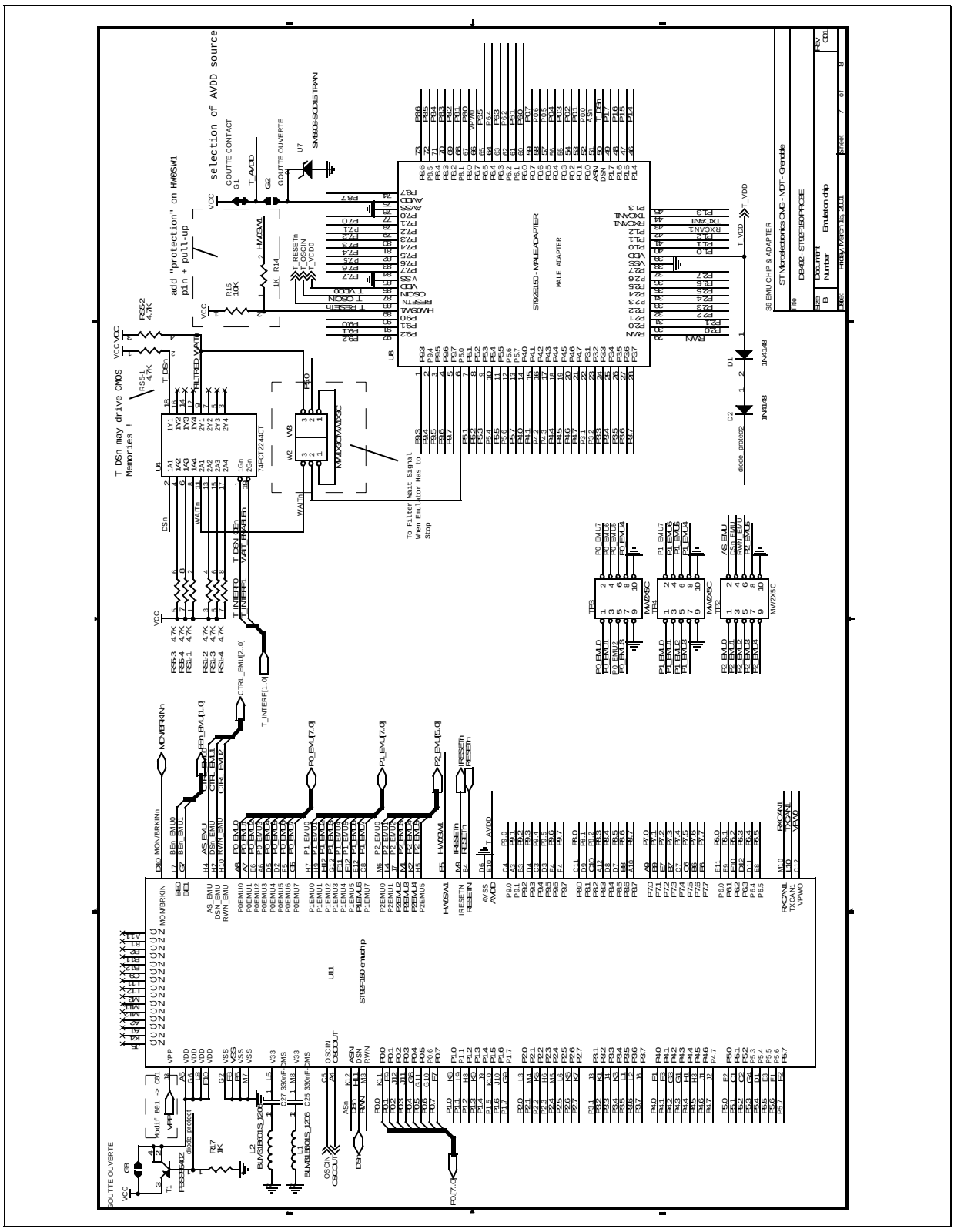


Figure 23: Emu chip and adapter schematic

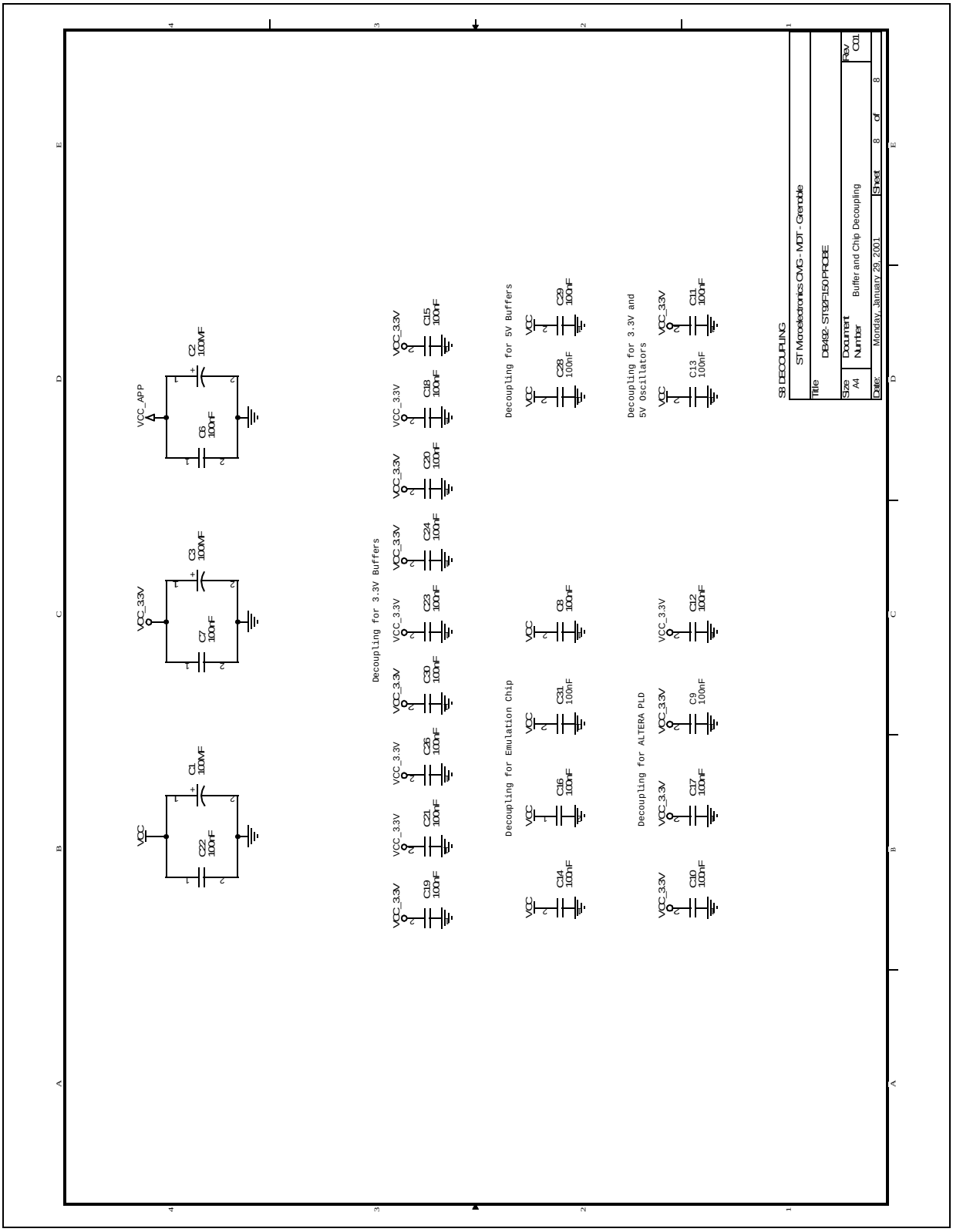


Figure 24: Decoupling schematic

PRODUCT SUPPORT

If you experience any problems with this product or if you need spare parts or repair, contact the distributor or ST sales office where you purchased the product.

Getting prepared before you call

Collect the following information about the product before contacting ST or your distributor:

1 Name of the company where you purchased the emulator kit.

2 Date of purchase.

3 Order Code: Refer to the side of your emulator kit box. The order code will depend on the region for which it was ordered (i.e. the UK, Continental Europe or the USA).

4 Serial Number: The serial number is located on the rear panel of the emulator box.

5 Target Device: The sales type of the ST9 microcontroller you are using in your development.

Contact list

*Note: For **American and Canadian customers** seeking technical support the US/Canada is split in 3 territories. According to your area, contact the following sales office and ask to be transferred to an 8-bit microcontroller Field Applications Engineer (FAE).*

Canada and East Coast

STMicroelectronics
Lexington Corporate Center
10 Maguire Road, Building 1, 3rd floor
Lexington, MA 02421
Phone: 781-402-2650

Mid West

STMicroelectronics
1300 East Woodfield Road, Suite 410
Schaumburg, IL 60173
Phone: 847-517-1890

West coast

STMicroelectronics, Inc.
30101 Agoura Court
Suite 118
Agoura Hills, CA 91301
Phone: 818-865-6850

Europe

France (33-1) 47407575
Germany (49-89) 460060
U.K. (44-1628) 890800

Asia/Pacific Region

Japan (81-3) 3280-4120
Hong-Kong (852) 2861 5700
Sydney (61-2) 9580 3811
Taipei (886-2) 2378-8088

Software updates

You can get software updates from the ST Internet web site **http://mcu.st.com**. For information on firmware and hardware revisions, call your distributor or ST using the contact list given above.

Hardware spare parts

SMB/BNC adapters

You can order the following SMB/BNC adapters from **RADIALL**:

- Ref.: 191214. Adapter SMB female / BNC male.
- Ref.: 191215. Adapter SMB female/ BNC female.

For the name of a Radiall distributor close to you, visit Radiall's website at:

www.radiall.com

Yamaichi sockets

You can order additional Yamaichi QFP sockets directly from Yamaichi at:

http://www.yamaichi.de/Pu/quad_flat_pack/spec/a21-ic149.htm

Index

Index

S

T

W