inDART®-ST7C In-Circuit Debugger for STMicroelectronics ST72CXXX FLASH Devices

User's Manual

SofTec® MICROSYSTEMS

DC00326

SofTec Microsystems

E-mail (general information): info@softecmicro.com

E-mail (marketing department): marketing@softecmicro.com

E-mail (technical support): support@softecmicro.com

Web: http://www.softecmicro.com

Written by Paolo Xausa

Contents

1. Overview

What is inDART-ST7C?

inDART-ST7C is a powerful entry-level tool for STMicroelectronics ST7-based systems. inDART-ST7C takes advantage of STMicroelectronics' STVD7 (STMicroelectronics Visual Debug) Integrated Development Environment and the ISP (In Situ Programming) feature to program the FLASH memory of the ST7 family of microcontrollers. Together with STVD7, inDART-ST7C provides you with everything you need write, compile, download, in-circuit emulate and debug user code. Full speed program execution allows you to perform hardware and software testing in real time. inDART-ST7C is connected to the host PC through a parallel port, while the 10-pin probe of the product fits into the target's standard ISP connector. On Design Kits packages, a full-featured experiment board for a specific ST7 microcontroller is also included.

inDART-ST7C offers you the following benefits:

- Real-time code execution without probes–works with all packages;
- In-circuit debugging;
- Built-in FLASH programmer;
- 3.3 V to 5.0 V operating voltage;
- Operation from voltage supplied by the target application;
- Standard chip used–no bondouts, 100% electrical characteristics guaranteed;
- Working frequency up to the microcontroller's maximum;
- STMicroelectronics Visual Debug user interface (the same user interface of all STMicroelectronics ST7 tools), with integrated C compiler and assembler and source level and symbolic debugging.

Naming Conventions

Throughout this manual, the following naming conventions are used:

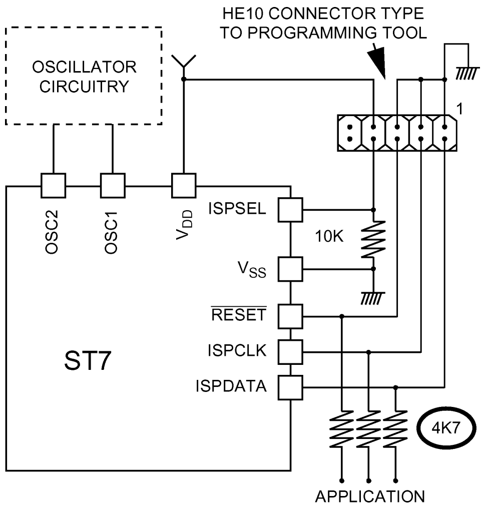
- **inDART-ST7C** is the inDART in-circuit debugger (specific for the STMicroelectronics ST72CXXX devices);

- **inDART-ST7 user interface** is the user interface common to all of the inDART-ST7 series in-circuit debuggers.

What is In Situ Programming (ISP)?

The ISP feature allows you to update the content of FLASH program memory when the chip is already plugged on the application board. ISP programming uses a serial protocol to interface a programming tool like inDART. The ISP feature can be implemented with a minimum number of added components and board area impact.

inDART-ST7C uses the standard, 10-pin ST7 ISP connector to program and in-circuit emulate the target device. You must therefore provide such connector (see the diagram below) on your target board.



The ST7 ISP Interface

ISP Mode Pin Name	ISPDATA	ISPCLK	ISPSEL	RESET	V_{SS}	V_{DD}
ISP Connector Pin #	2	4	8	6	1, 3, 5	7

The ST7 ISP Connector

Supported Devices

inDART-ST7C currently supports all EEPROM-like ST7 FLASH devices, in all packages.

Device	Package	Notes
ST72C104G1B	SDIP32	ST7 FLASH MCUs (FLASH=4Kx8)
ST72C104G1M	SOIC28	ST7 FLASH MCUs (FLASH=4Kx8)
ST72C104G2B	SDIP32	ST7 FLASH MCUs (FLASH=8Kx8)
ST72C104G2M	SOIC28	ST7 FLASH MCUs (FLASH=8Kx8)
ST72C124J2B	SDIP42	ST7 FLASH MCUs (FLASH=8Kx8)
ST72C124J2T	TQFP44	ST7 FLASH MCUs (FLASH=8Kx8)
ST72C171K2B	SDIP32	ST7 FLASH MCUs (FLASH=8Kx8)
ST72C171K2M	SOIC34	ST7 FLASH MCUs (FLASH=8Kx8)
ST72C215G2B	SDIP32	ST7 FLASH MCUs (FLASH=8Kx8)
ST72C215G2M	SOIC28	ST7 FLASH MCUs (FLASH=8Kx8)
ST72C216G1B	SDIP32	ST7 FLASH MCUs (FLASH=4Kx8)
ST72C216G1M	SOIC28	ST7 FLASH MCUs (FLASH=4Kx8)
ST72C254G1B	SDIP32	ST7 FLASH MCUs (FLASH=4Kx8)
ST72C254G1M	SOIC28	ST7 FLASH MCUs (FLASH=4Kx8)
ST72C254G2B	SDIP32	ST7 FLASH MCUs (FLASH=8Kx8)
ST72C254G2M	SOIC28	ST7 FLASH MCUs (FLASH=8Kx8)
ST72C314J2T	TQFP44	ST7 FLASH MCUs (FLASH=8Kx8)
ST72C314J4B	SDIP42	ST7 FLASH MCUs (FLASH=16Kx8)
ST72C314J4T	TQFP44	ST7 FLASH MCUs (FLASH=16Kx8)
ST72C314N2B	SDIP56	ST7 FLASH MCUs (FLASH=8Kx8)
ST72C314N2T	TQFP64	ST7 FLASH MCUs (FLASH=8Kx8)
ST72C314N4B	SDIP56	ST7 FLASH MCUs (FLASH=16Kx8)
ST72C314N4T	TQFP64	ST7 FLASH MCUs (FLASH=16Kx8)
ST72C334J2B	SDIP42	ST7 FLASH MCUs (FLASH=8Kx8, EEPROM=256x8)
ST72C334J2T	TQFP44	ST7 FLASH MCUs (FLASH=8Kx8, EEPROM=256x8)
ST72C334J4B	SDIP42	ST7 FLASH MCUs (FLASH=16Kx8, EEPROM=256x8)
ST72C334J4T	TQFP44	ST7 FLASH MCUs (FLASH=16Kx8, EEPROM=256x8)
ST72C334N2B	SDIP56	ST7 FLASH MCUs (FLASH=8Kx8, EEPROM=256x8)
ST72C334N2T	TQFP64	ST7 FLASH MCUs (FLASH=8Kx8, EEPROM=256x8)
ST72C334N4B	SDIP56	ST7 FLASH MCUs (FLASH=16Kx8, EEPROM=256x8)
ST72C334N4T	TQFP64	ST7 FLASH MCUs (FLASH=16Kx8, EEPROM=256x8)

inDART-ST7C Supported Devices

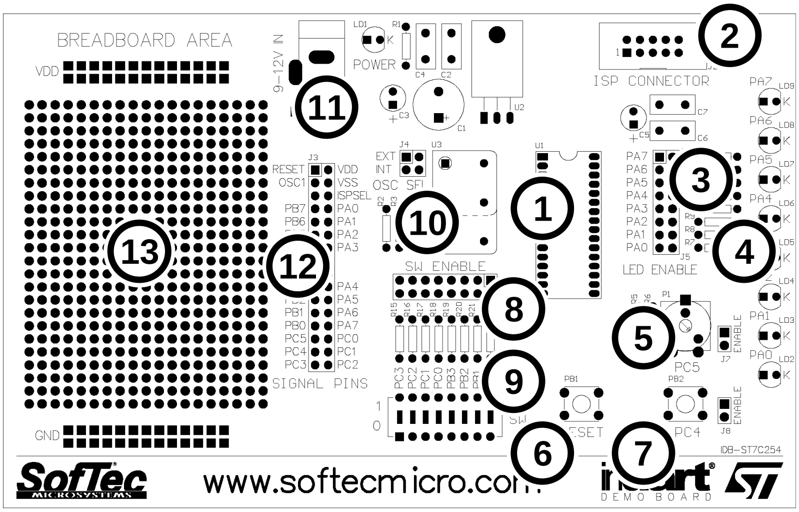
Demo Boards

On Design Kits packages, a full-featured experiment board for a specific ST7 microcontroller (ST7C254 or ST7C334) is also included. Each demo board can be used for evaluation/experiments in the absence of a target application board.

IDB-ST7C254 Demo Board

The IDB-ST7C254 Demo Board has the following hardware features:

1. An ST72C254G2B microcontroller (in SDIP32 package, already programmed with a demo application—in addition, you can also use the following microcontrollers: ST72C104G1B, ST72C104G2B, ST72C215G2B, ST72C216G1B, ST72C254G1B);
2. A standard ISP connector;
3. Eight jumpers to connect/disconnect each of the eight LEDs to/from their respective Port A pin;
4. Eight high-efficiency (low-current) LEDs connected to Port A;
5. A potentiometer, together with a jumper to connect/disconnect it to/from PC5;
6. A push-button switch connected to RESET;
7. A push-button switch, together with a jumper to connect/disconnect it to/from PC4;
8. Eight jumpers to connect/disconnect each of the eight DIP-switches to/from their respective Port A/Port B pin;
9. Eight general-purpose DIP switches (four connected to PB and four connected to Port C);
10. A provision for an external oscillator, together with a jumper to select the microcontroller's internal oscillator or the external oscillator;
11. A connector for a 9-12V, 100 mA power supply;
12. A connector area to access the I/O pins of the microcontroller for expansion prototyping;
13. A prototyping area.



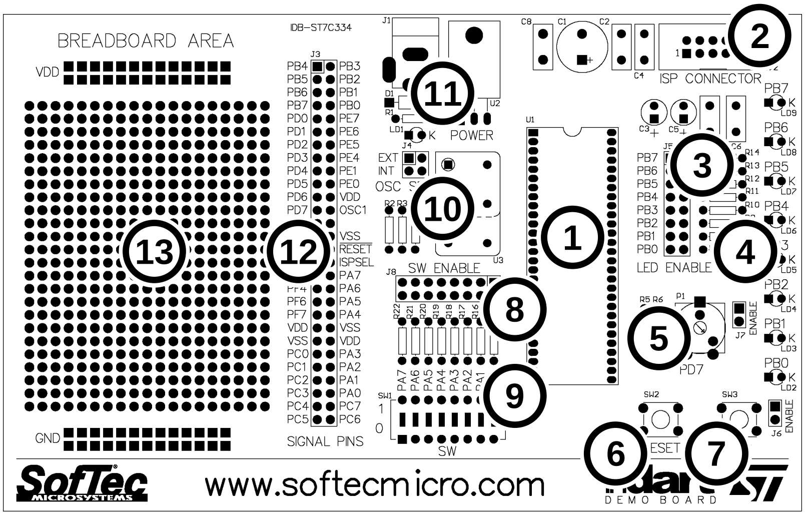
The IDB-ST7C254 Demo Board

IDB-ST7C334 Demo Board

The IDB-ST7C334 Demo Board has the following hardware features:

1. An ST72C334N4B microcontroller (in SDIP56 package, already programmed with a demo application—in addition, you can also use the following microcontrollers: ST72C314N2B, ST72C314N4B, ST72C334N2B);
2. A standard ISP connector;
3. Eight jumpers to connect/disconnect each of the eight LEDs to/from their respective Port B pin;
4. Eight high-efficiency (low-current) LEDs connected to Port B;
5. A potentiometer, together with a jumper to connect/disconnect it to/from PD7;
6. A push-button switch connected to RESET;
7. A push-button switch, together with a jumper to connect/disconnect it to/from PD0;
8. Eight jumpers to connect/disconnect each of the eight DIP-switches to/from their respective Port A pin;
9. Eight general-purpose DIP switches connected to Port A;

10. A provision for an external oscillator, together with a jumper to select the microcontroller's internal oscillator or the external oscillator;
11. A connector for a 9-12V, 100 mA power supply;
12. A connector area to access the I/O pins of the microcontroller for expansion prototyping;
13. A prototyping area.



The IDB-ST7C334 Demo Board

STVD7 Integrated Development Environment

The inDART-ST7 user interface is based on the ST7 Visual Debug Integrated Development Environment (STVD7). STVD7 enables programs to be executed and stopped where desired, while viewing the memory contents. It offers the ability to step through and examine code at the C source level and the Assembly instruction level. You can introduce breakpoints and run or single-step the executable, while viewing the source and observing current program values. All registers and memory locations are accessible for both read and write operations. This documentation covers the basic setup and operation of the STVD7, but it does not cover all of its functions. For further information, please refer to the STVD7 on-line help.

Hiware and Cosmic Demo Versions

These two third-party companies have developed a C compiler for use with ST7 microcontrollers. A demo version of each compiler can be directly installed on the PC using the STMicroelectronics “MCU ON CD” CD-ROM.

Recommended Reading

This documentation describes how to use inDART together with the STVD7 Integrated Development Environment. Additional information can be found on the following documents:

- **STVD7 On-line Help**—The ST7 Visual Debug on-line help.
- **AST7-LST7.PDF**— This user’s guide describes how to use the STMicroelectronics assembler, linker, formatter and librarian for the ST7 family.
- **ST7 FAMILY 8-BIT MCUs Programming Manual**—Programming reference containing the description of the full ST7 instruction set.
- **ST7 Family Data Sheets**—The complete set of device data sheets can be found on the “MCU ON CD” CD.
- **RELEASE.TXT**—Contains notes and known problems about the STVD7 user interface.

Software Upgrades

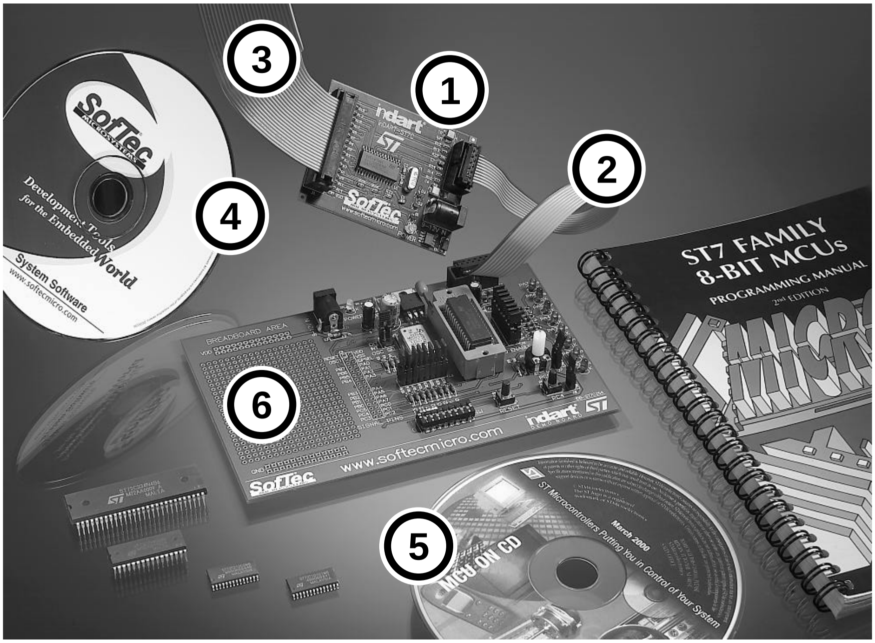
The latest version of the inDART-ST7 user interface is always available for free at our download page on the web:
http://www.softecmicro.com/download.html.

2. Getting Started

inDART-ST7C Components

The inDART-ST7C package includes the following items:

1. The inDART-ST7C in-circuit debugger;
2. A 20-cm, 10-conductor ISP cable;
3. A parallel cable;
4. The inDART-ST7 CD-ROM;
5. The STMicroelectronics “MCU ON CD” CD;
6. On Design Kits packages, a full-featured experiment board for a specific ST7 microcontroller (ST7C254 or ST7C334) is also included.



The inDART-ST7C package

Host System Requirements

The inDART-ST7C in-circuit debugger is controlled by a PC user interface running under Windows. The following hardware and software is required to run the inDART-ST7 user interface:

- An Intel Pentium 100 or above running Windows 95, Windows 98, Windows 2000 or Windows NT;
- 32 MB of random-access memory (RAM) plus 20 MB of available disk space.

Installing the Hardware

The inDART-ST7C in-circuit debugger is connected through the parallel port to a PC which runs the inDART-ST7 user interface as explained later. Connection steps are numbered below in the recommended flow order:

1. Turn off the PC;
2. Insert the male D-Sub connector of the parallel cable into a free PC parallel interface (LPT1 or LPT2);
3. Insert the other end of the parallel cable into the LPT connector on the inDART-ST7C board;
4. Insert one end of the ISP cable into the ISP connector on the inDART-ST7C board;
5. Insert the other end of the ISP cable into the ISP connector of the demo board or target application;
6. Turn on the PC;
7. Turn on the power to the demo board/target application which also powers the inDART-ST7C board.

Note: *inDART-ST7C can be powered by the demo board/target application via the ISP cable. In this case, the power supply is taken from the Vcc pin of the ISP connector, and MUST BE 5 V, 10 mA. In the case your target application is not able to provide the required voltage and current, you must power the inDART-ST7C board via its power connector. inDART-ST7C uses a 9-12 V DC, 10 mA (unregulated) wall plug-in power supply with a 2.1 mm pin and sleeve plug with positive in the center and sleeve as ground.*

Installing the Software

The inDART-ST7 user interface setup program is located on the SofTec Microsystems "System Software" CD-ROM provided with the instrument. The setup program will copy the required files to your hard drive. Additionally, an uninstall program will be copied, giving you the option to uninstall the inDART-ST7 user interface at any time.

To install the inDART-ST7 user interface:

1. Insert the **"System Software"** CD-ROM into your computer's CD-ROM drive.
2. A startup window should automatically appear (if the startup window doesn't appear automatically, manually run the **`Setup.exe`** file located on the CD-ROM root). Choose **"Install Instrument Software"** from the main menu.
3. A list of available software should appear. Click on the **"Install inDART-ST7 Toolchain"** option.
4. Follow the on-screen instructions.

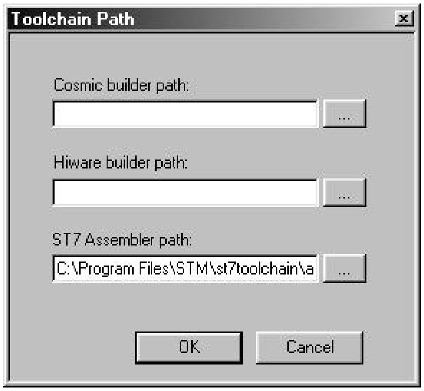
Note: *if you are installing the inDART-ST7 user interface from Windows NT, you must have logged in as Administrator.*

Application Tutorial

This section will provide a step-by-step guide on how to launch your first inDART-ST7 project and get started with the inDART-ST7 user interface. The example provided requires that inDART-ST7 be used with the IDB-ST7C254 Demo Board or the IDB-ST7C334 Demo Board. The sample application configures the A/D peripheral to convert on the A/D channel connected to the potentiometer and displays the results on the LEDs.

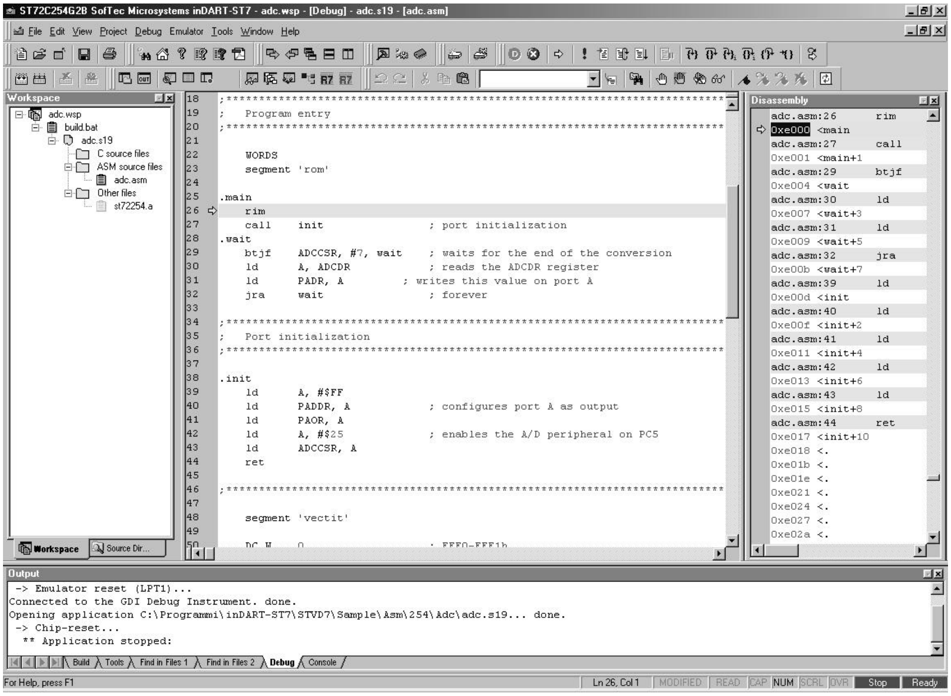
Note: *this tutorial is based on an Assembly example. Additional examples in C (based on the Cosmic C compiler) are also provided.*

- Ensure that inDART-ST7 is connected to the PC (via the parallel cable), to the demo board (via the ISP) connector, and that the demo board is powered on.
- Make sure that the “OSC SEL” jumper on the demo board selects the “INT” mode.
- Make sure that all of the “LED ENABLE” jumpers and the “POTENTIOMETER ENABLE” jumpers are inserted.
- Start the inDART-ST7 user interface by selecting **Start > Programs > SofTec inDART-ST7 > inDART-ST7**. The first time you launch the inDART-ST7 user interface you are prompted to enter the toolchain paths to be used by STVD7’s integrated development environment. Click **“Yes”**. The following dialog box will appear.



The *Toolchain Path* Dialog Box

- This step-by-step tutorial requires that the ST7 Assembler path be defined. Make sure that the ST7 Assembler path that appears on the **Toolchain Path** Dialog Box corresponds to the location where the ST7 Assembler has been installed (please note that you may modify the toolchain path at any time from within STVD7—simply select **Project > Toolchain Paths** from the main menu to access the dialog box above). Click **"OK"**. The inDART-ST7 user interface will open.
- From the main menu, choose **File > Open Workspace**. Select the **"adc.wsp"** workspace file that is located under the **"\Program Files\inDART-ST7\STVD7\Sample\Asm\254\Adc"** directory if you are using the IDB-ST7C254 Demo Board. If you are using the IDB-ST7C334 Demo Board, look for the same file under the **"\334\Adc"** directory. Click **"Open"**.
- As the application has already been assembled and the executable file generated, from the main menu, choose **Emulator > Emulator Settings** and select the LPT port the inDART-ST7C board is connected to. Click **"OK"**.
- From the main menu, choose **Debug > Start Debugging**. The user interface will display the source code with the Program Counter pointing to the first instruction, alongside of the *Disassembly* window.



```
;*************************************************************************
;   Program entry
;*************************************************************************

    WORDS
    segment 'rom'

.main
    rim
    call    init                ; port initialization
.wait
    btjf    ADCCSR, #7, wait    ; waits for the end of the conversion
    ld      A, ADCDR            ; reads the ADCDR register
    ld      PADR, A         ; writes this value on port A
    jra     wait                ; forever

;*************************************************************************
;   Port initialization
;*************************************************************************

.init
    ld      A, #$FF
    ld      PADDR, A            ; configures port A as output
    ld      PAOR, A
    ld      A, #$25             ; enables the A/D peripheral on PC5
    ld      ADCCSR, A
    ret

;*************************************************************************


    segment 'vectit'

    DC.W    0                   ; FFE0-FFE1h
```

```
adc.asm:26      rim
0xe000 <main
adc.asm:27      call
0xe001 <main+1
adc.asm:29      btjf
0xe004 <wait
adc.asm:30      ld
0xe007 <wait+3
adc.asm:31      ld
0xe009 <wait+5
adc.asm:32      jra
0xe00b <wait+7
adc.asm:39      ld
0xe00d <init
adc.asm:40      ld
0xe00f <init+2
adc.asm:41      ld
0xe011 <init+4
adc.asm:42      ld
0xe013 <init+6
adc.asm:43      ld
0xe015 <init+8
adc.asm:44      ret
0xe017 <init+10
0xe018 <.
0xe01b <.
0xe01e <.
0xe021 <.
0xe024 <.
0xe027 <.
0xe02a <.
```

```
 -> Emulator reset (LPT1)...
Connected to the GDI Debug Instrument. done.
Opening application C:\Programmi\inDART-ST7\STVD7\Sample\Asm\254\Adc\adc.s19... done.
 -> Chip-reset...
  ** Application stopped:
```

Debugging Session Started

- From the main menu, select **Debug > Run**. The program will be executed in real-time. By rotating the potentiometer, you affect the results of the A/D conversion, and the binary value of each conversion is displayed on the LEDs.
- From the main menu, select **Debug > Stop Program**. The application will stop, and the Program Counter arrow will point to the next instruction to be executed.
- From the main menu, select **View > ST7 Registers**. A small window displaying the current value of all of the ST7 registers (Program Counter, Stack Pointer, Index Registers, etc.) will appear.
- From the main menu, select **View > Peripheral Registers**. A small window displaying the current status of all of the ST7 built-in peripherals (I/O ports, Timers, A/D converter registers, etc.) will appear.
- On the source code window, set a breakpoint on the "`ld A, ADCDR`" instruction. To do so click on the line containing that instruction and then,

from the main menu, select **Edit > Insert/Remove Breakpoint**. A solid, red circle will appear on the leftmost column indicating that the breakpoint has been set.

- From the main menu, select **Debug > Run**. The application will restart, and will automatically stop at the previously set breakpoint.
- From the main menu, select **Debug > Step Into**. This command will execute the "`ld A, ADCDR`" instruction. On the *ST7 Registers* window, you can see how that instruction affected the value contained on the Accumulator. This value is the result of the A/D conversion.
- Issue another Step Into command (**Debug > Step Into**). The Accumulator value will be displayed on the LEDs.

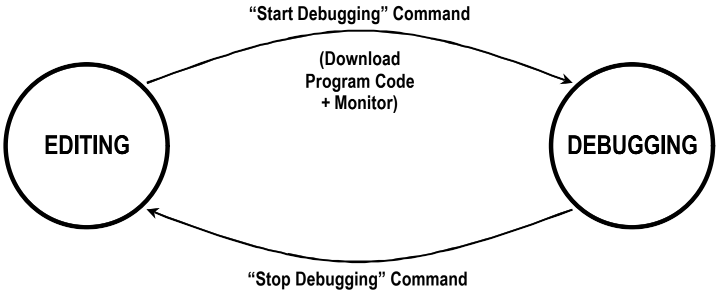
Congratulations! You have successfully completed this tutorial! You can continue to experiment with the inDART-ST7 user interface and discover by your own its potentialities. For an in-depth guide of all of the user interface features, select **Help > Search** from the main menu.

3. inDART-ST7C Operations

inDART-ST7C Working Principles

inDART-ST7C is an in-circuit debugger as well as a programming tool. It programs files into the ST72CXXX microcontrollers and offers debugging features like real-time code execution, stepping, and breakpoint. Its debugging features are achieved thanks to a small portion of monitor code which is automatically and transparently added to the user code and programmed into the target microcontroller. The monitor communicates with the host PC through a bi-directional command-based protocol via the ISPDATA and ISPCLK lines of the microcontroller. For this reason, the ISPDATA and ISPCLK lines (which are the standard lines used by the microcontroller's ISP peripheral) are reserved during debugging sessions. The same two lines are also used during device programming.

Contrariwise to traditional in-circuit emulation (where the target application is executed and emulated inside the emulator), inDART-ST7C uses the very same target microcontroller to carry on in-circuit execution. This means that all microcontroller's peripherals (timers, A/D converters, I/O pins, etc.) are not reconstructed or simulated by an external device, but are the very same target microcontroller's peripherals. Moreover, the inDART-ST7C debugging approach ensures that the target microcontroller's electrical characteristics (pull-ups, low-voltage operations, I/Os thresholds, etc.) are 100% guaranteed. The trade-off, however (in electronics miracles don't exist, after all—do they?), is that the target microcontroller must be properly configured and ready to execute target applications.



Monitor Download

Limitations

Since inDART-ST7C is based on the ISP feature of the ST7, some on-chip resources are wasted for debugging purposes. In particular:

- 7 stack levels (bytes) are wasted;
- 288 bytes are reserved for the monitor (from address FEC0H to address FFE0H);
- ISPDATA and ISPCLK lines are reserved for programming and in-circuit debugging (i.e., PB5 and PB6 I/O lines are reserved on ST72C254 devices; PC4 and PC6 lines are reserved on ST72C334 devices; PB3 and PB5 lines are reserved on ST72C171 devices);
- SPI peripheral and the SPI interrupt vector are reserved (since the SPI I/O lines share the ISPDATA and ISPCLK lines);
- The TRAP instruction and the TRAP interrupt vector are reserved for the monitor;
- Global interrupts must be enabled to allow the STOP command to be implemented;
- Peripherals run even in STOP mode.

Breakpoints Notes

inDART-ST7C can handle an unlimited number of breakpoints within program memory. When you set a breakpoint on a source code line, inDART-ST7C automatically (and transparently) replaces the opcode of the instruction where the breakpoint is set with the opcode of the TRAP instruction (this explains why the TRAP instruction—and the TRAP vector—is reserved and should not be used in user applications).

The difference between inDART-ST7C and a standard emulator is that the latter can set/reset breakpoints under all conditions, while inDART-ST7C (since it needs to change the opcode of the instruction where the breakpoint is set with the TRAP instruction) needs to program the FLASH memory of the target microcontroller to perform this operation. To program the FLASH memory, inDART-ST7C needs to enter the ISP mode and, consequentely, to reset the target microcontroller. inDART-ST7C actually programs breakpoints only when an execution command that resets the target microcontroller (e.g., the **"Run"** command) is issued. In all of the other cases breakpoint information is stored inside the host PC, program execution switches to “step mode” (see below) and breakpoint programming occurs next time an execution command that resets the target microcontroller is issued.

Program Execution Notes

inDART-ST7C executes in real-time mode or in step mode.

- Step mode execution occurs when you launch the emulation, then stop it, then set/modify breakpoints, and then continue the emulation;
- Real-time execution occurs in all other cases.

The following table summarizes (for the most common emulation commands) which command causes inDART-ST7C to execute in real-time mode or in step mode.

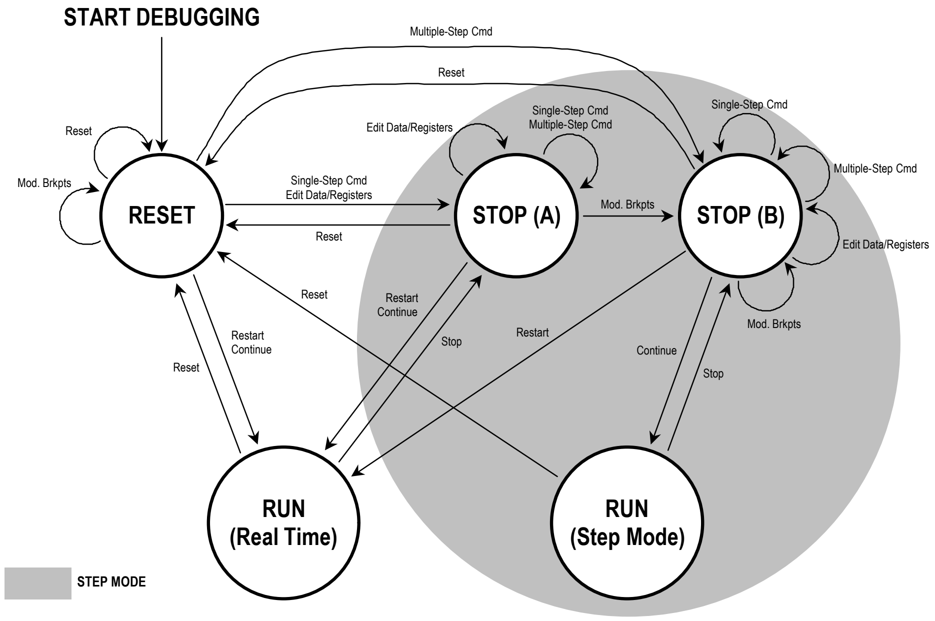
Emulation Command	Execution Mode
Run	Real-time
Restart	Real-time
Continue	Real-time if no breakpoints have been set/modified since the “Stop Program” command. Step mode otherwise.
Step Into	Step mode
Step Over	Step mode
Step Out	Step mode
Run to Cursor	Step mode

Execution Mode Summary

The following FSM (Finite State Machine) diagram illustrates in detail the behaviour of inDART-ST7C. In the diagram, each state transition is caused by one or more events, explained below.

- **Reset.** The Reset event is caused by the **“Chip Reset”** command, an external Reset event, or a peripheral Reset event.
- **Restart.** Include either the **“Run”** command or the **“Restart”** command.
- **Stop.** The Stop event is caused either by the **“Stop Program”** command or by a breakpoint hit.
- **Continue.** Is caused by the **“Continue”** command.

- **Single-step command.** Is caused by the **"Step Into"** and **"Step Into ASM"** commands.
- **Multiple-step command.** One of the following commands: **"Step Out"**, **"Step Over"**, **"Step Over ASM"**, **"Run To Cursor"**. Please note that, in some cases, the **"Step Over"** and **"Step Over ASM"** commands have the same effect of the **"Step Into"** and **"Step Into ASM"** commands, respectively, depending on whether there is a function call to be executed or not.
- **Modify Breakpoints.** One of the following commands: **"Insert/Remove Breakpoint"**, **"Enable/Disable Breakpoint"**, **"Remove All Breakpoints"**.
- **Edit Data/Registers.** One of the following events: memory editing, registers editing, watch variables editing, peripherals editing, Program Counter editing.



inDART-ST7C Execution Modes

Real-Time Execution

When inDART-ST7C runs in real time, instructions are executed just as the microcontroller would without the debugger. inDART-ST7C executes in real time until a breakpoint is encountered or until the **“Stop Program”** command is issued. Subsequent **“Continue”** commands are still performed in real time only if you do not set/modify breakpoints.

Step Mode Execution

In step mode, program execution is supervised by the host PC and performed (automatically) step by step.

Note: *in step mode, peripherals are still truly handled by the target microcontroller. However, interrupts are not handled (that is, interrupt events never occur). This affects execution of instructions like* **`wfi`** *and* **`halt`***—since these instructions require interrupts to work, in step mode their behaviour is different than in real-time mode. In particular, the* **`wfi`** *instruction loops forever on itself (no interrupts ever occur), and after executing an* **`halt`** *instruction, program execution cannot continue (same reason).*

Note: *in step mode, program execution speed is affected by the host PC speed—in particular, in step mode, program execution is slower than real-time program execution. This can affect the program execution speed when, for example, commands such as* ***“Step Over”****,* ***“Step Out”*** *or* ***“Run to Cursor”*** *are executed, since it can take a long time to perform the action (especially when the code contains loops to be executed many times).*

Online Assembler Notes

During a debugging sessions, every time assembly instructions are modified in the *Assembly* window (through the Online Assembler), a reset condition occurs (even though the Program Counter arrow still points to the same instruction). Subsequent debugging commands will be therefore executed from a reset state.

Stop Execution Notes

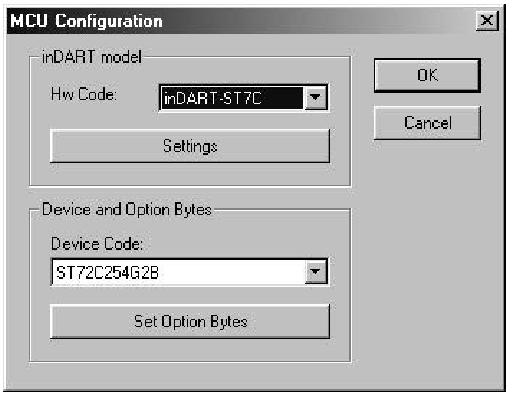
During debugging, user program execution will stop if one of the following conditions occur:

1. A breakpoint has been encountered (or a TRAP instruction has been executed);
2. The microcontroller RESET line has been driven active;
3. A microcontroller internal RESET state has been triggered (by, for example, a Watchdog event, or an internal low-voltage detection);
4. The **"Stop Program"** emulation command has been issued from the inDART-ST7 user interface. Please note that, during real-time execution, the **"Stop Program"** command only works if interrupts are enabled in the user application.

Note: *when executing in real-time mode, the **"Stop Program"** command only works if interrupts are enabled on the user program (the **RIM** instruction must be present at the beginning of user's code). The **RIM** instruction is only needed if users want to stop the program by using the **"Stop Program"** from the user interface—and does not affect any other debugging feature. In particular, the **RIM** instruction does not affect breakpoints.*

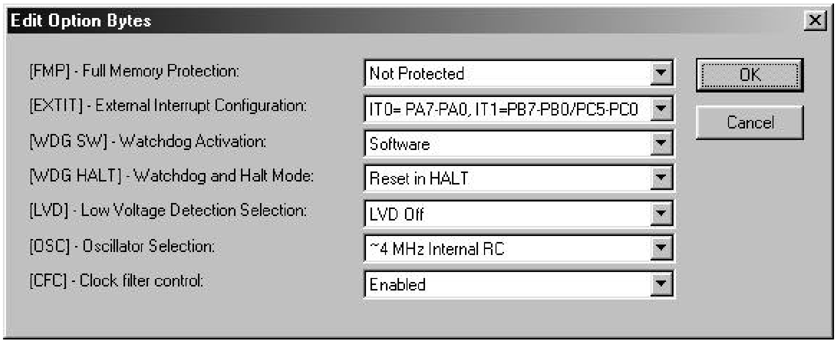
Configuring the MCU

Before to start a debugging session, you must define and configure the target device (MCU) you wish inDART-ST7C to emulate. The target device is defined and configured from the *MCU Configuration* dialog box. To access it, select **Tools > MCU Configuration** from the main menu. The following dialog box will appear.



The *MCU Configuration* Dialog Box

- First of all, verify that the inDART model corresponds to "inDART-ST7C".
- By clicking the **"Settings"** button you can modify the "RESET Rise Time" parameter to make it suitable for your target application. This parameters depends on your target application's reset circuitry. If you use inDART-ST7C in connection with an inDART demo board, you don't need to change this parameter. For more information on this parameter, please refer to *"Appendix A: Target Reset Topics"*.
- The "Device Code" parameter specifies the target microcontroller used in your target application. You must specify the exact device code of the microcontroller you are working with.
- The **"Set Option Bytes"** button allows you to access the *Edit Option Bytes* dialog box. The following figure illustrates an example of the *Edit Option Bytes* dialog box (every microcontroller has its own specific Option Bytes).



[FMP] - Full Memory Protection:	Not Protected
[EXTIT] - External Interrupt Configuration:	IT0= PA7-PA0, IT1=PB7-PB0/PC5-PC0
[WDG SW] - Watchdog Activation:	Software
[WDG HALT] - Watchdog and Halt Mode:	Reset in HALT
[LVD] - Low Voltage Detection Selection:	LVD Off
[OSC] - Oscillator Selection:	~4 MHz Internal RC
[CFC] - Clock filter control:	Enabled

The *Edit Option Bytes* dialog box

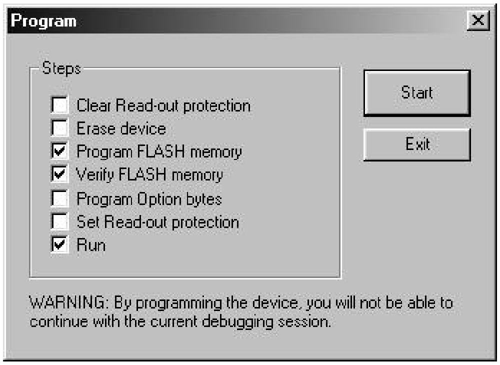
Particular attention must be paid in correctly setting each of the Option Bytes parameters. Improper settings may cause the target microcontroller not to work correctly (or not to work at all). In particular, the following parameters must be carefully set:

- **FMP (Full Memory Protection).** Must be always set to “Not Protected”.
- **WDG SW (Watchdog Activation).** If set to “Hardware”, you must refresh the Watchdog in your target application. You can leave it to “Software” when working with the provided sample applications.
- **OSC (Oscillator Selection).** Specify the oscillator source used by your target application. inDART-ST7C demo boards, by default, work with the internal, 4-Mhz RC oscillator.
- **CFC (Clock Filter Control).** When enabled, allows an internal, security clock to be used should the main oscillator source (specified by the OSC parameter) not be present, or set incorrectly, or not work correctly. It is suggested to set this parameter to “Enabled”—since otherwise, if due to a bad OSC parameter setting (or other reason) the target microcontroller can’t work, it will not be possible to reprogram the Option Bytes.

Note: *the Option Bytes value specified in the* Edit Option Bytes *dialog box will be used when programming the device (see* “Programming Capabilities” *below).*

Programming Capabilities

inDART-ST7C feature ISP programming capabilities. A programming utility is built-in in the inDART-ST7 user interface. When in Debugging mode, the **Tools > Program** command in the main menu is enabled, and allows you to program the target microcontroller with your target application code but without any debugging code. A typical programming procedure is made of a series of programming steps, as indicated in the figure below. You can choose whether to perform or not each single step. Please note that the programming steps are performed in the exact order as shown in the *Program* dialog box.



WARNING: By programming the device, you will not be able to continue with the current debugging session.

The *Program* Dialog Box

- **Clear Read-Out Protection.** Clears the Read-Out Protection bit. Check this programming step when you are programming a device which has been programmed with the Read-Out Protection bit set.
- **Erase Device.** Erases the entire Code memory area.
- **Program FLASH Memory.** Writes the application code into the Code memory area.
- **Verify FLASH Memory.** Verifies that the application code has been correctly written into the Code memory.
- **Program Option Bytes.** Programs the Option Bytes according to the parameters specified in the *Edit Option Bytes* dialog box.
- **Set Read-Out Protection.** Set the Read-Out Protection bit. Check this programming step if you want to protect the application code from external reading operations.

- **Run.** After programming, resets the microcontroller and executes the application program.

Note: *after programming the device, you will not able to continue with the current debugging session. To continue debugging, you must stop the current debugging session and start a new one.*

DataBlaze Programming Utility

A standalone, full-featured programming utility (DataBlaze) is also provided with inDART-ST7. To start the DataBlaze utility select **Start > Programs > SofTec inDART-ST7 > Programmer > DataBlaze**.

DataBlaze offers the following advanced features:

- Code memory editing;
- Data memory (EEPROM) editing;
- Blank check/erase/verify operations on Code memory, Data memory and Option Bytes;
- Read operations from Code memory, Data memory and Option Bytes;
- Project handling.

4. Troubleshooting

Installing the inDART-ST7 User Interface under Windows NT

In order to install the inDART-ST7 user interface under Windows NT, you must have logged in as *Administrator*.

Common Problems and Solutions

Communication can't be established with inDART-ST7C

1. Make sure the inDART-ST7C in-circuit debugger is powered on. inDART-ST7C can be powered by the demo board/target application via the ISP cable. In this case, the power supply is taken from the Vcc pin of the ISP connector, and MUST BE 5 V, 10 mA. In the case your target application is not able to provide the required voltage and current, you must power the inDART-ST7C board via its power connector. inDART-ST7C use a 9-12 V DC, 10 mA (unregulated) wall plug-in power supply.
2. Make sure that the parallel cable is connected. Also make sure you selected the LPT port number to which the instrument is connected to. To view/change the LPT port in use, choose **Emulator > Emulator Settings** from the inDART-ST7 user interface's main menu.
3. Make use you are working with the correct inDART hardware model. To view/change the inDART hardware model in use, choose **Tools > MCU Configuration** from the inDART-ST7 user interface's main menu.
4. Make sure the demo board/target application board is powered on and the target microcontroller is working. Programming and debugging rely on an ISP serial communication between the inDART-ST7C board and the demo board/target application. This means that, in order to work correctly, the target microcontroller must be running. In particular, make sure that:
   - The ISP cable is connected to the demo board/target application's ISP connector.
   - The target microcontroller is in place.
   - All of the ISP connector signals (ISPSEL, ISPDATA, ISPCLK, RESET, VDD and VSS) are correctly tied to the target microcontroller.

- The microcontroller's Reset circuitry is working (for more information please refer to "Appendix A: Target Reset Topics").
- The oscillator circuitry is working according to the Option Bytes specifications. To view/change the Option Bytes, choose **Tools > MCU Configuration** from the inDART-ST7 user interface's main menu. From the dialog box which will appear, click the **"Set Option Bytes"** button.

A communication error is returned on a program execution command (Run, Continue, Step, etc.)

Make sure that your program doesn't use the SPI peripheral and that the I/O bits corresponding to the ISPDATA and ISPCLK ISP signals are set to input mode.

When debugging, the program runs too slowly

This problem can have two causes:

- If not correctly set (via Option Bytes), the target microcontroller may run at a frequency lower than that specified. On some ST7 microcontrollers (if specified on the Option Bytes), the working frequency is automatically set to a safe value (< 1 MHz) when the specified oscillator source can't be found (or doesn't work properly). In this conditions, the low working frequency affects the ISP communication and, therefore, the inDART-ST7 user interface performances.
- Under some conditions, program execution is not performed in real time. As a consequence, program execution may slow down. Please read *"Program Execution Notes"* on Chapter 3 for more information.

The "Stop Program" command doesn't work

When executing in real-time mode, the **"Stop Program"** command only works if interrupts are enabled on the user program (the **RIM** instruction must be present at the beginning of user's code). The **RIM** instruction is only needed if users want to stop the program by using the **"Stop Program"** from the user interface—and does not affect any other debugging feature. In particular, the **RIM** instruction does not affect breakpoints.

Also make sure that your program doesn't use the SPI peripheral and that the I/O bits corresponding to the ISPDATA and ISPCLK ISP signals are set to input mode.

The program execution stops at the beginning of user’s code

A Reset condition occurred. This can be due to an external Reset condition (microcontroller’s RESET line driven low) or an internal Reset condition (e.g., due to a Watchdog event). For more information on causes that can trigger a Reset condition, please refer to the specific ST7 microcontroller device data sheet.

Interrupt handling routines are not executed

Under some conditions, program execution is not performed in real time. As a consequence, interrupts are not handled. Please read *“Real-Time Execution Notes”* on Chapter 3 for more information.

The program seems not to work correctly

When debugging your program with inDART-ST7C, some target microcontroller’s resources are reserved for debugging purposes. In particular, 288 bytes are reserved for the monitor (from address FEC0H to address FFE0H). Therefore, you must pay attention that your program doesn’t use this range of memory locations. Also, please remember that 7 stack levels are reserved by inDART-ST7C as well.

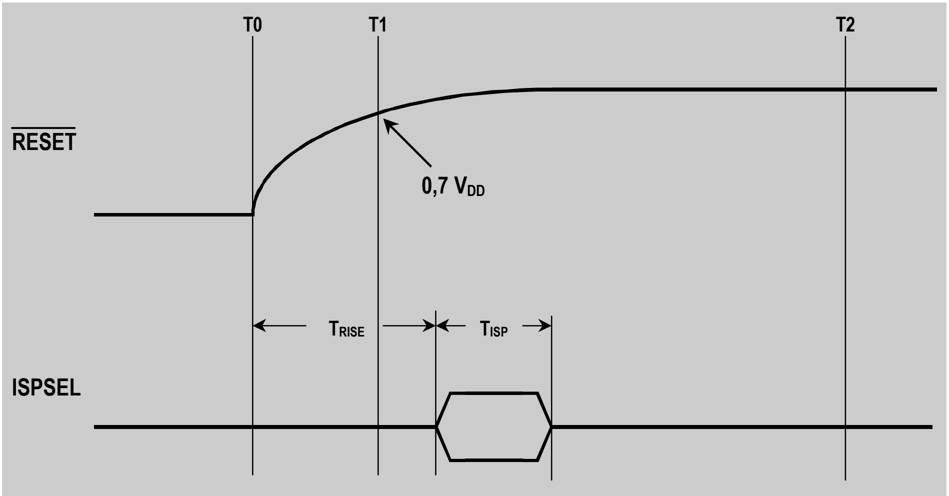
Getting Technical Support

Technical assistance is provided free to all customers. For technical assistance, documentation and information about products and services, please refer to your local SofTec Microsystems partner.

SofTec Microsystems offers its customers a free technical support service at *support@softecmicro.com.*

Appendix A: Target Reset Topics

During debugging and programming, inDART-ST7C communicates with the target microcontroller through the target microcontroller's ISP capability. Both debugging and programming capabilities thus require that the ISP mode be selected on the target microcontroller. In order to enter the ISP mode, the microcontroller ISPSEL line must be correctly driven during the microcontroller RESET phase. inDART-ST7C automatically and transparently performs such operation: however, since inDART-ST7C must drive the ISPSEL line in the correct timings, you must specify (in the inDART-ST7 user interface—see *"Configuring the MCU"* on chapter 3) a parameter called "RESET Rise Time" (T_{RISE} in the diagram below), according to the limits depicted on the diagram itself.



T2 – T1 = 256 CPU Clocks (32 us @ 16 MHz)
T_{ISP} = 4 us
$T1 < T_{RISE} < T2 - T_{ISP}$

T_{RISE} Parameter Constraints

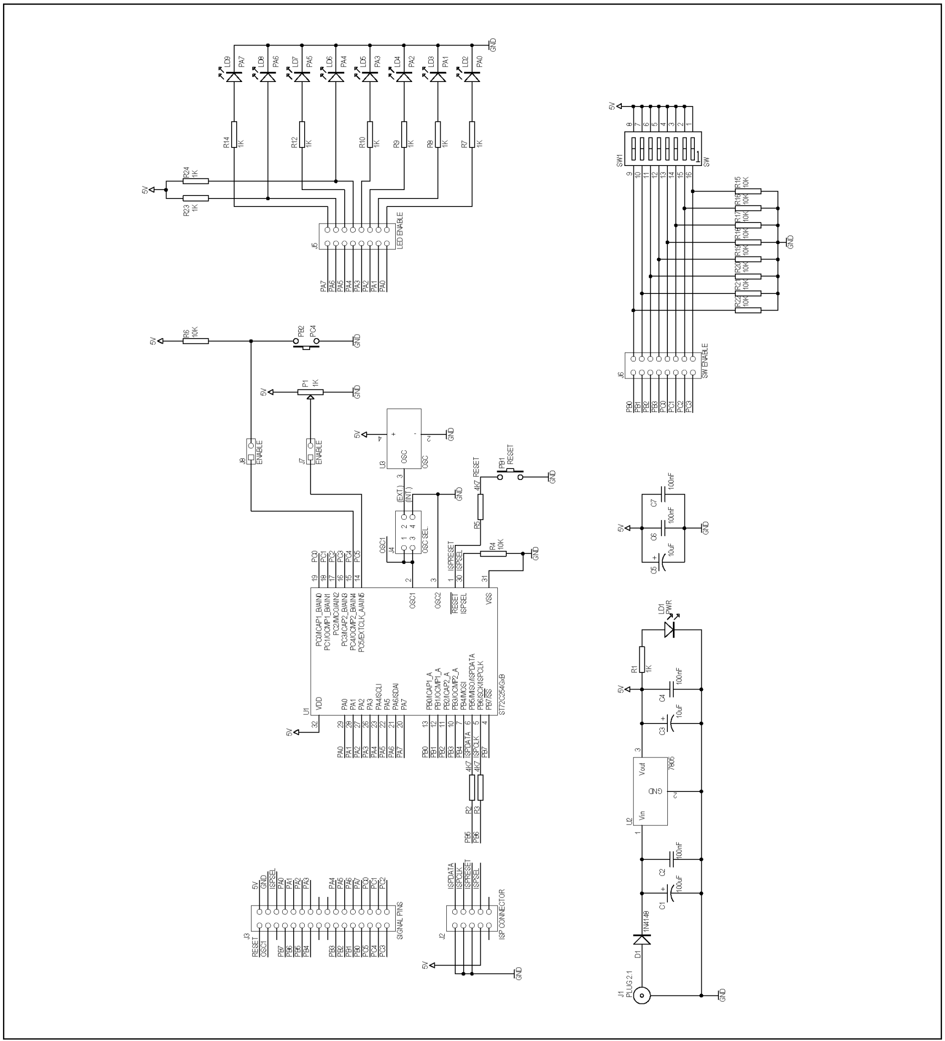
The following table lists some values of the "RESET Rise Time" parameter based on the most common RC configurations. In the table, Rs is the serial resistor, which we suggest to use.

f = 16 MHz, Vcc = 5 V					
RC	RESET Rise Time	Rs	RC	RESET Rise Time	Rs
4,7 K * 100 pF	10 us	4,7 K	4,7 K * 100 pF	10 us	0
4,7 K * 10 nF	10 us	4,7 K	4,7 K * 10 nF	35 us	0
4,7 K * 22 nF	10 us	4,7 K	4,7 K * 22 nF	76 us	0
4,7 K * 47 nF	10 us	4,7 K	4,7 K * 47 nF	202 us	0
4,7 K * 100 nF	10 us	4,7 K	4,7 K * 100 nF	490 us	0
4,7 K * 1 uF	10 us	4,7 K			

"RESET Rise Time" Vs. RC Values

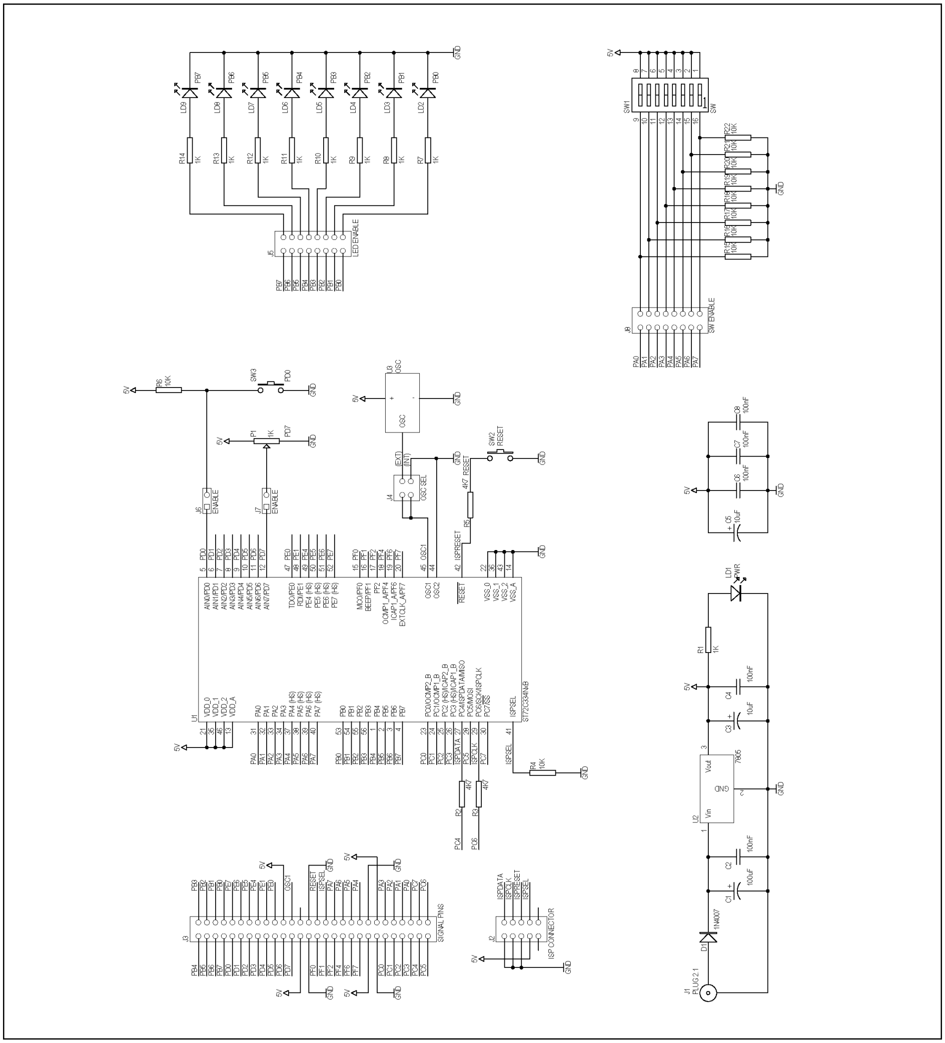
A

Appendix B: IDB-ST7C254 Demo Board Schematic



IDB-ST7C254 Demo Board Schematic

Appendix C: IDB-ST7C334 Demo Board Schematic



IDB-ST7C334 Demo Board Schematic