

Quick Start Guide

Hitex Starter Kit for netX

with

Hilscher NXSB 100 Board



04/2006 - 002 - 5.2 - 30.17.0019.0

© Copyright 2006 Hitex Development Tools GmbH

All rights reserved. No part of this document may be copied or reproduced in any form or by any means without prior written consent of Hitex Development Tools. Hitex Development Tools retains the right to make changes to these specifications at any time, without notice. Hitex Development Tools makes no commitment to update nor to keep current the information contained in this document.

Hitex Development Tools makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hitex Development Tools assumes no responsibility for any errors that may appear in this document.

DProbe, Hitex, HiTOP, Tanto, and Tantino are trademarks of Hitex Development Tools. All trademarks of other companies used in this document refer exclusively to the products of these companies.

Hitex Development Tools GmbH

Greschbachstr. 12

D-76229 Karlsruhe

Germany

Tel: +49 721 9628 - 0

Fax: +49 721 9628 - 149

E-mail: Sales@hitex.de Sales@hitex.com
Support@hitex.de Support@hitex.com

Internet: <http://www.hitex.de> <http://www.hitex.com>

04/2006 - 002 - 5.2

Contents

1	Introduction	4
2	Installing Software and Hardware	5
3	Starting a Project in HiTOP	7
4	The First Steps of Debugging	11
5	Modifying an Application	16
6	rcX Example	19
7	Profibus Example	20
8	rcX LED Demo	22
9	Recommended Settings	23

1 Introduction

The goal of this starter kit is to give the user a fast introduction to the Hilscher netX communication controller family. Included are examples demonstrating the development of ARM code and the debugging of this code using the debug hardware **Tantino for ARM 7-9** with HiTOP environment.

The best way getting started is to follow the instructions within this small guide.

To get more information about the **Hitex Tool chain for ARM**, refer to http://www.hitex.com/perm/arm_toolchain.html.

A comprehensive introduction to the HiTOP5 user interface will be given in our video tutorials. Feel free to download the movies on http://www.hitex.com/perm/video_tutorials.html.

To get more information about the Hilscher netX product line, refer to <http://www.hilscher.com>

We wish you a fast and high quality embedded development using the Hilscher netX communication controller.

Your Hitex Team

2 Installing Software and Hardware

Installing the Software

To install HiTOP please insert the CD and follow the instructions on the screen. If the auto run feature of your CD drive is disabled, open the CD drive folder and start **setup.exe**.

Click on



In order to use this starter kit, you must install both

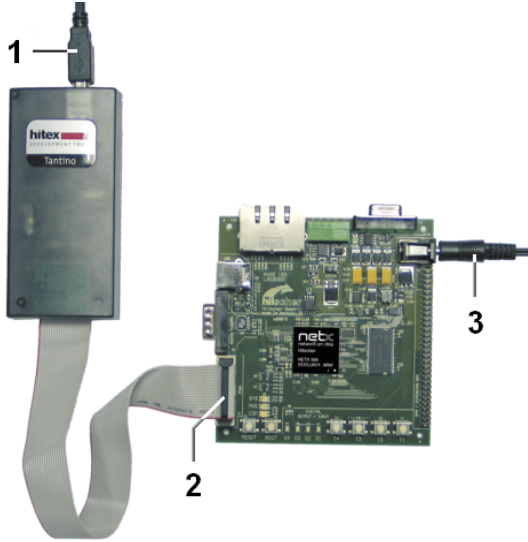
- **HiTOP for ARM® / Intel® XScale**, and the
- **GNU Compiler for ARM®**.

After installing the software, the debug hardware Tantino and the evaluation board may be connected as described on the next page.

Installing the Hardware

Tantino for ARM 7-9

- Connect the Tantino to the PC's USB port via the USB cable (1).
- Connect the 20-pin cable to the board's JTAG connector (2).
- Connect the power supply to the NXSB 100 board (3).



The power supply should deliver 9 to 24 Vdc with 5 VA.

3 Starting a Project in HiTOP

Start HiTOP by double clicking the desktop icon:



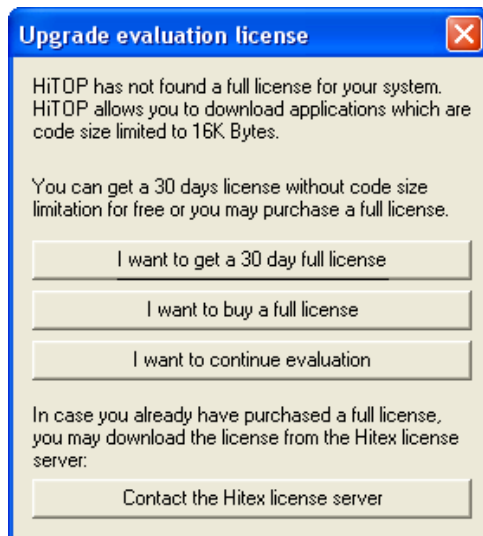
The tip of the day is presented which contains useful information when working with HiTOP.

Select and open a project file using the **Project > Open** command.

If HiTOP was started for the first time:

- Choose the debugger hardware in the **Tool selection** dialog (**Tantino for ARM 7-9** according to the bottom side label of your Tantino system), and click on **Next**.
- In the **Port selection** dialog, select as communication port **USB** and enter the serial number of your Tantino, without leading zeros. The serial number can be found on the bottom side of the Tantino. Click on **Next** and HiTOP will be connected to the Tantino.

The netX Starter kit is delivered with a code size limited edition of HiTOP. After opening the project, you will be prompted with the following selection:



The examples delivered with the starter kit can all be used with the code size limited version. Please select

I want to continue evaluation

to use the examples.

More information on how to get a license can be found in HiTOP's Online Help System.

Information on the licenses installed can be found in the "HitopLicense.lic" file in the HiTOP52-ARM directory.

Starting a netX Example


- Open an example project using the **Project > Open** command.
- In the `..\Examples\netX\netX100\GNU\Example01` folder, select the following project file:


`netX_example01.htp`

- In the **Download Applications** dialog box, click **OK**.


The ELF (Executable and Linkable Format) is now downloaded to the target. The code is loaded into the netX memory and HiTOP retrieved the debug information.

Next we will have to set the program counter to the beginning of code execution:

- Click on the  icon and select the **GotoStart.scr** script file located in the same directory as the example program, or use the following icon to execute this script file:

 GotoStart

This script will set the program counter to the `Start_init_s` label in the startup code of the GNU library.

- Now click on the **Go** icon: 

The netX starts executing the program code making the board's two-color LED continuously change its color state.

Project Loaded

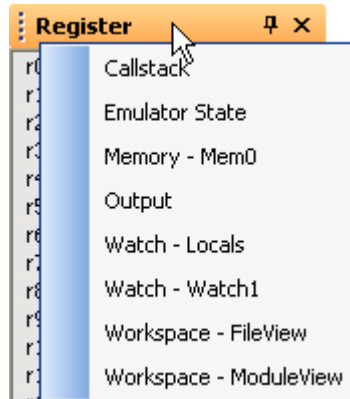
After having loaded the project successfully, proceed as follows:

- Click on the **Stop** icon:  to halt the program execution.

You may now arrange the HiTOP windows according to your needs.

- Right click on the Title bar of any Docking window. This will show the list of windows not yet opened.

Note, that by using the **View > Save screen layout** and **View > Load screen layout** commands, different window layouts can be stored and later reloaded.¹




Do not forget to save the project after such changes (**Project > Save**).

¹ More about the handling of HiTOP is shown in the online help system and in the video tutorials.


4 The First Steps of Debugging

In the previous steps, a program was loaded, the program counter was set to the starting point, and execution was started and stopped.


In the **Source window**, start debugging with some single steps

- by clicking on the **Step Into** icon , or
- using the **Debug > Step Instruction** command, or
- via the **F9** function key.

It is also possible to run the program up to a desired code line by moving the cursor over the window's left side column.


The cursor will get a different shape: 

Click on the blue rectangle using the left mouse button. The program will be executed until this program line is reached. When it stops, the green line and the yellow arrow mark the current position of the program counter:

```
 for (j=0; j<1000; j++)
```

The different text modules used to build the program can be displayed in the **Source window** by double-clicking on the module name in the **Workspace window's** module view:

ModuleView


To return to the current program counter (PC) location, use the **Show PC** icon (**Debug > Show PC Location** command): 

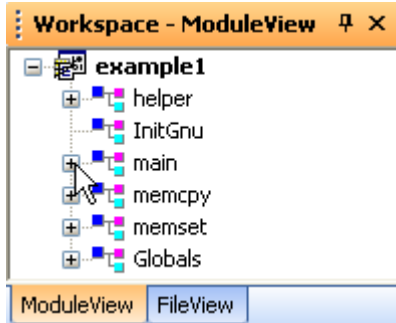
You can now examine the code and the variables on source level.

The ARM core has two hardware breakpoint resources. This is why only two hardware breakpoints (code or data breakpoints) can be set when debugging in ROM/FLASH.



Note that all example programs of the netX starter kit execute in RAM. With code in RAM, HiTOP can use an unlimited number of software code breakpoints.

To show the features of these breakpoints, do the following:

In the **Workspace window** click on the plus sign (+) of the 'main' module (if it is a  sign, skip this action).



Now the symbols of the 'main' module are displayed. With the right mouse button click on the function 'main' and select the context menu's **Set Breakpoint** command.

Click on the  icon to make sure that we start this program from the beginning, then click on the **Go** icon: .

The emulation will stop at the breakpoint.

Open the **Source window's** context menu by right clicking on the green bar and select **Disable Breakpoint** to temporarily disable the breakpoint or **Remove Breakpoint** to permanently remove it.

Next, mark the text "xyz.a", just a few lines below the start of the main function. The text 'xyz.a' now has a black background:

```
/* now we do
xyz.a = 0;
xyz.c = 'A';
```

Open the **Breakpoint window** via **Debug > Breakpoints** and move it to a position that the marked 'xyz.a' is not hidden by this new window.


Select the Breakpoint window's **Data** tab of the and drag&drop the marked text into the Breakpoint window. This sets a **Data Breakpoint** on a write access to the 'xyz.a' variable.

Now, the default settings of this breakpoint have to be modified.


Open the **Breakpoint window's context menu** right clicking on the data line and select **C**hange. Click on the **A**dvanced button.

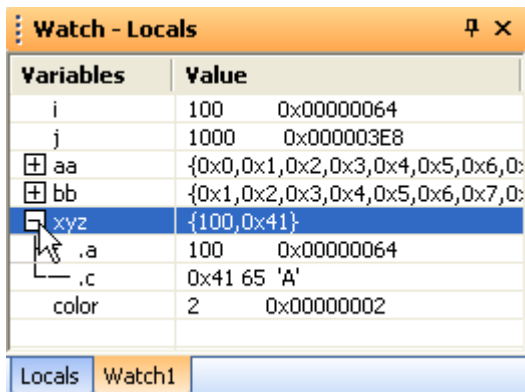
The Start and the End of the address range should both be "xyz.a".

In the **Value Range**, set the **Minimum** for the condition to 100 and leave the **Type** set to DWORD. Click on **OK**.

Click on the  icon to start the program execution. After a few seconds, the CPU will stop indicated by changing the text "Running" to "Halted" in the lower right of the status bar.

Next, refer to the **Watch window** and select the **Watch1 tab**.

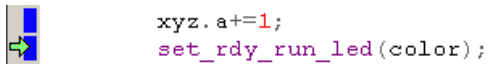
Expand the "xyz" structure by clicking on the  node:



Variables	Value
i	100 0x00000064
j	1000 0x000003E8
aa	{0x0,0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8,0x9}
bb	{0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8,0x9}
xyz	{100,0x41}
.a	100 0x00000064
.c	0x41 65 'A'
color	2 0x00000002

In the highlighted data line you see that "xyz.a" has a value of 100.

The current location of the program counter is just one line below the line where the value of "xyz.a" had been incremented:



```

xyz.a+=1;
set_rdy_run_led(color);

```

With **Code Breakpoints**, the ARM controller will halt before the code at the breakpoint position is executed.

With **Data Breakpoints**, the ARM controller is halted because it accessed the data as defined in the breakpoint conditions. As mentioned before, the ARM CPU supports a maximum of two Data Breakpoints.

5 Modifying an Application

To compile and link an application, HiTOP relies on an external compiler. For this purpose we installed the GNU compiler (see p. 5). In the next steps we will:


- configure HiTOP to use the compiler,
- modify the example of the previous chapter,
- rebuild the application,
- load the application into the netX controller,
- change the program counter,
- execute the program.

But first of all there is an issue with the HiTOP environment that the user must be aware of:

If the target is in running state, loading a new image into the memory of the target or even closing HiTOP may lead to unpredictable results.

Make sure that the lower right corner of the status line displays

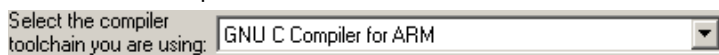


If necessary halt the target by clicking on the  icon.

Execute the **Project > Settings** command, select



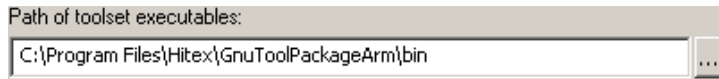
and the GNU compiler:



Click on the



button and enter the correct path to your GNU Compiler¹ installation:



Use the  button to browse.

Click the **OK** button and save the project using the **Project > Save** command.

The file "netX.Ink" defines the object files that will be linked together to create the application. In this file you also find a SEARCH_DIR entry specifying where the linker can find the GNU standard C library files. Please modify these lines or add new SEARCH_DIR lines to match your installation path.

¹ Details about the GNU compiler can be found in the documentation included in the compiler's installation files.

To change the code of the application, the **Source window's** editor can be used. Double-clicking the "main.c" file in the **FileView** mode of the **Workspace** window will add a new tab "main.c" in the **Source window**. Click on the new **main.c tab** in the Source window.


Right-click into the **Source window** and select the context menu's

Switch to Edit Mode 

Move the cursor to the line

```
for (j=0; j<1000; j++)
```


and change the value 1000 into 3000.

You may now rebuild the application using the **Rebuild** icon  in HiTOP's toolbar: The results of the build process can be seen in the **Output** window's **Build** tab. Following a successful rebuild, a **Download dialog** will show up. Confirm the download clicking on the **OK** button.

Now set the Program Counter to the start of the GNU library code by clicking on

 GotoStart

and then on the **Run** icon: 

Before continuing with the next example, remember to click the **Stop** icon: 

6 rcX Example

The second example for the netX controller will use the ROM based rcX operating system. This example project is called **SimplercXexample**.

Open the project (**Project > Open** command) and confirm the **Download dialog**.

In the **ModuleView** of the **Workspace window**, double-click the "InitGnu" entry. In the **Source window**, scroll down until you see the **start_init_s** label at line 95. Right-click the line just below the label and select the context menu's

Set **N**ew Program Counter

The program counter is now set to the start of the initialization code.

The file "SimpleExample.c" defines the code of the task to be executed. In the file "Config_SimpleExample.c" you will find the rcX initialization and configuration defining the task to execute. For a detailed description, refer to the rcX manuals.

Click on the **Run** icon  to start the application.

7 Profibus Example

The third example **Profibus_slave** is a PROFIBUS-DP slave. A GSD file describing the slave being emulated by the netX can be found in the example directory. A description of the Profibus Slave can be found in the DOC subdirectory of the example.

In the example, the slave is configured as 'netX' with a station address of 32, 2 bytes input and 2 bytes output. Only 4 of these inputs and 4 of the outputs are mapped to I/O on the board.

After loading and executing the example, the slave is waiting for communication from a Profibus master. With SyCON, the slave information will look like this:

Slave Configuration

General

Device: netX Station address: 32

Description: Slave1

Activate device in actual configuration

Enable watchdog control GSD file: HIL_QA12.GSD

Max. length of in-/output data: 4 Byte Length of in-/output data: 4 Byte

Max. length of input data: 2 Byte Length of input data: 2 Byte

Max. length of output data: 2 Byte Length of output data: 2 Byte

Max. number of modules: 1 Number of modules: 1

Module	Inputs	Outputs	In/Out	Identifier
2 byte input/output	2 Byte	2 Byte		0x21, 0x11

Assigned master: Station address 0 Master0

0 / CIF60-PB

Actual slave: Station address 32 Slave1

32 / netX

Slot	Idx	Module	Symbol	Type	I Addr.	I Len.	Type	O Addr.	O Len.
1	1	2 byte	Module1				QB	0	2
1	2	2 byte	Module1	IB	0	2			

Buttons: OK, Cancel, Parameter Data..., DPV1 Settings..., Append Module, Remove Module, Insert Module, Predefined Modules, Symbolic Names

8 rcX LED Demo

The fourth example demonstrates the handling of externally generated events under rcX. For this purpose it utilizes the four push buttons on the netX 100 Starter Board as inputs. Depending on their state, the MSN and COM LED will be switched on. The following table shows the assignment to the LED colors.

Push Button	I1	I2	I3	I4
LED color	COM orange	COM red	MSN green	MSN red

The example project is called **LedDemo**.

Open the project (**Project > Open** command) and confirm the **Download dialog**. Before the download takes place the script **sdram_config.scr** will initialize the SDRAM controller. After the download, the program counter will be automatically set by the script **reset_pc.scr**.

Finally, click on the **Run** icon  to start the application.

The file layout confirms to the "Task Layer Reference". So the file "LedDemo_Process.c" defines the code of the task to be executed. In the file "Config_netX.c" you will find the rcX initialization and configuration defining the task to execute. Please refer to the rcX manuals for a detailed description.

9 Recommended Settings

The following is a list of recommended HiTOP configurations for the netX controller:

Clock	
In the Project > Settings > Emulator Settings dialog, Tap Clock settings, use the following options:	
Clocking Mode	Fixed
Clock Frequency	10.0 MHz
rcX Operating System	
Always configure "little endian" when using the netX controller with the rcX operating system.	
Cache	
In the Project > Settings > Processor Settings dialog, Cache settings, use the following options:	
Cache Mode	automatic
Cache Code address	0x0

Assembler- / Compiler- / Linker Options

The following options are recommended:

Assembler	Assembler options: <code>-marmv4t -gdwarf2 -mthumb-interwork</code>
Compiler	Preprocessor definitions: <code>_NETX_HITOP_; __RCX__</code> Compiler options: <code>-gdwarf-2 -Wall -O1 -mapcs -march=armv4t -mthumb-interwork -c -mthumb -fshort-enums</code>
Linker	Linker options: <code>-static -lc -lm -lgcc -Map=output/example.map</code>

Hint

To change compiler or linker options of a project currently loaded, select the target in the **Project > Settings > Applications** dialog before using the **Tool Settings** button:



If you forget to do this, you will only change the general tool configuration for new targets, not for the one already defined.