# Support for remote debugging

| Applies to: | | | No. L6-001 |
|---|---|---|---|
| Platform | OS version | SDK version | Classification |
| LinPAC series | All version | LP-8x4x: V1.12 or later<br>Others: All version | Linux Applications |

Remote debugging is the process of debugging a program running on a different system (called target) from a different system (called host as local PC).

A debugger can pause your program and you can watch the values of the variables that you have defined. To perform remote debugging, the following utilities are needed:

- gdbserver – Run this on your target system- LinPAC controller for example.
- gdb – Execute this on your host system to connect to your target system-local PC for example.

| Model Name | GDB Command |
|---|---|
| PXA270 Series: LinCon/LP-51xx/8x3x/8x4x | arm-linux-gdb.exe |
| AM335x Series: LP-52xx/8x2x/9x2x | arm-linux-gnueabihf-gdb.exe |
| X86 Series: LP-8x8x/8x8x-Atom/LX-8000/9000 | gdb.exe |

This tutorial gives you easy-to-follow instructions works on two different interface.

## Section 1: Working with the Command line interface

The following example is based on an LP-8x4x controller.
The LP-8x4x SDK is based on the Cygwin interface and provides a Linux-like environment for Microsoft Windows systems, which can be executed within the LinPAC-8000 Build Environment in order to connect to the target system.

**Step 1: Compile the getlist.c file to getlist.exe**

As illustrated in the Fig. 1, the option **-g** must be added to the command line instruction in order to compile a .c file to an executable.

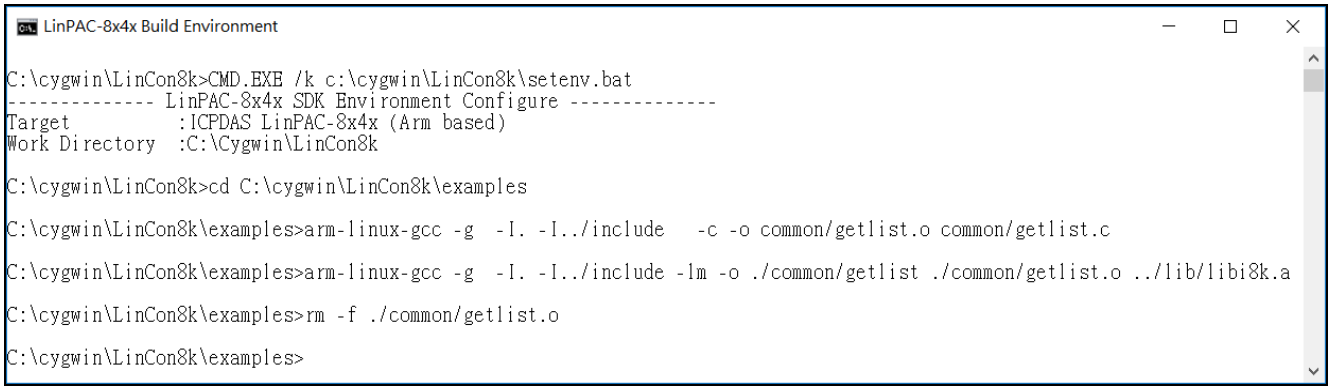



Fig. 1

 Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. 

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

**Step 2: Upload the "getlist.exe" file to the LP-8x4x controller**

In your FTP program, FileZilla is used in this example, locate the getlist.exe file in the LP-8x4x and then right click the file icon to open the context menu.

Select the Permissions option. In the Permissions dialog box, type "777" into the Numeric value textbox, and then click the OK button, as illustrated in the Fig. 2.

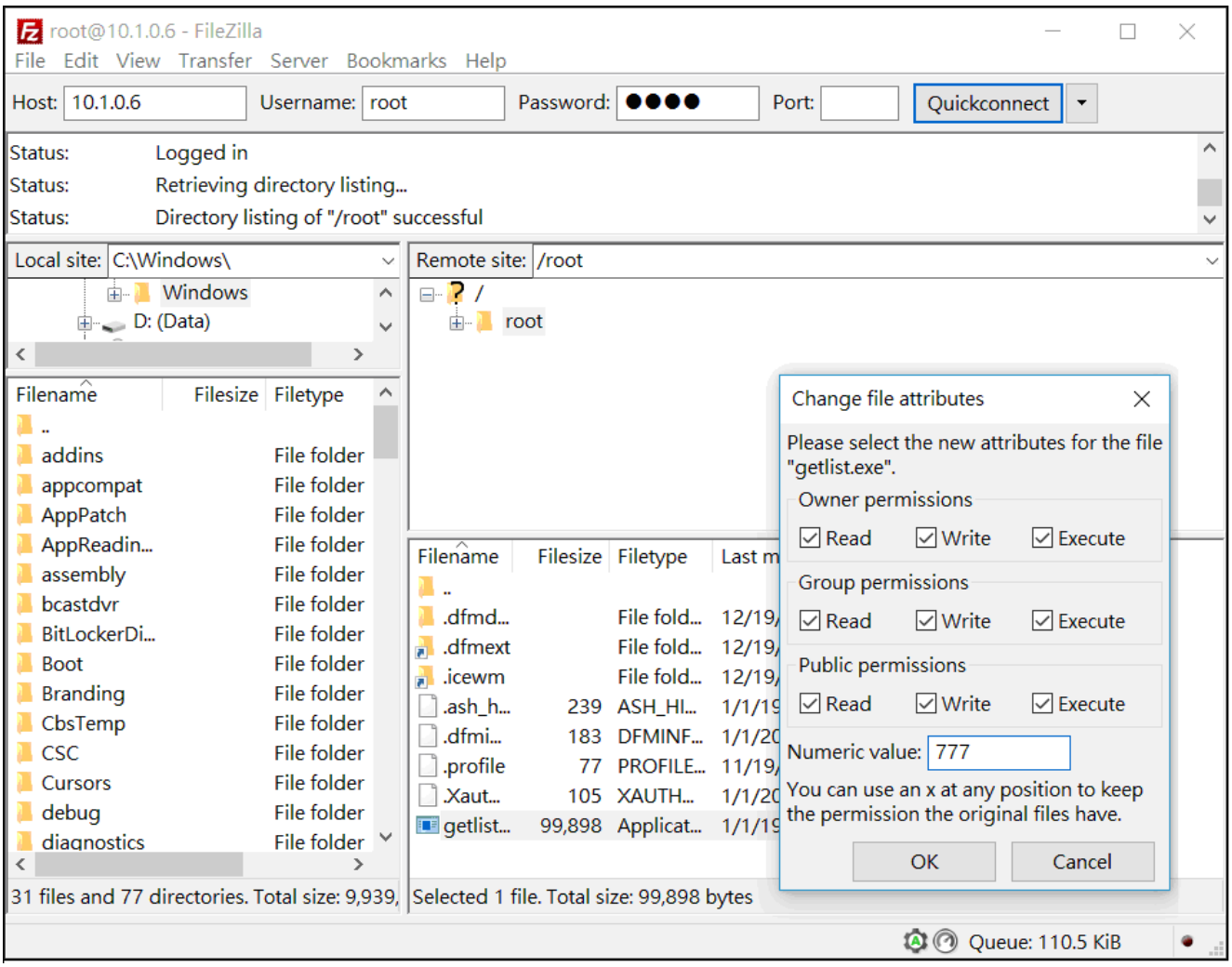



Fig. 2

**Step 3: Running gdbserver on the LP-8x4x controller.**

To use the server, log on to the target system, and run the gdbserver program.

When gdbserver is executed on the LP-8x4x, it passively waits for the host gdb to communicate with it, as illustrated in the Fig. 3. The usual syntax is:

**Command:** gdbserver 10.1.0.61:2345 getlist.exe

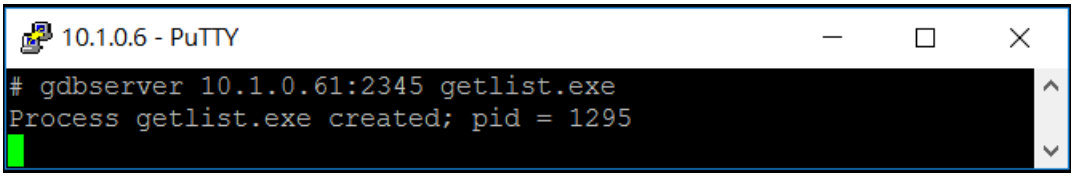



Fig. 3

 Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. 

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

**Step 4: Remotely connecting to LP-8x4x Debugging using gdbserver**

To perform remote debugging, start gdb and connect to the gdbserver.

The figures below illustrate the commands and processes used for debugging with gdb (Refer to Figs. 4 and 5 for more details).

**Command:** arm-linux-gdb getlist.exe




Fig. 4

**(gdb) target remote 10.1.0.6:2345**

To open a remote debug connection, debug using a TCP connection to a port on the IP address.

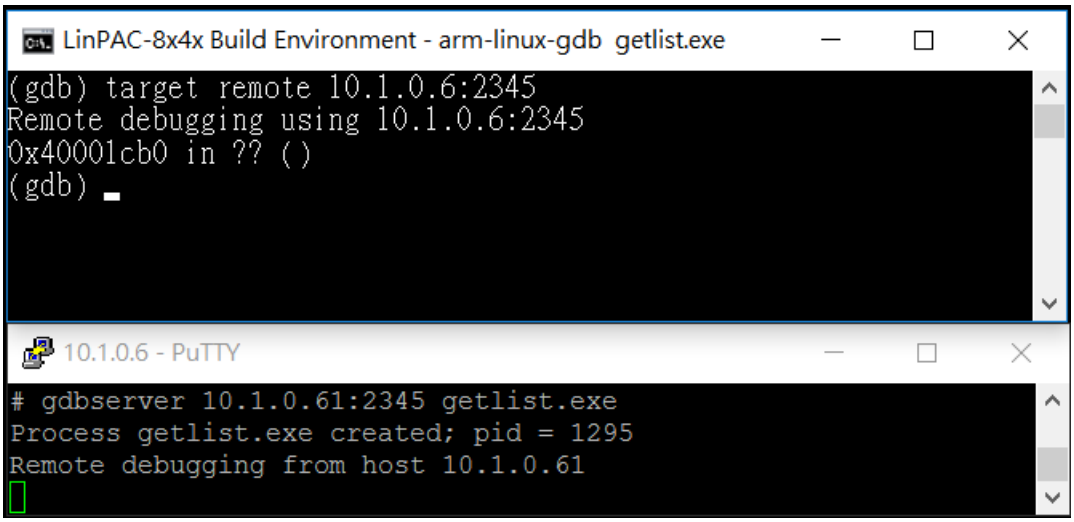



Fig. 5

Breakpoints “b 30” and “b 6“ have been included in the program (refer to Figs. 6 to Fig. 9), if we continue to execute the program, the output will be displayed on the target machine (refer to Fig. 10). To exit the gdb, type "q" (Quit).

 Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. 

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

**(gdb) list**

The "**list**" command allows you to display the demo code. By default, GDB prints ten source lines with any of these forms of the list command.




Fig. 6

**(gdb) b 66**

This command is used to set a breakpoint at line 66. To set a breakpoint on a different line, change the value "66" to the desired line number.




Fig. 7

**(gdb) info break**

This command is used to retrieve the status of user-defined breakpoints contained in the code and the breakpoint number.

 Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. 

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

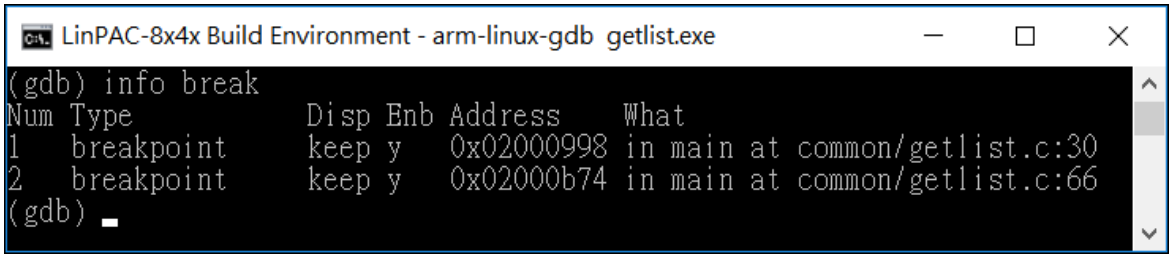



Fig. 8

**(gdb) c**

This command is used to continue debugging the program after a signal or breakpoint has been reached.

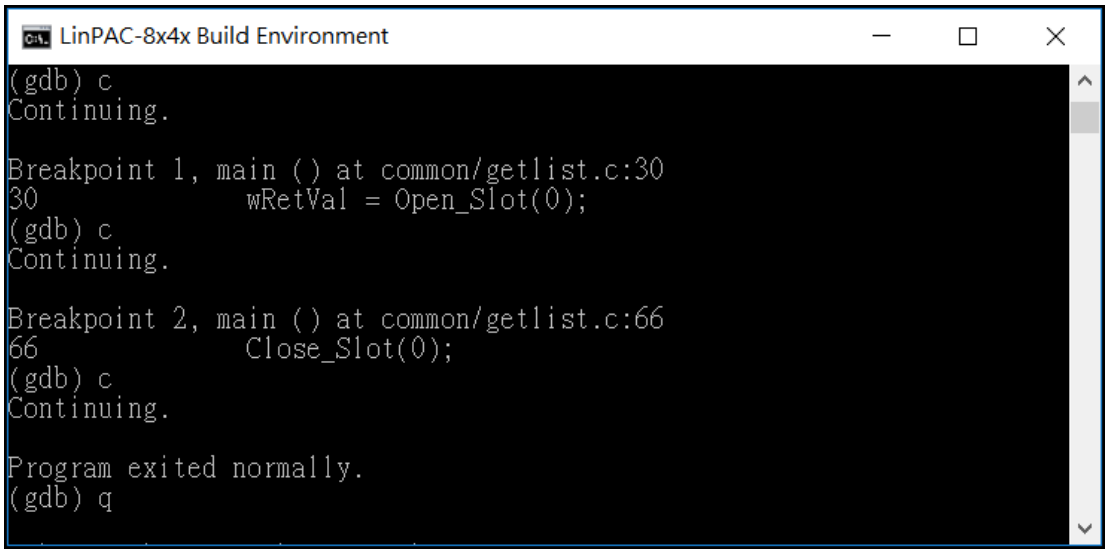



Fig. 9

Once the end of the program is reached, the output will be displayed on the target machine.

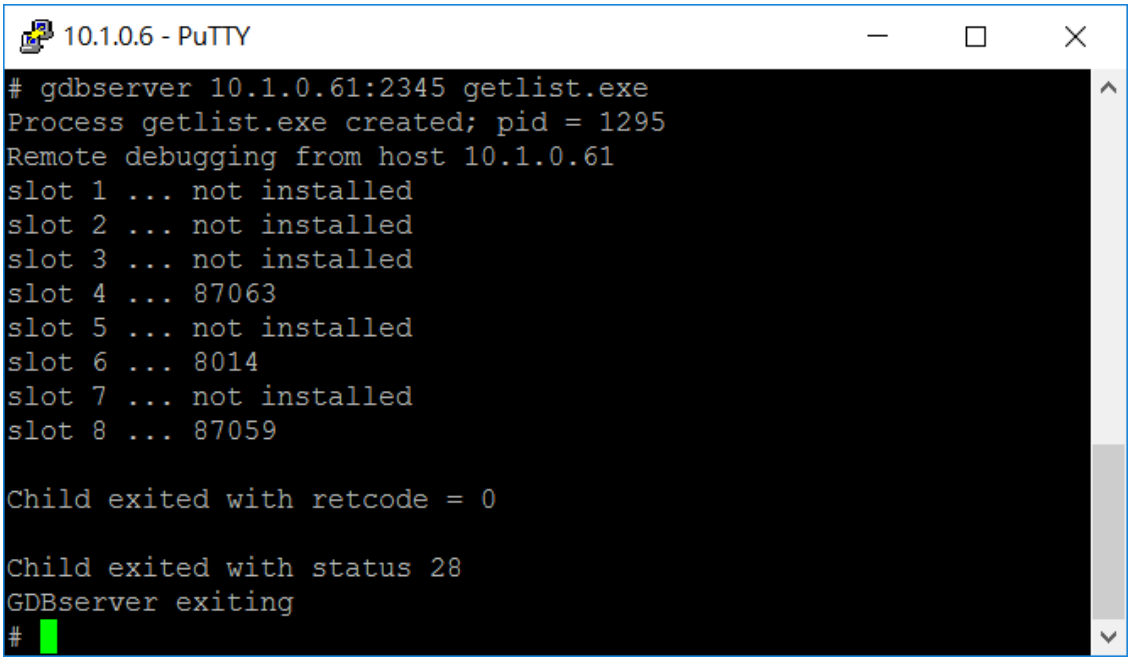



Fig. 10

 Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. 

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

## Section 2: Working with the Code::Blocks IDE interface

At the first time, user needs to make sure the LinPAC controller have gdbserver tool. In the LP-8x2x, user can type command '**apt-get update**' and '**apt-get install gdbserver**' for install gdbserver tool, as illustrated in Fig. 11 and 12 below.

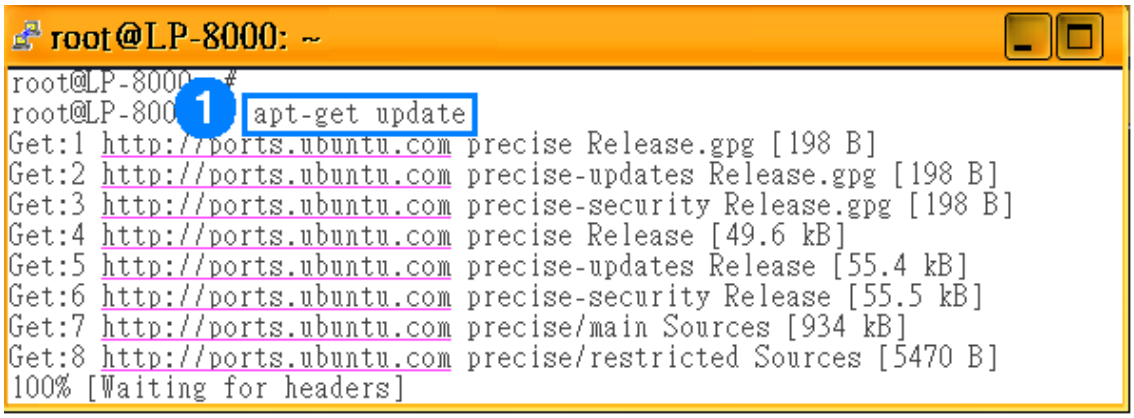



Fig. 11

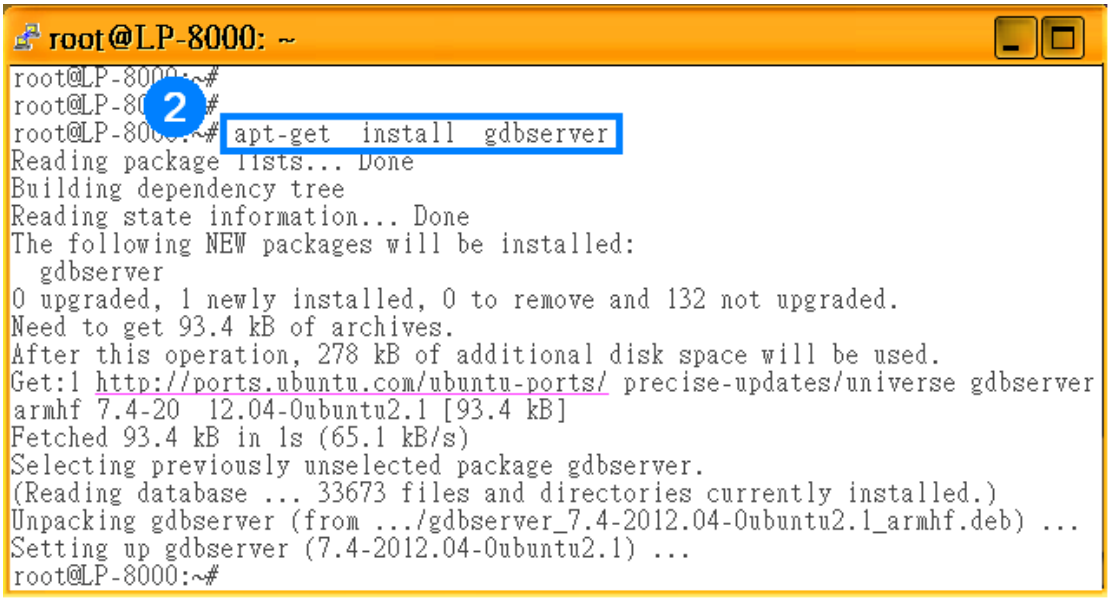



Fig. 12

The following C program example will be used to demonstrate the remote debugging on the LP-8x2x with Code::Blocks IDE platform (http://www.codeblocks.org/home).

**Step 1**: To create a new project for remote debugging.

Click on the **File** pull-down menu, open **New** and then **Project**, and it will bring up the New from template window, as illustrated in Fig. 13 below.

 Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. 

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

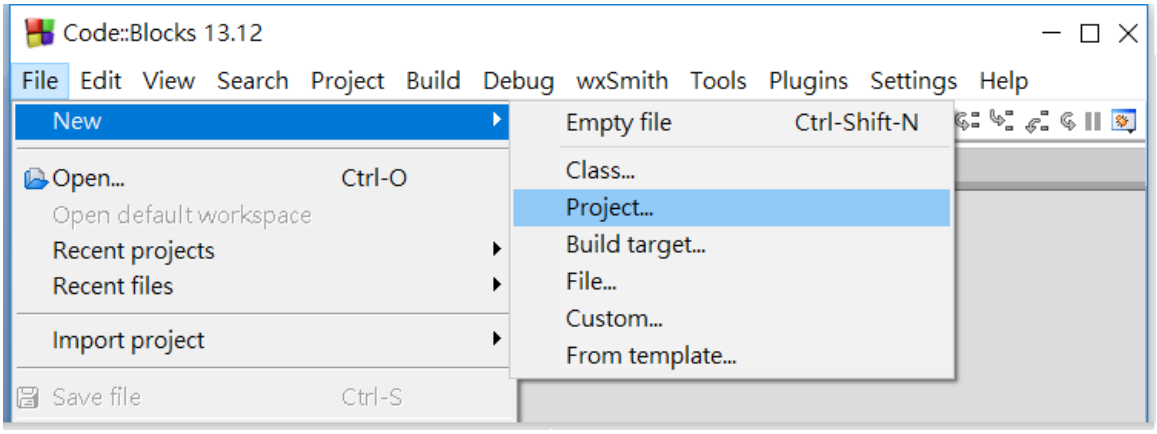



Fig. 13

Opening (clicking on) **Console Application** will then allow you to write a program on the console. The next window allows you to choose the language that you will use. Select the language as **C**, then press Finish, as illustrated in Fig. 14 below.

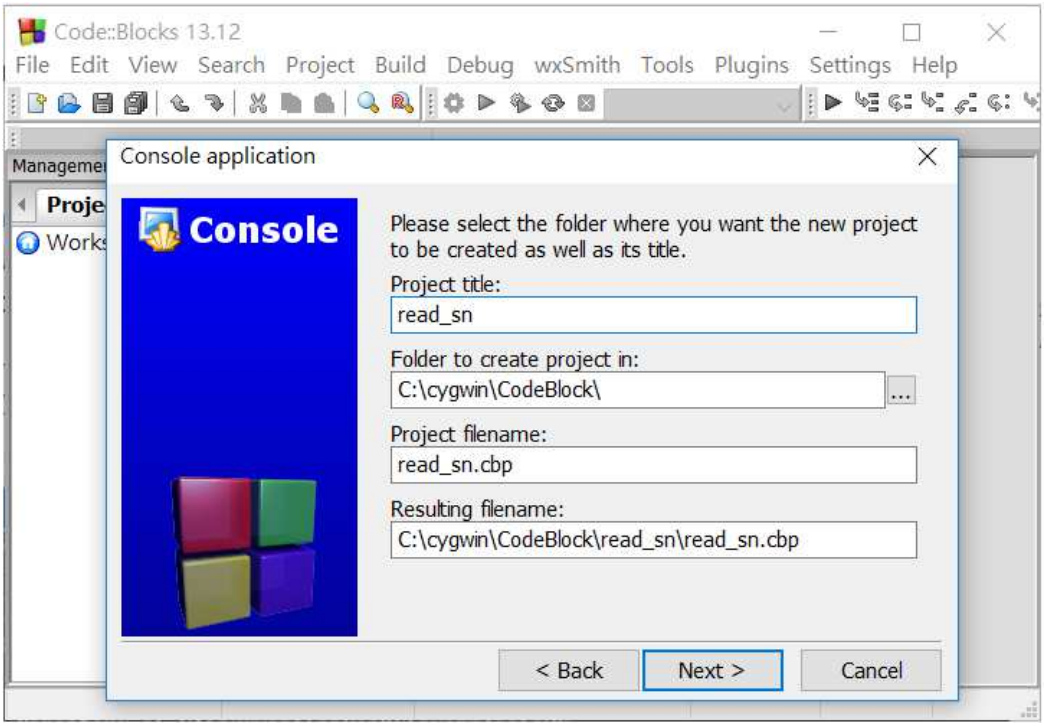



Fig. 14

**Step 2**: Setup a custom makefile.

To copy and modify '**Makefile**' file from C:\cygwin\LP-8x2x_SDK\ folder to debugger project (C:\cygwin\CodeBlock\read_sn) folder. Compile the C program with debugging option **-g**, as illustrated in Fig. 15 below.

 Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. 

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

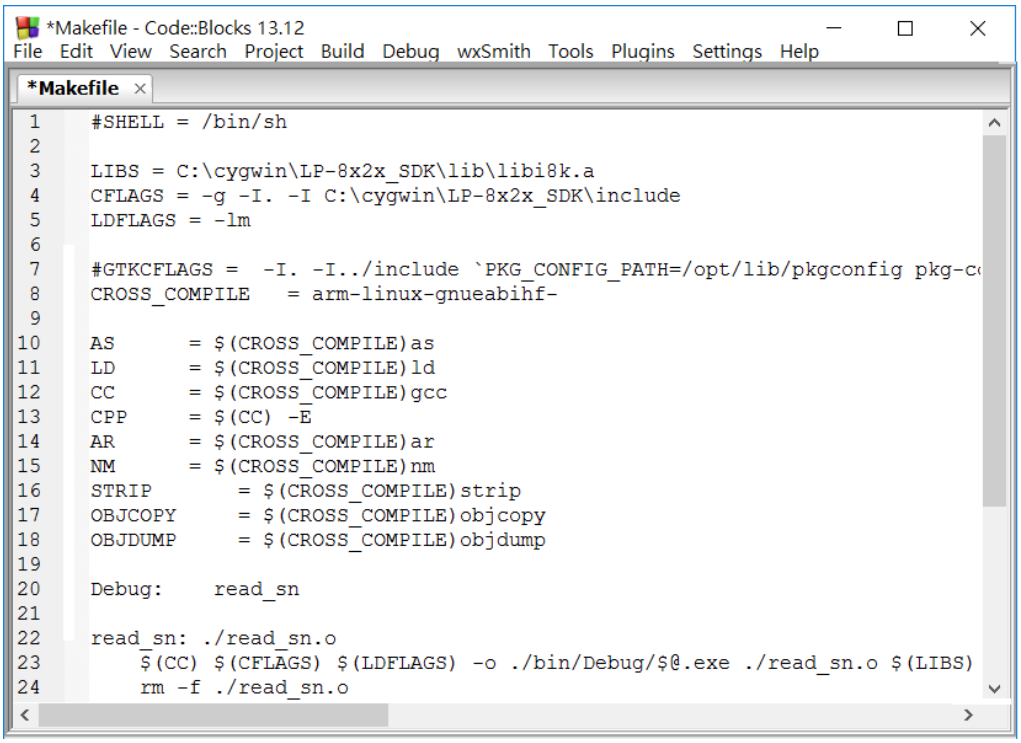



Fig. 15

Open '**Project--> Properties**' to access the main properties of the active project. And user will see a tick box for 'this is a custom makefile'.

Tick this box, make sure the name just above it is the one you want for your makefile, and change the current working directory, as illustrated in Fig. 16 below.

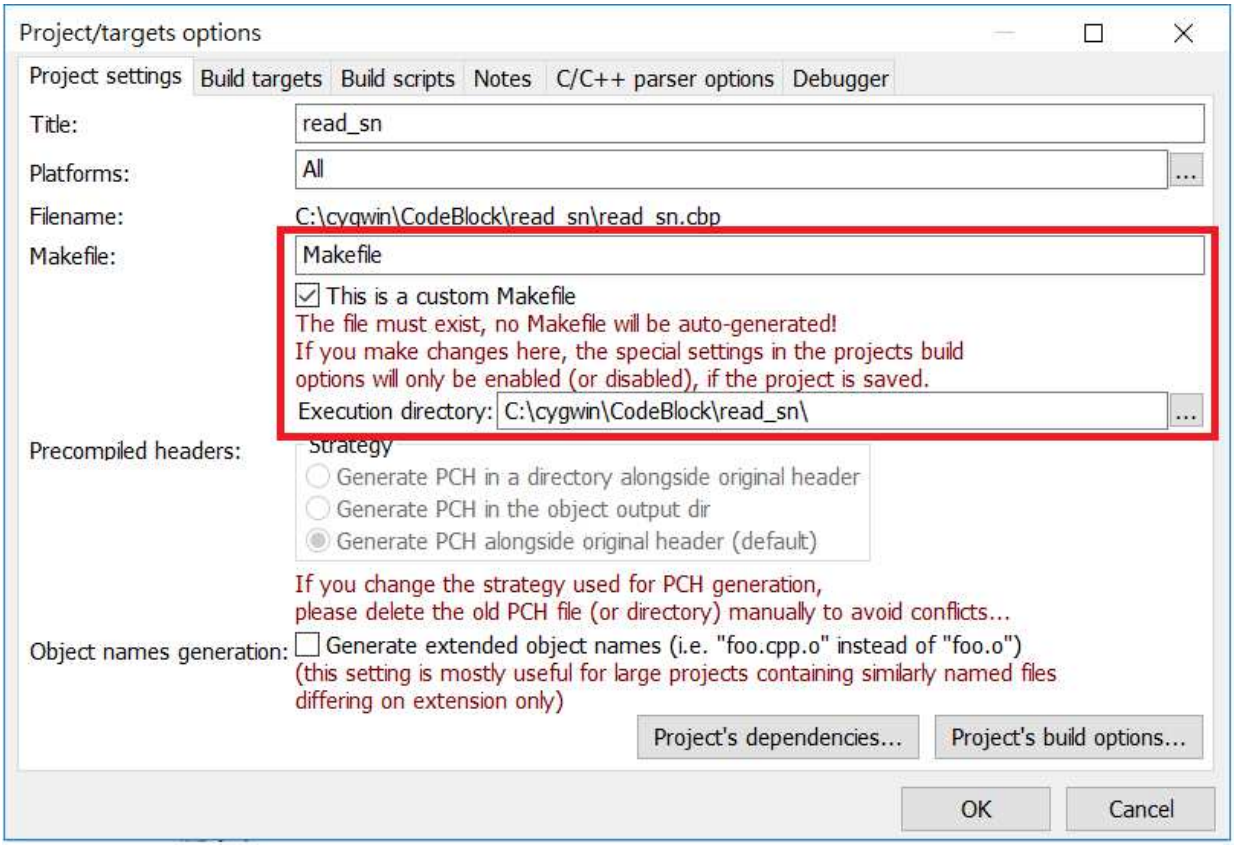



Fig. 16

 Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. 

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

In addition to customize your IDE, user also double click the default debug project file - read_sn, as illustrated in Fig. 17 below.

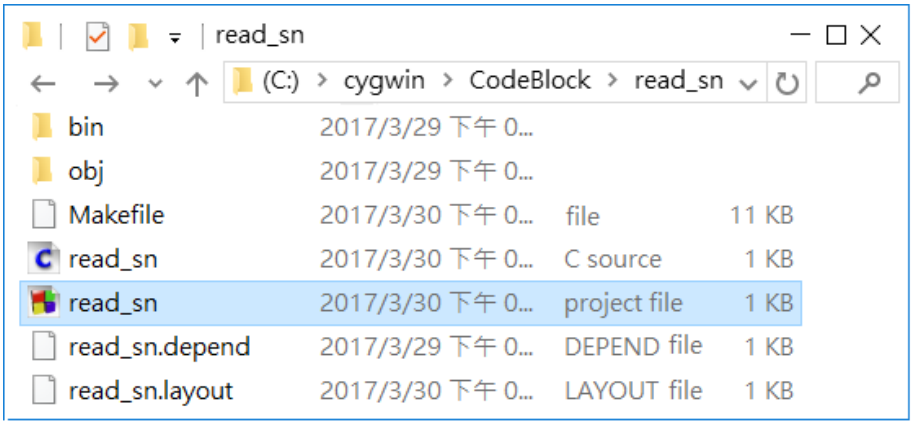



Fig. 17

**Step 3**: Click '**Settings->Debugger**' to make sure the executable path of default debugger is set to **arm-linux-gnueabihf-gdb.exe** (as illustrated in Fig. 18).

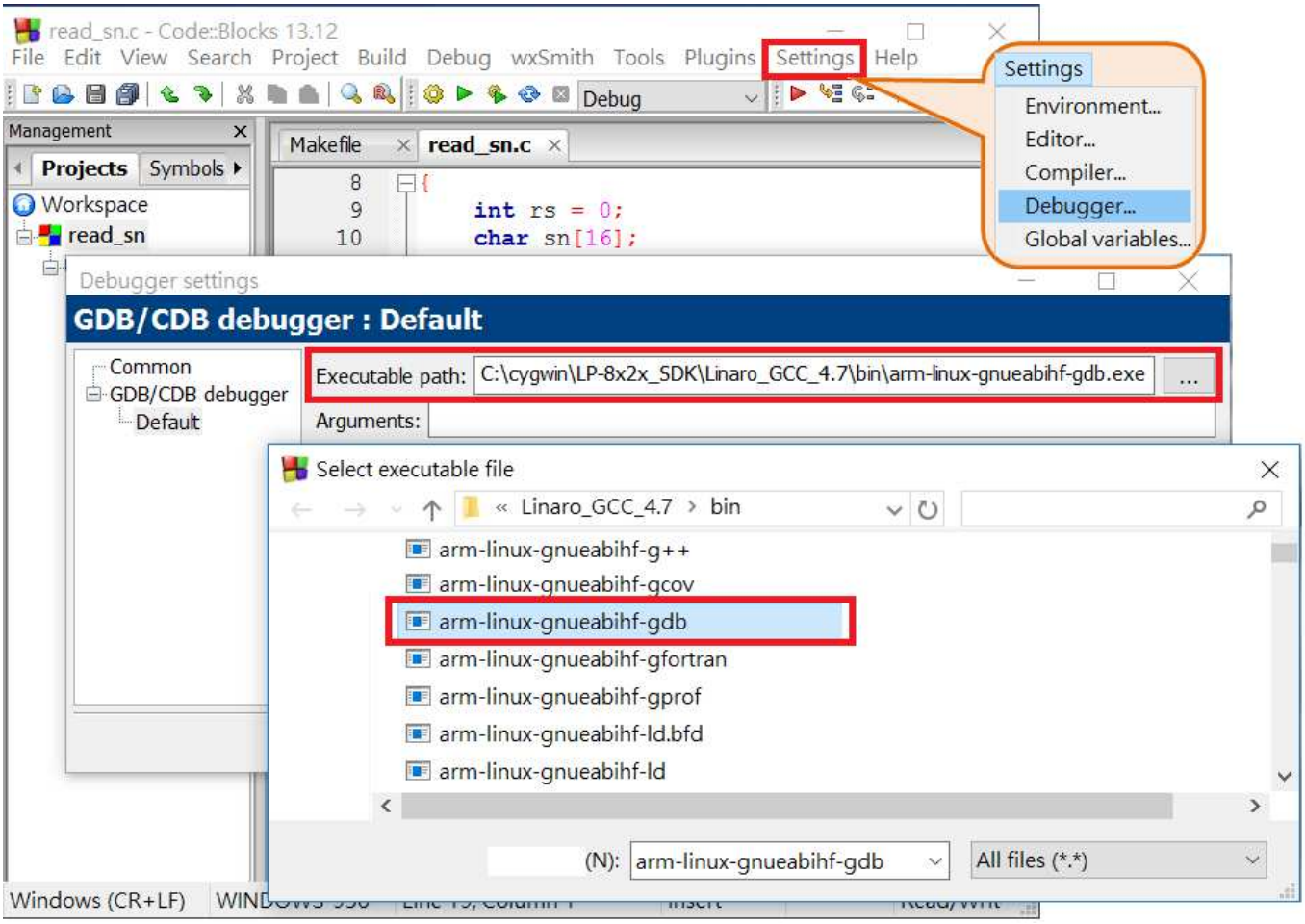



Fig. 18

 Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. 

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

**Step 4**: As your programs become more complicated, there will be a need to trace the program execution step by step or place break points where you wish the program to pause, as illustrated in Fig. 19 below.

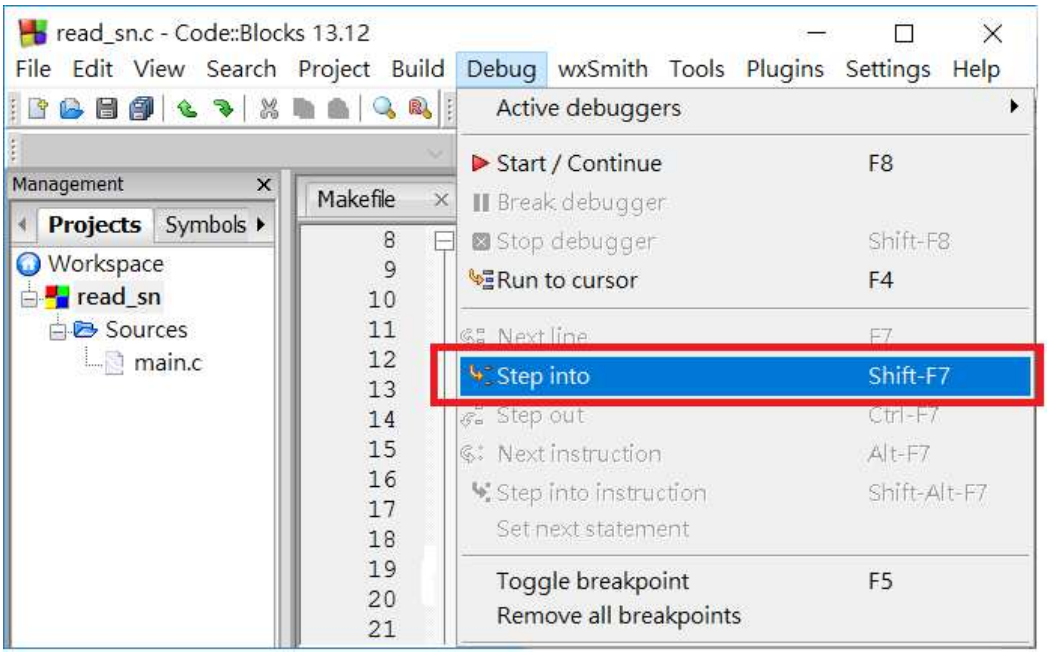



Fig. 19

**Step 5**: Make sure the remote debugger is running on the target machine.

To start remote debugging, a debugger running on host machine (local PC) connects to a program which is running on the target (LinPAC controller) via network (as illustrated in Fig. 20 and 21), that can be traced "line by line" while watching what happens as each line of code is executed(as illustrated in Fig. 22), type the command as follows:

- On target system (LP-8x2x): gdbserver 10.1.0.37:1234 read_sn.exe
- On host system (local PC): target remote 10.1.0.109:1234

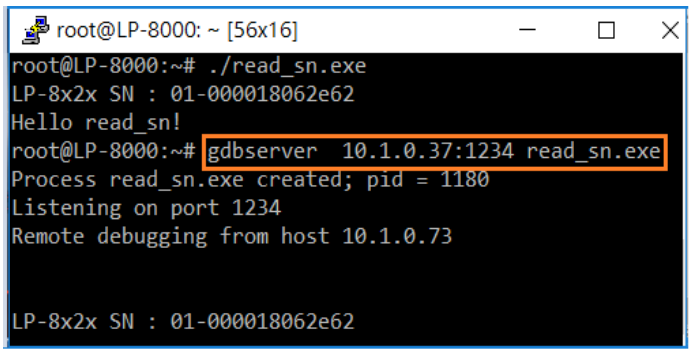



Fig. 20

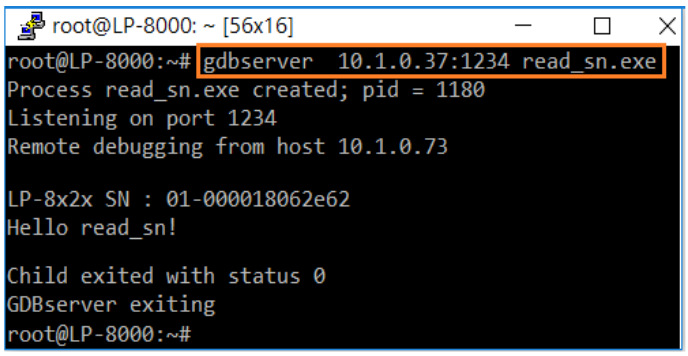



Fig. 21

 Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. 

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

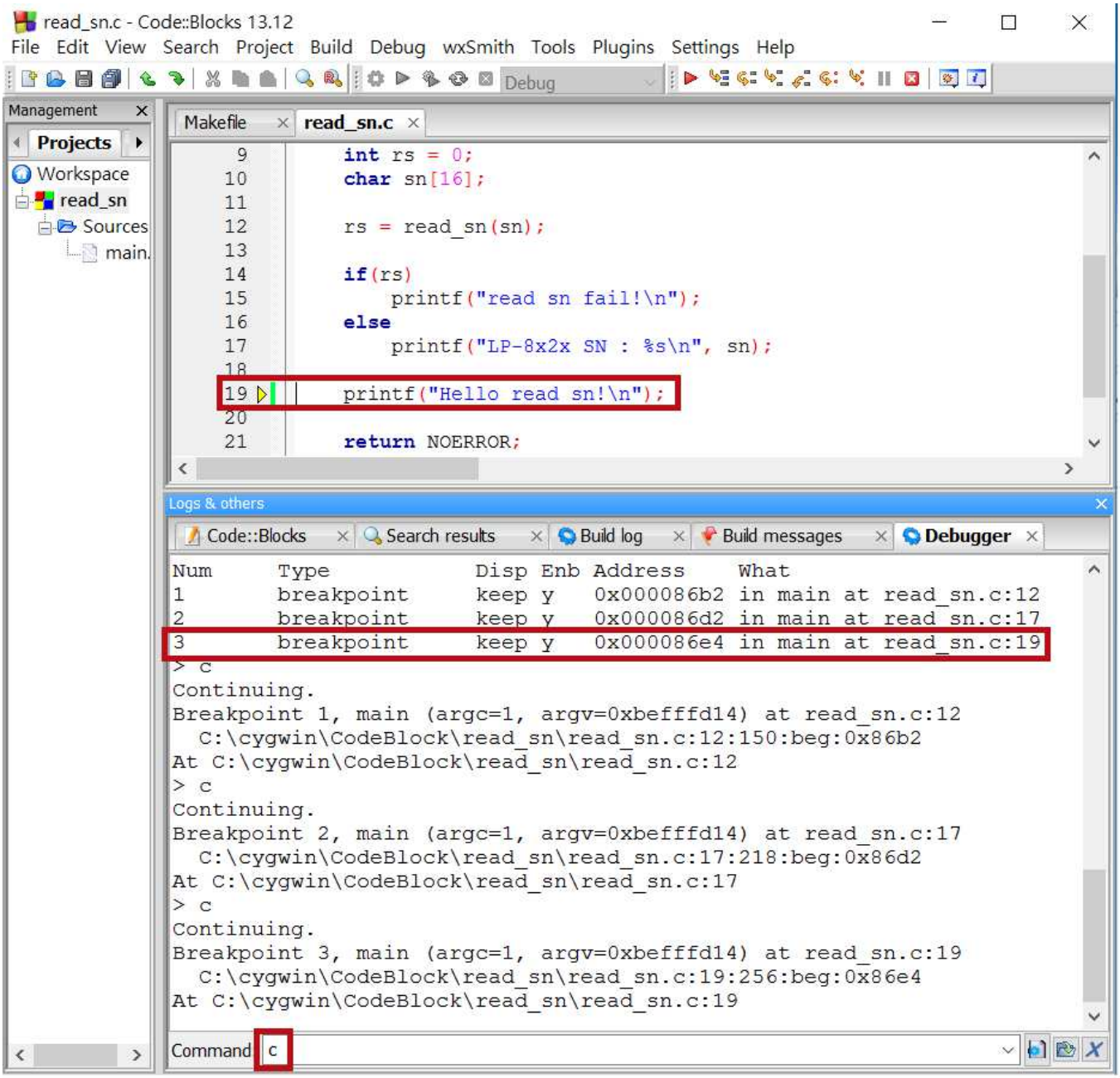



Fig. 22

 Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. 

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000