

Fido

How to create .ini files

This material, including documentation and any related computer programs, is protected by copyright controlled by Nokia. All rights are reserved. Downloading and copying of the material for use with the Fido tracebox is permitted. Redistribution of any or all of this material requires a prior written consent of Nokia. This material also contains confidential information, which may not be disclosed to others without the prior written consent of Nokia.

CONTENTS

1. INTRODUCTION	4
1.1. How to read this document	4
1.2. What is the Fido .ini file	4
1.3. How Fido handles its .ini file	4
1.3.1. How to specify the .ini file on the command line	4
1.3.2. How Fido uses the information stored in the .ini file	5
1.3.3. Read-only .ini files	5
1.3.4. Specific use case: the .ini file is missing or incorrect	5
1.3.5. Specific use case: several software instances running on the same computer	5
1.4. When you need to write your own .ini file	6
2. .INI FILE CONTENTS	7
2.1. Some general rules	7
2.2. Example	7
2.3. .ini file version	9
2.4. Description of Fido boxes	9
2.5. Description of ports	10
2.5.1. Identification and general parameters	10
2.5.2. The <i>interfaces</i> block	11
2.5.3. The <i>protocols</i> block	12
2.5.4. Connecting Fido boxes to ports	14
2.6. Description of configuration	15
3. APPENDIX: PATTERN	17
4. REFERENCES	19

CHANGE HISTORY

Version : 1.0
Status : Draft
Date : 22-Aug-2011
Owner : Viktor Leppikson / Liewenthal Electronics Ltd.
Comments : Document was created.

Version : 1.1
Status : Draft
Date : 23-Aug-2011
Owner : Viktor Leppikson / Liewenthal Electronics Ltd.
Comments : Upgraded to cover the .ini file version 8

1. Introduction

1.1. How to read this document

When you read this document, you may meet unknown terms or not understandable sections. Therefore it is advisable to read it with open Fido Help /3/. Below we'll rather often refer to Fido Help and controls located on the panels of Fido Graphical User Interface. Remark that Fido Help is online: simply move the focus onto a control and press *F1*. In addition, the most of controls have tooltips.

If you are in hurry, see the pattern from 3 first. If you find it suitable, you need not to read chapter 2. The pattern is supplied with comments explaining how to edit it to get an .ini file exactly matching your use case.

1.2. What is the Fido .ini file

When you work with Fido, you connect Fido boxes to Fido software ports (i.e. you create a tracing network) and select values for various setup parameters. The Fido .ini file (or configuration file) is an ASCII text file that stores the description of you network and the values of your setup parameters.

1.3. How Fido handles its .ini file

1.3.1. How to specify the .ini file on the command line

The default .ini file is named as *Fido.ini* in Windows and *fido.ini* in Linux. In Windows the .ini file default folder is *My Documents\Liewenthal\Fido*. In Linux the ini file default folder is specified by the *\$FIDO_USER_CONFIGURATION_HOME* environment variable (about details read /4/). Fido installation procedure creates the default folder automatically.

If you start the Fido software without specifying the configuration file, the default .ini file located in the default folder is assumed. To specify another .ini file, use the *-name* or *-file* command line parameters:

-name = <filename>

is applied when the specified .ini file is located in the Fido default folder.

-file = <path to configuration file>

is applied when the .ini file is located in some other folder. File specified with relative path or without path is searched relatively to the current folder.

In both cases the .ini extension may be omitted.

Examples:

1. Configuration file is *MyConfig.ini* located in the default folder
Fido -name=MyConfig
2. Configuration file is *test.ini* located in folder *c:\data*
Fido -file=c:\data\test.ini
3. Default configuration file is used:
Fido

1.3.2. How Fido uses the information stored in the .ini file

When you exit the Fido software, the current tracing network description (i.e. which Fido boxes were used, how they were connected to software ports, etc.) and the current values of setup parameters are stored in the .ini file (see also 1.3.3). When you next time launch the Fido software, Fido tries to restore the previous state of your work:

- the setup parameters are assigned with values read from the .ini file;
- if possible, Fido boxes are connected to the ports according to the network description retrieved from the .ini file¹.

1.3.3. Read-only .ini files

As you could see from 1.3.2, on the end of tracing session right before the exit Fido overwrites the .ini file. To obstruct it, you may

- Using the operating system facilities declare the .ini file as read-only.
- Insert the *readonly;* expression into the .ini file (see example in 2.2).
- When launching the Fido software, use the *-ro* command line parameter. In that case the read-only feature is valid only during the current session.

1.3.4. Specific use case: the .ini file is missing or incorrect

For Fido, an incorrect or missing .ini file does not cause instant exit. If the .ini file does not exist or is unreadable (for example, due to serious syntactic errors), Fido acts in the following way:

- all the setup parameters are assigned with default values;
- as there is no any information about previous connections, attempts to restore the tracing network are skipped.

If the .ini file contains errors but is readable, Fido ignores the incorrect sections and completes the missing information using the approach presented above.

Report about .ini file abnormalities is shown in the Graphical User Interface logbook window². After that Fido is ready to work.

If the original .ini file was missing and the *-ro* command line parameter was not used, the tracing session ends with creating a new .ini file. Remark, however, that the folder for the new file must exist because Fido does not create new folders. Exception: the missing Fido default folder will be created.

If the original .ini file was incorrect, Fido overwrites or not overwrites it as presented in 1.3.2 and 1.3.3. Remember that if the original file is incorrect, the

readonly;
expression may remain unnoticed.

1.3.5. Specific use case: several software instances running on the same computer

When two or more Fido instances are concurrently running on the same computer and each of them has its own configuration file then the following aspects are important:

¹ The overview of tracing network building procedure is in /3/ chapters *Automatic configuring* and *Manual configuring*. See also 2.5.4.

² See /3/ chapter *Logbook*.

- Each software instance must have its own TCP/IP port. 7654 is the default port, but only one of the concurrently running instances may use it. You may specify the TCP/IP port in the .ini file (see 2.6) or as the command line parameter³.
- Generally, the software instances operate independently and do not disturb each other. However, there is an exception. When you launch a Fido software instance, it uses the information from .ini file to establish connections between Fido boxes and ports (see 2.5.4). If you do not turn attention to this information, it may happen that one of the Fido instances grabs the Fido box you wanted to reserve for another instance. In other words, if you have several concurrently running Fido instances and several Fido boxes connected to your computer, you must very exactly specify your tracing networks.
- Naturally, different .ini files may use the same names, port IDs, etc. For example, if you have two instances with two .ini files, the both of them may contain port with ID 0x01. See also 2.5.1.

Several software instances may even use one common .ini file. In that case the different TCP/IP ports can be specified only on the command lines. Merely the instance that had started first has the right to overwrite the .ini file (see 1.3.2).

1.4. When you need to write your own .ini file

Generally, you may always prepare your tracing network and select the proper values for setup parameters using the Fido Graphical User Interface facilities. Sometimes, however, this way is too awkward. If you need to be sure that right after Fido software startup you will get the proper tracing network and settings (and consequently, you can start tracing immediately), you should modify the .ini file or create a new one yourself.

³ See /3/ chapter *Command line startup*.

2. .ini file contents

2.1. Some general rules

An .ini file consists of blocks surrounded by braces. The blocks include parameter expressions and/or other blocks. The parameter expressions are formatted as:

```
<parameter name> = <value>;
```

or

```
<parameter name> = <sequence of values>;
```

A parameter value may be a number, word or asterisk. The components of a sequence are separated by commas. A sequence of numbers may also include intervals. For example, the 1, 3, 4, 5, 8 sequence may be written as 1, 3 – 5, 8.

Fido .ini files have free-form format – you can place the components of your text as you like (exactly as in C-programs). The order of blocks and parameter expressions inside blocks is free. Do not forget, however, that if the .ini file is not read-only (see 1.3.3), right before the exit Fido overwrites it. The order of blocks and parameter expressions and the formatting (division of text into rows, etc.) in the original and in the overwritten file may not match.

The .ini file may contain comments. A comment must start with #. The end of line marks the end of comment. The comments are not kept – the overwritten .ini file does not contain them.

The overwritten file may contain some new blocks and/or parameter expressions that you did not include into your original. You should not turn any attention to them. Everything you definitely need to know to create your own .ini files is presented in this document. To obstruct overwriting use read-only.ini files.

2.2. Example

The example below presents an .ini file describing a tracing network that consists of one port and one Fido box. Sections from this example are used below to explain the rules that .ini file components must follow. The pattern you may apply for creating your own .ini files are in 3.

```
version = 8;                                # see 2.3

readonly;                                   # see 1.3.3

tracebox Tracebox1 {                       # see 2.4
    id = 0014E64B99902A75;
}

port Port1 {                                # see 2.5.1
    id = 1;                                  # see 2.5.1
    state = open;                           # see 2.5.1
    target_detection = 1;                   # see 2.5.1

    interfaces {                             # see 2.5.2
```

```

mipi60 {
    interface = dedicated;
    interfaces = *;
    protocol = xtiv3;
    protocols = *;
}
xti {
    protocol = xtiv2;
    protocols = *;
}
sti {
    protocol = sti;
    protocols = sti;
}
}

protocols {                                     # see 2.5.3
    xti* {
        max_message_pause = 100000;
        min_start_pause = 1000;
    }
    xtiv3 {
        return_channel = uart;
        swd {
            processors = default;
        }
        masters {
            default = big-endian;
        }
    }
    stpv2 {
        funneling {
            sources = 64 - 79;
            state = enabled;
        }
    }
    sti* {
        max_message_pause = 100000;
        min_start_pause = 20000;
        min_xmit_pause = 1000;
    }
    csti {
        adsp_csid_check = 1;
    }
}

traceboxes = *, Tracebox1;                     # see 2.5.4
}

```

```
fido fido {
  ip {
    port = 7654;
  }
  portcount = 8;
  ports = *, Port1;
  musti {
    ch1 = Port1;
  }
}
```

2.3. .ini file version

The current .ini file version is 8 and it is compatible with software version 1.9.0.1 and higher. Fido follows the forward compatibility concept: .ini files of older versions are always accepted and the overwritten .ini files already match the new standard.

2.4. Description of Fido boxes

When you connect a Fido box to your computer, Fido software detects it and then checks does the .ini file already contains the description of this box. If not, it creates a description block like this:

```
tracebox Tracebox1 {
  id = 0014E64B99902A75;
}
```

Here the word *tracebox* is the block ID. *Tracebox1* is the Fido box name⁴ and parameter *id* presents the box unique serial number⁵.

Thus, the .ini file stores the list of all the boxes you have ever used.

Names *Tracebox1*, *Tracebox2*, etc. are assigned by Fido software. If you do not like them you may use the Fido Graphical User Interface facilities and assign another names⁶. If you modify a .ini file or write a new one, you may also use any names. Remember that the names of Fido boxes registered in the same .ini file must be unique. To see the box names without opening the .ini file click on the Graphical User Interface connection tree nodes presenting a port or free device.⁷

When you edit the .ini files, you may create new tracebox blocks describing the Fido boxes you have. You may also delete the existing tracebox blocks. You should, however, keep the names of traceboxes in accordance with the information presented in the description of ports (see below).

⁴ Read /3/ chapter *Box name*.

⁵ Read /3/ chapter *Serial number*

⁶ Read /3/ chapter *Box renaming*.

⁷ Read /3/ chapter *Connection tree*.

2.5. Description of ports

2.5.1. Identification and general parameters

When the automatic configuring feature is allowed, Fido creates the ports automatically⁸. Ports may be also created manually using the Graphical User Interface⁹ or commands sent from external applications (details are in /2/). When you edit .ini files, you may create new port blocks and delete the existing ones.

In the .ini file each port is presented by port description block. This block includes the *interfaces* block, *protocols* block and several parameter expressions:

```
port Port1 {
  id = 1;
  state = open;
  target_detection = 1;
  .....
}
```

Here the word *port* is the block ID. *Port1* is the port name. If Fido creates a port automatically, it uses names like *Port1*, *Port2*,... If you create a new .ini file or edit an existing one, you may select any names you like. Remember that the names of ports registered in the same .ini file must be unique. You must keep the names of ports in accordance with the information presented in the *fido* block (see 2.6).

The Graphical User Interface does not show the port names. On the connection tree¹⁰ the ports are identified by numbers *01*, *02*, ... Those numbers are the port IDs¹¹. The

id = 1;

expression means that the port ID is *01*. Ports registered in the same .ini file must have unique ID. Fido inserts the port ID into the output data (about output data formats read /1/). All the commands from the Graphical User Interface and external control applications also use the port ID (read /2/).

The

state = open; *# Alternative: state = closed;*

expression means that the port is open¹² (i.e. the data flow through the port is allowed).

The

target_detection = 1; *# Alternative: target_detection = 0;*

expression means that Fido must detect the target presentation by the voltage on the far end of connection cable¹³.

⁸ Read /3/ chapter *Automatic configuring* and 2.6.

⁹ Read /3/ chapter *Manual configuring*.

¹⁰ Read /3/ chapter *Connection tree*.

¹¹ Read /3/ chapter *Ports*.

¹² Read /3/ chapters *Connection tree* and *Start/stop data flow*.

¹³ Read /3/ chapter *Target presence detection*

2.5.2. The *interfaces* block

Fido box can be connected to traced device via

- MIPI 60-pin QSH Trace Cable (see /5/).
- Cables designed to transport the both XTI and STI data, for example /6/ and /7/.
- Cables designed to transport only STI data.

The protocols¹⁴ that the trace data may follow are:

- XTI version3 / STP version 1 (the .ini file keyword is *xtiv3*).
- XTI version 3 / STP version 2 (the .ini file keyword is *stpv2*).
- XTI version 2 (the .ini file keyword is *xtiv2*).
- ST-XTI (the .ini file keyword is *stxti*).
- STI (the .ini file keyword is *sti*).
- C-STI (the .ini file keyword is *csti*).

If the MIPI-60 pin cable is used, two different pin sets¹⁵ are possible:

- Dedicated
- Muxed

The trace data protocol and pin set are Fido setup parameters and the user may set their values. The both parameters, however, have also value *Auto*. It means that Fido has to detect the proper trace data protocol¹⁶ and/or pin set itself.

The contents of *interfaces* block specifies the values for the above-mentioned two setup parameters. It specifies also the Fido behaviour in case of automatic detection. The block includes *mipi60* (for MIPI 60-pin QSH Trace Cable), *xti* (for cables designed to transport the both XTI and STI data) and *sti* (for cables designed to transport only STI data) inner blocks.

```
interfaces {
    mipi60 { ..... }
    xti { ..... }
    sti { ..... }
}
```

If you know the type of your cable, write only the block you do need.

```
mipi60 {
    interface = dedicated;
    interfaces = *;
    protocol = xtiv3;
    protocols = *;
}
```

Asterisk marks the automatic detection mode. In that case the *interface* and *protocol* expressions present the starting point of detection. Thus, in the current example Fido suggests that the pin set is dedicated and the protocol is XTI version3 / STP version 1. If problems appear, Fido switches over to another suggestions.

¹⁴ Read /3/ chapter *Supported protocols*.

¹⁵ Read /3/ chapter *Pin sets*.

¹⁶ Read /3/ chapter *Protocol detection*.

If the user wants to select the pin set and/or trace data protocol himself, the both *interface* and *interfaces* as well as the both *protocol* and *protocols* must specify the selected value. Example:

```
mipi60 {
    interface = muxed;    # no pin set automatic detection, the pin set is muxed
    interfaces = muxed;
    protocol = xtiv2;     # no protocol automatic detection, the protocol is XTI version 2
    protocols = xtiv2;
}

```

The *xti* and *sti* inner blocks follow the same rules. Examples:

```
xti {
    protocol =xtiv2;     # protocol automatic detection, XTI version 2 is the first suggestion
    protocols = *;
}

sti {
    protocol = sti;     # no automatic detection, the protocol is STI
    protocols = sti;
}

```

2.5.3. The *protocols* block

The *protocols* block presents the port setup parameters except the trace data protocol and MIPI 60-pin cable pin set specified in the *interfaces* block. The block includes the *xti** (parameters for all the XTI protocols), *xtiv3* (parameters for the XTI version 3 / STP version 1 and 2 protocols only), *stpv2* (parameters for the XTI version 3 / STP version 2 protocol only), *sti** (parameters for the both STI protocols) and *csti* (parameters for the C-STI protocol only) inner blocks.

```
protocols {
    xti* {.....}
    xtiv3 {.....}
    stpv2 {.....}
    sti* {.....}
    csti {.....}
}

```

You must include into your .ini file only those inner blocks that you do need. For example, if you know that you will work with XTI version 3 / STP version 1 protocol, you may omit the *stpv2*, *sti** and *csti* inner blocks.

The *xti** inner block specifies the extreme values for pauses in XTI trace data stream¹⁷:

```
xti* {
    max_message_pause = 100000;
    # Maximal pause allowed between trace data messages during composing of PhoNet or OST messages
    # unit: μs. 100000 is the factory default
    min_start_pause = 1000;
    # Minimal pause in trace data stream needed for resynchronization after data errors
}

```

¹⁷ Read /3/ chapter *Pauses in trace data stream.*

```
# unit:  $\mu$ s. 1000 is the factory default
}
```

The *xtiv3* block contains the *masters* and *swd* inner blocks and the

```
return_channel = uart; # Alternative: return_channel = swd;
```

parameter expression specifying whether the user wants to apply the SWD technology¹⁸. Remark, that in the other Fido documents instead of the *return channel* the *reverse channel* term is used.

The *swd* block must contain one of the expressions presented below:

```
swd {
    processor = 1; # to specify the index of processor selected by the user
}
```

or

```
swd {
    processors = default; # Fido has to select the processor with which it will communicate
}
```

If the *swd* block must contain the both expressions like

```
swd {
    processor = 1;
    processors = default;
}
```

the *default* mode is valid and processor index is ignored.

If you are not going to utilize the SWD technology, you may omit this block.

The *masters* block specifies the masters STP endianness¹⁹ and filtering²⁰. If all the masters provide big-endian or little endian transport and the filtering are suppressed, this block contains only one expression:

```
masters {
    default = big-endian; # Alternative: default = little-endian
}
```

If there are masters providing non-default transport and/or the output data of some masters must be discarded (i.e. their filters must be turned to state *on*), you have to list those particular masters. Example:

```
masters {
    default = big-endian;
    little-endian = 0, 2 - 4, 6; # masters 0, 2, 3, 4 and 6 provide little endian transport
    filter = 7 - 255; # discard output data of masters with index 7 and higher
}
```

The *stp2* block contains the *funneling*²¹ inner block that specifies the used trace sources and the trace formatter state. Example:

¹⁸ Read /3/ chapter *Serial wire debug*.

¹⁹ Read /3/ chapter *STP endianness*.

²⁰ Read /3/ chapter *Filters*.

²¹ Read /3/ chapter *Trace formatter and trace sources*.

```

stp2 {
    funneling {
        sources = 64 - 79;           # sources 64, 65, ..., 79 are in use
        state = enabled;           # Alternative: state = disabled;
    }
}

```

The *sti** inner block specifies the extreme values for pauses in STI trace data stream²²:

```

sti* {
    max_message_pause = 100000;
    # Maximal pause allowed between trace data messages during composing of PhoNet or OST messages
    # unit: μs. 100000 is the factory default
    min_start_pause = 20000;
    # Minimal pause in trace data stream needed for resynchronization after data errors
    # unit: μs. 2000 is the factory default
    min_xmit_pause = 1000;

    # Minimal allowed pause between bytes transmitted from Fido to traced device
    # unit: μs. 1000 is the factory default
}

```

The *csti* block specifies whether Fido has to use the cellular system identifier²³.

```

csti {
    adsp_csid_check = 1;           # Alternative: adsp_csid_check = 0;
}

```

2.5.4. Connecting Fido boxes to ports

Fido .ini file stores the information about Fido boxes (see 2.4) and about ports. It stores also the information about connections between boxes and ports. When you start a new tracing session, Fido tries to establish the connections described in the .ini file.

The port connections are presented in the following way:

```
traceboxes = Tracebox1;
```

means that this port can be connected only with Fido box named as *Tracebox1*. If *Tracebox1* is connected to the computer, Fido automatically connects it to the current port. If *Tracebox1* is not found, the port stays not connected.

```
traceboxes = Tracebox1, Tracebox2;
```

means that this port can be connected only with Fido boxes named as *Tracebox1* or *Tracebox2*. If the both Fido boxes are found, the selection is random. If neither *Tracebox1* nor *Tracebox2* is found, the port stays not connected.

```
traceboxes = *;
```

means that when Fido detects not connected boxes (called also as free boxes), it will connect one of them to the current port.

²² Read /3/ chapter *Pauses in trace data stream..*

²³ Read /3/ chapter *Cellular system identifier(CSID) usage.*

```
traceboxes =*, Tracebox1;
```

means that if *Tracebox1* is connected to the computer, Fido automatically connects it to the current port. If *Tracebox1* is not found but there are some other free boxes, Fido connects one of them.

2.5.5. Redefining the port default settings

In most cases Fido creates the new ports automatically²⁴. Occasionally, the user may need to create a new port himself²⁵. By default, the new port, whether created automatically or manually, has setup parameters with factory default values. Would you like to get a new port with another settings, complete your .ini file with an additional a *port* block in which the port name is replaced by asterisk:

```
Port * {
.....
}
```

However, you cannot specify the new port ID (see 2.5.1) and connections to Fido boxes (see 2.5.4). The *id* and *traceboxes* expressions in the *Port ** block are ignored.

2.6. Description of configuration

The configuration description block presents some general parameters and lists the ports:

```
fido MyConfig {
  ip {
    port = 7654;
  }
  portcount = 8;
  ports = *, Port1;
  musti {
    chl = Port1;
  }
}
```

The .ini file must contain one and only one configuration description block. Word *fido* is the block ID and *MyConfig* is the configuration name. You can see the configuration name besides the Graphical User Interface connection tree root node²⁶. When Fido has created the .ini file, the configuration name matches with the filename. Would you like, you may select any other name.

The *ip* inner block specifies the Fido TCP/IP connection port²⁷. 7654 is the default value. Remember that several Fido software instances concurrently running on the same computer cannot have the same TCP/IP port (see also 1.3.5). TCP/IP port specified by the command line overrides the port specified in .ini file.

The *portcount* expression presents the allowed maximal number of ports. In the current version it must be 8. If you violate this rule, Fido behaviour may be unpredictable.

²⁴ Read /3/ chapter *Automatic configuring* and /2/.

²⁵ Read /3/ chapter *Port creating*.

²⁶ Read /3/ chapter *Connection tree*.

²⁷ Read /3/ chapter *Command line startup*.

The *ports* expression presents the list of ports (see 2.5). The asterisk means that Fido has the right to create new ports. If the asterisk is missing, you must create the new ports manually²⁸. Each port mentioned in the list must have its own description block.

The *musti* block specifies the ports that must operate in Musti mode²⁹. For example:

```
musti {      # none of the ports operates in Musti mode
}

musti {      # Port1 emulates Musti channel CH1
    ch1 = Port1;
}

musti {      # Port1 emulates Musti channel CH1 and Port2 emulates CH2
    ch1 = Port1;
    ch2 = Port2;
}
```

²⁸ Read /3/ chapter *Automatic configuring* and *Manual configuring*.

²⁹ Read /3/ chapter *Operating in Musti mode*.

3. Appendix: pattern

This pattern is helpful for those who have several Fido boxes connected to the computer but want to launch a Fido software instance that deals with only one particular of them. For example, suppose you have two traced devices and two Fido boxes. You want to launch two Fido software instances – one for tracing the first device and the other for tracing the second device. In that case you need two .ini files – one of them must contain the serial number of the first Fido box and the other the serial number of the second Fido box. You can use this pattern to for creating those files.

```
# Trace data protocol: XTI version 3 / STP version 1
# Trace cable: any except cables designed for STI protocols only
# Number of Fido boxes: 1, the serial number is known
# Number of ports: 1
# You want to start the tracing immediately, without any preliminary settings.
# Fido must automatically connect this particular box to the port.

version = 8;
readonly;          # It means that Fido will not overwrite your .ini file.
                  # If necessary, delete this expression.

tracebox Tracebox1 { # Tracebox1 is the name of your Fido box. You may use any
                    # other name.
    id = 019D31D224432355; # Replace with the serial number of your Fido box
}

port Port1 {

    id = 1;          # It means that your single Fido port ID is 0x01. You may use
                    # IDs 0x01...0x08. The ID identifies the port in all the
                    # commands.

    state = open;
    target_detection = 1;

    interfaces {

        mipi60 {      # If your trace cable is not MIPI 60-pin QSH, ignore this block
            interface = dedicated;
            interfaces = *;

                                # If you do know that your pin set is dedicated,
                                # those expressions should be
                                # interface = dedicated; interfaces = dedicated;
                                # If you do know that your pin set is muxed, those
                                # expressions should be
                                # interface = muxed; interfaces = muxed;
                                # If you are uncertain, do not edit them.

            protocol = xtiv3;
            protocols = xtiv3;
        }

        xti {          # If your trace cable is MIPI 60-pin QSH, ignore this block
            protocol = xtiv3;
            protocols = xtiv3;
        }
    }
}

protocols {
```

```

xti* { # These values here are the defaults. If necessary, replace them.
    max_message_pause = 100000;
    min_start_pause = 1000;
}

xtiv3 {
    masters {
        default = big-endian;
        # If you have some little-endian masters, list them.
        # For example:
        # little-endian = 0, 2 - 4, 6;
        # If you want to discard the output data from some
        # masters, list them. For example:
        # filter = 1, 5;
    }
    return_channel = uart;
}
}

traceboxes = Tracebox1; # It means that Fido can connect your port only with
# Fido box named as Tracebox1 (see above). If Fido
# detects the presence of this box, it automatically
# connects it to the port. If the box is not found,
# the port stays not connected.
}

fido PatternV8 { # Replace the name "PatternV8" with the name you
# have chosen for your .ini file

    ip {
        port = 7654; # If you have several Fido software instances
# concurrently running on the same computer, each
# of them must have its own TCP/IP port
    }

    portcount = 8;
    ports = Port1;
    musti {
        chl = Port1; # Necessary if you are using software designed
# for Musti. If not, this block is ignored.
    }
}
}

```

4. References

1. Fido Data Processing Reference. The PDF-version is downloadable from <http://www.liewenthal.ee/projects/fido/documents>
2. Fido Programmer's Reference. The PDF-version is downloadable from <http://www.liewenthal.ee/projects/fido/documents>
3. Fido Help. Accessible from the Fido Control Panel *Help* menu.
4. README for Linux. Accessible on <http://www.liewenthal.ee/projects/fido/download/>.
5. MIPI 60-pin QSH Trace Cable Datasheet. The PDF-version is downloadable from <http://www.liewenthal.ee/projects/fido/documents>
6. MIPI 34-pin FTSH Trace Cable Datasheet. The PDF-version is downloadable from <http://www.liewenthal.ee/projects/fido/documents>
7. MIPI MicroSD Trace Cable Datasheet. The PDF-version is downloadable from <http://www.liewenthal.ee/projects/fido/documents>