FLEXGUIUG
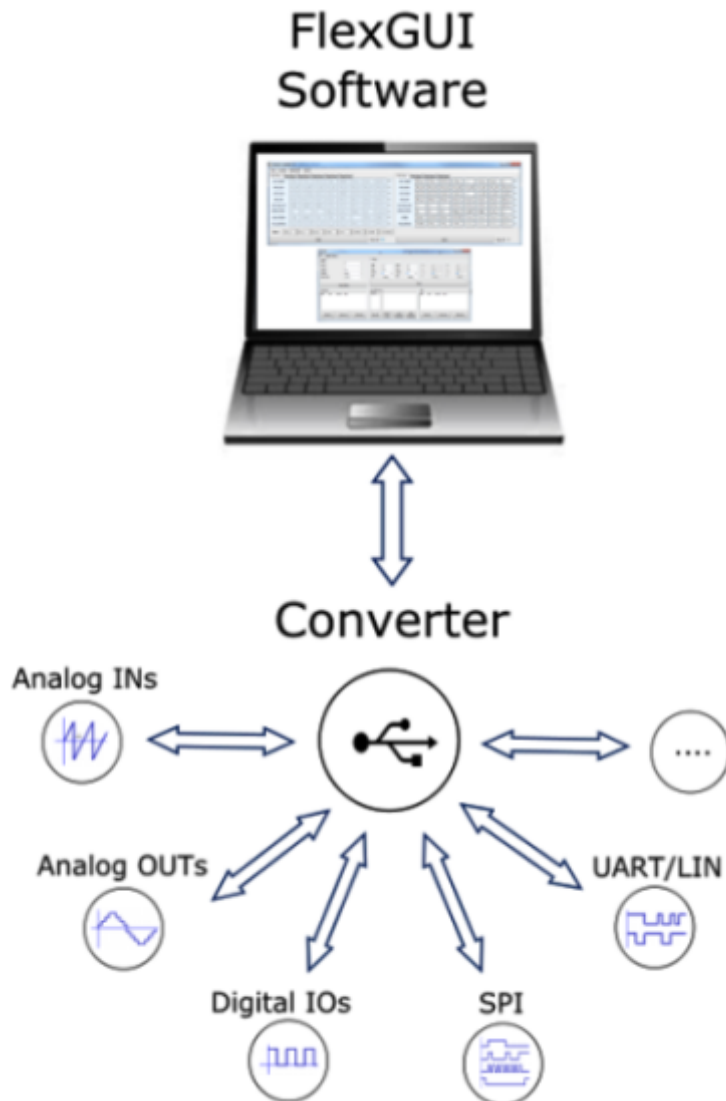
Flexible Graphical User Interface (FlexGUI)

Rev. 1 — 13 May 2016   **User guide**

1   Flexible Graphical User Interface Principle

FlexGUI
Software

Converter

Analog INs

Analog OUTs

Digital IOs

SPI

UART/LIN

2   Introduction

FlexGUI is a set of hardware and software tools that allow the user to program and configure a dedicated integrated circuit.

The hardware part (converter) converts analog and digital signals and different communication protocols to USB. It can function as a multiplexer:
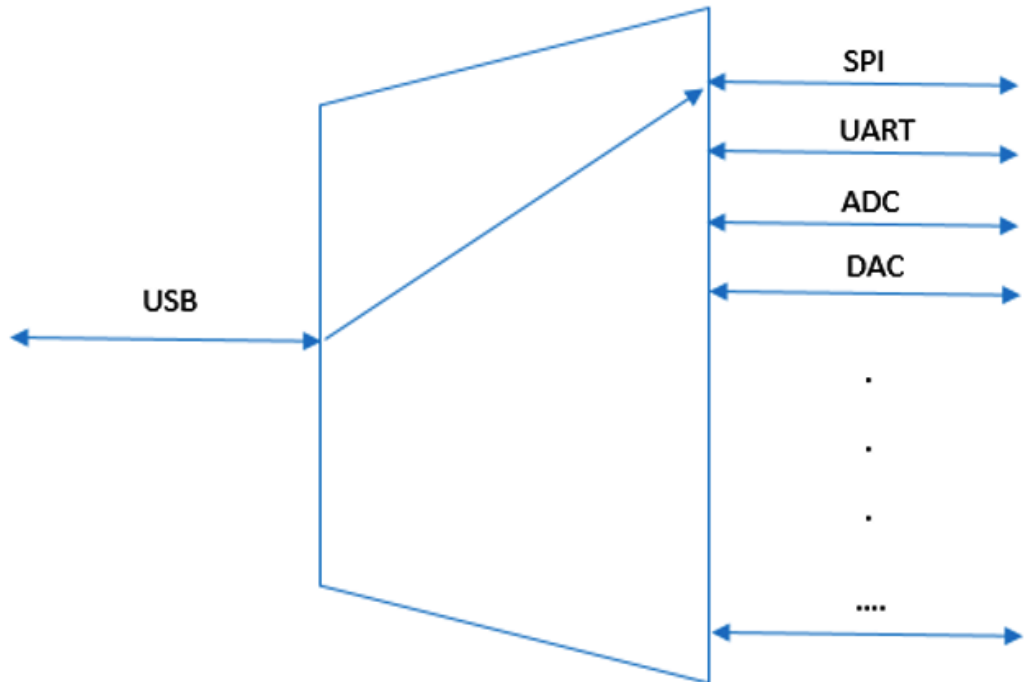
This hardware can be implemented as a stand-alone board or integrated on any other evaluation board. The block schematics is shown here below. Communication with the hardware can be established easily by using the FlexGUI software or just the Converter DLL.
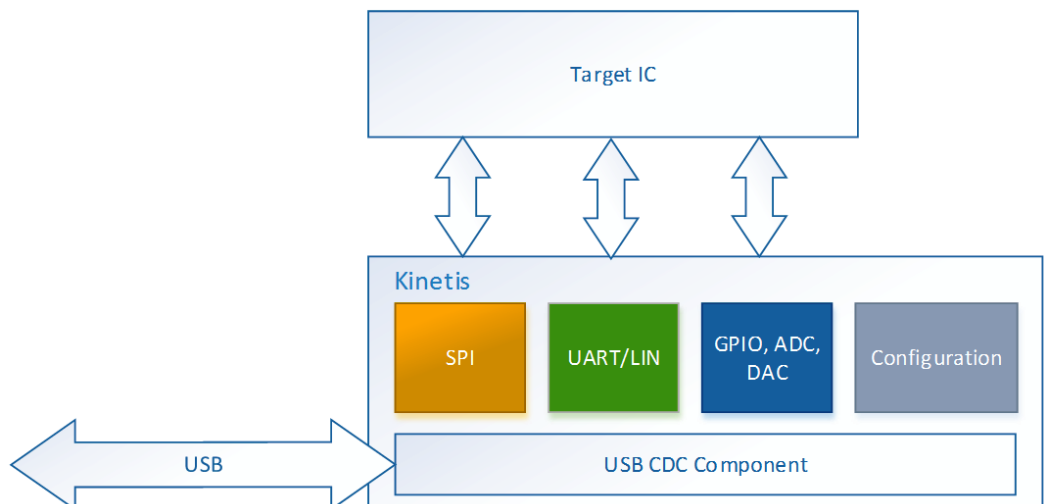
The software part consists of two different layers. The lower layer (Converter DLL) handles communication with the hardware and translates signals. Communication is established via a virtual serial COM port. The higher layer (User Interface) uses functions that are provided by the Converter DLL to communicate with the target device. The Converter exists and can be used as an independent GUI as well. In this case it can serve to convert different types of signals and protocols to USB.
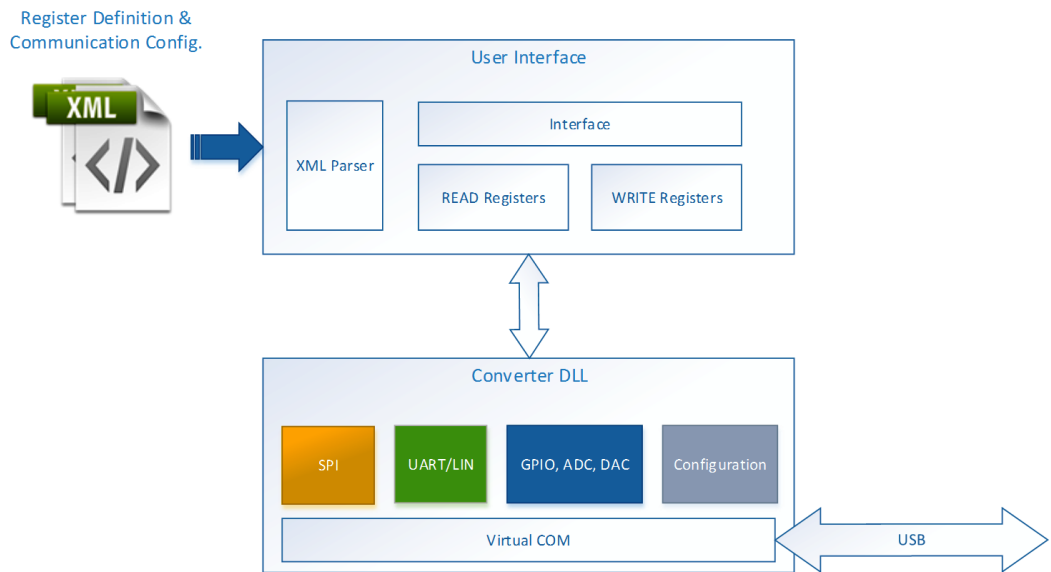

2.1 System requirements

FlexGUI requires the following setup:

• Windows compatible PC
• .NET framework 4.0 or higher
• Microsoft Office (Microsoft Excel is used to generate and store special sequences. The sequencing functionality is not supported if MS Excel is not installed).
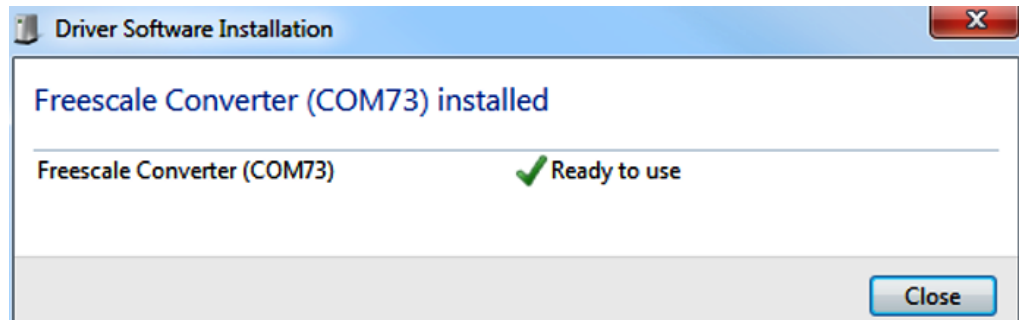
2.2 Installing the FlexGUI on Your Computer

FlexGUI is compatible with Windows operating systems (Windows 7 and higher).

1. To install the FlexGUI application and necessary drivers, launch **setup.exe** and follow the instructions.
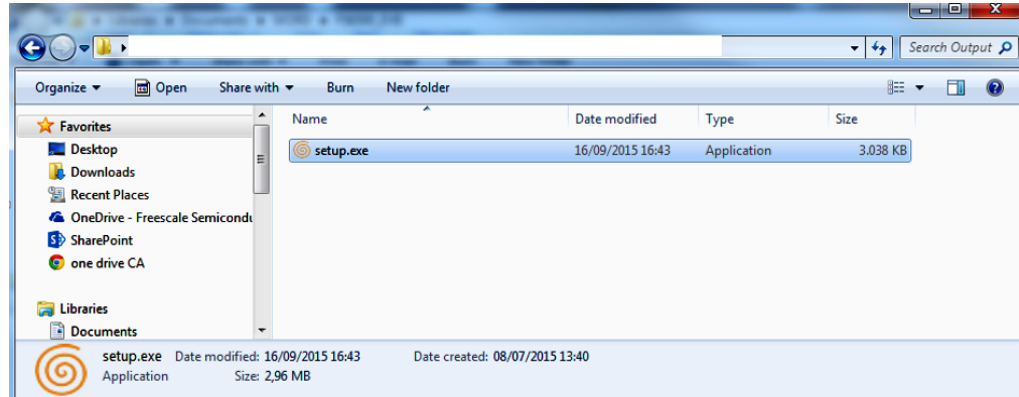
2. Connect hardware to your PC and wait for its installation. When the hardware is installed properly, the following screen is displayed.

If the driver is not signed by Microsoft, it cannot be installed on machines where the signature is mandatory. On Windows 7, there is a possibility during the installation process to omit this signature (necessary for proper installation). On Windows 8, the user has no choice during installation process and the signature is requested (default Windows 8 configuration). This setting on Windows 8 can be changed in advanced options (for details, see the Windows 8 driver installation troubleshooting).

3. Launch the **FlexGUI** application.

Note: Do not launch the installed application before the hardware is connected and installed.
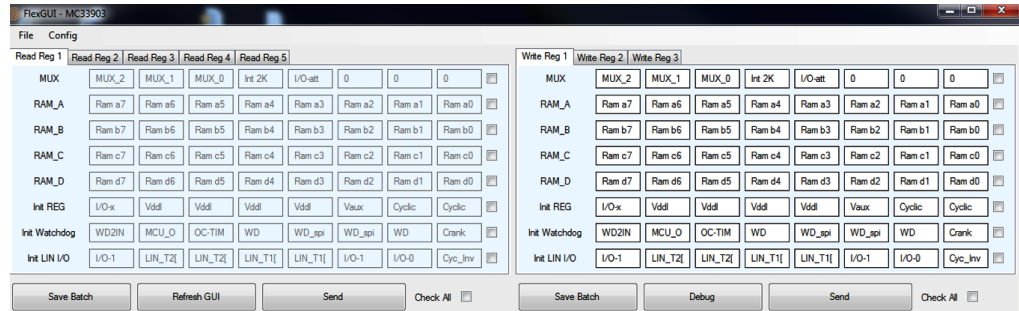
3 Getting Started With the FlexGUI

The GUI allows the user to program all SPI features by using a friendly interface as well as modifying the register table manually for advanced users. GUI consists of two separate windows; **Converter** and the **FlexGUI**. The **Converter** window enables to convert different protocols such as SPI, UART, GPIOs or analog signals to USB. FlexGUI is an extension of the converter that enables sending commands that are predefined for each IC in a special .XML file.

3.1 Launching FlexGUI

1. Launch the **FlexGUI.exe**.

FLEXGUIUG

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2016. All rights reserved

User guide

Rev. 1 — 13 May 2016

4 / 23

2. Select your device (XML configuration file). Navigate to the right folder (by default the configuration files are placed in **C:\Program Files (x86)\FlexGUI \Sequences&Config**).

3. If the hardware works properly, init config is loaded and two windows appear.

FLEXGUIUG

User guide

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2016. All rights reserved

Rev. 1 — 13 May 2016

5 / 23

4   Converter

The Converter hardware and software can be used in any application and in a stand-alone mode. In this case the FlexGUI interface is not used. The Converter software can be launched directly by launching the **Converter.exe**.

4.1   SPI converter

4.1.1   Features

Table 1. SPI converter features

Parameter	Setting	Description
CPOL	0	Clock polarity
	1	
CPHA	0	Clock Phase
	1	
Length	8	Frame length (in bits)
	16	
	24	
	32	
Start bit	MSB	Bit starting the communication
	LSB	
Baud rate	25 MHz	SPI baud rate
	12.5 MHz	
	7.69 MHz	
	4 MHz	
	2 MHz	
	1 MHz	
	500 kHz	
	250 kHz	
	125 kHz	
	62.5 kHz	
	31.25 kHz	
	15.625 kHz	

4.1.2   Using the interface

SPI Converter has an internal buffer with a default size of 1024 bytes. Depending on the configured SPI length, the user can write a number of SPI frames in one write access Table 2. One write access means the maximum length of the sequence sent (when the GUI is used) or the maximum length of a message written to the virtual COM port (when a terminal is used instead of the standard GUI).

FLEXGUIUG

User guide

All information provided in this document is subject to legal disclaimers.

Rev. 1 — 13 May 2016

© NXP B.V. 2016. All rights reserved

6 / 23

Table 2. SPI buffer size

SPI length	Max . number of frames written in one write access
8	1024
16	512
24	256
32	128

4.1.3 Delay between SPI commands

The delay between two SPI commands depends on the software execution time (for example, function, calls) and is not defined. An additional delay is can be induced by adding a "zero" command. An SPI command that contains only zeros is considered to be a delay of **xy**.

4.1.4 Sending a sequence

SPI Converter allows the user to create and load sequences that are stored in an Excel format. Clicking the **Excel Log** button causes a new sequence file to be created using the current values in the **Received** and **Sent** textboxes. This Excel file can then be saved and loaded again when necessary. Loading the file can be done by clicking on the **Load Sequence** button.

A sequence that has already been loaded can be re-loaded by double-clicking on the sequence name (for example, InitSequence.xlsx from the previous image). By loading several sequences in the same time, sequences can be combined as needed. A newly loaded sequence is always put after the previously loaded one.

Sequences can be generated by the program or created manually in Excel. The important requirement for the Excel sheet is the format of the first column and name of the worksheet. The name Converter_log must be used for the worksheet where the sequence is located. The data to send must be placed in the first column in a hexadecimal format without any special formatting as shown in Figure 1. The other columns are used for information only and are not important when the data is sent through SPI.
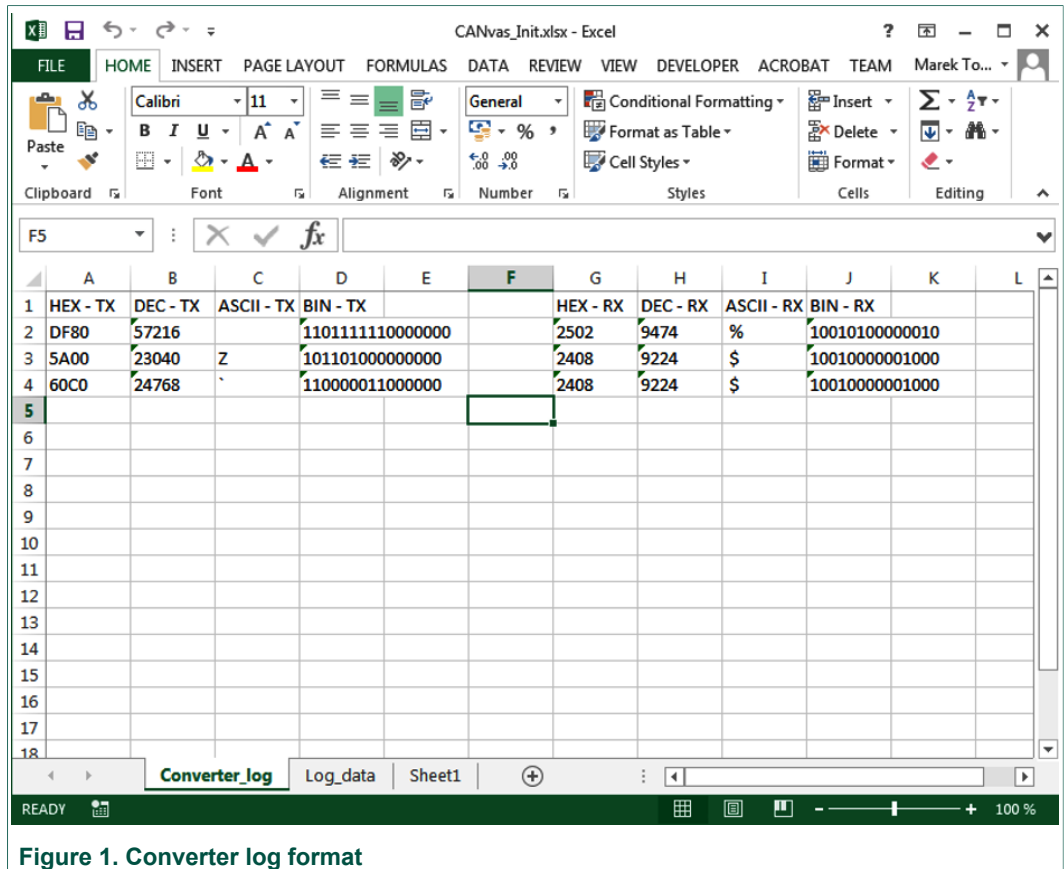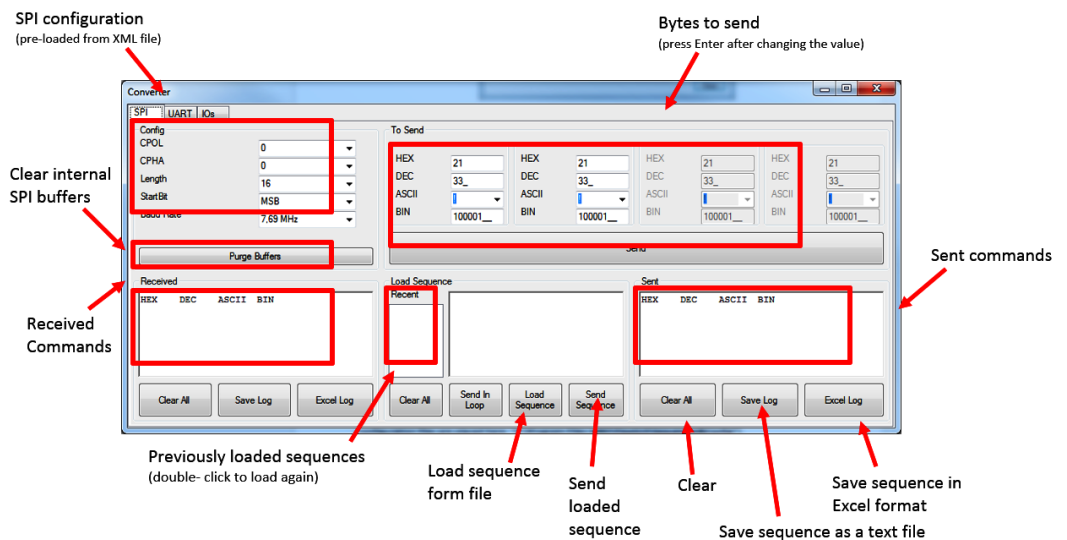
Figure 1. Converter log format

4.1.5 Converter SPI description

FLEXGUIUG

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2016. All rights reserved

User guide

Rev. 1 — 13 May 2016

8 / 23

4.2 UART converter

4.2.1 Features

UART Converter allows the user to receive and transmit asynchronous frames with the following settings:

Table 3. UART converter settings

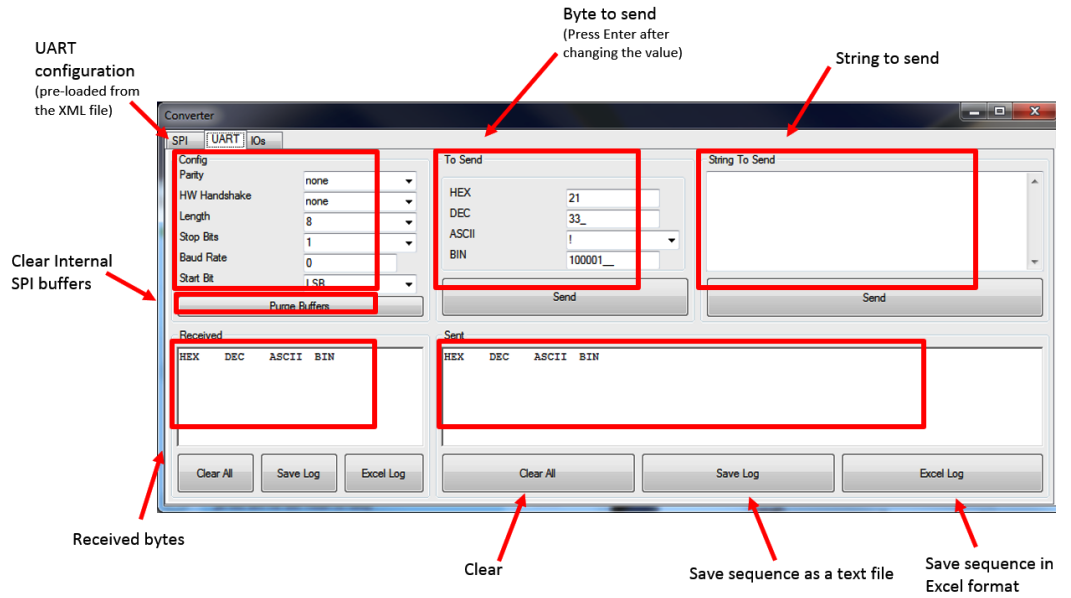| Parameter	Setting	Description
Parameter	None	Frame Parity
	Odd	
	Even	
HW handshake	None	No hardware handshake used (fixed configuration)
Length	8	Frame length (in bits) – fixed configuration
Stop bits	1	Number of stop bits
	2	
Start bit	MSB	Bit starting the communication
	LSB	
Baud rate	From 1 466 to 11 999 999	UART Baud Rate. Baud rate can be set to any value in the specified range. To set the value, write the value to the edit box and press "Enter".

4.2.2 Using the interface

UART Converter works as a standard non-buffered UART/USB converter. Before operation, the converter must be configured properly. The baud rate must be entered in the edit box and validated by pressing **Enter**. If the baud rate is not within the acceptable limits, the converter will not take it into account. Data can be sent in a byte mode or as a string of characters. Received data can be stored in a text or Excel format.

Note:

The value of the Send data may be written in any format (HEX, DEC, ASCII, BIN), but the **Enter** *key must always be pressed to validate the choice. When* **Enter** *is pressed, the other fields synchronize automatically.*

FLEXGUIUG

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2016. All rights reserved

User guide

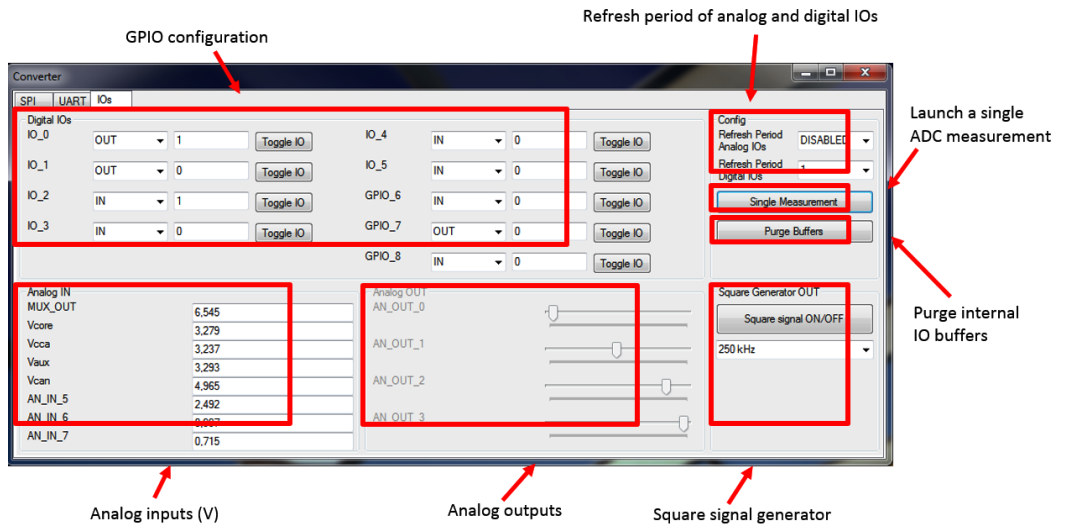**Rev. 1 — 13 May 2016**

9 / 23

4.2.3   Converter UART description

4.3 IO converter

The IO Converter tab includes all special IO functions such as analog inputs, analog outputs, digital IOs, and square generator.

4.3.1 IO converter description

FLEXGUIUG

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2016. All rights reserved

User guide

Rev. 1 — 13 May 2016

11 / 23

4.3.2 Digital IOs

IO Converter provides nine general purpose IOs. These can be configured as inputs or outputs. Default pin direction and the pin name are loaded from the configuration file. The pin direction can be changed any time. The change is applied immediately once the selection is made in the combo box. The current value of each digital output can be changed by pushing the **Toggle IO** button.
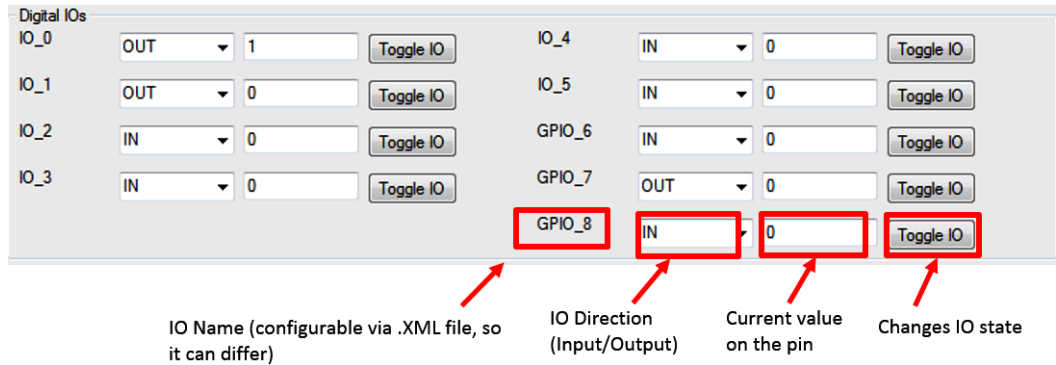

Note:

Be careful when configuring the pins as outputs. Configuring an input pin as an output may damage the MCU. Protection of the digital IOs can differ on different types of hardware. See the appropriate schematics for details.

4.3.2.1 Refreshing IOs

Current state of the digital IOs is read (refreshed) periodically only if the refresh period is enabled. To enable this refreshing, select one of the pre-defined period in the appropriate combo box as shown below. Refresh period is a product of the following equation:

$$T_{REF} = x * T_{CYC}$$

Where "x" is the selected number and T_{CYC} is the cycle time.
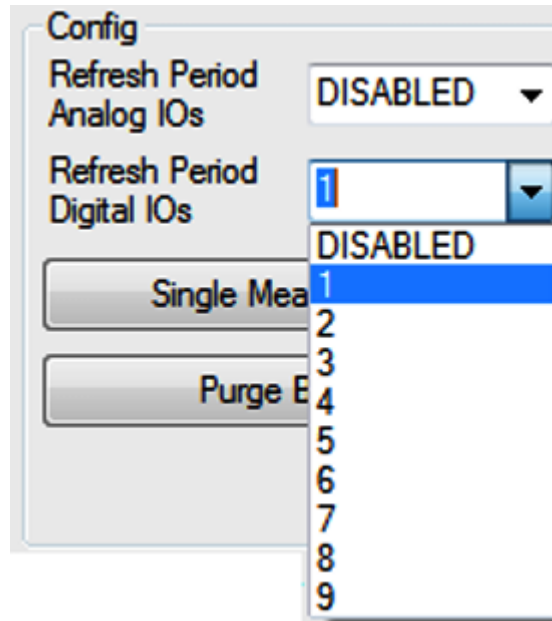
4.3.3 Analog inputs

Eight analog inputs are available in the **Analog IN** group box. The analog signals must be amplified before going to the ADC. This can be done using an operational amplifier as shown in Figure 2. The amplifier has a gain G, which must be defined in the configuration file.
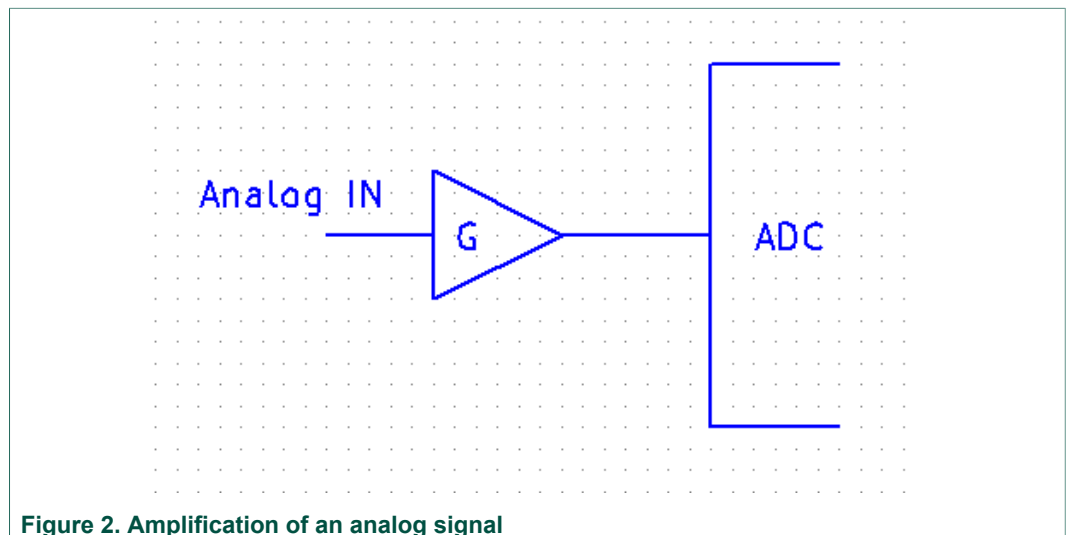

Figure 2. Amplification of an analog signal

The final value of each analog input displayed in the GUI is then computed using the formula shown in Figure 3. The constants used in this formula are defined in the .XML configuration file. The number coming from the ADC is "x". An example is given for x = 2646.

FLEXGUIUG

User guide

All information provided in this document is subject to legal disclaimers.

Rev. 1 — 13 May 2016

© NXP B.V. 2016. All rights reserved

13 / 23

$$V_{DISP} = \frac{ADC_{REF}}{ADC_{RES}} \cdot x \cdot AN\_IN\_RATIO\_n$$

$$V_{DISP1} = \frac{3,3}{4096} \cdot 2646 \cdot 3,049 = 6,5\ V$$

```
<boardConfig>
    <DAC_REF>3,3</DAC_REF>
    <DAC_RES>4096</DAC_RES>
    <ADC_REF>3,3</ADC_REF>
    <ADC_RES>65536</ADC_RES>
</boardConfig>
<configAnalogINs>
    <AN_IN_RATIO_0>3,049</AN_IN_RATIO_0>
    <AN_IN_RATIO_1>1,826</AN_IN_RATIO_1>
    <AN_IN_RATIO_2>1,826</AN_IN_RATIO_2>
    <AN_IN_RATIO_3>1,826</AN_IN_RATIO_3>
    <AN_IN_RATIO_4>1,826</AN_IN_RATIO_4>
    <AN_IN_RATIO_5>1,826</AN_IN_RATIO_5>
    <AN_IN_RATIO_6>1</AN_IN_RATIO_6>
    <AN_IN_RATIO_7>1</AN_IN_RATIO_7>
</configAnalogINs>
```
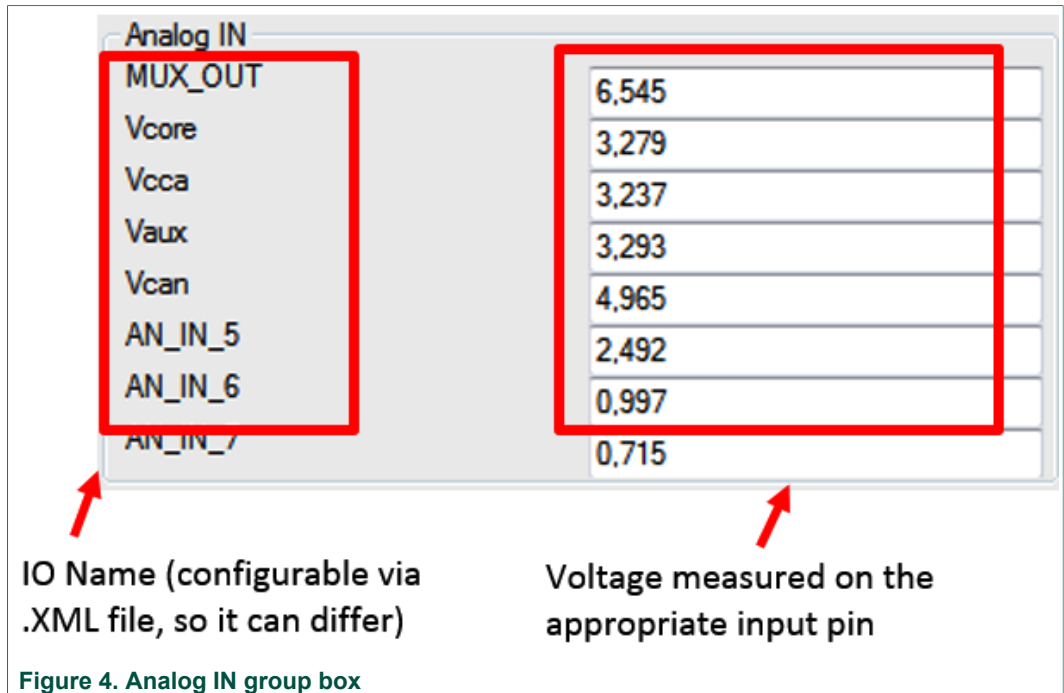
Figure 3. ADC constants in the .XML configuration file

Group box **Analog IN** shows the values previously measured (see Figure 4). All values are the final values in Volts (already multiplied by the appropriate coefficient).

FLEXGUIUG

User guide

All information provided in this document is subject to legal disclaimers.

Rev. 1 — 13 May 2016

© NXP B.V. 2016. All rights reserved

14 / 23

Figure 4. Analog IN group box

Note:

*Analog values can be refreshed manually (using the **Single Measurement** button) or automatically with a refreshing period.*

4.3.4 Analog outputs

Converter provides four analog outputs generated by a 12-bit DAC. The out-going value can be changed using a slider. Actual value of the appropriate output is displayed in a tooltip help. The first value gives the number of points from the 12-bit scale and the second value gives the output voltage.

Figure 5. Analog outputs
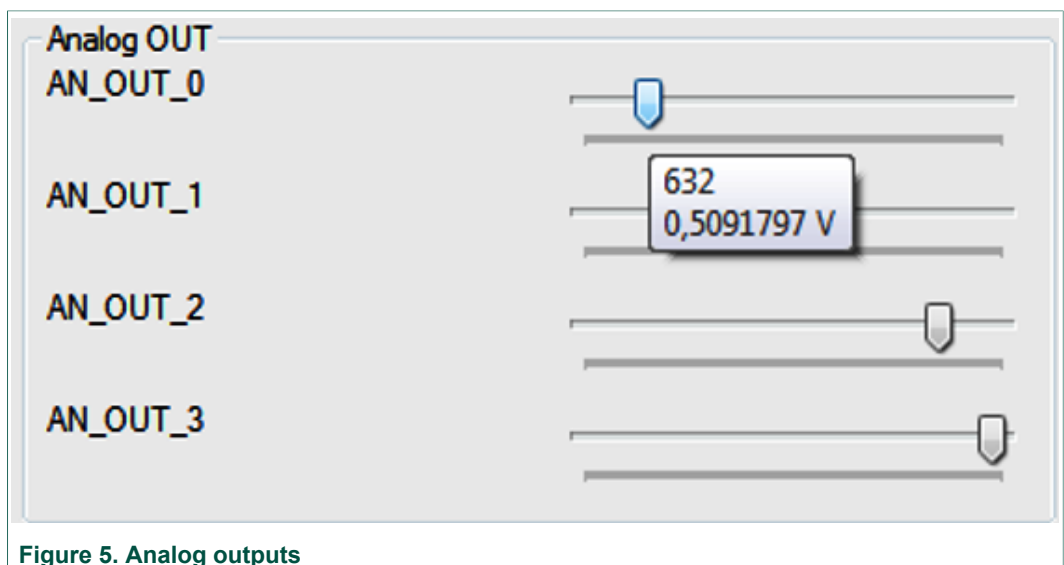
4.3.5 Square Signal Generator

Converter includes one square signal generator with several fix frequencies. The frequency can be changed using the combo box.
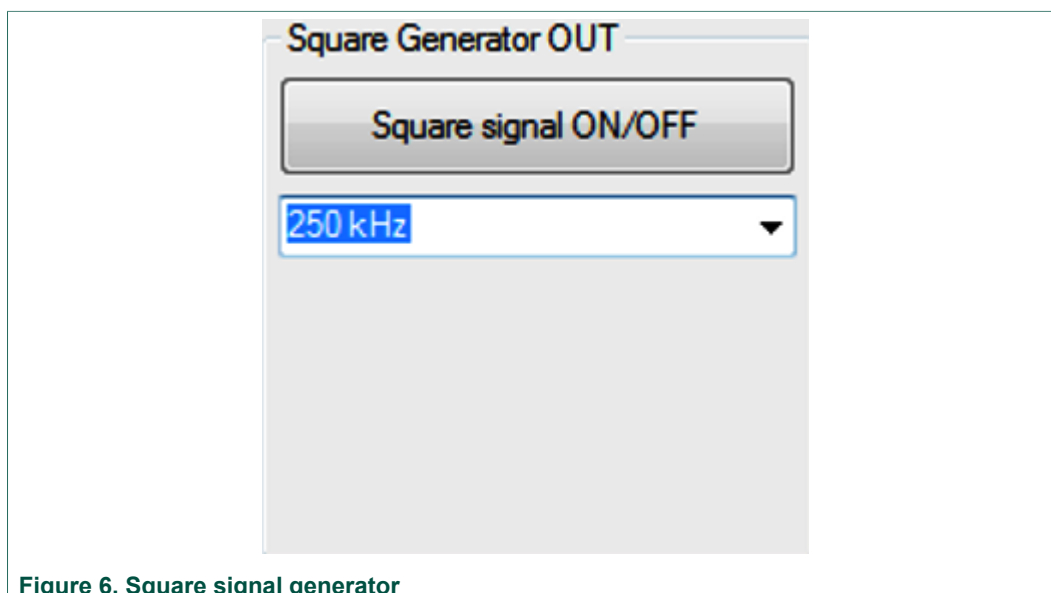

Figure 6. Square signal generator

4.3.6 Configuration

Analog and digital input values can be read with different periods. These periods are predefined and set by using the appropriate combo boxes. Analog values can also be measured on request using the **Single Measurement** button. The reading period is a product of multiplication.

FLEXGUIUG

User guide **Rev. 1 — 13 May 2016**

16 / 23
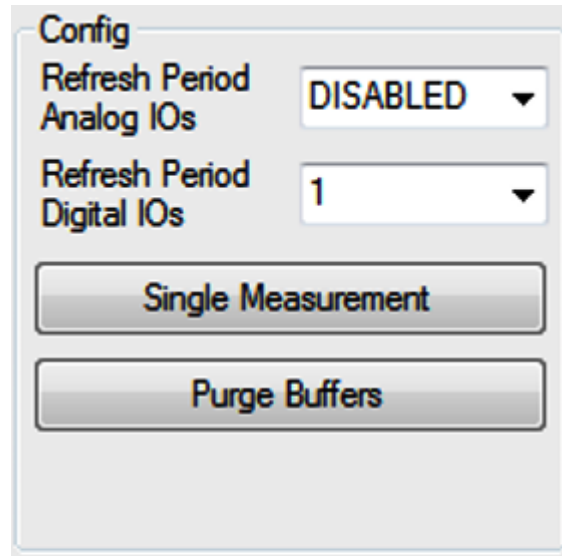
$$T_{REF} = x * T_{CYC}$$

5 FlexGUI

5.1 Using FlexGUI interface

FlexGUI is an interface that can be adapted to the needs of a specific IC without compiling code. The target IC is fully described in an XML file. The XML file specifies the configuration of communication interfaces, registers, names of signals and so forth. This file is then selected during the FlexGUI start-up.

Figure 7. XML configuration file
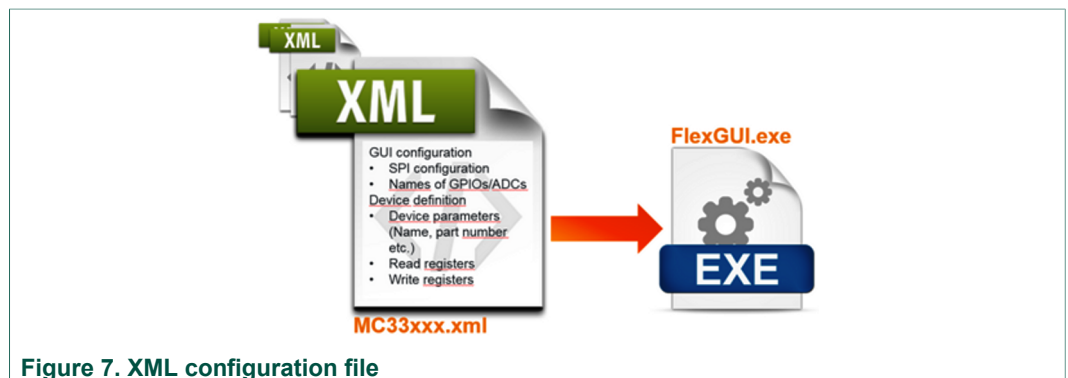
5.2 XML configuration file

The XML configuration file is easy to read and to modify. All the .XML files that are delivered with the GUI can be found in the **Sequences&Config** folder (in the default installation directory). Structure of the .XML file is defined and must be maintained.

XML parameter		Description
`<device>`		
	`<configuration>`	SPI, UART, GPIOs, ADC, DAC configuration
	`<parameters>`	Partnumber, version, date etc.
	`<COMconfig>`	SPI length
	`<readRegsParameters>`	Parameters of the read registers (number of bits etc.)
	`<writeRegsParameters>`	Parameters of the write registers (number of bits etc.)
	`<readTabNames>`	Names of the read tabs
	`<readRegistersOrders>`	Order of the read registers in the GUI
	`<readRegisters>`	Definition of all read registers
	`<writeTabNames>`	Names of the write tabs
	`<writeRegistersOrders>`	Order of the write registers in the GUI
	`<writeRegisters>`	Definition of all write registers
`</device>`		

Figure 8. XML file structure

5.3 Interface description

FlexGUI application is divided into two parts: Read registers and Write registers. Read registers are placed on the left side, Write ones on the right side. Both types of commands can be sent by clicking the **Read**" or "**Write** buttons.

Only the commands that are checked and that are on the active tab are sent. All commands that were sent are stored in the **Received** and **Sent** textboxes in the Converter application. Read commands cannot be configured using GUI. A Read command consists of an address and a mask as defined in the .XML file. A Write command can be changed by clicking directly on the appropriate bit.
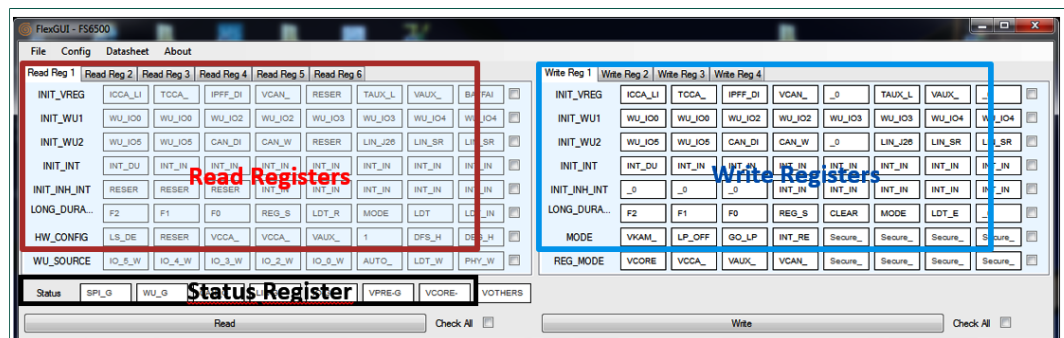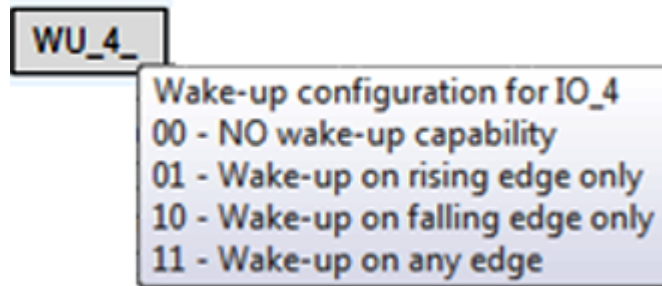

Figure 9. FlexGUI Read, Write and Status registers

Note:

Tooltip help - Description for each bit can be displayed by putting mouse over the specific bit.

FLEXGUIUG

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2016. All rights reserved

User guide

Rev. 1 — 13 May 2016

18 / 23

6   Windows 8/10 driver installation and troubleshooting

By default, Windows 8 requires verification of the driver's signature. The provided driver is not verified by Microsoft, so the verification always fails and the driver cannot be installed. To install this driver successfully, this verification must be disabled.

For Windows 8 systems, follow the below instructions to disable the verification:

1. Hold the **Shift** key down and click **Restart** (under Charms on the login screen) to reboot the computer.
2. Once the computer has rebooted, choose the **Troubleshoot** option.
3. Choose **Advanced options**.
4. Choose **Startup Settings**.
5. Click **Restart**. Computer reboots.
6. After rebooting, change the startup settings.
7. Change **Disable driver signature enforcement** by pressing **F7**. The PC reboots.

The driver can now be installed manually. Follow, the below instructions to install the driver:

1. Connect the Converter hardware.
2. Go to **Control Panel** > **System** > **Device Manager**, and then select **Ports (COM & LPT)**.
3. Select the **Freescale Converter** and right-click.
4. Select **Update Driver Software…**.
5. Select **Browse my computer for driver software**.
6. Browse the FlexGUI installation folder and select **drivers/cdc.inf**. By default, the FlexGUI path is: **C:\Program Files (x86)\FlexGUI** or **C:\Program Files\FlexGUI**.
7. Update the driver.

7   Revision history

Revision	Date	Description of changes
1.0	5/2016	Initial release

8   Legal information

8.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

8.2 Disclaimers

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit,

and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/ or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/salestermsandconditions.

8.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

NXP — is a trademark of NXP B.V.

the NXP logo — is a trademark of NXP B.V.

Freescale — is a trademark of NXP B.V.

the Freescale logo — is a trademark of NXP B.V.

SMARTMOS — is a trademark of NXP B.V.

Tables

FLEXGUIUG

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2016. All rights reserved

User guide

Rev. 1 — 13 May 2016

21 / 23

Figures

Contents