
NCF Tool User Manual

For 8/16/32-bit LIN Stack

Document Number: NCFTOOLUSERMANUAL

Rev1.3.3 1/2017



Table of Contents

Chapter 1	Introduction	3
1.1	Revision History	4
1.2	Definitions, Acronyms, and Abbreviation	5
1.3	References	6
Chapter 2	Overview	7
2.1	Configuration data flow in LIN Stack	8
2.2	Extend NCF to Eclipse plug-in.....	9
	2.2.1 Software requirements	9
	2.2.2 Limitations	9
Chapter 3	How to use Plug-in.....	10
3.1	Plug-in Setup	11
3.2	Eclipse Plug-in text editor	12
	3.2.1 LDF files Editor	12
	3.2.2 NPF files Editor.....	16
3.3	Eclipse Plug-in graphical editor	19
	3.3.1 LDF file graphical editor.....	19
	3.3.2 NPF file graphical editor	26
	3.3.3 Generate configuration code	29

Chapter 1

Introduction

This document describes in details how to use an NCF tool which contains three different methods aimed at generation configuration files for FSL 8/16/32 bit MCU LIN Stack.

The information in this document is subject of change without notice and does not represent a commitment on the part of NXP Semiconductors. The software describes in this document is furnished under a license agreement and may be used or copied in accordance with the terms of that license agreement. No part of this manual may be reproduced in any form or by any means, electronically or mechanically, including photocopying and recording for any purpose without the express written permission of NXP Semiconductors.

1.1 Revision History

Table 1-1. Revision history

Revision	Date	Author	Description
1.0	July 15, 2011	Cong Tran B26340	Initial release
1.1	October 12, 2011	Cong Tran B26340	Add new icons for save history features New feature for multi timer selection for S12X
1.2	September 23, 2013	Cong Tran B26340	Add user manual for Wiondows command prompt and standalone GUI
1.3	August 05, 2014	Cong Tran B26340	Add steps to add plug-in to CW compiler
1.3.1	June 01, 2015	Lan Bui B39392	Changed name of the LIN Driver Package to LIN Stack Package
1.3.2	November 18, 2015	Lan Bui B39392	Add max_message_length in global definition and support_sid field to network definition to npf files
1.3.3	January 13, 2017	Dat Bui – B54556	Update to NXP copyright

1.2 Definitions, Acronyms, and Abbreviation

BSP	Board Support Package
LDF	LIN Definition File
LIN	Local Interconnect Network
NCS	Node Configuration Script
NPF	Node Private file

1.3 References

- [1] LIN Specification Package, rev. 2.1, November 24, 2006
- [2] LIN Stack User Manual Package, rev. 2.0.1, April , 2011

Chapter 2

Overview

This chapter provides a general description on configuration data flow of LIN Stack and the role NCF tool in generation configuration files.

This chapter contains sections:

- Configuration of data flow in LIN Stack
- Extension NCF to Eclipse plug-in.

2.1 Configuration data flow in LIN Stack

The configuration data flow is described on [Figure 2-1](#) where the node configuration .c and .h files based on LDF file (see chapter 3.1 in [2]) and NPF file (see chapter 3.1 in [2]) are inputs for Node Configuration Tool to generate Node configuration Code.

These node configuration .c and .h files are then together with Stack code (Stack source files and Stack header files) compiled by the compiler to get the node Target image which is downloaded to the target MCU.

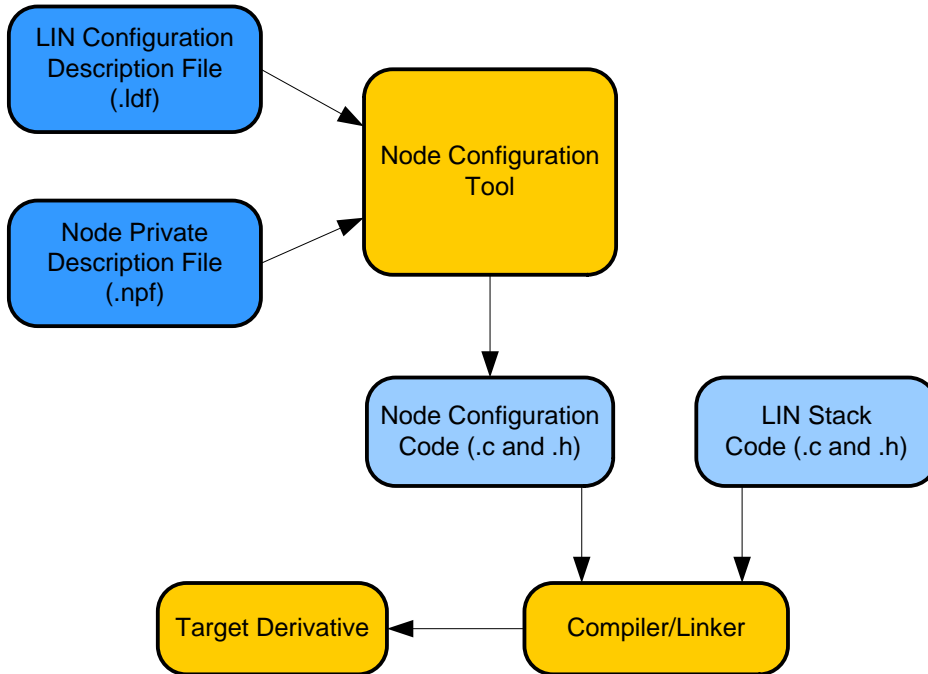


Figure 2-1. Configuration data

The Node Configuration Tool contains three scripts used to generate Node Configuration Code (.c and .h) Window Command Line, Standalone GUI and Eclipse Plug-in.

This document is aim at providing user how to use Eclipse Plug-in for both purpose of create new LDF and NPF files by text or graphical and generate Node Configuration Code.

For further understanding how to use two remains scripts, refer to LIN release package version 3.0.

2.2 Extend NCF to Eclipse plug-in

The Plug-in developed in Eclipse environment contains text and graphical editors used to create new LDF and NPF files or import existing these files in a certain folder.

2.2.1 Software requirements

The Eclipse used in this document based on version 3.4 and the application is compatible with Eclipse CDT 3.4 or later / Code warrior 10 or later.

2.2.2 Limitations

In comparison with session 9.2 of LIN 2.1 Specifications [1], currently NCF plug-in GUI editor for LDF files does not support definitions which are listed below

- Channel_name_def
- Node_composition_def
- Signal_groups_def
- Signal_encoding_type_def
- Signal_representation_def

Chapter 3

How to use Plug-in

The Eclipse plug-in contains text and graphical editor for user to create, modify existing configuration files and generate configuration code. This section describes in detail two user interfaces text and graphical editor to create LDF and NPF configuration files and generate configuration code.

The chapter contains sections:

- Plug-in setup
- Plug-in text editor
- Plug-in graphical editor

3.1 Plug-in Setup

The Eclipse plug-in for NCF tool is combined in the software package at the directory: LIN_Package\NCFTool\Eclipse_Plugin.

To use it, those are steps:

1. Copy the plug-in (.jar file) to the folder of CW: \Freescale\CW MCU v10.6\eclipse\plugins
2. Run CW execution file by Command prompt from Windows with –clean command:

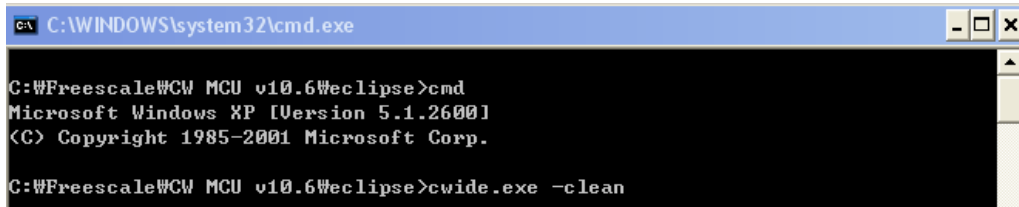
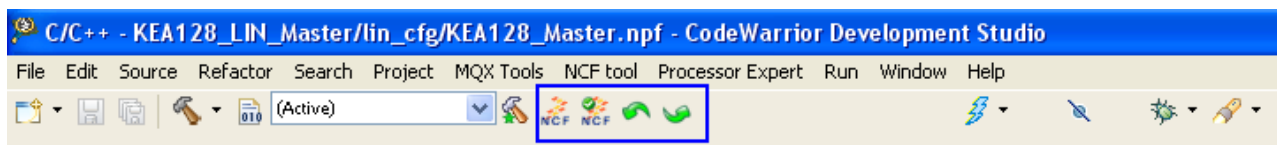


Figure 3-1: CodeWarrior 10 clean command

Note:

This NCF plug-in supports the both Eclipse CDT and CodeWarrior 10, but in this document we just use Eclipse CDT 3.4 r2 to demonstrate (everything in CodeWarrior 10 is similar).

The Code Warrior is displayed below with some new icons for plug-in tool:



3.2 Eclipse Plug-in text editor

3.2.1 LDF files Editor

Two input files which are required for generation configuration code are LDF and NPF files. In order to create NPF files, the LDF must be existed or created first. If the LDF file or NPF is created in the Plug-in text editor, they will be saved in the working folder with current working project.

3.2.1.1 Create LDF file by using LDF template

The Plug-in support a template for create ting LDF file which is based on LIN Configuration Specification in LIN Spec2.1 [1]. The steps below show how to import the LDF template in text editor.

1. Create an empty project name “DEMO_XEP100” by click “New ->Project...-> Select C Project”

The screen for adding name of project is shown in the [Figure 3-2](#)

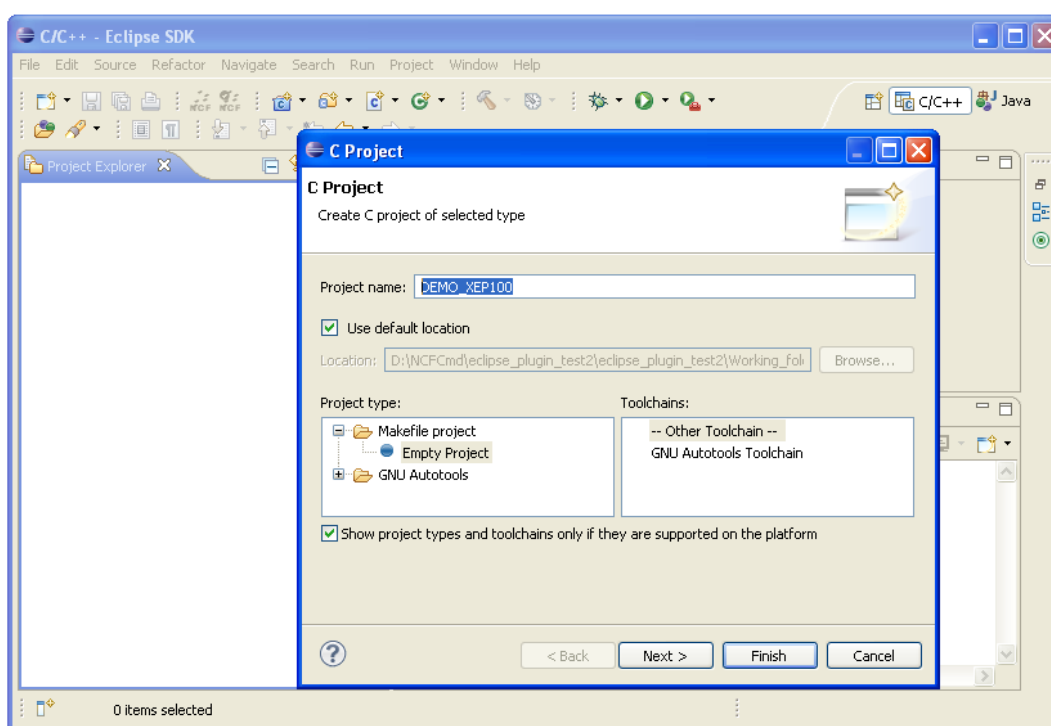


Figure 3-2: Empty working project with name as DEMO_XEP100

2. Open a LDF template by right click to DEMO_XEP100 in Project Explore “New->Other...”
In the next Wizard, select LDF file in the LIN Node Configuration Files pop-up as in [Figure 3-4](#).

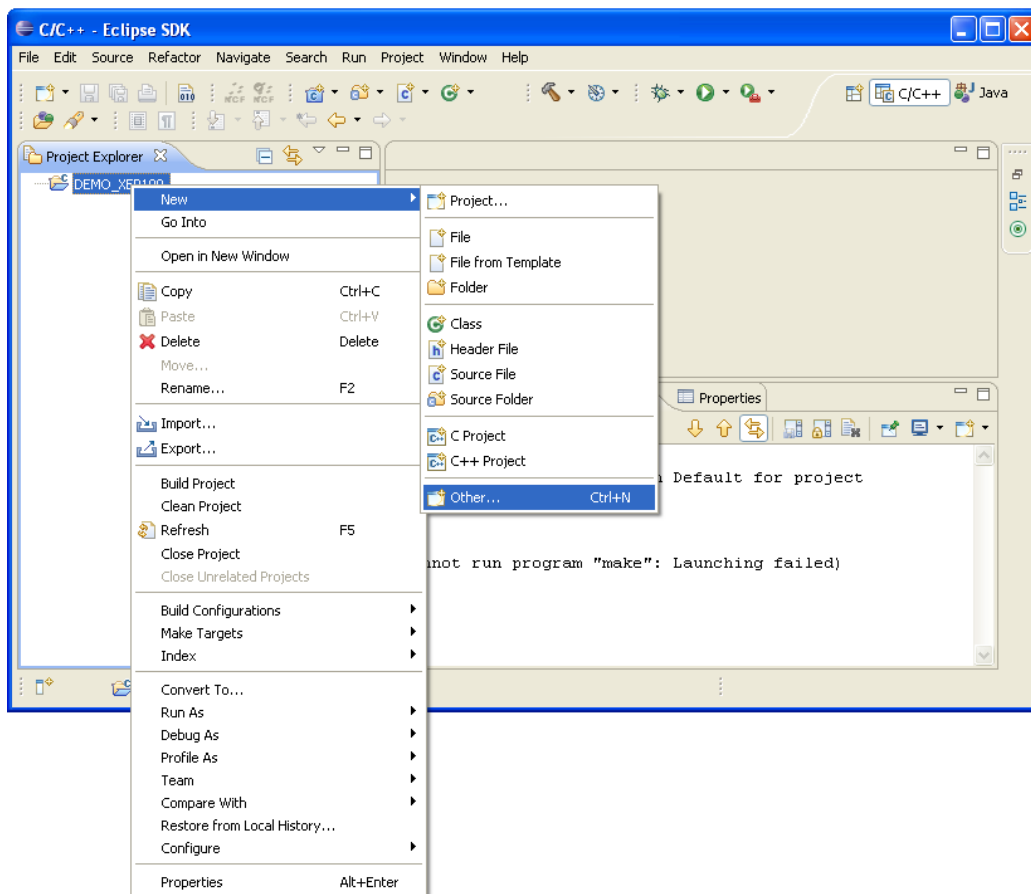


Figure 3-3: Select new working file to working project

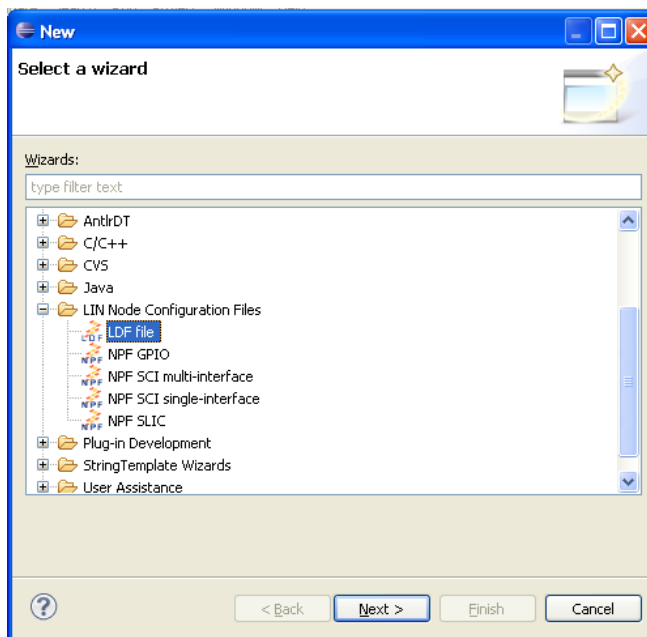


Figure 3-4: Select LDF template

3. Click “**Next >**” button and name the LDF file as LIN_Example.ldf. Click “**Finish >**” button to create new template file. The template displayed in text editor is shown below

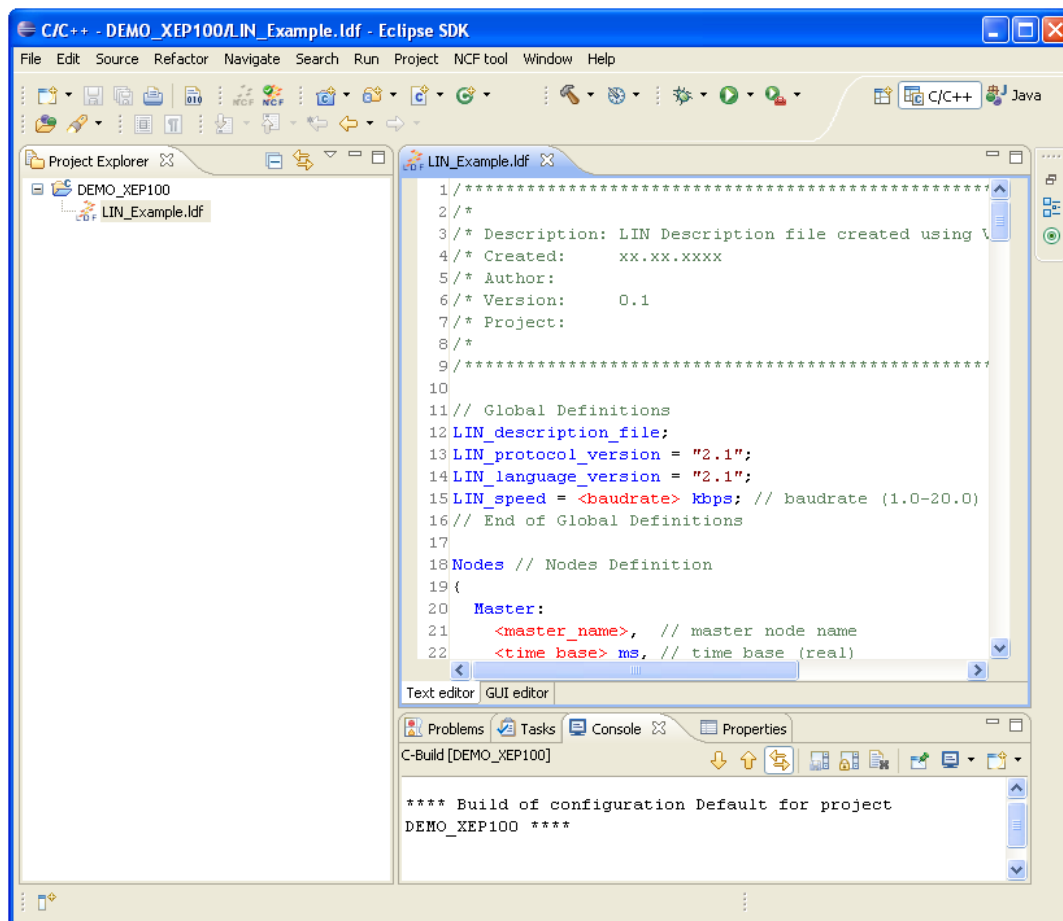


Figure 3-5: A LDF template in text editor

4. After LDF template is opened in text editor mode, it is ready for editing the file by your own.

3.2.1.2 Open an existing LDF file to Project Explorer

If an LDF file exists somewhere in your computer, it is easy to import that file to the working project. The mechanism of plug-in is to create the link of the file and add that link to Project Explorer of Eclipse.

The example below shows adding a LDF file named “LIN21.ldf” stored in the directory “D:\NCFCmd\SampleInputFile\DiagClass1”

1. Click to “File” in menu tool bar
2. Select “Open File”
3. Select the file in the directory and click “Open”

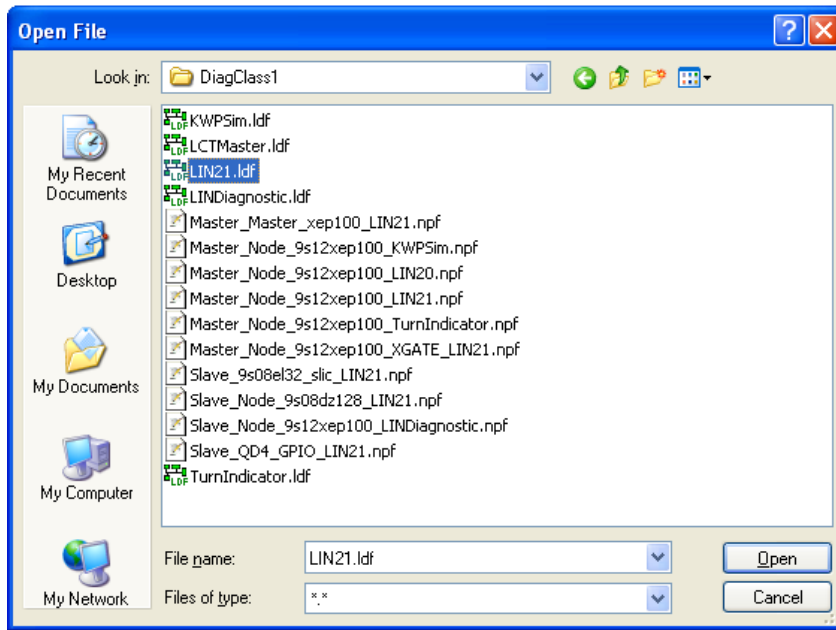


Figure 3-6: Dialogue to select external LDF file

4. Confirm to add the file to working project.

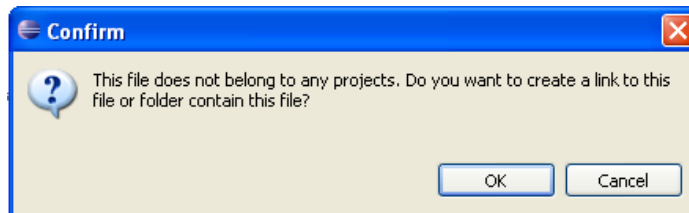


Figure 3-7: Confirm open the file in internal or external mode

If click “OK” the plug-in will create a link to a file to the project and serve as internal mode.

If click “Cancel”, the plug-in will open the file and serve as external mode.

Note:

The internal or external modes of LDF file are not different in term of using. This category is only valuable when applied for NPF file where the configuration code is located in the same working folder or external folder.

5. Select folder to store the link

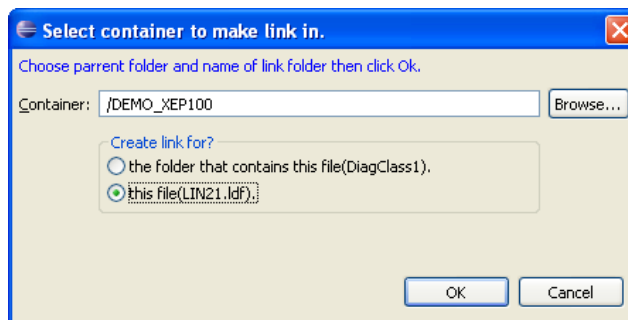


Figure 3-8: Select folder to store link in working project

The dialogue contains two options: create the link for all other files contains in the same folder and create the link for this file only. The default option is for this file only.

After step 5, the text editor screen is shown as **Figure 3-9** below

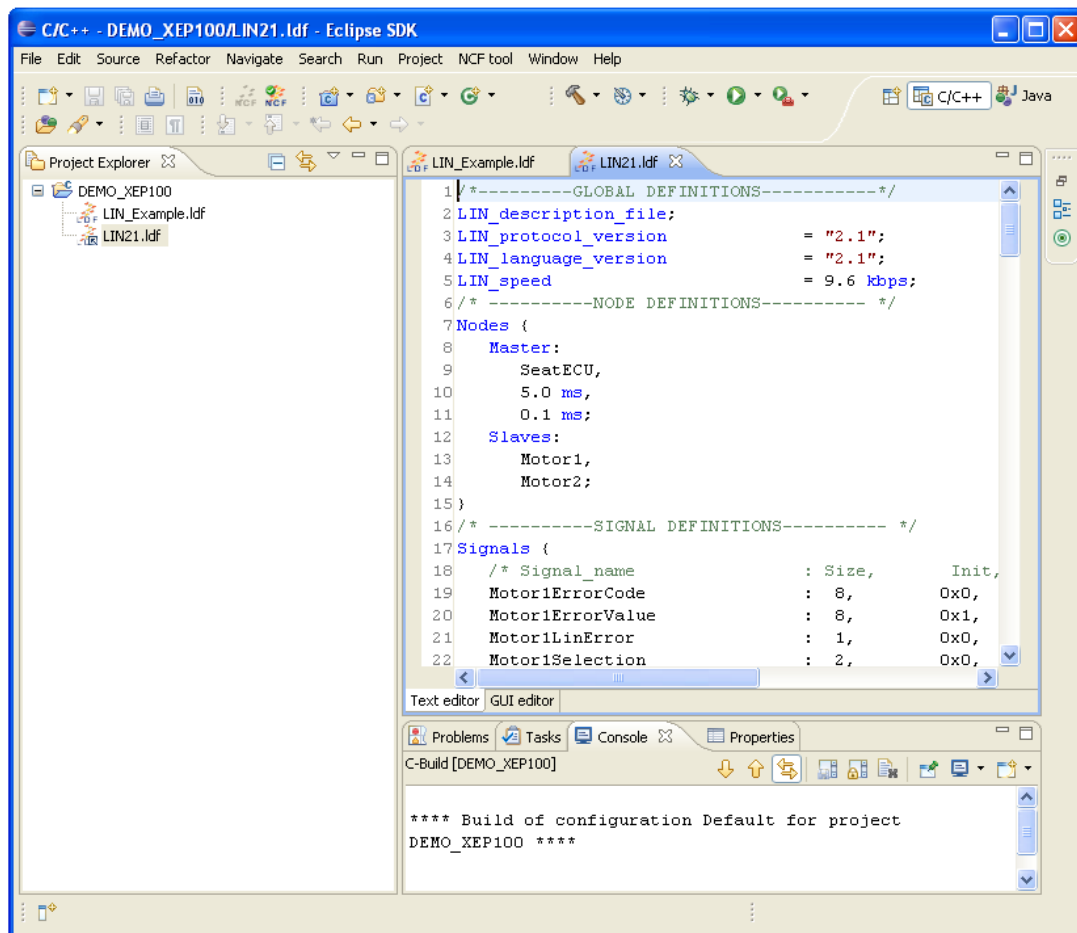


Figure 3-9: External LIN21.ldf file in text editor mode

3.2.2 NPF files Editor

Once LDF file has been created, it is ready to create NPF file. The reason why LDF file must be created first is the NPF file will include the file to a field in configuration.

3.2.2.1 Create NPF files by using template

The plug-in supports some templates for each type of NPF file which is listed below

- NPF for GPIO interface
- NPF for SCI interface with single LIN channel
- NPF for SCI interface with multiple LIN channels
- NPF for SLIC interface

To create a NPF template:

1. **Right click** to working project in Project Explorer where the file will be located.
2. Select **"New > Other "** or short key **"Ctrl + N"**
3. In the list of template, select suitable template and click **"Next"**
4. Add suitable name for NPF file and click **"Finish"** button

In this example, the NPF file is named as **"XEP100_Master"**

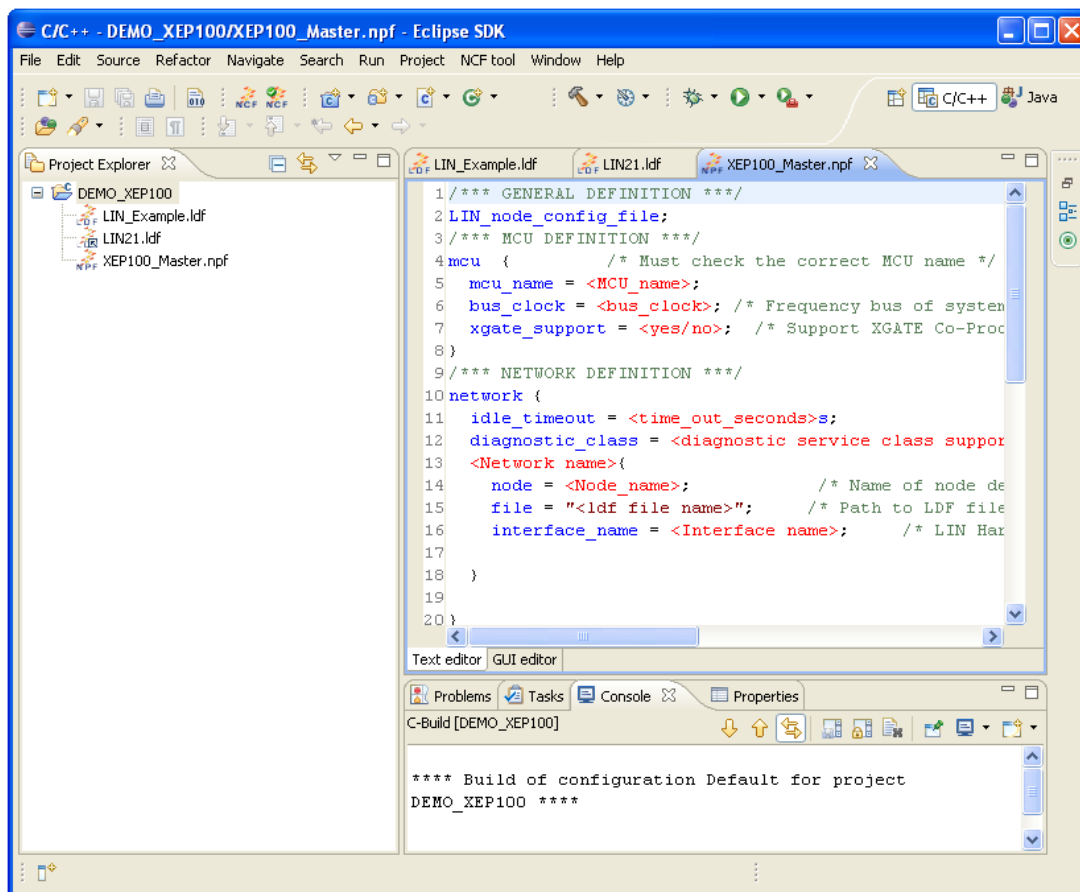


Figure 3-10: NPF file template in text editor mode

3.2.2.2 Open an existing NPF file to Project Explorer

Similar to section 3.1.1.2, user can open an existing NPF file to working project in Project Explorer. The example below shows example of opening the file named “Master_Node_9s12xep100.npf”

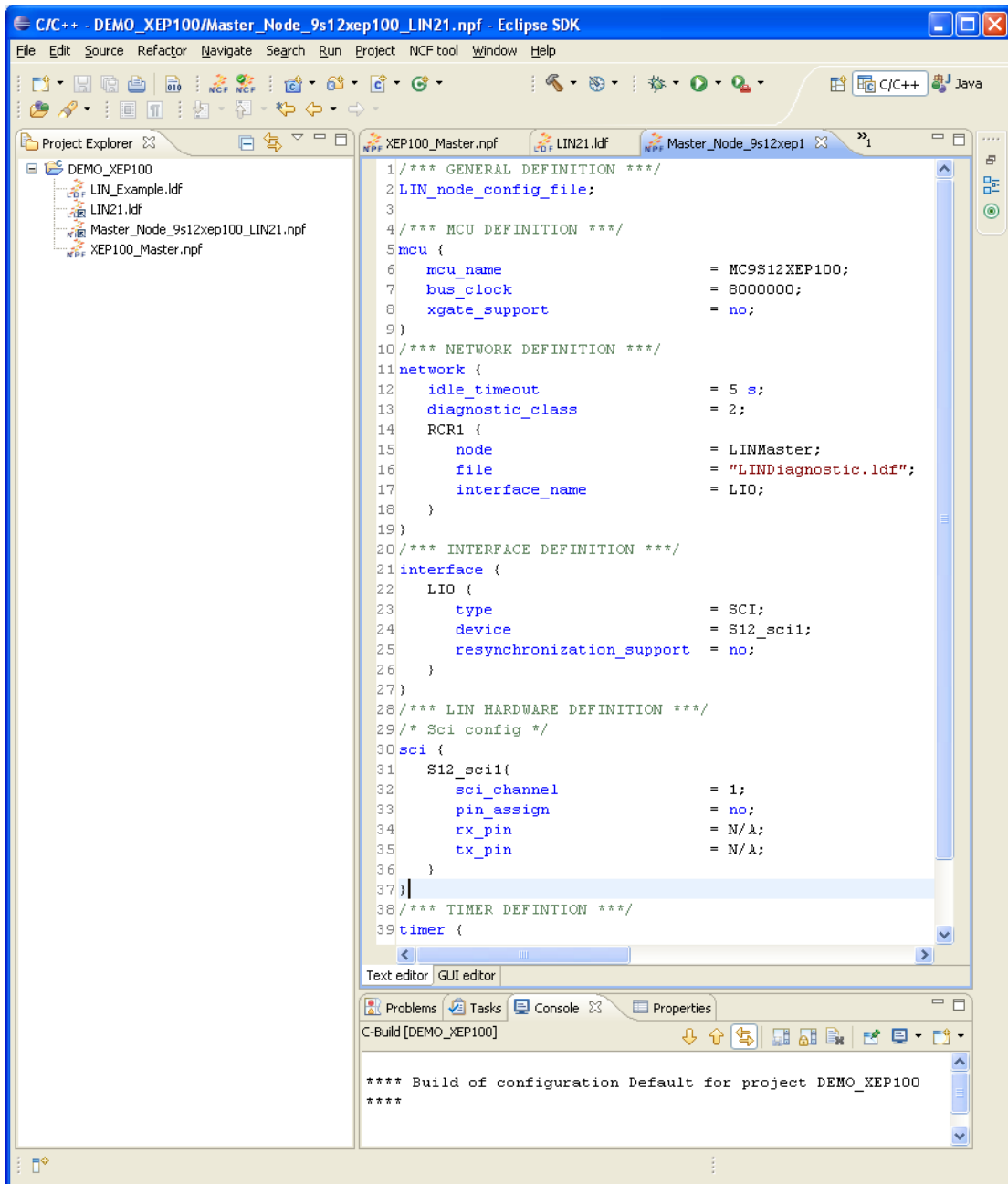


Figure 3-11: External NPF file in text editor mode.

3.3 Eclipse Plug-in graphical editor

Graphical editor is one of outstanding feature in the Plug-in where user can convert text editor to graphical modes and via verse. This section aims at showing graphical editor for LDF and NPF files.

3.3.1 LDF file graphical editor

Based on LDF file template has been created in the section [3.1.1](#), click tab to change mode from text to GUI editor at the bottom of editor screen. The steps below show completed example to create LDF file by graphical editor mode.

1. Click to “GUI editor” tab

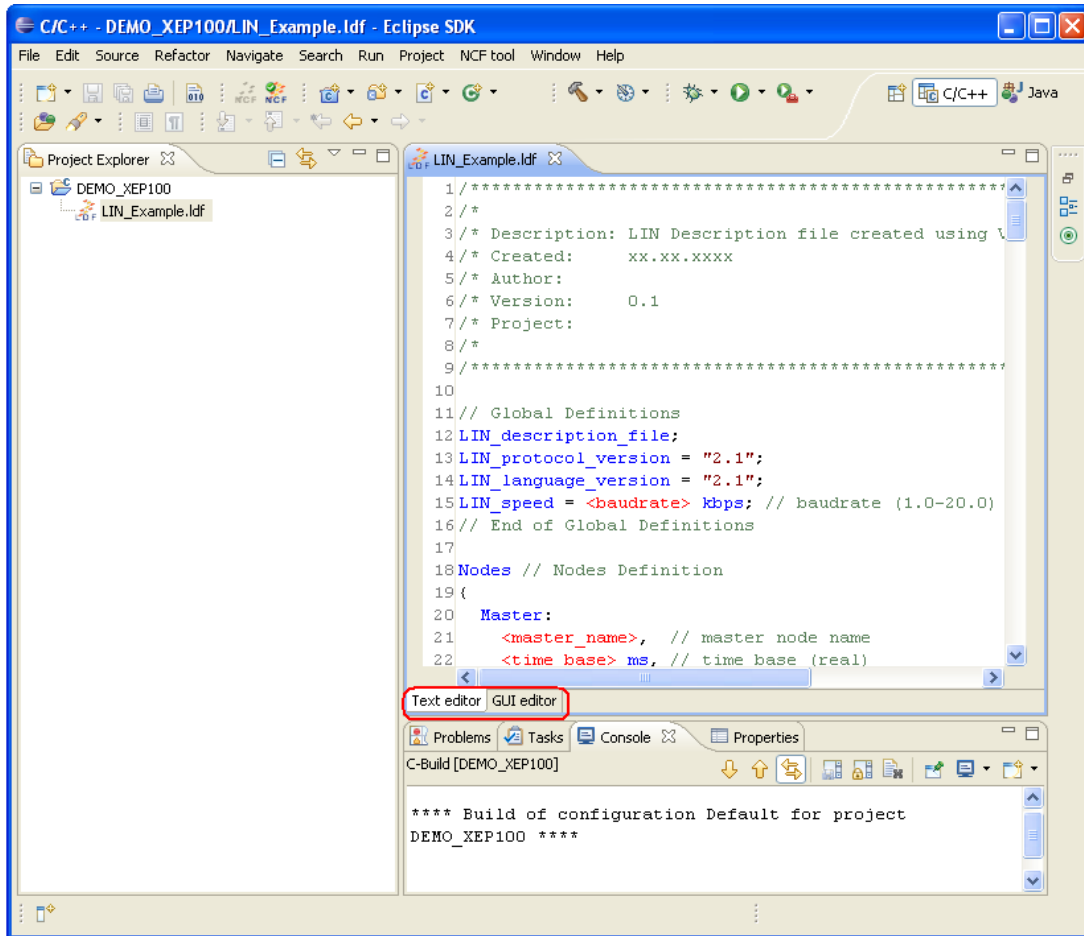


Figure 3-12: Select GUI editor tab

2. In the first dialogue of GUI editor, an error appears as in [Figure 3-13](#) .The reason that the LDF file in text editor mode has no information or information is incorrect.

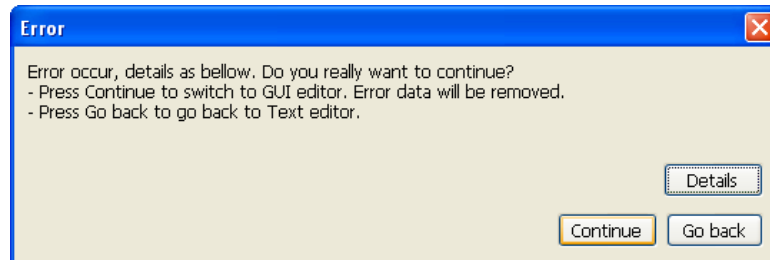


Figure 3-13: Error dialogue in first time

3. Click “Continue” button to continue in GUI mode. The GUI editor displays as

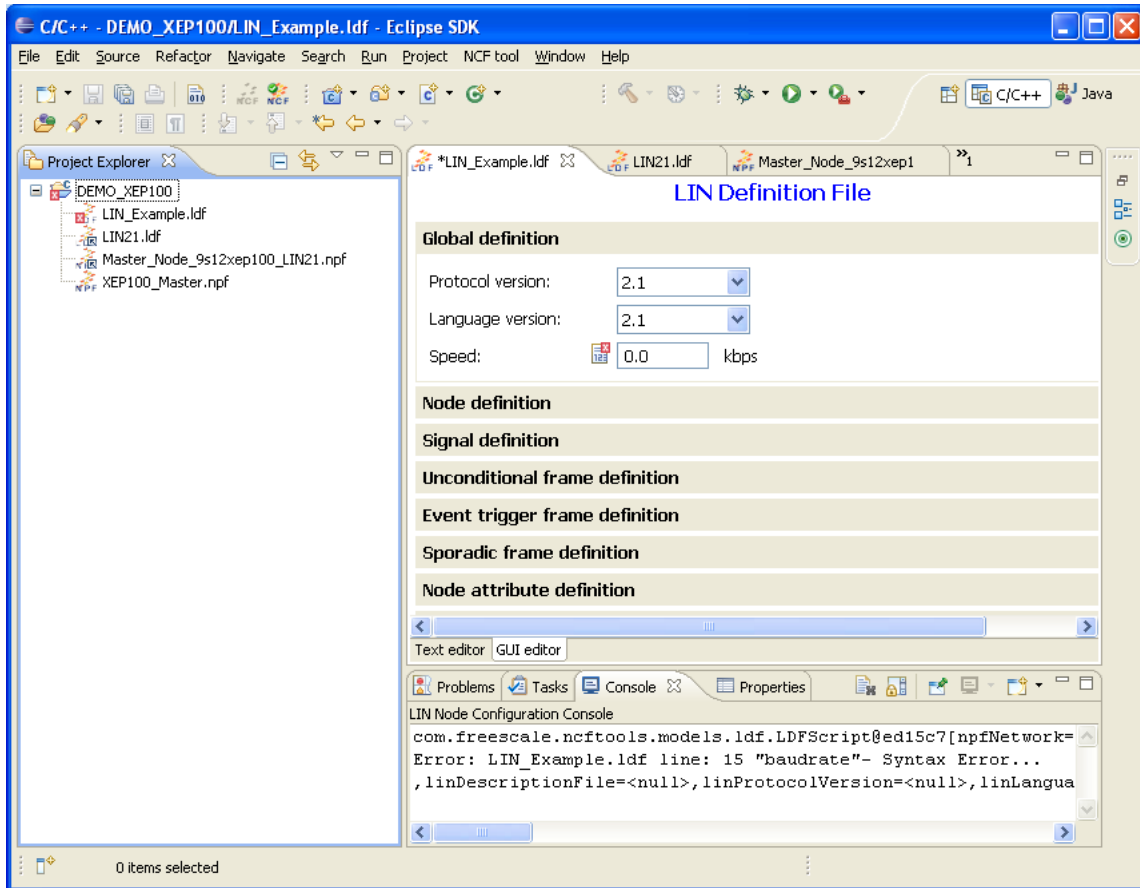


Figure 3-14: GUI editor for LDF file

3.3.1.1 Global definition

- In the “Global definition” tab, select value for “protocol version = 2.1”, “Language version = 2.1” and “Speed = 9.6kbps”.

Note:

Speed for J2602 protocol is fixed with value 10.417 kbps

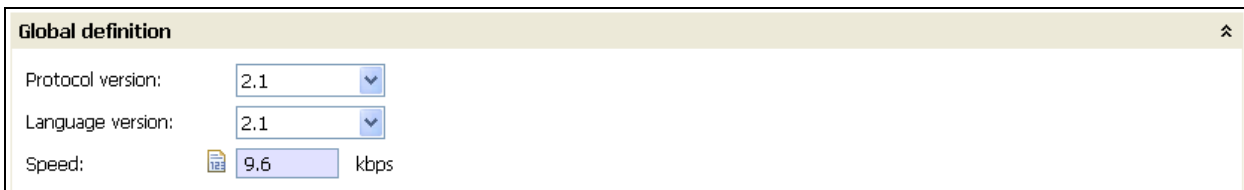


Figure 3-15: Global definition editor

3.3.1.2 Node definition

- Select tab “Node definition” and add name for master node and “time base”, “jitter” values

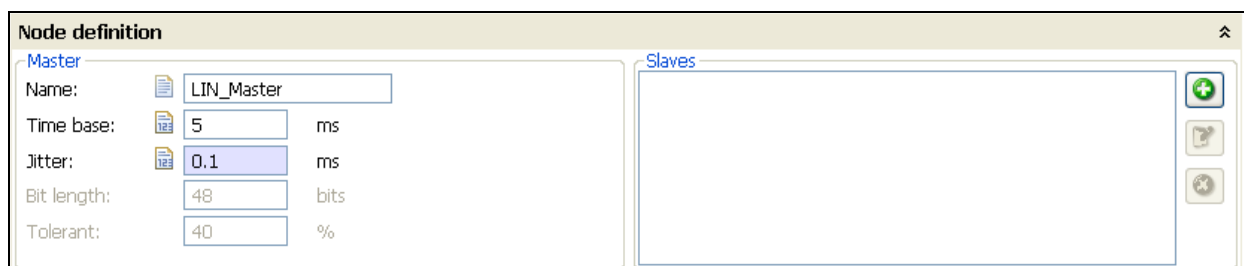



Figure 3-16: Add information for master node

Note:

The fields “Bit length” and “Tolerant” are enable only in J2602 protocol.

- Click button  to create slave nodes. In this example, two slave nodes are created with name Motor1 and Motor2

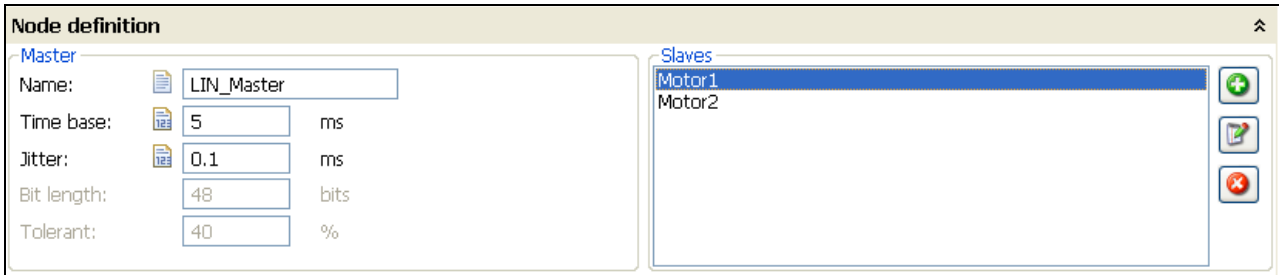


Figure 3-17: Create slave nodes in the network

3.3.1.3 Signal definition

- Select tab “**Signal definition**”, click button  to create new signal data. In the new dialogue, add information for Name of signal, signal type, initial value, Publisher.

For selection Subscriber, click button  to add Subscriber data as below

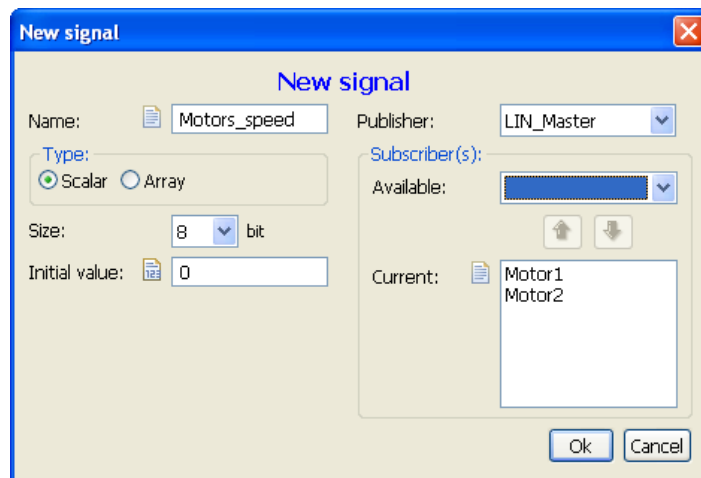



Figure 3-18: Editorial signal dialogue

In this example, the list of signals created is shown as in the [Figure 3-19](#).

Name	Size...	Type	Publisher	Subscriber(s)	Init value
Motor1_temp	8	Scalar	Motor1	LIN_Master	0
Motor2_temp	8	Scalar	Motor2	LIN_Master	0
Motor1_pos_error	1	Scalar	Motor1	LIN_Master	0
Motor2_pos_error	1	Scalar	Motor2	LIN_Master	0
Motors_direction	2	Scalar	LIN_Master	Motor2, Motor1	0
Motor2_pos	16	Array	Motor2	LIN_Master	0,0
Motor1_pos	16	Array	Motor1	LIN_Master	0,0
Motor2_speed_setting	16	Array	LIN_Master	Motor2	0,0

Figure 3-19: List of signal definition

3.3.1.4 Unconditional frame definition

- Select tab “**Unconditional Frame definition**”, the expanding item for frame editor is appeared, click  button to create new frame for the network.

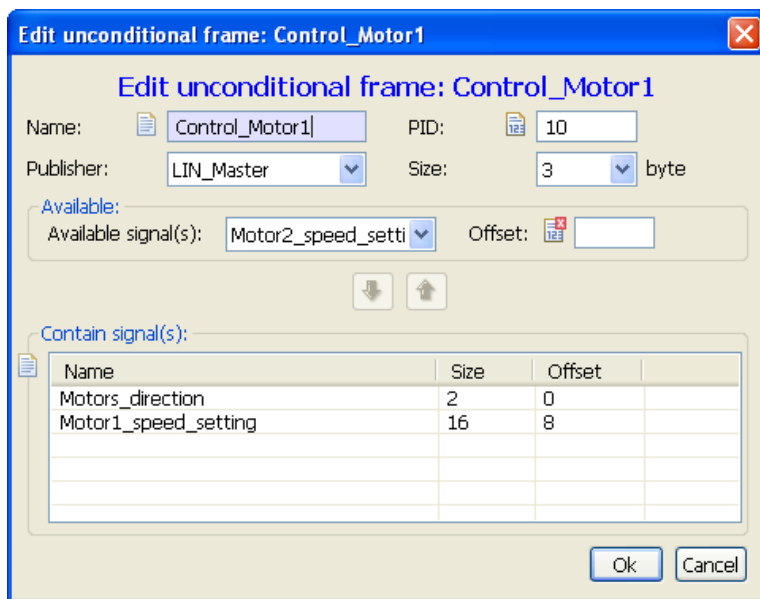



Figure 3-20: Editorial unconditional frame dialogue

The completed frame definition in this example is listed below

Unconditional frame definition					
Name	ID	Published by	Size...	Signal(s)	
Control_Motor1	10	LIN_Master	3	Motors_direction, Motor1_spe...	
Motor1_update	11	Motor1	3	Motor1_temp, Motor1_pos	
Motor2_update	12	Motor2	3	Motor2_temp, Motor2_pos	
Motor1_ETFCollision	30	Motor1	4	Motor1_pos_error, Motor1_pos	
Motor2_ETFCollision	31	Motor2	4	Motor2_pos_error, Motor2_pos	
Control_motor2	13	LIN_Master	3	Motors_direction, Motor2_spe...	

Figure 3-21: List of frame definition

3.3.1.5 Event trigger frame definition

- Select tab “**Event trigger frame definition**”, the expanding item for frame editor is appeared, click  button to create new frame for the network.

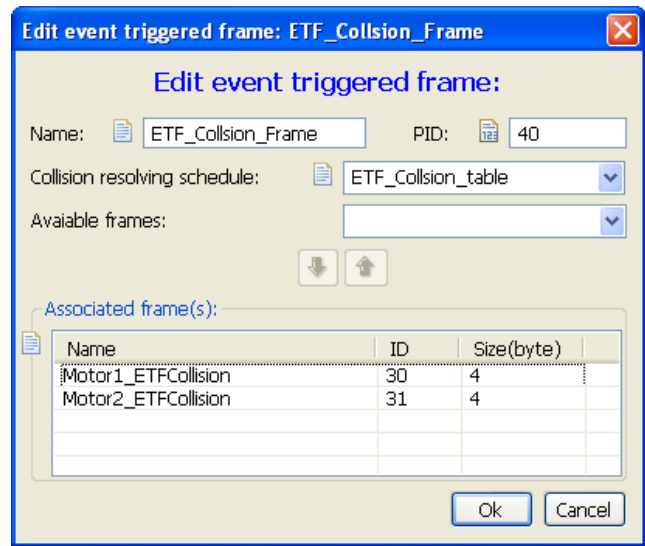


Figure 3-22: Editorial event trigger frame dialogue

Note:


The “Collision resolving schedule” can be edited or selected from the existing a schedule in the “Schedule table” definition.

The completed frame definition in this example is listed below

Event trigger frame definition			
Name	Resolving schedule	ID	Associated frame(s)
ETF_Collision_Frame	ETF_Collision_table	40	Motor1_ETFCollision, Motor2_ETFCollision

Figure 3-23: List of event trigger frame definition

3.3.1.6 Sporadic frame definition

Similar to event trigger editor, select tab “Sporadic frame definition”, the expanding item for frame editor is appeared, click  button to create new frame for the network

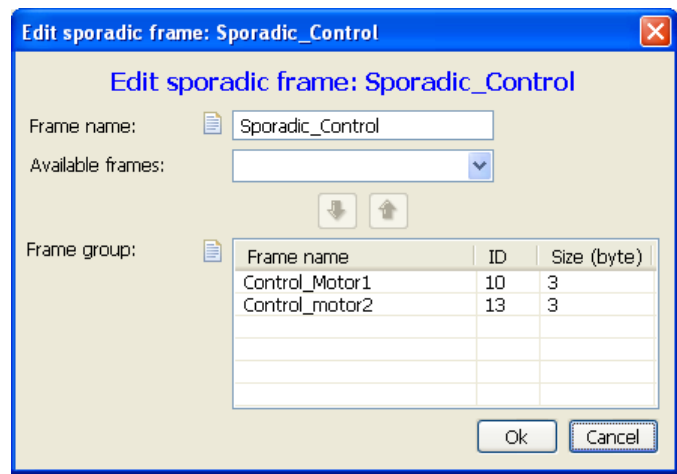



Figure 3-24: Sporadic frame dialogue

The completed frame definition in this example is listed below

Name	Associated frame(s)
Sporadic_Control	Control_Motor1,Control_motor2

Figure 3-25: List of sporadic frame definition

3.3.1.7 Node attribute definition

10. Select tab “Node attribute definition”, the expanding item for frame editor is appeared, click  button to create new frame for the network.

Edit node attribute: Motor1

Name: Configured NAD:

Protocol version: Initial NAD:

Product ID

Supplier ID: Fault state signals:

Function ID:

Variant:

Response error:

P2 min: ms N_As timeout: ms

ST min: ms N_Cr timeout: ms

Configurable frames

Available frames:

Message ID:

Frame name	PID	Message...
Control_Motor1	10	
Motor1_update	11	
Control_motor2	13	
ETF_Collision_Frame	40	
Motor1_ETFCollision	30	

Ok Cancel

Figure 3-26: New node attribute dialogue

The completed Node attribute definition in this example is listed below

Name	Protocol	NAD	Product ID	Response error
Motor1	2.1	1	1000, 10, 1	Motor1_pos_error
Motor2	2.1	2	2020, 20, 1	Motor2_pos_error

Figure 3-27: List of Node Attribute definition

3.3.1.8 Schedule table definition

Schedule table definition contains features for creating normal schedule table and Node Configuration and Identification.

11. Select tab “Schedule table definition”, the expanding item for frame editor is appeared, click



button to create new schedule tables for the network.

12. Type name of table in the “Name” filed
13. In the combo box of Item type, select type of frame will be added to this table as shown in **Figure 3-28**

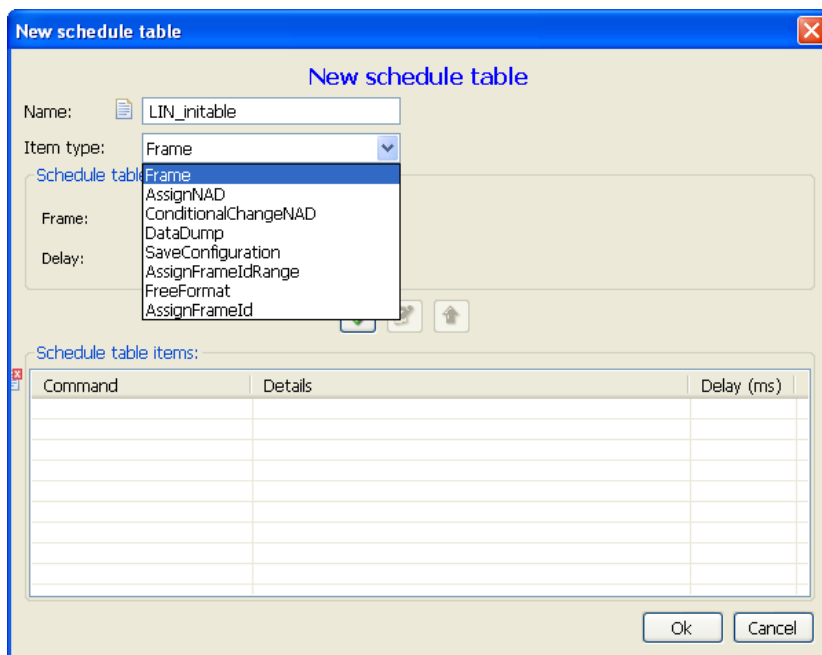
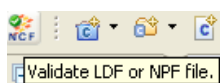


Figure 3-28: New schedule table dialogue

The completed Schedule table definition in this example is listed below

Schedule table definition		
Name	Frame(s)	
MotorControl	Control_Motor1,Control_motor2	
ETF_Collision_table	Motor1_ETFCollision,Motor2_ETFCollision	
MotorsUpdate	Motor1_update,Motor2_update	
LIN_Initable	AssignNAD,SlaveResp,AssignNAD,SlaveResp	

Figure 3-29: Completed schedule table definition



14. Select icon “NCF” to validate the LDF file we have created already.
15. Convert to text editor to see the file in text mode.

Note:

As mentioned in section [2.2.2](#), some fields of LDF files will be further implementation with the reason that the data in these definitions are not utilized in LIN Stack currently.

3.3.2 NPF file graphical editor

Based on NPF file template has been created in the section 3.1.2, click tab to change mode from text to GUI editor at the bottom of editor screen. The steps below show completed example to create NPF file by graphical editor mode.

1. Convert to graphical mode Text editor GUI editor

In first glance, you will see an error due to no information in the NPF file from text editor as

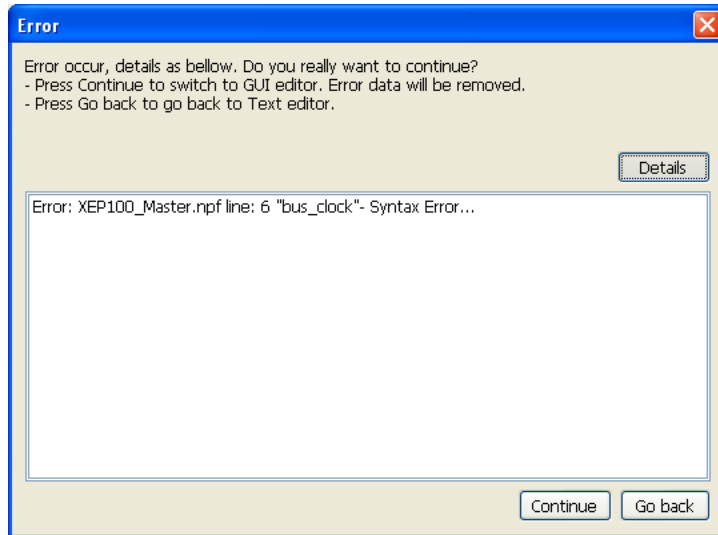


Figure 3-30: Error message for NPF file in the first glance

2. Click to “Continue” button to edit NPF file in graphical mode.

Note:

All errors must be fixed before switch from text to GUI and GUI to text.

3.3.2.1 Global definition

3. In the first tab of graphical editor, add general information for NPF file.

The example as shown in Figure 3-31 defines the NPF for MC9S12XEP100 MCU with bus clock of 8MHz, no XGATE support, diagnostic class 3, Idle timeout for LIN bus is 5s, Autobaud support not supported, Multi-timer channel disabled and Message length (that is max_message_length) 106 bytes.

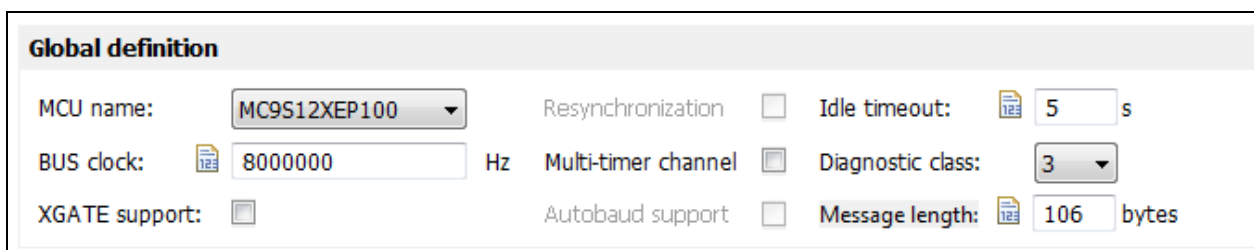


Figure 3-31: Global definition

3.3.2.2 Hardware definition

4. Select “Hardware definition” tab to create HW interface and channel if existed. HW interface include SCI, SLIC and GPIO. In SCI, some MCUs contain more than one channel.

The list of available SCI channel for each MCU is defined in the selected list of creating new HW dialogue.

How to use Plug-in


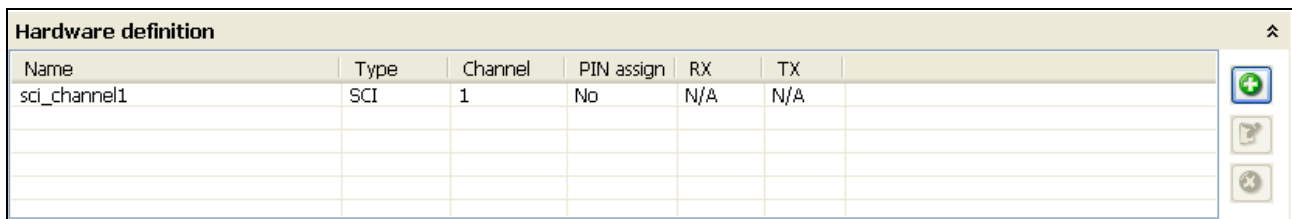
5. Click to  button to create new hardware channel



Figure 3-32: New hardware dialogue

6. If you are expecting to create this NPF file for multiple LIN channels, create more than one hardware channel in this definition. The scope of this example is to create a note in single LIN channel which is linked with LIN_example.ldf file we have created previously.
7. Click "OK" button, the list of hardware channel is shown in [Figure 3-33](#)

A table titled "Hardware definition" with a scroll bar on the right. The table has columns: Name, Type, Channel, PIN assign, RX, and TX. The first row contains: sci_channel1, SCI, 1, No, N/A, N/A. There are three empty rows below. On the right side of the table are three icons: a plus sign, a refresh icon, and a minus sign.

Name	Type	Channel	PIN assign	RX	TX
sci_channel1	SCI	1	No	N/A	N/A

Figure 3-33: List of hardware definition

3.3.2.3 Interface definition

8. Select "Interface definition" tab to create new interface for this node

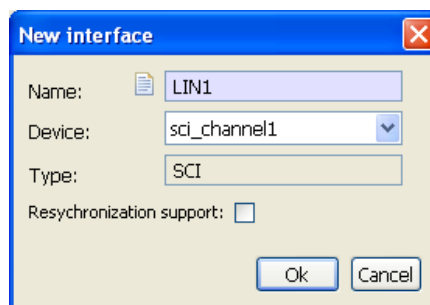
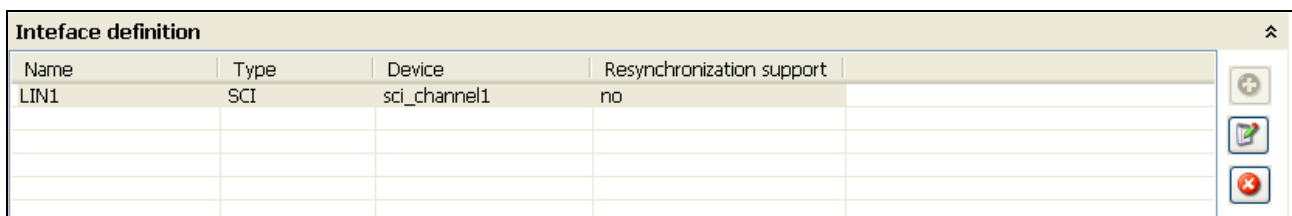


Figure 3-34: New interface dialogue

The interface definition dialogue contains a field for section "Resynchronization support" which is applicable for slave node only and with specific MCUs: 9S08DZ60, 9S08DZ128, 9S08EL32, 9S08SG32, 9S08MP16.

9. Click "OK" button, the list of interface is shown in [Figure 3-35](#)

A table titled "Interface definition" with a scroll bar on the right. The table has columns: Name, Type, Device, and Resynchronization support. The first row contains: LIN1, SCI, sci_channel1, no. There are three empty rows below. On the right side of the table are three icons: a plus sign, a refresh icon, and a minus sign.

Name	Type	Device	Resynchronization support
LIN1	SCI	sci_channel1	no


Figure 3-35: List of interface definition

3.3.2.4 Network definition

How to use Plug-in

This is the final step to create NPF file.

10. Select **“Network definition”** tab to create new network for this node

11. Create new network by clicking button .

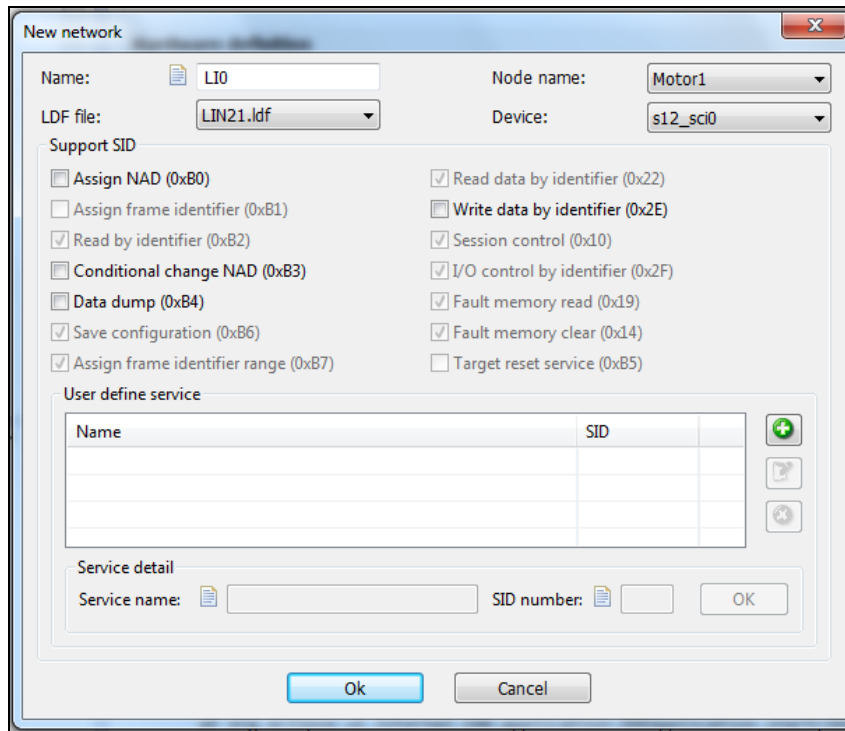



Figure 3-36: New network dialogue

The dialogue shows information for the node to select from the Combo box. Click to pop-up menu to see available for LDF file, name of the node in LDF file that current node participate in, and Interface utilized in the node.

12. User can choose SID that the current node should support. Some SIDs are dimmed because LIN Stack support them by default, since these SIDs are mandatory according to the diagnostic class of the current node. Users can choose to support optional SID (e.g. Assign NAD SID) by checking the option box of that SID. If users do not want to support optional SID (e.g. Assign NAD SID), they just leave the box of that SID unchecked as default.

For Diagnostic Class III, users also can define User Defined Diagnostic Services. Create new

User Defined SID by clicking button . Users can input Service names and SID numbers as decimal values, e.g. User_SID_01 is 174.

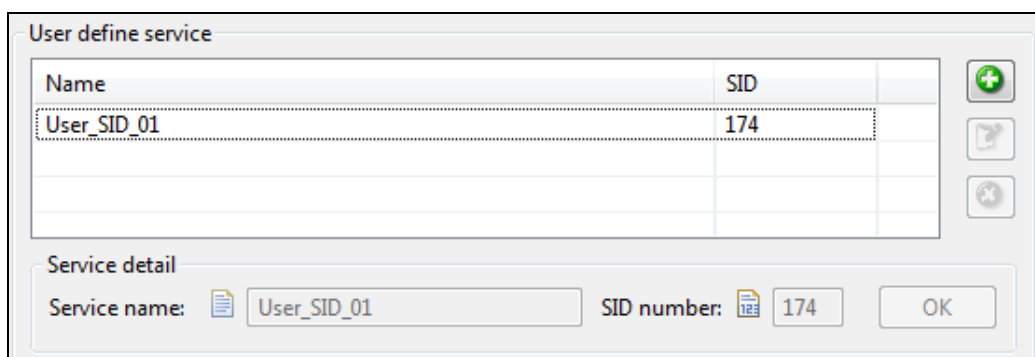


Figure 3-37 User Defined SID

13. Click **“OK”** button, the list of network is shown in [Figure 3-37](#)


How to use Plug-in

Network definition				
Name	LDF file	Node	Device	SID
LI0	LIN21.ldf	SeatECU	s12_sci0	0xB2, 0xB6, 0xB7, 0x22, 0x10, 0x2F, 0x19, 0x14, 0xB0, 0xB3, 0xAE

Figure 3-37: List of network definition

3.3.3 Generate configuration code

After completion creating NPF file, it is time to create configuration code (.h and .c).

14. Click the button  in the tool bar and select destination folder to store output file

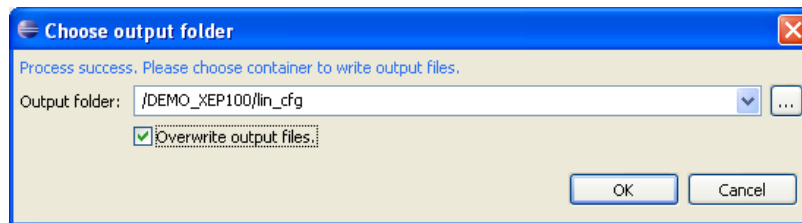


Figure 3-38: Generation configuration code dialogue

15. Click "OK" button, the final configuration code is shown in Figure 3-39

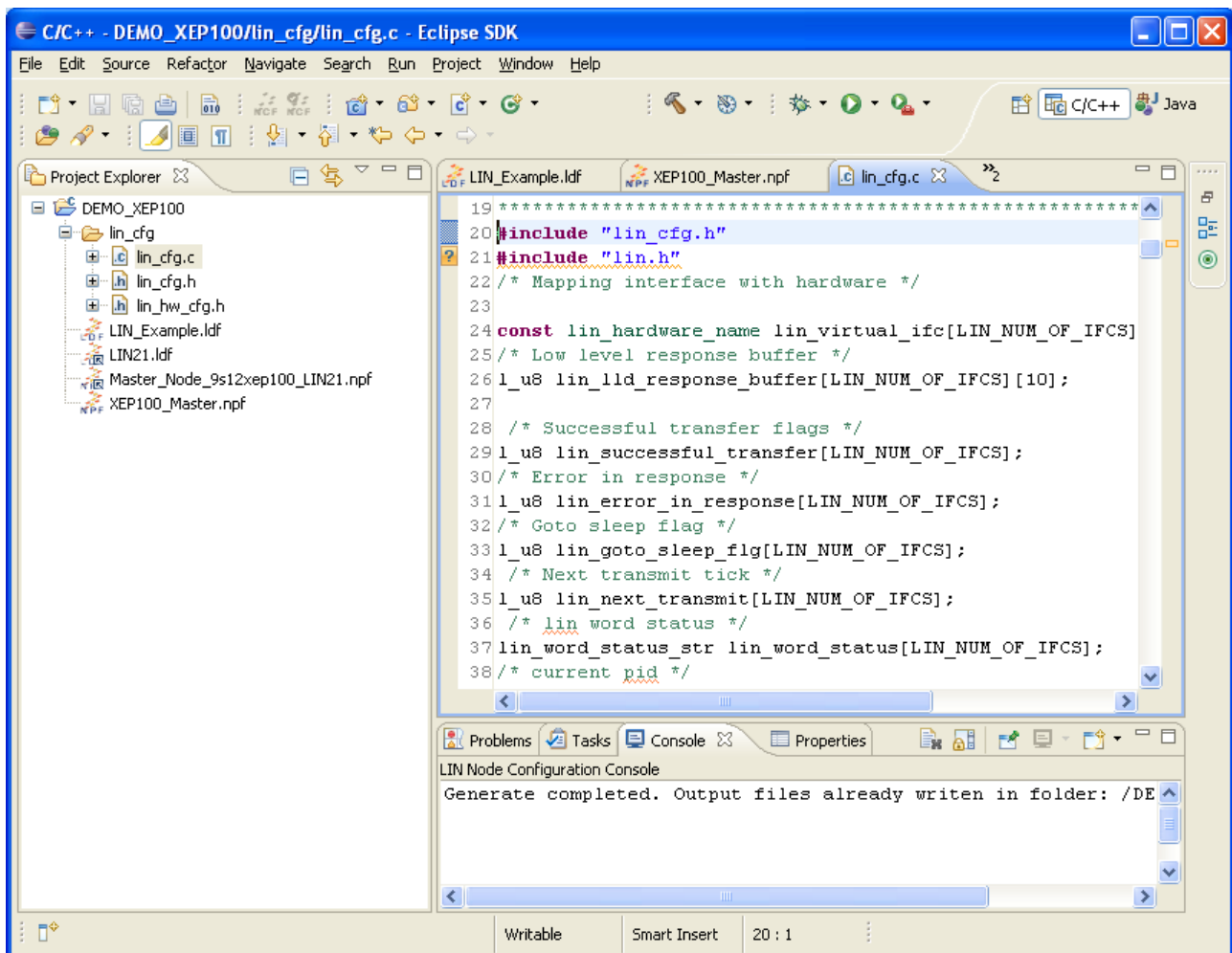


Figure 3-39: Configuration code for XEP100

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

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.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTest, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and μ Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2017 NXP B.V.

Document Number: NCFTOOLUSERMANUAL
Rev. 1.3.3