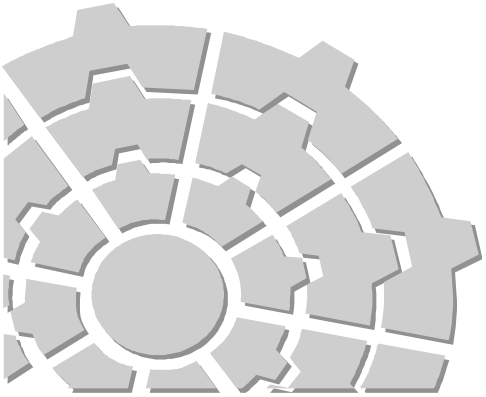
DeltaLink Redundancy Module for FactoryLink

User Manual



Version 3

Printed: 26 juni, 1994

Table of Contents

This page is left blank intentionally.

1. Introduction

Thank you for buying the Redundancy Module! We hope you will enjoy using this product.

1.1. Scope of this document

This manual is written for a technician who is familiar with the FactoryLink® IV software. This document can be used both as a training manual as well as a reference manual.

> ***Note:*** *Please check the contents of the shipment with the list as described in the next chapter.*

The first section of this manual deals with the installation of the software in your FactoryLink workstation. This part is split into a platform independent and a platform specific part. Please read carefully through both parts to make sure both hardware and software are installed correctly.

The second part explains the operation principles of the redundancy module. All functionality's supplied by the module will be discussed in this section such as the heart beat, tag copy, dataset duplication, external database synchronisation and mode change actions.

The third part explains the exact tables associated with the redundancy module. This part is useful only to FactoryLink programmers and can be used as a reference. All tables entries will be explained with a summary of valid entries.

The fourth section will explain the integration of the redundancy module in an application. It is very important to configure an application in the right way because the module depends on other standard FactoryLink tasks. Also project risks will be discussed in this section when using the redundancy module.

The last part are the Appendices which contain summarised data.

DeltaLink bv, Pioenroosstraat 26, 5241AB Rosmalen, The Netherlands,

tel + 31 73 5231234
fax +31 73 5231222

1.1.1. Contents of shipment

Please check the package you received with the checklist below. Should there be an item missing, contact DeltaLink to correct the problem. There is a limit of 90 days after shipment to report problems !

This package includes the following:

1 diskette labelled "DeltaLink Redundancy Module"

Files:	\AC\FL_RED.AC	
	\BIN\FL_RED.EXE	(Windows - OS/2)
	\BIN\FL_RED	(Unix)
	\CTGEN\FL_RED.CTG	
	\KEY\DE\FLR_ACT.KEY	
	\KEY\DE\FLR_MOD.KEY	
	\KEY\EN\FLR_ACT.KEY	
	\KEY\EN\FLR_MOD.KEY	
	\KEY\FR\FLR_ACT.KEY	
	\KEY\FR\FLR_MOD.KEY	
	\MSG\DE\FL_RED.TXT	
	\MSG\DE\FL_REDAC.TXT	
	\MSG\EN\FL_RED.TXT	
	\MSG\EN\FL_REDAC.TXT	
	\MSG\FR\FL_RED.TXT	
	\MSG\FR\FL_REDAC.TXT	
	\HELP\DE\FL_RED.HLP	
	\HELP\EN\FL_RED.HLP	
	\HELP\FR\FL_RED.HLP	
	\OPT\DELTA.OPT	
	\INSTALL.BAT	(Windows)
	\INSTALL.CMD	(OS/2)
	\INSTALL	(UNIX)
	\INST_SEQ.EXE	
	\FLBUILD.ID	
	\FLXMEDIA	(Windows - OS/2)
	\FL_RED.$$$	

❷ 1 DeltaLink authorisation sequence containing FL_RED option. In case this sequence code has been delivered with the floppy then it will be in the */OPT/DELTA.OPT* file

This manual (Which seems to be present).

You should also have:

☑ A correct installed and running FactoryLink workstation.

This page is left blank intentionally.

2. Installation

2.1. Installation of the FactoryLink software.

To install the FactoryLink task and its related tables please follow the following steps.

Before installing

Before installing the redundancy module on the system FactoryLink must have been installed error free. It is very important that all the environment settings are made for the FactoryLink system such as the *FLINK*, *FLOPT* etc because the installation procedure depends on it.

First:

The redundancy module can be installed by running the install utility that is placed in the root directory of the install media. This utility will take every action that is needed to install the module. However the next steps describe what the installation utility does so that the user can check if the module has been installed error free. Normally this first step is sufficient for installation, the next steps can be used for reference.

The installation procedure differs for UNIX and Windows-OS/2 systems.

For Windows-OS/2 systems follow the next procedure:

```
a:\↵
install↵
```

For UNIX systems first an install directory must be created. The files on the install floppy or tape must be first copied to the install directory using the *tar* command. From this directory the install utility can be run.

For UNIX systems follow the next procedure:

```
mkdir install
cd install
tar xv↵
install↵
```

Second:

The installation automatically appends the *fl_red.ac* entry into the *{FLINK}/AC/titles* file[1]. The place of this entry is also the place where the option appears in the FLCM Main Menu. Therefore check the validity of the entry and move it to the place where it must appear in the Configuration manager. The entry must match the *fl_red.ac* entry in the following table:

file: {FLINK}\ac\titles:

...
spool.ac
itimer.ac
etimer.ac
fl_red.ac
sinec_h1.ac
tiway_1.ac
...

Third:

To make sure all the Configuration Tables (CT's) are generated after a change, the install utility automatically adds the *fl_red* entry at the end of the *{FLINK}/ctgen/ctlist* file. The place of this entry is not important. Check if this entry has the same format as in the next table:

file: {FLINK}\ctgen\ctlist:

...
rp: rptovr rpthdr
sinec_h1: sinech1m sinech1x sinech1p sinech1d
fl_red: flr_def flr_hist flr_mci flr_act flr_tag
timer: itimer etimer
...

This completes the installation of the FactoryLink (software) parts.

[1]{FLINK} is the working directory for the FactoryLink programs.

Fourth:

To enable the help fields in the redundancy module type the following command:

{FLINK}\BIN\mkhelp↵

Fifth:

The redundancy module must be entered in the FactoryLink Configuration Manager (FLCM) System Configuration table. An entry of an existing task which will not be used at run-time can be overwritten or a new entry can be made with as a minimum the following data:

Task Name	*Description*	*....*	*Executable File*
FL_RED	***DeltaLink Redundancy Module***		***bin/fl_red***

The *Task Name* and *Executable File* name are fixed and should not be altered by the user.

This completes the installation of the FactoryLink (software) parts.

2.2. Installation of the authorisation code

The redundancy module has been protected with an authorisation code. This code must be present in the DeltaLink option file which must reside in the {FLINK}\OPT directory. This file contains authorisation sequence codes for DeltaLink modules. The protection is linked to the serial number of the FactoryLink package.

2.2.1. The DeltaLink option file

The installation disk of the redundancy module contains a delta.opt file in the '\opt' directory. In case for protection with an option file then this file contains the unique authorisation sequence which enables the redundancy module to run. The install utility automatically copies the authorisation sequence to the {FLOPT}\delta.opt file. It is also possible to enter the authorisation sequence manually into the {FLOPT}\delta.opt file. For more information on the delta.opt file refer to *Appendix B*.

Note that the driver will only run on the FactoryLink system with the same serial number.
The delta.opt file on the installation diskette contains the serial number of FactoryLink.

2.2.2. Demo installation.

It is possible to install the redundancy module without an authorisation code. This will be done with the normal installation diskette with the difference that the authorisation code is not present in the *delta.opt* file. The module will start up but will only run in demo mode. This means that the redundancy module runs only for a period of consecutive time (usual five hours). After this period has expired the module will shutdown automatically and can not be restarted before the total FactoryLink system has been restarted.

After installation of this demo diskette an authorisation code or protection key can be ordered and installed which enables the driver to run with full functionality. No new software has to be installed, just the authorisation code must be entered manually in the {FLOPT}\delta.opt file. After entering the code the redundancy will start up in normal mode and will have no limitations. For more information on entering the authorisation code in the delta.opt file refer to *Appendix B*.

The limitations of a demo version of the redundancy module are:

- five hours of consecutive run-time
- not restartable (FactoryLink application must be restarted)

3. Principles of operation

The redundancy module implements a toolbox to create a full redundant SCADA system using two separate FactoryLink systems. It is not the intention of the Redundancy module to implement redundancy functionality at a low level which is not visible for the application programmer, but at a high level. This means that a FactoryLink application must be redundant aware. The application programmer has to implement the redundancy functionality by configuring the right tags at the right panels. The redundancy module gives the application programmer a toolbox which makes the integration of redundancy functionality very easy in an application.

3.1. Redundancy definition

The definition of redundancy according to the redundancy module is: to keep parts of the Real Time Database (RTDB) and the data logged in external databases consistent between two functional identical FactoryLink applications at any time so that in case of a fault on one system the other system can take over control. The redundancy module has the duty to detect a fault condition on one of the systems. The module initiates internal actions when taking over control but also supplies tools for external action which must be programmed in the application.

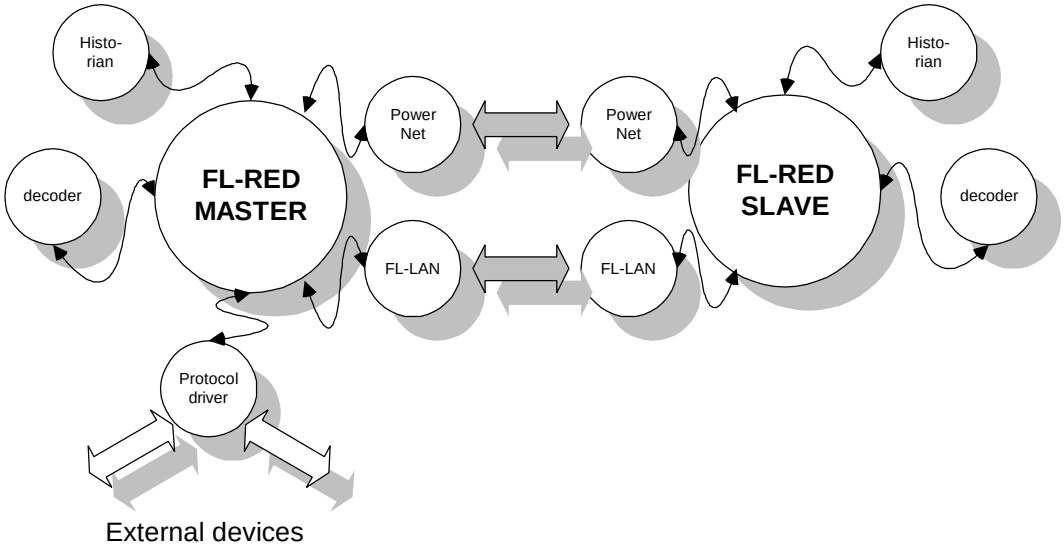


Figure 4.1 Redundant system

A functional identical FactoryLink application is an application which has been duplicated and adjusted at minor parts for implementing redundancy with the DeltaLink redundancy module.

It is very important for two redundant systems to control the flow of data. In case of FactoryLink, the flow of data always starts in external devices and comes in through protocol drivers. In a redundant system one system controls the source of the data, the other system will be updated with this data. For this reason two modes has been defined:

- Master mode
- Slave mode

In a normal situation one system runs in the master and the other in slave mode where the master system controls the data flow. Figure 4.1 displays the master and slave system. The protocol driver is active on the master system which then also updates the slave system.

A redundant system also runs in certain state, independent of the mode. Both systems usually run in the same state at run-time. The following states exist:

- Initialization state
- Running state
- Synchronization state
- Fault state

The definitions described above implies the following functional areas:

- Fault detection with heart beat functionality (Master/Slave)
- FactoryLink RTDB synchronization
- External database synchronization
- IMX / MCI data set duplication
- Action initiating on a state change

It is possible to implement the same redundant functionality in an application in more than one way. For example: to synchronise the RTDB by synchronising individual tags or by using IMX / MCI for duplication of data sets. This means that an application programmer can implement redundancy in different ways in the FactoryLink application and still have the same functionality in the end.

3.1.1. Redundancy dependencies

The figure below shows a FactoryLink application in a redundant environment with all the necessary tasks to implement all functional areas of the redundancy module.

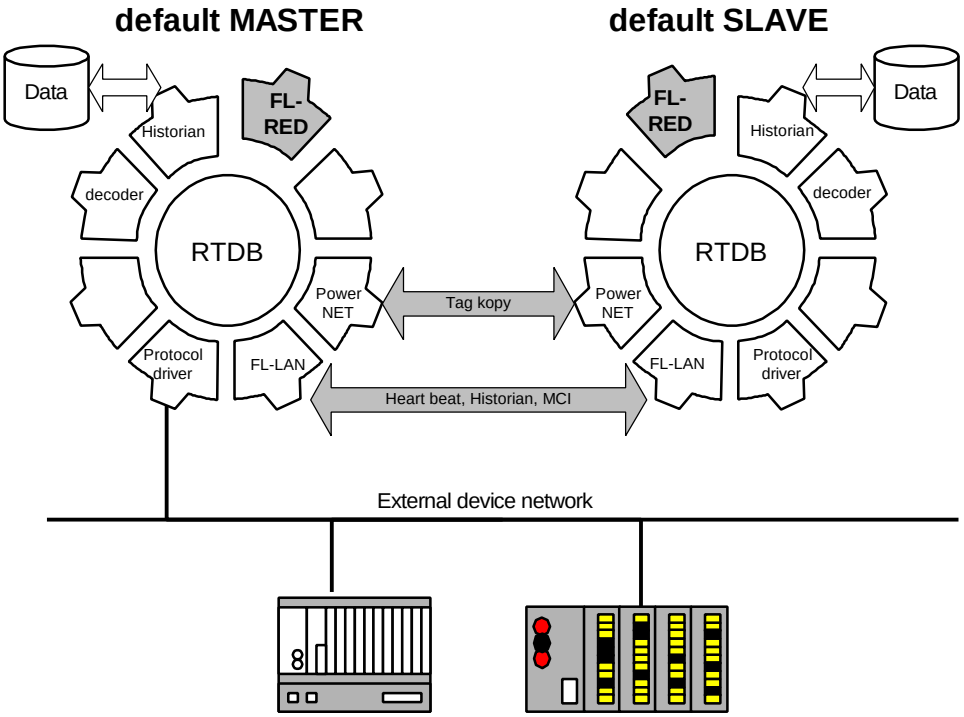


Figure 4.2 Schematic of a redundant application

Figure 4.2 shows two FactoryLink applications on different systems which form together a redundant system. The system consist of almost two identical FactoryLink applications connected through a Local Area Network (LAN) and an external device network which can be of any type. The applications are functional identical as described in the redundancy definition section.

As we see in the figure, the redundancy module depends on other tasks for its operation. These tasks are standard FactoryLink tasks:

- FL-LAN task
- Power NET task
- Historian task
- Communication drivers

The *FL-LAN* task is required for exchanging mailbox messages over the LAN between the two FactoryLink applications. Almost all functionality's in the redundancy use the FL-LAN task. For example the heart beat for exchanging heart beat messages, historian synchronisation for exchanging data when recovering from a fault and IMX / MCI data-set duplication for re-routing data sets from one system to another.

The *PowerNet* task is required for synchronisation of the FactoryLink Real Time Database (RTDB) on basis of individual tags. This functional area is covered in the Tag Copy function of the redundancy module.

The *Historian* task is required for communication with external databases. In a normal situation, the redundancy module does not have to communicate with historians. Communication with the external databases is needed after a fault period. In this case the redundancy module will synchronise the databases on both machines.

Communication drivers are required for communicating with external devices. Three types of drivers exist in the FactoryLink system: EDI drivers, third party drivers and the IMX / MCI drivers. The redundancy module operates with all type of drivers. The difference between EDI, third party and IMX / MCI drivers is that EDI and third party drivers update data directly on tags and MCI / IMX drivers generate mailbox messages which contain data sets. IMX / MCI based protocol drivers are the most suitable for use with the redundancy module because they are very easy to configure in the redundancy module and to implement redundant functionality and they have the best performance at run-time.

3.2. Redundancy functional areas

The redundancy module covers a number of functional areas which are required for operation or which can be used as a tool for creating two redundant systems. In this section we will discuss all functional areas and explain their use.

3.2.1. Heart Beat

The basic function of the redundancy module is to check the state of the remote system. At normal operation both systems are running and one system acts as master and the other as slave. The master or slave must be able to detect whether the remote system is running or not. The heart beat function covers the detection mechanism to check the state of the remote system.

Two methods of detection are implemented: the LAN heart beat and the I/O heart beat. The LAN heart beat is the most important one whereas the I/O heart beat can be used as an extension for fault detection.

In a redundant system two redundancy modules communicate with each other via the LAN. Heart beat packets will be exchanged over the LAN to detect if the other machine is running. These packets contain information about the mode and state of the sender. The redundancy module sends heart beat request and expects to receive a heart beat response within a certain interval. If, after a number of retries no response has been received then the other system is likely to be in a fault state. In this case the system will change to the fault state.

The LAN heart beat can be extended with the I/O heart beat , which is optional, for better fault detection. Note that this is an extension and not a replacement, the LAN heart beat is always needed. The I/O heart beat travels through the I/O communication of a protocol driver. The redundancy module supplies two tags: one for writing its own I/O heart beat and one for receiving the remote I/O heart beat. Usually each system (master and slave) will toggle a bit in the external device's memory space. This toggle will signal the remote system an alive beat which means that the system is still running.

The I/O heart beat can be necessary in case the LAN heart beat fails due to an error on the LAN, for example the cable has been broken. In this case both system are running and do not see each other on the network. When no I/O heart beat is used then both systems will change to the master mode and fault state. This means that at this point two systems control the data flow. In case the I/O heart beat is used then the redundancy module can check that the other system is still running and that only the LAN connection has been lost. In this case only the state will changed to the fault state on both systems. The mode of the slave system stays the same and the data flow will be controlled by one system, the master system

3.2.2. RTDB synchronisation

A very important issue with redundant systems is to synchronise data on both systems so that in case of a fault either system can take over control. For FactoryLink this means that parts of the Real-Time Database, the heart of FactoryLink, have to be synchronised. It is not necessary to synchronise the complete database because not all parts are critical for taking over control. It is application dependent which parts of the database have to be made redundant.

To implement to RTDB synchronisation the tag copy function will be used.

3.2.2.1. The Tag Copy function

The redundancy module implements a tag copy function which can be used for different purposes such as the *RTDB Synchronisation* and *Trigger Tag Disable* functionality's.

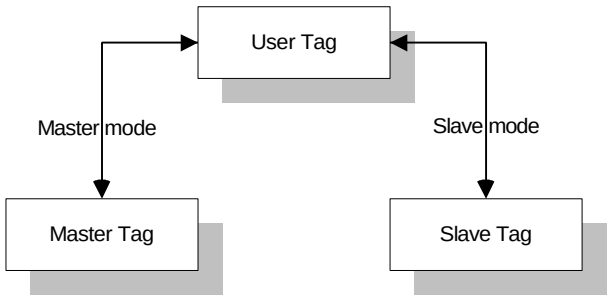


Figure 4.3.2.1 Tag synchronisation function.

The tag copy function implements a simple principle as shown in figure 4.3.2.1. When the system is running in master mode then the redundancy module will copy all tag changes from the master tag to the user tag and vice versa. When the system is running in slave mode then the tag changes of the slave tag will be copied to the user tag and vice versa. The user tag must be used in the application, for example in the graphics or as trigger for communication.

Different methods of use can be thought of for the tag copy function, depending on the application. The two most important and most used are the *RTDB Synchronisation* function in conjunction with PowerNet and the *Trigger Tag Disable* implementation.

3.2.2.2. RTDB synchronisation with tag copy

The most important implementation of the tag copy function is the RTDB synchronisation. For this function PowerNet will be used with the goal to synchronise the user tags on both systems. The source of the data flow will be the value of the master tag on the system which is running in master mode. This value has to be copied to the user tag on both the master and slave system.

The following schematic shows the principle of the tag synchronisation function.

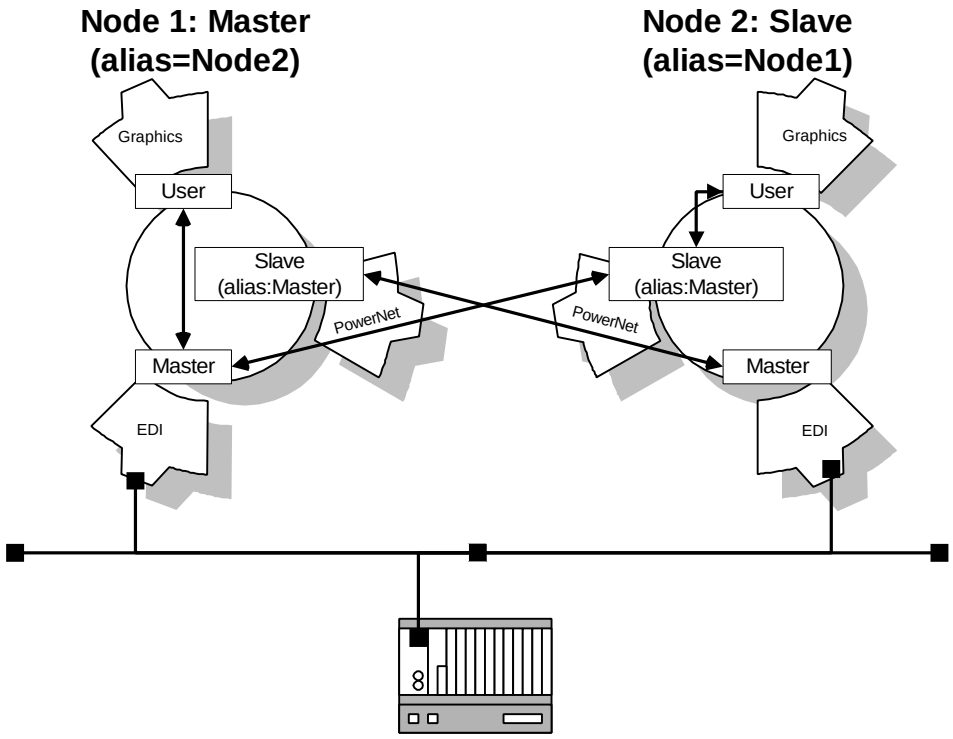


Figure 4.3.2.2 Tag synchronisation function.

The RTDB synchronisation uses the principle of PowerNet of automatic updating tags defined on another system on the LAN. For example when a tag has been defined on Node1 called *temperature* then this tag can be accessed on Node2 as *Node1:temperature*. PowerNet updates the *temperature* tag both ways so *temperature* and *Node1:temperature* will always have the same value.

The figure shows the relation of the tags of PowerNet and how they will be linked to the tag copy function for a tag *temp*.

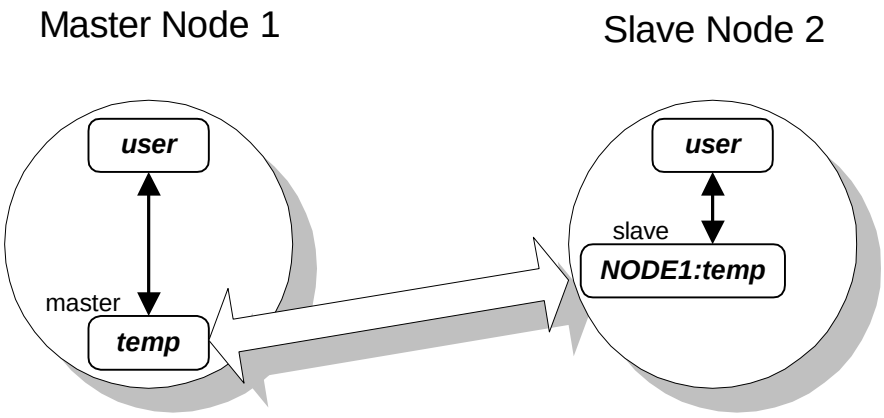


Figure 4.3.2.3 PowerNet tags.

The locally defined tag will always be used as the master tag in the tag copy panels of the redundancy module. The PowerNet counterpart of the tag will always be specified as the slave tag. This means that in master mode always the local tag will be copied to the user tag and in slave mode the tag which remains on the remote machine will be copied. This way the user tags will be synchronised because the master and the slave tags are synchronised by PowerNet.

The principle can be best explained with an example: In this case Node1 is running in master mode and Node2 is running in slave mode. The data flow must be controlled by Node1 according to the definition of redundancy.

When the *temp* tag on Node1 changes then the tag copy function will copy the *temp* tag to the *user* tag. The *temp* tag will also be copied by PowerNet to Node2 on the *Node1:temp* tag. The tag copy function of Node2 detects a change of the slave tag, in this case *Node1:temp* and copies it to the *user* tag. This way the *user* tags on both systems have the value of the master tag on Node1.

This principle also works the other way around. In case the *user* tag on Node2 changes then the tag copy function will copy the value to the *Node1:temp* tag. PowerNet will synchronise the value with the *temp* tag on Node1. The tag copy function in turn detects a change of its master tag and copies the *temp* tag to the *user* tag. The *user* tags on both systems have the same value and if the tag has been specified in a protocol driver it will also be sent to the external device.

Note that the values of *Node2:temp* on Node1 and of *temp* on Node2 will be discarded in this situation by the tag copy function. They become active in case both systems switch from mode, Node1 becomes slave and Node2 becomes master.

This means that the flow control with the tag copy function is always controlled by the master system and that the slave system still can write values to external devices via the master system.

3.2.3. External database synchronisation

On a redundant FactoryLink system both systems log data in external databases via historian tasks to the specific external database. Normally, the source of data for logging in external database comes from the real time database of FactoryLink. This means that the two real time databases of the two redundant systems must contain the same data for logging the same data. This functionality will be handled by the tag copy function for the real time database as described in the previous section.

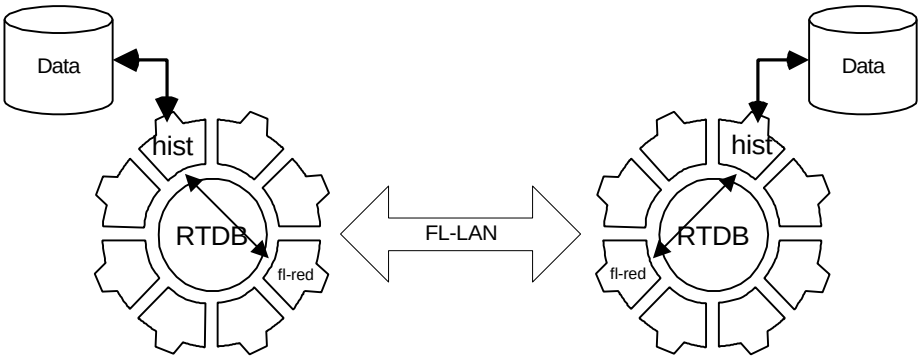


Figure 4.3.3.1 External database synchronisation.

Data will be logged on both systems in normal operation and because both real time databases are the same, the data logged will be the same. At this stage the extrnal database synchronisation function of the redundancy module is idle.

Synchronisation of the external databases is needed after a failure situation. When one system has stopped running for some reason then a gap will exist in its external database during the fault period. Or when the LAN connection has been broken then different data will be logged on both systems because the real time databases are not being synchronised any more.

After a fault period the two redundancy modules will switch to the synchronisation mode. In this mode the two external databases will be synchronised. The redundancy module communicates with standard historian tasks for reading from and writing to the external database. The complete fault period will be checked on both systems and synchronised. Data will be transferred from one redundancy module to the other via the LAN.

The external database synchronisation function expects that on both sides of the system exactly the same external databases are running. This means they must be of the same brand and the databases must contain the same tables with the same schema's. When following the procedure of creating a redundant application, which globally means just copying the application then this requirement will be automatically fulfilled.

3.2.3.1. IMX / MCI Dataset Routing

Drivers within FactoryLink can be divided into two types of drivers as described earlier, The first category updates the data directly on tags, the second generates blocks of data which are called data sets and which travel through mailboxes. These protocol drivers support the IMX / MCI protocol for communicating with decoders. The redundancy module supports the routing of data sets between two redundant systems in order to control the source of the data flow. This principle is covered in the IMX / MCI data set routing function.

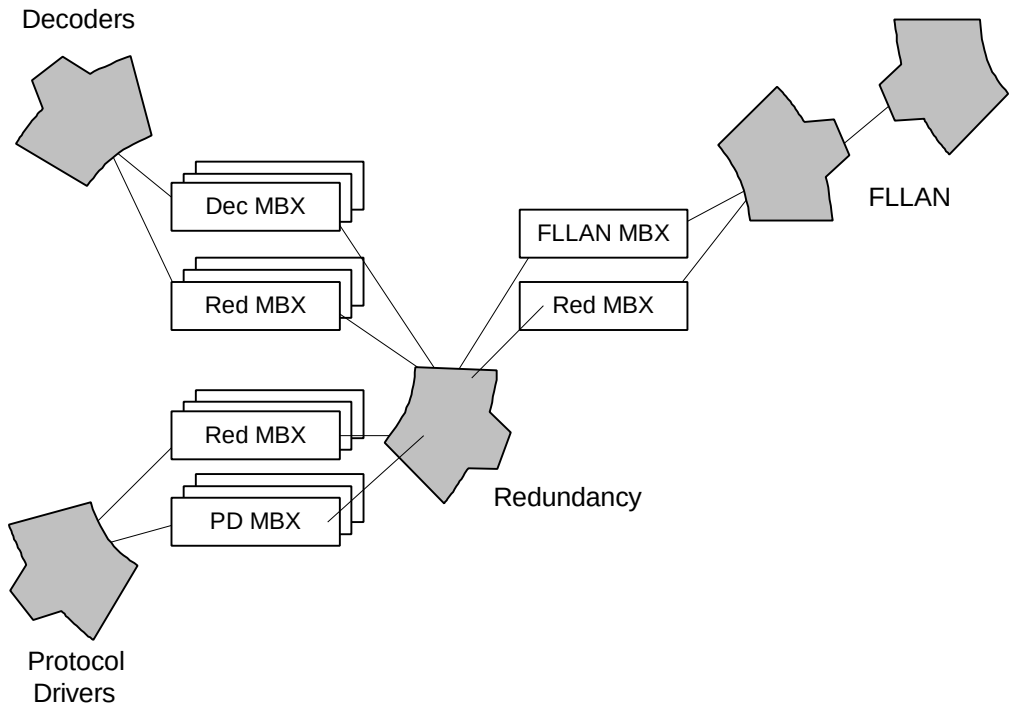


Figure 3.3.4.1 IMX / MCI data set duplication function.

The figure shows the relation of the redundancy module and the other tasks. What actually happens is that all traffic of data sets on the system flow through the redundancy module. The redundancy module is placed in between the decoder and the protocol driver. It decides depending on the mode and the state of the system to which mailbox the data sets will be routed. The right mailboxes at the right places have to be configured to make sure that all data sets flow through the redundancy module,.

In a redundant system with IMX / MCI protocol drivers only the protocol driver on the master system will be used for communication by the redundancy module. Incoming data on the master protocol driver will be duplicated to both systems and outgoing data will always be routed via the master protocol driver. All communication with the slave protocol driver will be discarded by the redundancy module. Both protocol drivers are running at all times, only the slave driver has been blocked. The advantage of both drivers running is that in case of a fault switching communication is very fast.

We will now describe a few routes of data sets in a redundant system. For example a read job will be triggered in the decoder on the master system. In this case the decoder will generate a IMX / MCI read command message which will be sent to the redundancy module via the mailbox. The redundancy module will route the command to the local protocol driver because it is running in master mode.

The local protocol driver will execute the command and receive the data from the external device. It generates a data set with the information and sends this data set to the redundancy module. The redundancy module interprets the received data set and concludes that it has received a response on a read command of the local system. It then will forward the data set to the local decoder from which it originated.

In order to keep the two systems synchronised the data set has also to be sent to the slave system. But the slave system didn't trigger any read action. In this case the redundancy module will mark the data set as unsolicited and route the data set also to the slave system. Both systems have received the same data set and are synchronised.

In the other case the slave system wants to write a block of data to the external device. In this case the decoder on the slave system generates a data set with a block write command and sends it to the redundancy module. The redundancy module then will route the data set via the LAN to the redundancy module on the master system because the local protocol driver has been blocked. The redundancy module on the master system routes the data set to the master protocol driver which executes the command.

The master protocol driver generates an acknowledgement when the write command has been completed. This message will be sent back to the redundancy module. The origination of the write command will be checked which is the slave system in this case. The message will be routed to the redundancy module on the slave system which the in turn sends it to the slave decoder. The write command has been completed.

This means that both machines in a redundant system have full functionality with IMX / MCI based protocol drivers. All communication will be routed through the master protocol driver by the redundancy module.

3.2.4. Mode change actions

The redundancy module is a toolbox for implementing redundancy functionality at the level of application programming. A part of this toolbox is the *mode change action* functionality.

The mode change actions supply the possibility of triggering different actions in the application in case the system changes from mode in case of a fault. These actions can be used for example to disable triggers of certain tables or trigger some actions within an application. The redundancy performs some kind of action on a tag in case of a mode change. This tag can be used in the application for specific defined functionality.

This page is left blank intentionally.

4. Configuring the redundancy module

In the Configuration Manager Main Menu, select **DeltaLink Redundancy Task**. Five tables appear, with the titles of all panels visible for direct access. To access a specific panel position the cursor on a visible area and press the left mouse-button, or use the Next/Prev buttons.

Note: For general information about entering data in FactoryLink configuration tables, refer to Chapter 5 of the FactoryLink Software System User Manual.

4.1. Redundancy Setup Defintion

The redundancy setup definition panel must be used to specify global redundancy information. The panel looks as follows:

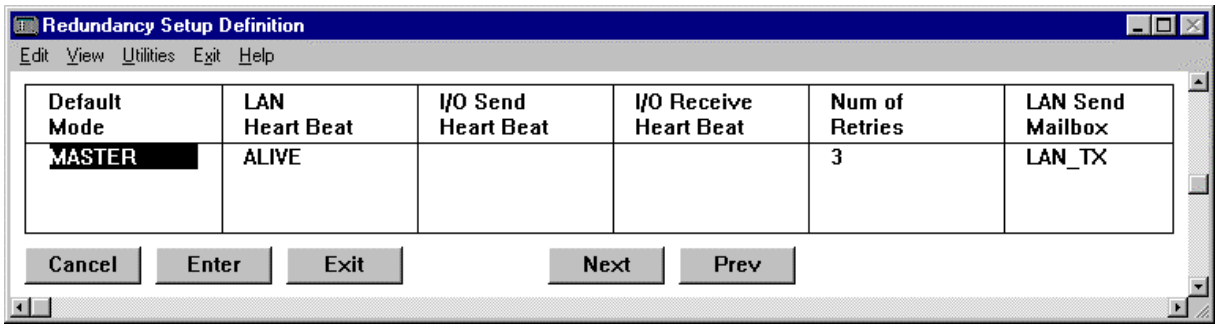


Figure 5.1.1 First part of the redundancy set-up definition panel

- **Default Mode** **Required**

  The default running mode of the system where the application is running. The system will start up in this mode and return to this mode after a fault period.

  valid entry: MASTER or SLAVE.

- **LAN heart beat** **Required**

  This is the LAN Heart Beat trigger which triggers the heart beat functionality in the redundancy module. This tag must be triggered with a constant interval. The length of the interval must be defined by the user. Therefore it is advised to define this tag in the timer module.

  valid entry: TAG of type digital

- **I/O Send Heart Beat** **Optional**

  This field specifies the trigger used with the sending part of the heart beat which travels via the external device. This is the active part of the I/O heart beat and signals the remote system that we are alive. The interval of the I/O heart beat has been coupled to the interval of the LAN heart beat. This tag must be configured in a protocol driver. For configuring the IO heart beat refer to the section *I/O heart beat configuration*.

  valid entry: TAG of type digital

- **I/O Receive Heart Beat** **Optional**

  This field specifies a tag which receives the I/O heart beat of the remote system. This tag must be configured in a protocol driver to receive the heart beat. It will checked at run-time for detecting if the remote system is still alive. For configuring the IO heart beat refer to the section *I/O heart beat configuration.*

  valid entry: TAG of type digital

- **Num of Retries** **Required**

  The number of retries specifies the total number of retries will be made in case no response on a heart beat has been received. When this number has been reached then the state of the system will be changed to the fault state.

  valid entry: decimal number from 1 to 999

- **LAN Send Mailbox** **Required**

  This Mailbox is used for sending data via the FL-LAN module over the LAN to the remote system. This mailbox must match the mailbox as specified in the FL-LAN panels

  valid entry: TAG of type mailbox

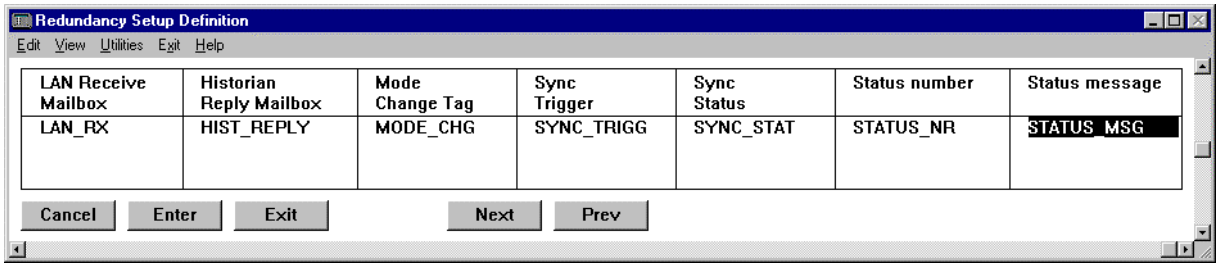


Figure 5.1.2 Second part of the redundancy setup definition panel

- **LAN Receive Mailbox** **Required**

  This mailbox is used for receiving data via the FL-LAN module over the LAN from the remote system. This mailbox must match the mailbox as specified in the FL-LAN panels.

  valid entry: TAG of type mailbox

- **Historian Reply Mailbox** **Optional**

  This mailbox must be specified in case historian tasks are used with the redundancy module. Responses from historians will be replied in this mailbox. Any mailbox can be specified because it is not linked with other panels.

  valid entry: TAG of type mailbox

- **Mode Change Tag** **Optional**

  It is possible to force a mode change on a system while it is running in normal. When this tag is triggered then the redundancy tries to invert the modes on both systems from master to slave and vice versa. Note that this only can be done when running without a fault.

  valid entry: TAG of type digital

- **Sync Trigger** **Optional**

  The synchronisation trigger can be used for starting a synchronisation cycle when running in normal operation. This functionality will be used after a erroneous synchronisation cycle which could not be resolved at the time. As a consequence the databases are not synchronised and a new cycle has to be started.

  valid entry: TAG of type digital

- **Sync Status** **Optional**

  The synchronisation status tag displays the status of the synchronisation cycle. In case this tag holds a value of zero then the two systems are synchronised. In case the value is non-zero then the two system are not synchronised.

  valid entry: TAG of type analog

- **Status Message** **Optional**

  The status message displays the current mode and state of system.

  valid entry: TAG of type message

4.2. Database Synchronisation Definition

The database synchronisation definition panel must be used to specify parameters for the external database synchronisation function. The panel looks as follows:

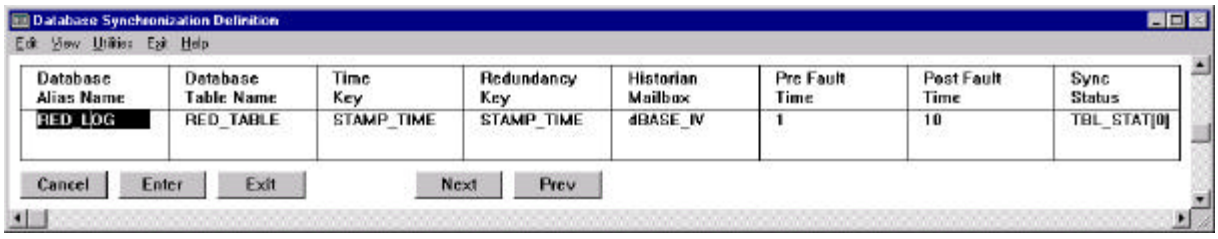


Figure 5.2 Database synchronisation definition panel

- **Database Alias Name** **Required**

  The field specifies the name of the database which must exist.

  valid entry: ASCII name of maximum 16 characters

- **Database Table Name** **Required**

  This field specifies the table name which exist in the database of the previous field.

  valid entry: ASCII name of maximum 16 characters

- **Time Key** **Required**

  The time key will be used as the stamp time for the samples in the external database. This field name must match the record name as defined in the schema of the specific table. In case of synchronisation, this key will be used for selecting data from a database.

  valid entry: ASCII name of maximum 16 characters

- **Redundancy Key** **Required**

  This field can be used to assign a column as the unique data of a sample. Usually this column matches the value as entered in the Index Information panel of the Database Schema Creation table. The key will be used when updating records in a database.

  valid entry: ASCII name of maximum 16 characters

- **Historian Mailbox** **Required**

  The historian mailbox field must be filled in with the mailbox of the specific historian. The redundancy module uses this mailbox for communication with the historian. This field must match the *Historian Mailbox* field in the historian panels.

  valid entry: TAG of type mailbox

- **Pre Fault Time** **Required**

  Before a fault period will be detected by the redundancy module then fault period has already begun. This time represents the fault time before the period has been detected, usually the number of retries multiplied with the heart beat interval. The pre fault time will be used when synchronising the external databases. The start detection time of the fault period will be subtracted with the pre fault time.

  valid entry: decimal number in seconds

- **Post Fault Time** **Required**

  The system needs a period of time to change to the normal running state after a fault period. The post fault time represents this time. The end detection of the fault period will be added with this time when synchronising. The length of the post fault time depends on the application.

  valid entry: decimal number in seconds

- **Sync Status** **Optional**

  The synchronisation status displays the status of this table. In case the status tag holds the value zero then the table is synchronised with its remote counterpart. In case non-zero then the table is not in sync.

  valid entry: TAG of type analog

4.3. Dataset duplication definition

The dataset duplication definition panel must be used in case IMX / MCI based protocol drivers will be used with the redundancy module.

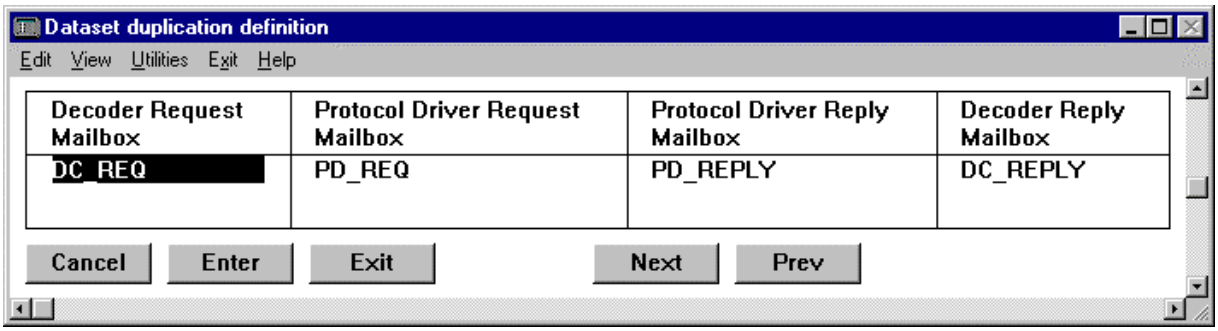


Figure 5.3 Dataset duplication definition panel

- **Decoder Request Mailbox** **Required**

  This mailbox will be used by the decoder task for writing IMX / MCI commands. The mailbox configured here must be the same as the mailbox configured in the *DeltaLink Protocol Driver Definition* field in the decoder task.

  valid entry: TAG of type mailbox

- **Protocol Driver Request Mailbox** **Required**

  This mailbox will be used by the redundancy module for writing requests to a specific protocol driver. The mailbox configured here must be the same as the mailbox configured in the *Protocol Driver Mailbox Tag* field in the specific protocol driver.

  valid entry: TAG of type mailbox

- **Protocol Driver Reply Mailbox** **Required**

  This mailbox will be used by the protocol driver to return responses of executed command to the redundancy module. The mailbox specified in this field must match the value in the *Decoder Mailbox Tag* fields in the panels of the specific protocol driver.

  valid entry:

- **Decoder Reply Mailbox** **Required**

  This field will be used by the redundancy module for routing responses of the protocol driver to the decoder task. The mailbox specified in this field must match the mailbox configured in the *DeltaLink Decoder Mailbox Definition* panel in the decoder task.

  valid entry: TAG of type mailbox

4.4. Mode Change Actions Definition

The mode change action definition panel must be used when actions have to be defined in case the system changes from mode, for example if the mode changes from master to slave.

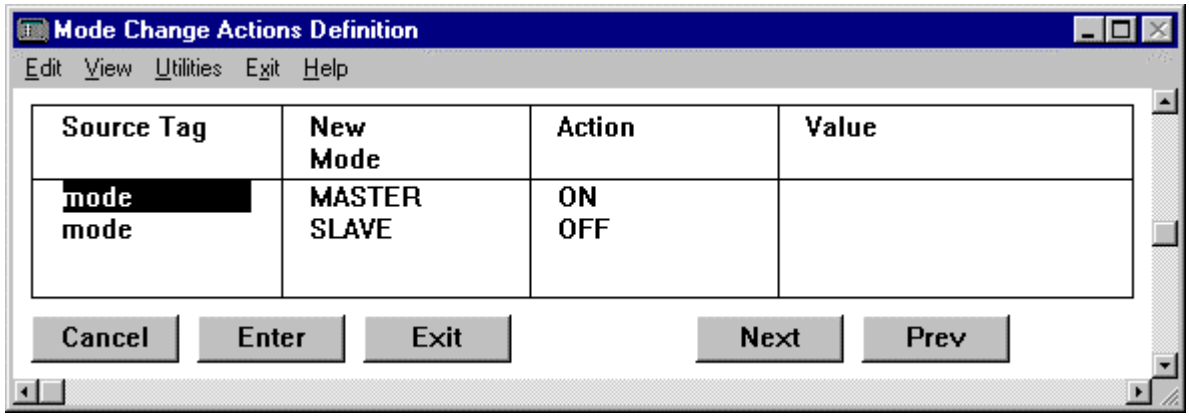


Figure 5.4 Mode change actions definition panel

- **Source Tag** **Required**

  The source is a user tag on which the action will be performed in case of a mode change.

  valid entry: TAG of any type except mailbox

- **New Mode** **Required**

  The new mode specifies the mode transition for the action to be taken. For example if the new mode is master then the action will be performed when the system changes from slave to master.

  valid entry: MASTER or SLAVE

- **Action** **Required**

  The action specifies what kind of action will be performed in case of a mode change.

  valid entry:
  NULL
  OFF
  ON
  TGL
  SET
  ADD
  SUB

- **Value** **Optional**

  The value will be used in conjunction with the action field. For example the SET action uses the value for setting the tag. Not all actions use the value parameter.

  valid entry: character string of maximum 80 characters

4.5. Tag Copy Definition

The tag copy function can be used for more than purpose but the most important is the synchronisation of the real-time databases of FactoryLink on both systems.

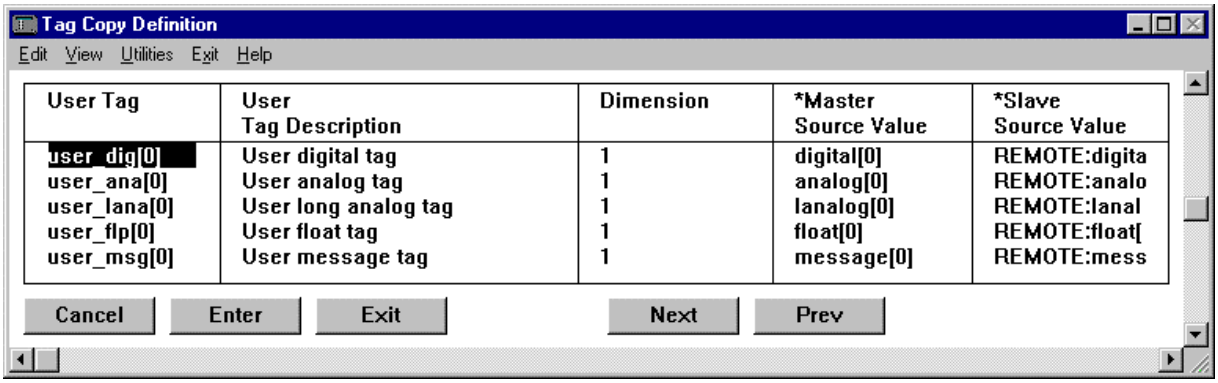


Figure 5.5 Tag copy definition panel

- **User Tag** **Required**

  The user tag must be used in the application and is always the source or destination of the other two tags depending on the mode of the system. For example when the system is running in master mode then the tag copy function will copy between the user tag and the master tag.

  valid entry: TAG of any type

- **User Tag Description** **Output**

  This field displays the description as defined with the specific tag

  valid entry:

- **Dimension** **Required**

  Defines the number of elements in case an array has been used. Note that the user, master and slave tags must have the dimension as specified in this field.

  valid entry: TAG of any type

- **Master** **Required**

  The master tag will be exchanged with the user tag in case running in master mode. The master tag can be a tag or a constant. In case the value is a constant then this value will be copied once at initialisation and in case of a mode change.

  valid entry: TAG of any type or a constant value

- **Slave** **Required**

  The slave tag will be exchanged with the user tag in case running in slave mode. The slave tag can be a tag or a constant. In case the value is a constant then this value will be copied once at initialisation and in case of a mode change.

  valid entry: TAG of any type or a constant value

5. Application considerations.

The redundancy module is a tool kit for implementing redundancy functionality in a FactoryLink application. It depends on several other tasks for correct operation. This section discusses the configuration of these tasks together with the application for an optimal result.

The file *master.mps* has been installed in the {FLINK}/mps directory with the installation of the redundancy module. This is a multi-platform save file of an example application which can be restored. This file contains the master application of the demo. In case a test environment has to be tested then this application can also be restored on the slave system. After restoring on the slave system minor changes have to be made which are described in the application duplication section.

5.1. Application notes

The application note specifies which configurations are necessary with the redundancy module.

5.1.1. System Configuration

In the *system configuration* panel of the configuration manager all modules are defined which will run in the application. It is possible to pass program arguments to each module by defining these arguments at the *program arguments* field in the entry of the module.

In case protocol drivers are used with the redundancy module then some parameters have to be passed to the redundancy module, the decoder and the protocol. These parameters are the local and remote station identification.

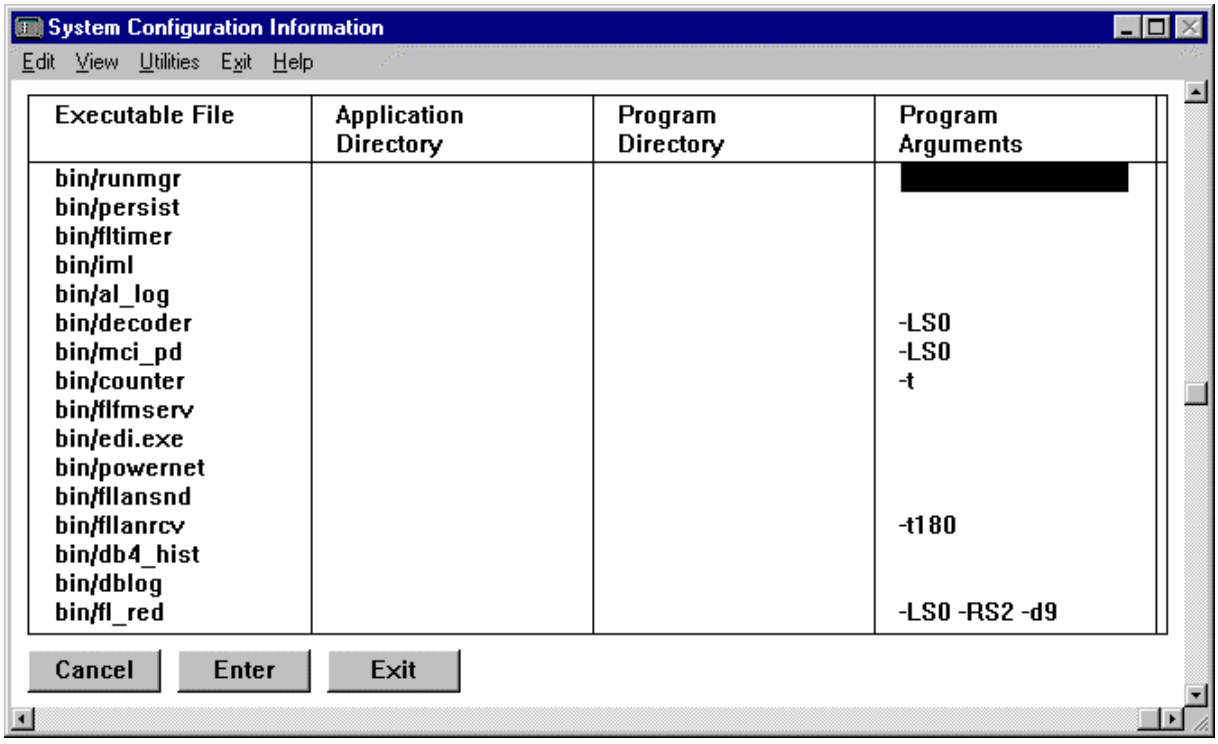


Figure 6.1.1.1 Local and remote station id numbers.

The figure shows which parameters have to be specified. The parameter for the local station id is **-LS<number>** and for the remote station id **-RS<number>**. The station id of a system must be unique on the network and can be freely choosen by the application programmer.

The redundancy module needs both the local and the remote station id. In the example the local station id is '0' and the remote station id is '2' so the parameters will *-LS0* and *-RS2*. Both, the decoder and protocol driver need only the local station id so only the *-LS0* will be specified with these tasks. The same parameters have to be specified on the remote system only the local and remote id numbers are swapped.

Note that the specific protocol driver must be IMX / MCI based and be able to accept the local station id parameter. To be sure contact your protocol driver supplier.

Also the '-S' parameter can be passed as a program argument. When this argument has been configured then the synchronization cycle will be skipped.

5.1.2. PowerNet

The PowerNet module is used for real-time database synchronisation in conjunction with the tag copy function of the redundancy module. The configuration of the PowerNet module must be actually done at different places. The basic configuration exists in the *External Domain Definition* and all tags that are used with PowerNet can be specified freely throughout the application.

For the redundancy module this means that a domain has to be specified in the *External Domain Defintion* for accessing tags on the remote system. Normally this will be a domain with the *Update Type* on *Exception* and the *Data Transfer* on *ReadWrite*. It is possible to define more than domain and use these domains in the redundancy module. The following picture shows an example of the basic PowerNet configuartion.

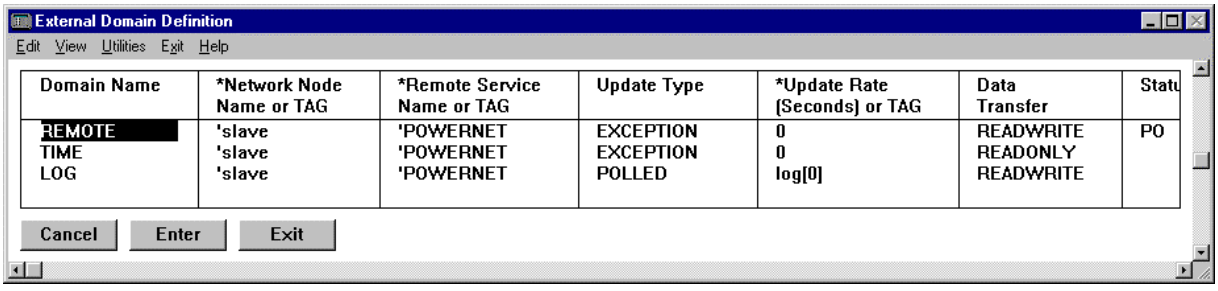


Figure 6.1.2.1 Basic PowerNet configuration.

The second part is to define the tags which have to be synchronised by the tag copy function in the panel of the redundancy module. The section *Tag Copy Definition* in the previous section shows an example of how to configure PowerNet tags for use with the tag copy function. What basically happens is that the local tag will be configured in the master field and the same tag on the remote system with the right domain in the slave tag field.

5.1.3 FL-LAN

The redundancy module makes use of mailboxes for transferring messages over the LAN. At the moment the PowerNet module does not support the use of mailboxes so the FL-LAN module will be used for this purpose. Two mailboxes have to be defined: a FL-LAN send and receive mailbox. These mailboxes must be configured in Redundancy Setup Definition.

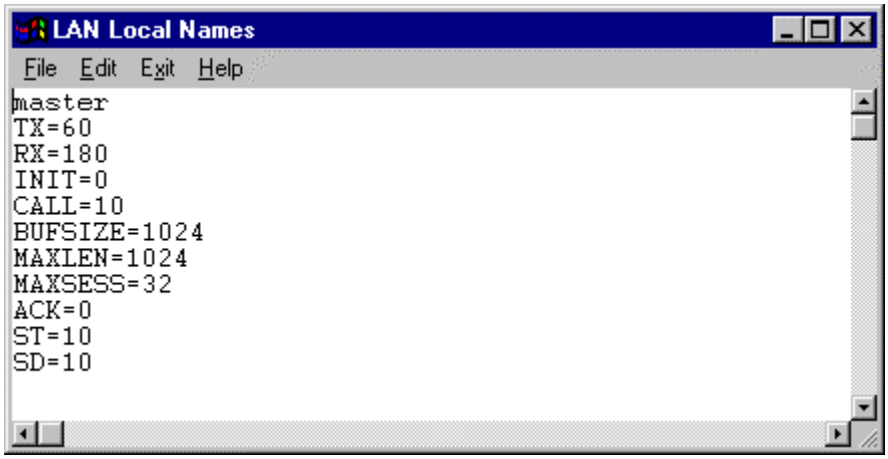


Figure 6.1.3.1 LAN Local Names.

It is advisable to configure all parameters in the LAN Local Names panel of the FL-LAN module. The picture above shows an example of this. In this case also a parameters has to be supplied for the *fllanrcv* module, the receiving module of the FL-LAN. The parameter, as shown in figure 6.1.1.1 is *-t180*, the same as the *RX* entry. Note that these values can be adjusted to the needs of the application and are described in the FactoryLink manual at the FL-LAN section.

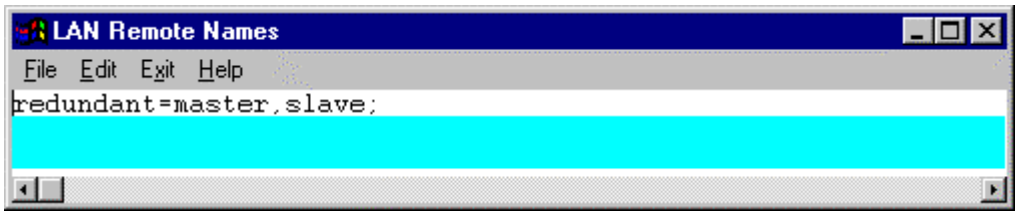


Figure 6.1.3.2 LAN Remote Names.

The next step is to define a group of nodes in the FL-LAN. In this case we have defined two node with the names *master* and *slaves* which are members of the group *redundant*. This group will be used in the other panels of the FL-LAN.

The other panels may look like the examples below. Changes in the send mailbox will be sent on exception which means that messages will be sent directly. The send mailbox will be coupled to the remote receive mailbox by configuring the network alias name.

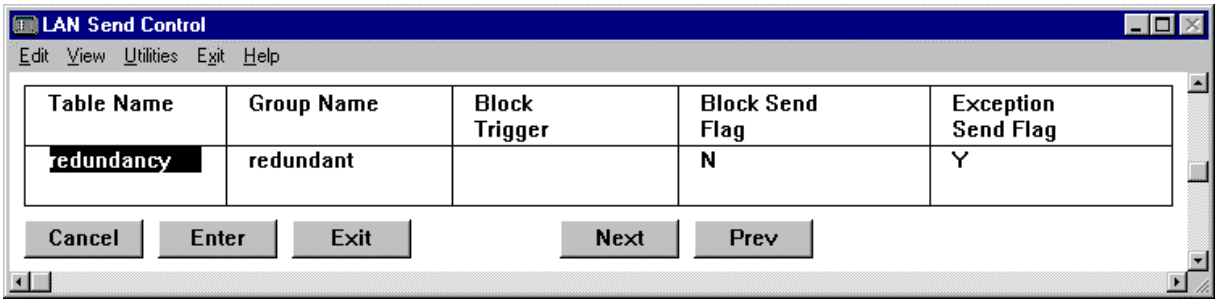


Figure 6.1.3.3 LAN Send Control.

The network alias field must contain the same name as the receive mailbox defined in the *LAN Receive Panel*.

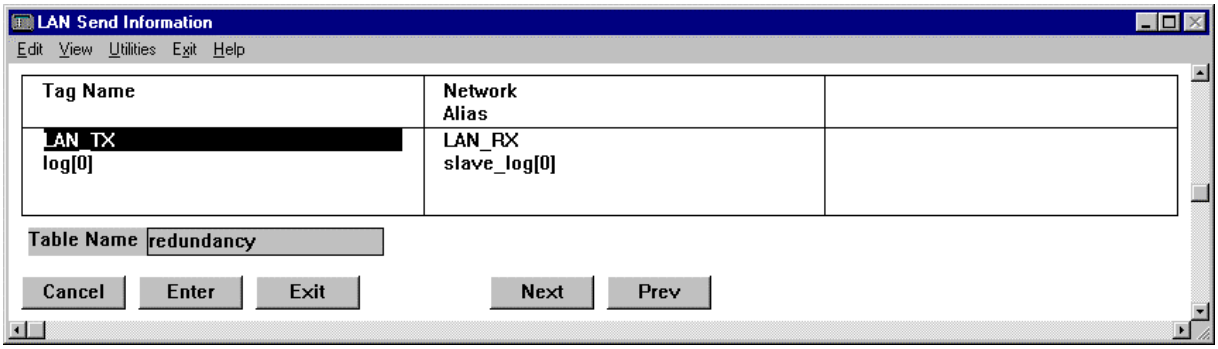


Figure 6.1.3.4 LAN Send Information.

The panels for receiving messages may look like the following panels.

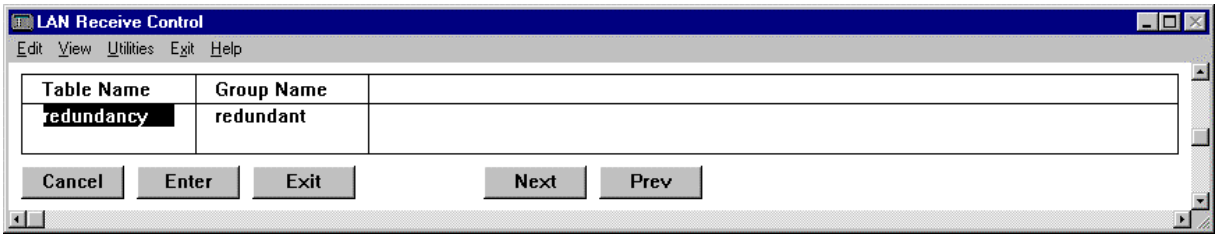


Figure 6.1.3.5 LAN Receive Control.

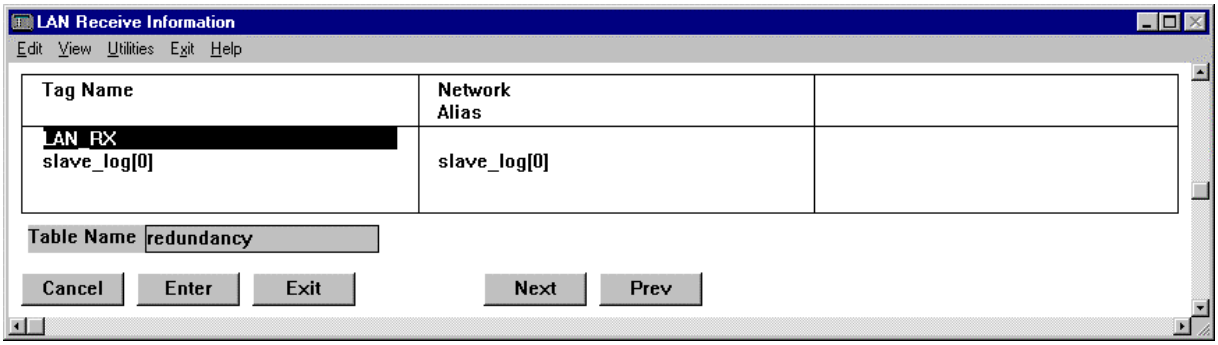


Figure 6.1.3.6 LAN Receive Information.

5.1.3. Historian tasks

No special configurations have to be made for using historians with the redundancy module. The historians have to be configured normally as described in the FactoryLink manuals. The only necessity exists in creating a schema for use with the redundancy module. This schema must always contain a column which logs the stamp time of the samples in that table. This column must be specified in the *Time Key* field of the *Database Synchronisation Definition* panel of the redundancy module.

The time key is very important when synchronising database after a fault period. The redundancy module assumes that the two systems log the same stamp time for every sample. This way the two external databases can be synchronised with the same samples after a fault period.

5.1.4. IMX / MCI based protocol drivers

It is very efficient to use IMX / MCI based protocol drivers for use with the redundancy module. To use the redundancy module then all data sets have to be routed through this module. This needs special configuration for the mailboxes used with the decoder and protocol drivers.

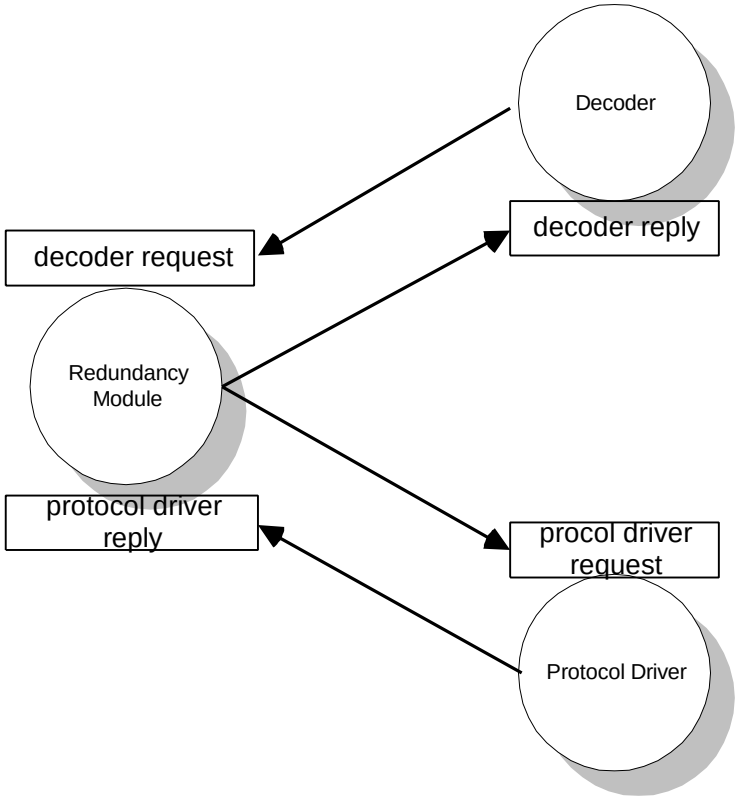


Figure 6.1.5.1 IMX / MCI mailbox relations.

The picture shows the relation between mailboxes of the three tasks. The names of the mailboxes are the same as the fields in the *Dataset Duplication Definition* panel in the redundancy module. The position of the mailboxes specify in which task the mailbox has to be defined. For example the decoder *reply mailbox* is the mailbox of the decoder task and the *protocol driver request* mailbox is the mailbox of the specific protocol driver. Thus the *decoder reply mailbox* has to be defined in the *DeltaLink Decoder Mailbox Definition* panel of the decoder in case the DeltaLink decoder module has been used. The redundancy module defines two additional mailboxes, the decoder request and the protocol driver reply mailbox, through which all messages are routed.

5.1.5. Task start up

It is very important to configure the sequence of start up in case IMX tasks are used. The redundancy module needs information from the ioxlator and the protocol driver at start up. Therefore the redundancy module must be the first task to be started. This can be manipulated through the *Start Order* column in the configuration manager. The number in this column must be lower than the number for the ioxlator and protocol driver.

For the tag copy functionality another start-up order must be configured. The tag copy functionaity uses the standard FactoryLink Powernet module for copying tags to the other machines. At start up Powernet synchronises all tags on both machines. This action must be completed before the redundancy task will be started. This means that the redundancy module must be started after Powernet has synchronised all tags. This can be done within the math&logic module. One disadvantage is that it is not possible to determine when the Powernet module has finished the synchronization. The delay before starting the redundancy module must be determined from experience.

5.2. Time synchronisation

The local time on systems is a very important item in the redundant systems. For a good operation the time must be synchronised between the two systems. If the time has been synchronised then identical samples with identical time stamps will be logged. Identical databases will be generated on both systems at run-time. After a fault period only the samples which are missing on one system are updated on the other system.

In case the time of two systems are not in sync then two databases will exist which contain different samples. After a fault period this may result in doubling the amount of samples in the databases because they contain samples which do not exist on both databases.

Therefore it is important to synchronise the system time on both systems. The best way to do this is to utilise the possibilities of the operating system or from third party developers. For example, for most platforms special time cards exist for synchronising the system time. Triggers for logging together with stamp times are defined locally in this case.

An alternative could be to synchronise the system time via the redundancy module on the two systems. However this may be very inaccurate because the time has to be transferred via the PowerNet module over the LAN. The start and the length of this transfer period is not determined.

Another possibility can be to use stamp times which are generated in external devices. For example, in case the DeltaLink High Speed Logger module has been used then it is possible to send a burst of samples together with time stamps to the FactoryLink application. These samples are bundled in a dataset and will be duplicated by the redundancy module to both systems. The databases will log exactly the same information because only one stamp time per sample has been generated in the external device.

5.3. Application duplication

The easiest way to create a redundant system with the redundancy module is to develop a redundant aware application and duplicate this application. However only copying the application from one system to another is not enough. Some minor changes have to be made which will be discussed in this section.

5.3.1. Redundancy module

The only field that has to be changed in the redundancy module id the *Default Mode* field in the Redundancy Setup Definition panel. This field must be changed to the default mode of the system which can be master or slave. All other panels stay the same.

5.3.2. PowerNet module

The next module to change is the PowerNet module. In the External Domain Definition panel the field *Network Node Name* has to be changed to the node name of the remote system. In the example the two nodes are called *master* and *slave* so when duplicating this field has to be changed to master.

All other tags used with PowerNet, such as the tags specified in the tag copy panels of the redundancy modules need not to be changed because the domain name stays the same.

5.3.3. FL-LAN module

In the FL-LAN module the next table has to be changed: the *LAN Local Names* tables of the *Local Area Network Groups* entry. This table contains the local name of the system for FL-LAN on the first line of the table. This name must be the name of the local system as described in the next panel: the definition of the group names In the example the *master* name has to be changed to the *slave* name.

All other panels in the FL-LAN module stay the same. In case the *Network Monitoring* panel also has been configured the *Station Name* has to be changed to the local station name.

5.4. IO Heart beat configuration

Some special configurations have to be made for proper operation of the I/O heart beat. The I/O heart beat is an alternative way to detect if both systems, master and slave, are still running in case the normal heart beat is not working. The I/O heart beat makes use of I/O points in the external device which means that it uses a protocol driver. Data from the process which must be redundant also uses a protocol driver and is routed through the redundancy module on one system: the master system.

The I/O heart beat can only work properly in case the I/O send and receive tag are not routed via the redundancy module. This means that these I/O points are being sent and received through the local protocol driver. The local protocol driver must always be running. The configuration can be done as follows.

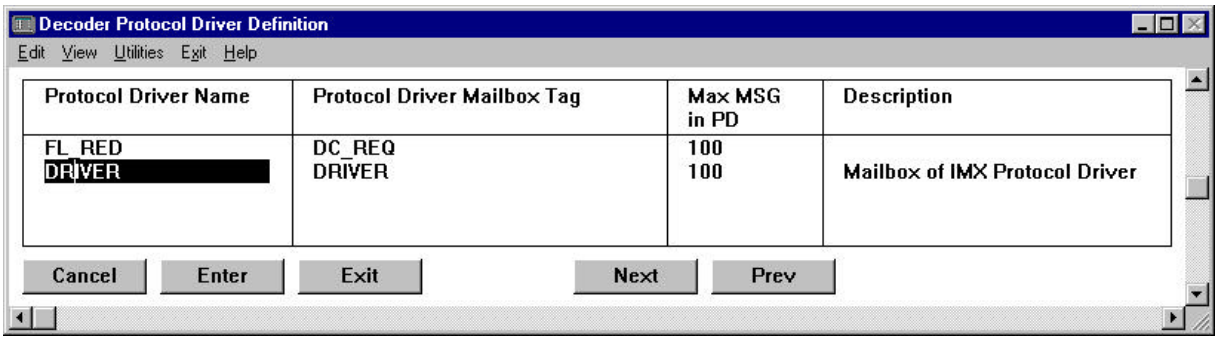


Figure 6.4.1 IO heart beat configuartion.

An additional entry in the *Decoder Protocol Driver Definition* must be made which points directly to the protocol drivers mailbox and therby bypasses the redundancy module. Underneath this entry a dataset must be defined which contains the I/O send and receive tags.

The I/O points will always be read and written through the local protocol driver by bypassing the redundancy module, even if the ethernet connection is broken.

This page is left blank intentionally.

Appendix A. The delta.opt file

In order to run the redundancy module permanently with full functionality an authorisation sequence code is needed. This appendix describes the authorisation sequence code together with the option file.

The authorisation sequence code of every independent DeltaLink module must reside in the *{FLOPT}/delta.opt* file in order to take effect. In case the redundancy module has been ordered with an authorisation sequence code then this code resides on the install diskette (in the *opt\delta.opt* file) and will be automatically added to the *{FLOPT}/opt file* with the installation. In case the redundancy module has been previously installed from a demo diskette and the authorisation sequence code has been purchased later then the code must be added manually to the *{FLOPT}\delta.opt* file.

The format of the delta.opt file consists of two parts. The first part is the comment header. This part remains always at the beginning of the file. Every line of the comment part starts with an '*' character. The second part contains the authorisation sequence codes of every independent DeltaLink module. Every line must hold one sequence code and must apply to a strict format.

An example of the option file looks like this:

Protection file: *delta.opt*

```
*
* Copyright 1994 DeltaLink bv. All Rights Reserved
*
* DeltaLink bv
* Pioenroostraat 26
* NL-5241 AB Rosmalen
* The Netherlands
* Tel: (int) 31 073 523 1234
* Fax: (int) 31 073 523 1222
*
*
* FactoryLink Serial Number: 12345OS2
*
* DeltaLink module option:
*
* MODULE = task name of always 8 characters with trailing spaces
* .      = <space>
* X      = authorisation code supplied by DeltaLink
*
* MODULE..XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX
FL_RED    1D63 F829 F51B 5269 1E1A 1588 0F36 2E3F 5E62
```

In case a full installation diskette with authorisation code has been ordered then the header in the option file on the diskette contains the serial number of FactoryLink system. The redundancy module will run only on the FactoryLink system with this serial number. If the serial number is not listed in the *{FLOPT}\delta.opt* file due to a previous demo installation then this number can be added in the header of this file.

The authorisation code must exactly match the format as listed in the header. If this is not the case the module will not recognise the authorisation sequence code and start up in demo mode.

The format of an authorisation sequence line is as follows:

MODULE<s><s>XXXX<s>XXXX<s>XXXX<s>XXXX<s>XXXX<s>XXXX<s>XXXX<s>XXXX<s>XXXX<CR><LF>

The **MODULE** field contains the DeltaLink module name in this case FL_RED. This field must always be 8 characters long. If the module name is shorter then 8 characters then the name must be filled out with spaces to 8 characters.

After the MODULE field one space must be entered;

After the space field 9 records must be specified with the authorisation code. One record is build up of one leading space (ASCII 0x20) and four sequence codes. The sequence codes must be entered exactly as specified by DeltaLink.

After the authorisation code records a carriage return (ASCII 0x13) and linefeed (ASCII 0x10) must follow.

There may be no empty lines between the specification of more than one module. A normal editor can be used to add an authorisation sequence code. If all modules are specified with the right authorisation codes according to the format described above then the modules will start with full functionality enabled.

Appendix B. Command line parameters

The Redundancy module accepts several command line parameters, these can be configured with the configuration manager in the 'System Configuration' table, column 'Program Arguments'. An argument consists out of first a minus sign ('-'), followed by the a letter specifying the option. After the letter an optional number can be present, if this is supported by the option.

Option	Description
-d*n*	Debug option, the level of debug information is set with the number *n*. The range of this number is from 1 until 4. If no number is specified the default level will be 1. The debug output will be visible in the 'window' of the protocol driver, that of the run time manager.
-l*n*	Same as the previous option, difference is the output device. For this option an ASCII log file is generated, being the file: {FLAPP}/{FLNAME}/{FLDOMAIN}/log/fl_red.log
-LS*n*	Local Station id with number *n*, needed to identify the different nodes (or FactoryLink workstations). This must be an unique (station) number for the network.
-RS*n*	Remote Station id with number *n*, needed to identify the different nodes (or FactoryLink workstations). This must be an unique (station) number for the network.
-S	Using this argument will skip the synchronisation cycle. To skip the synchronisation cycle this argument must be present on at least the master or slave system.

Appendix C. Error codes

The error code is returned to the user in a user-defined status tag. These error codes will also be printed with the message of the DeltaLink Redundancy task in the run-time manager. The errors can be generated from within different parts of the task which will be listed here:

Error #	General errors
0x101	dataset not yet queried on this system

Error #	FactoryLink errors
401	Internal error
402	Out of memory
403	Operating system error
404	Initialisation not successful
405	Initialisation not successful
406	Incorrect function
407	Incorrect argument
408	Incorrect data
409	Bad tag
410	Null pointer assignment
411	Change flag not set
412	Procedure table full
413	Bad procedure name
414	Bad user name
415	Bad option
416	Incorrect checksum
417	No options
418	No key
419	Bad key
420	No port available
421	Port busy
422	FL already active
423	No lock
424	Lock failed
425	Lock expired
426	Wait failed
427	Termination flag set
428	Q-size to big
429	Q-size changed
430	No tag list
431	Tag list changed
432	Wakeup failed
433	No signals
434	Signalled
435	Not a mailbox
436	No messages
437	Access denied
438	Attribute failure
439	Invalid attribute
440	Attribute not defined
441	Application exists
442	RTDB does not exist
443	No task bit
444	Not a lite task

Appendix D. Error messages

If an error condition occurs in the decoder task during run-time mode, a message to that effect will appear on the runtime manager graphics screen to the right of "FL_RED". The error messages that may be displayed are as follows:

Running in DEMO mode

The redundancy has been started in demo mode because no valid authorisation code has been found. Check the {FLOPT}/delta.opt file.

Demo shutdown, licensed to run -+ 5 hours

The module has been running in demo mode. This means that 5 hours of consecutive run-time is supplied. The application has to be restarted again for running the module. Check the {FLOPT}/delta.opt file for a valid authorisation code.

Demo restart prohibited, restart FactoryLink

The module has been running in demo mode and cannot be restarted after the total application has been restarted. Check the {FLOPT}/delta.opt file for a valid authorisation code.

DeltaLink protection key missing

The DeltaLink authorisation code has not been installed for this module. Check the {FLOPT}/delta.opt file for the sequence code.

Can't open CT file

The module was unable to open the configuration table file, generally because it does not exist. Check first the {FLINK}/ctgen/ctlist file for the entry of the redundancy module. When the entry exists try to run '*ctgen -v fl_red.ctg*' for creating a new configuration table file and try to start the application again.

Out of RAM

There is not enough RAM memory to load the complete configuration and/or task.

Error reading CT index

An error occurred during reading a Configuration Table index, normally this means the CT-file is corrupted. This is a fatal error. Try to generate new configuration table files using the *ctgen* module.

Error reading CT header

An error occurred during reading a Configuration Table header, normally this means the CT-file is corrupted. This is a fatal error. Try to generate new configuration table files using the *ctgen* module.

Error reading CT record

An error occurred during reading a Configuration Table record, normally this means the CT-file is corrupted. This is a fatal error. Try to generate new configuration table files using the *ctgen* module.

Invalid tag number

The module encountered an invalid TAG number. This is a fatal error. Make a platform restore and a platform save of the application in order to clean up the real time database.

No definitions defined !

The DeltaLink Redundancy Setup definition panel has not been specified totally. Check this panel.

Only first definition processed !

More than one entry has been made in a panel which only accepts one definition. The first entry will be processed by the module.

Error retrieving sectime tag

The global tag sectime could not be retrieved from the real time database object list. Check the application on validity.

Waiting on the timer task to start

The module is waiting for the timer task to start. If this message doen't dissappear the check if the timer task is running.

Error waiting on the sectime tag %d

The mdoule encountered an error while waiting for the timer task to start up. Check the validity of the application.

Local station ID program argument missing (-L)

The local station id program argument has not be specified in the *System Configuration* panel in the Configuration Manager. Define a unique local and remote station id and specify at the program arguments with the -L<id> option.

Remote station ID program argument missing (-R)

The local station id program argument has not be specified in the *System Configuration* panel in the Configuration Manager. Define a unique local and remote station id and specify at the program arguments with the -R<id> option.

FL reading data error %d

FactoryLink error reading the real-time database. The error number is specified.

FL writing data error %d

FactoryLink error writing the real-time database. The error number is specified.

FL forced write error %d

FactoryLink error forced writing the real-time database. The error number is specified.

FL change read error %d

FactoryLink error reading the real-time database on change. The error number is specified.

FL set change bit error %d

FactoryLink error setting change bits. The error number is specified.

FL reset change bit error %d

FactoryLink error resetting change bits. The error number is specified.

FL clear change error %d

FactoryLink error clearing change bits. The error number is specified.

FL reading data in %s MBX error %d

Error reading data from the specified mailbox. The error code has been specified. Try to generate new configuration table file with the *ctgen* utility.

FL writing data in %s MBX error %d

Error writing data to the specified mailbox. The error code has been specified. Try to generate new configuration table file with the *ctgen* utility.

FL add event to event list %d

The module is not possible to add the event to the internal event list. Contact DeltaLink when this error occurs.

FL clear events from event list %d

The module was not able to clear an event from the event list. Contact DeltaLink when this error occurs.

FL getting number of msg in queue %d

Error retrieving the total number of messages in the queue. Try to make a multi platform save and restore for cleaning up the real time database.

FL querying LAN rx queue %d

Error querying the FL-LAN receive mailbox. Try to make a multi platform save and restore for cleaning up the real time database.

MCI init %d

MCI initialisation error, internal error of the module.

MCI send pd %d error %d

MCI error, internal error of the module. The record number of the protocol driver and the error number are specified.

MCI check number message %s %d

MCI error, internal error of the module. The protocol driver and the error number are specified.

MCI maximum msg in PD MBX %d reached

MCI error, the maximum number of allowed messages in a protocol driver mailbox tag is reached. The record number of the protocol driver mailbox is specified.

MCI unknown boundary %d

MCI error, internal error of the module, the protocol driver specified an unknown boundary. The boundary is specified.

MCI error packet %s: nr %d

MCI error, internal error of the module. The functionality and the error number are specified.

MCI index identification not supported %d

MCI error, protocol driver tried to identify a dataset by a number. This is not supported in this version of the module. The dataset number is specified.

MCI unknown message received %d

MCI error, internal error of the module. The number of an unrecognised command is specified.

MCI error response %s error %d

Communication error, the module received an error response for a read/write command. Specified are the functionality and the error number. The error number is received from a protocol driver, the meaning of the error depends on the driver (see the appropriate manual).

MCI unknown query response received %s

The query response received is not known to the system. Contact DeltaLink when this error happens.

MCI unknown type received %d

Unknown type of IMX / MCI message received. Contact DeltaLink when this error happens.

MCI dataset not queried yet !

Communication not yet possible via this route because the dataset has not been queried yet. Check if all decoders and protocol drivers are running on both systems.

HST Error describing table %s

The table specified could not be decribed by the redundancy module. Check the validity of the database table.

HST number of columns table %s don't match

The total number of columns of the database table on the two machines (local and remote) are not the same. For synchronizing the systems, two exactly the same tables must exist.

HST column names %s and %s differ !

The names of the columns of the two database tables (local and remote) are not the same. For synchronizing the systems, two exactly the same tables must exist.

HST column types %s differ !

The column types of the two database tables (local and remote) do not match. For synchronizing the systems, two exactly the same tables must exist.

HST manual cycle can only be started in master mode

The manual cycle for synchronizing all external databases must be started in master mode and when in the running state. Check the the mode and state of the system.

HST error reading the logged fault period

Data could not be retrieved from the database of the fault period. Check the database table for consistency.

HST error writing the current fault period

An error occured while trying to write the data of the fault period to the database table. Check the database table on consistency.

HST no fault period exists on this system

There is no fault period registered on this machine. This is not an error but a warning.

Unknown action specified at tag %d:%d

Unknown type of action specified at the listed tag. The tag segment:offset is listed which can be searched on in the object list in the configiration manager. Check the application on consistency or try to run the *ctgen* utililty for generating new CT files for the redundancy module.

Warning: the remote system should run in %d mode

The remote system should run in the opposite mode of the local system which is not the case at the moment. This is only a warning which will be automatically resolved.

Currently running in %s mode and %s

This message specifies the current running mode and state of the redundancy module.

Redundancy module normally stopped

The redundancy module has been shut down without error.

This page is left blank intentionally.