An increasing number of companies start to build up FlexRay clusters in order to get experience with FlexRay. The biggest barrier for successfully bringing up a FlexRay cluster is the startup phase. This paper should serve as a cookbook to ease this as much as possible. A system step by step approach is chosen which has proven to be efficient.

Starting up FlexRay

Some Advice from Practical Experience

With the publication of the FlexRay specifications version 2.1 another important milestone in the development and introduction of the communication protocol was reached. However hardware and software experience and basic knowledge of the FlexRay protocol is anticipated.

Introduction

A FlexRay cluster consists of a set of spatially distributed nodes connected by a communication network that consists of either a passive bus or an active star [1]. Each node contains a host processor and a communication controller (CC) that interfaces the host processor to the communication network through a physical bus driver. The interface between the host processor and the CC is referred to as the CHI. Each CC within the FlexRay cluster executes the FlexRay protocol.

The overall FlexRay protocol state machine is shown in **Figure 1** and defines the main steps of every controller implementation how a startup works [2]. Mutual message exchange between nodes is only possible in the normal active state (6). To reach this state the state sequence 1 – 2 – 3 – 5 – 6 must be passed through.

FlexRay controller states:
1 – default config
2 – config
3 – ready
4 – wakeup
5 – startup
6 – normal active
7 – normal passive
8 – halt

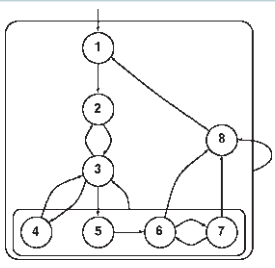

Figure 1: Overview protocol states

© automotive

After powering up the CC enters the state *default config* (1) in which the CC waits for the host command to enter the config state. In the *config state* (2) the host is allowed to write the configuration registers which are read only in all other states. Is the configuration completed the host can set the CC in the *ready state* (3). In the MFR4200 the config and the ready state are seen as only one state to the outside world. From the ready state the CC may enter the *startup state* (5) after the related host command. The startup state presents a super state with a number of substa-

tes. After a successful startup which is done by the CC itself without further interaction by the host the normal *passive state* (6) is entered and communication takes place. A precondition for a successful startup is a working configuration in all nodes of the cluster. A number of configuration pitfalls are given in this article which may prevent a cluster startup.

Step by step approach to bring up a FlexRay cluster

One of the key advantages of FlexRay, the mutual checks of the connected controllers, are at the same time the source for many problems. A network will not start up until the key parameters as schedule, cycle configurations, CRC values and slot booking are appropriately set. This encompasses a large set of parameters making it hard to identify a wrongly chosen parameter. The following explanation shall give a guideline how to systematically debug and startup a network. The step by step approach breaks down the entire set of parameters into manageable portions.

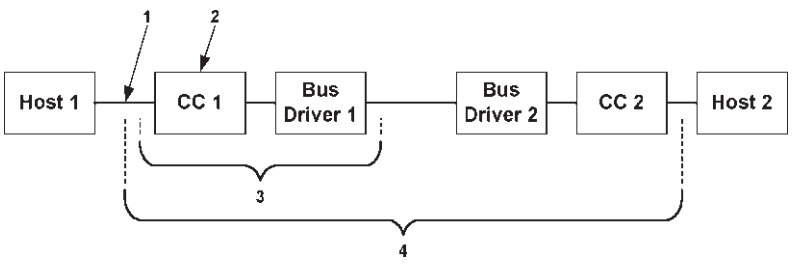

Figure 2: Two node FlexRay system and test steps.

© automotive

Test partitioning

A partitioning into five steps has proven to be efficient for starting a cluster. The last step requires a running FlexRay system. The scope of the steps are illustrated in **Figure 2**.

host – CC connection
Single CC configuration and startup
Single CC and physical layer
Networking with second node
Data exchange between host 1 and host 2

Checking host – CC connection

The base for all successive steps is the correct hook up of the FlexRay controller to the host. In case of the MFR 4200 this connection can be tested in two flavors:

- Reading the fixed value of the "magic number register" checks the correct addressing and data bus connection to the device.
- Writing to any write address with a successive read checks the correct write access. In case the device is not behaving as expected the independence of bus addresses should be checked. This can be done by writing multiple addresses at once and reading them back.

Checks on a single CC

For this step only a stand-alone controller is used. It is tested that the CC has received a valid configuration and is able to transmit. Doing so the controller discloses a specific footprint of the configuration at its TX pin, that can be evaluated. Intentionally this test step is limited to the TX pin of the controller and almost all buffer related settings do not matter. Only the startup/sync frame slot must be configured. This test step consists of two portions:

- *Download of the configuration data. It is absolutely required that the node is configured to be the leading coldstarter.*
- *Release the configuration step. The node will start to coldstart the network.*

During the coldstart attempts it can be check the correctness of cycle length, bus speed, sync frame position in the cycle and the number of coldstart attempts. By using debug or strobe signals on the MFR 4200 the user is able to measure macrotick and slot length.

Checks on a single CC and physical layer

The intention of this step is to validate the correct connection between the CC and the physical layer. In this step the same validation as in the previous step is performed on the physical bus. For this test care must be taken that the bus is correctly terminated.

Separate check of second node

This is an intermediate step checking that the other node gets the same configuration with a different sync slot booking.

Establish network connection

The ultimate goal for this check is to bring up an initial network communication but data exchange and buffer configuration are not checked. In this test are checked:

- Bus driver – CC RX connection
- Header CRC
- basic checks of the clock sync parameters

Hints:

The header CRC is only checked by the receiving node. It will discard a frame with a wrong CRC and will not integrate to the leading coldstarter.

Too tight clock sync parameters (namely max offset and rate correction) may prevent the receiving node to integrate to the imposed schedule of the leading coldstarter.

Establish network communication

This final step is related to buffer settings and buffer handling. So far data exchange was not checked. To cover the possibilities and necessities for buffer locking will exceed the scope of this article. However one aspect may cause confusion: this is the generation of a transmit interrupt. This interrupt is only generated when the respective buffer has been committed, otherwise a null frame will be sent in this slot without raising an interrupt. Furthermore performing the buffer handling in the transmit interrupt routine requi-

res an initial buffer commit in the main application. Otherwise this transmit interrupt will never be raised.

Typical issues in starting a FlexRay cluster

This section shall convey a sense for the underlying problems that arise when hardware has been set up successfully in the lab but fails in a wider application scope. As recognized by most users FlexRay offers many configuration parameters giving uncountable combinations. However some typical issues have been identified when providing customer support. In general this is related to a system setup with a too marginal system setup in regards to bus driver edge asymmetries, crystal tolerances and varying EMC impact.

Number of nodes configured as leading coldstarter

To startup a network one leading coldstarter would be sufficient. This urgently requires that another coldstart node (integrating node) is already waiting for the coldstart attempts sent by the leading coldstarter. If the integrating node(s) power up after the leading coldstarter, the leading coldstarter will not get any response and stop the coldstart attempts. To avoid the dependance on power up sequencing:

Rule: enable at least two nodes in a network to be leading coldstarters.

Hint: When considering that one leading coldstarter node may fail for any reason the minimum number should be increased to 3.

Number of coldstart attempts

Simply spoken, one coldstart attempts would be sufficient to start up the network. In a distorted environment the sync frame sent be the leading coldstarter may be disturbed. In this case another attempts must be made. The number of allowed attempts depends on the expected disturbances.

Rule: configure a sufficient number of allowed coldstart attempts.

Configuration of the transmission start sequence settings

The transmission start sequence (TSS) is transmitted ahead of any frame to activate star couplers and receivers. The length is checked during reception and a too long or too short TSS makes the frame invalid. The TSS can be shortened or enlarged by stars and receiver asymmetries. Inside the MFR 4200 the transmitted as well as the expected received TSS can be configured independently. One not very obvious detail is that any two nodes with differing clock frequencies have a slightly different perception of the same bit length. In case the receiving node has the shorter bit time it may sample the incoming TSS to be 1 bit longer. The resulting rules are:

Rule: TSS (rx) [Bit] >= TSS (tx) [Bit] + 1 [Bit]

Rule: TSS (tx) [Bit] >= 2 [Bit] (network with no stars)

Adverse effects of coldstart inhibit configuration

The coldstart inhibit flag prevents a node to become a leading coldstarter. When using this flag make sure that the required number of leading coldstarters is not endangered. The effects are already described in section 3.1.

Incorrect header CRC

The header CRC is not calculated correctly. The transmitting node sends its frames but the receiving nodes discard the frames because of the wrong header CRC. By using a configuration tool this problem may be avoided.

Summary

The article has shown a guideline to bring a FlexRay cluster to operation. A number of possible pitfalls are named and advices are given to avoid problems. First usage of the first FlexRay implementation – the MFR 4200 from Freescale – has shown in the lab and in the car a robust communication.

Dipl.-Ing. Leonhard Link and **Dr.-Ing. Mathias Rausch** work at Freescale Halbleiter Deutschland GmbH.