The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN205754

FM3 MB9BF506R 系列微控制器，带 LIN 模块

本应用笔记描述了介绍了 MB9BF506R 系列的 LIN 模块使用。本应用笔记介绍了如何使用配置 MB9BF506R 的 MFS 模块实现 LIN 的功能以及 LIN 模块的寄存器操作。

目录

1 概要 1
1.1 手册目的 1
1.2 术语及缩略语 1
1.3 章节概要 1
2 LIN 接口简介 2
2.1 LIN 的概念 2
2.2 LIN 的特点 2
2.3 LIN 协议层 2
2.4 帧的结构 3
3 样例代码及 API 说明 4
3.1 样例代码流程图 4
3.2 LIN 驱动说明 7
4 样例程序演示 24
5 附加信息 26
修改记录 27

1 概要

1.1 手册目的

本应用笔记介绍了 MB9BF506R 系列的 LIN 模块使用。

本应用笔记介绍了如何使用配置 MB9BF506R 的 MFS 模块实现 LIN 的功能以及 LIN 模块的寄存器操作。

1.2 术语及缩略语

MFS	Multi-function Serial Interface
LIN	Local Interconnect Network
API	Application Programming Interface

1.3 章节概要

本文档主要包括以下几个章节：

第二章简单介绍了 LIN 的原理以及应用范围。

第三章介绍了样例程序流程结构以及各 API 接口函数。

第四章演示了样例代码。

2 LIN 接口简介

LIN 接口简介

2.1 LIN 的概念

LIN 是 Local Interconnect Network 的缩写，是基于 UART/SCI（Universal Asynchronous Receiver-Transmitter/Serial Communication Interface，通用异步收发器/串行通信接口）的低成本串行通信协议。可用于汽车、家电、办公设备等多种领域。本文主要针对 LIN 在分布式的汽车电子网络系统中的应用。

2.2 LIN 的特点

LIN 具有以下特点：

1. 网络由一个主机节点和多个从机节点构成。
2. 使用 LIN 可以大幅度的削减成本，表现在以下方面：
   - ☐ 开放型规范：规范可以免费从官方网站获得。
   - ☐ 硬件成本削减：基于普通 UART/SCI 接口的低成本硬件实现，无需单独的硬件模块支持；从机节点无需高精度时钟就可以完成自同步；总线为一根单线电缆。
   - ☐ 装配成本削减：LIN 采用了工作流(Work Flow)和现成节点(Off-the-shelf Node)的概念，将网络装配标准化，并可通过 LIN 传输层进行再配置。
   - ☐ 缩短软件开发周期：LIN 协议将 API(Application Programming Interface，应用编程接口) 标准化。
3. 信号传输具有确定性，传播时间可以提前计算出。
4. LIN 具有可预测的 EMC（ElectroMagnetic Compatibility，电磁兼容性）性能。为了限制 EMI（ElectroMagnetic Interference，电磁干扰）强度，LIN 协议规定最大位速率为 20kbps。
5. LIN 提供信号处理、配置、识别和诊断四项功能。

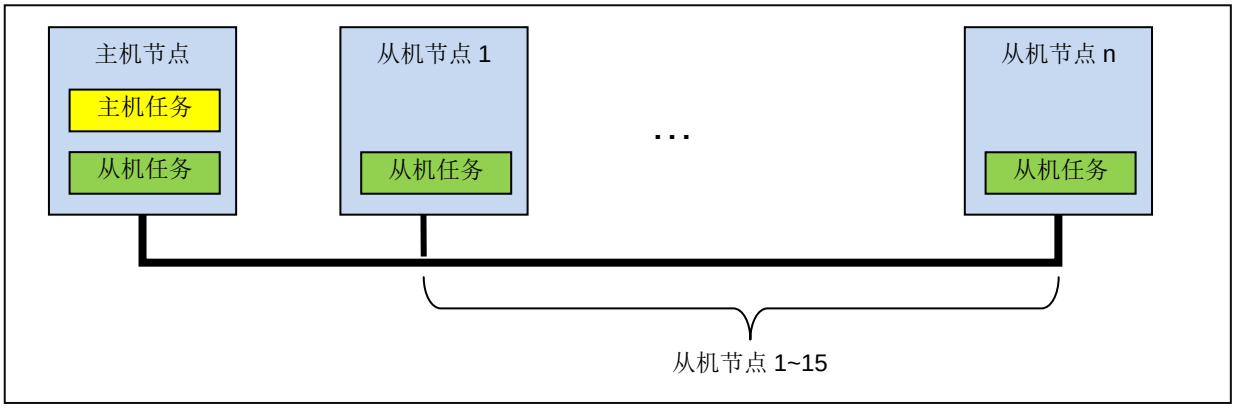
2.3 LIN 协议层

本节内容简单介绍了 LIN 的拓扑结构，帧结构。

LIN 的拓扑结构为单线总线，应用了单主机多从机的概念。总线电平为 12V，传输位速率最高为 20kbps。由于物理层限制，一个 LIN 网络最多可以连接 16 个节点，主机节点有且只有一个，从机节点可以有 1 至 15 个。

主机节点包含主机任务和从机任务；从机节点只包含从机任务。

图 1. LIN 总线拓扑图



主机任务包括：

- 调度总线上帧的传输次序
- 监测数据，处理数据
- 作为标准时钟参考
- 接收从机节点发出的总线唤醒命令

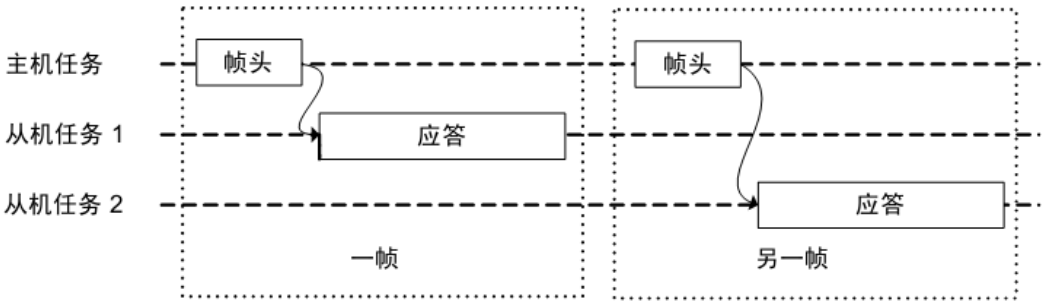
从机任务不能主动发送数据，需要接收主机发送的帧头，根据帧头所包含的信息判断：

- 发送应答（帧中除帧头外剩下的部分）
- 接收应答
- 既不接收也不发送应答

2.4 帧的结构

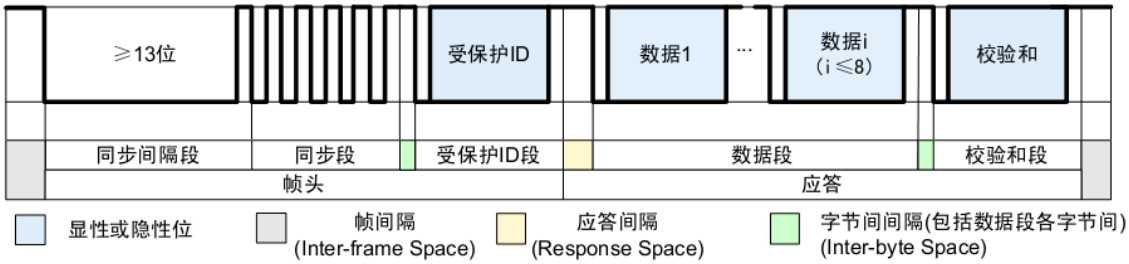
帧包含帧头和应答两部分。主机负责发送帧头；从机负责接收帧头并对帧头所包含信息进行解析，然后决定是发送应答还是接收应答，或是不作任何反应。

图 2. 帧在总线上的传输



帧头包括同步间隔段、同步段以及 PID（Protected Identifier，受保护 ID）段，应答包括数据段和校验和段，其中值“0”为显性电平（Dominant），值“1”为隐性电平(Recessive)，总线上实行“线-与”：当总线上有大于等于一个节点发送显性电平时，总线呈显性电平；所有的节点都发送隐性电平或不发送信息(不发送任何信息时总线默认呈隐性电平)时，总线才呈现隐性电平，即显性电平起主导作用。图中帧间隔为帧之间的间隔；应答间隔为帧头和应答之间的间隔；字节间间隔包括同步段和受保护 ID 段之间的间隔、数据段各字节间之间的间隔以及数据段最后一个字节和校验和段之间的间隔。

图 3. 帧的结构



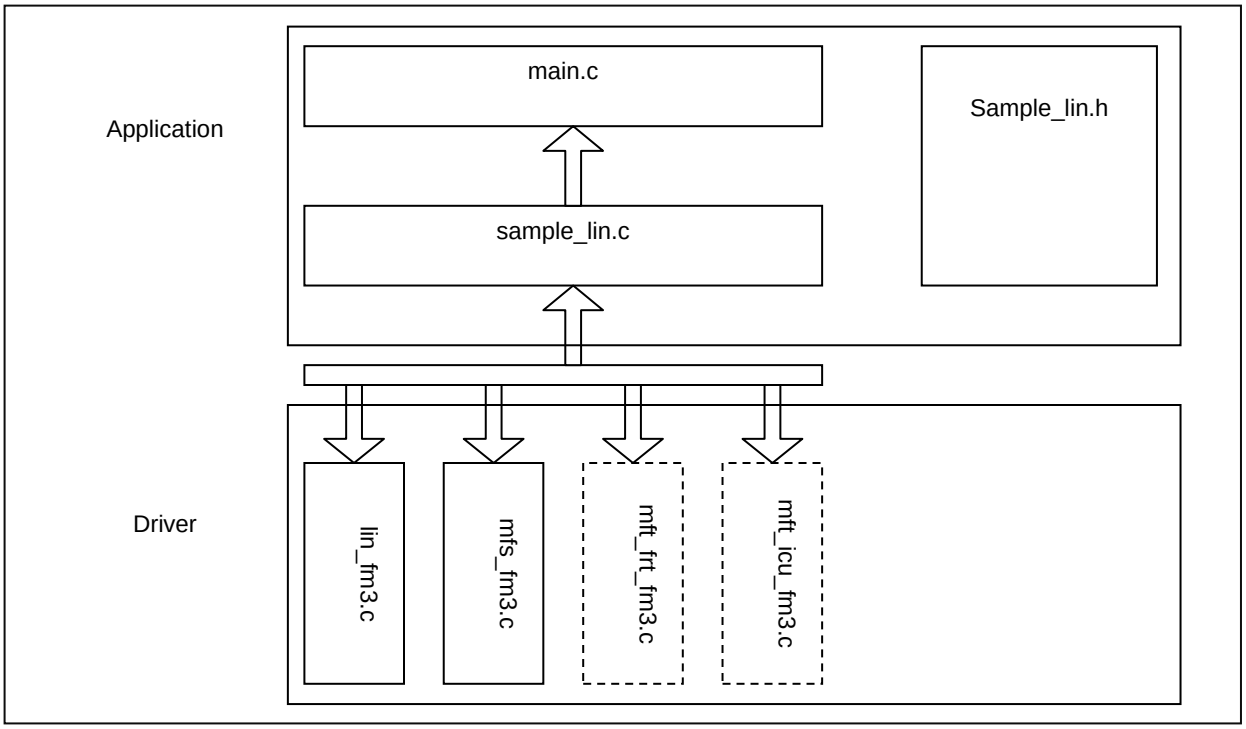
3 样例代码及 API 说明

本章介绍了样例代码以及各 API 的说明

3.1 样例代码流程图

此样例工程演示了 LIN 分别作为主机设备和从机设备时，包括同步场，波特率自动校正，数据域等的传输处理过程。程序结构如下：

图 4. 样例代码结构框图



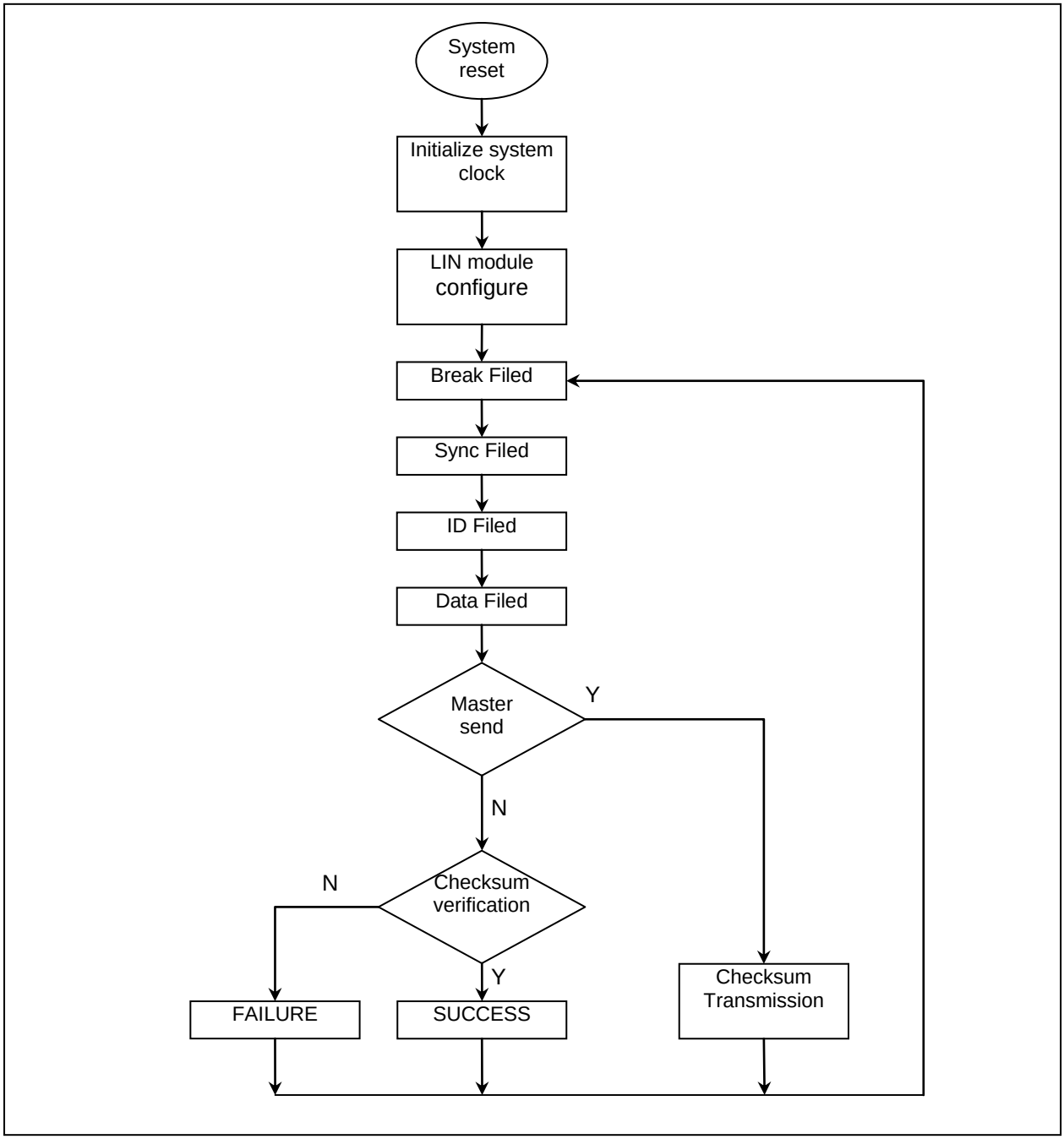
所有关于 LIN 总线的配置及操作，都在 sample_lin.c 文件里面。

**虚线框：从机设备使用*

3.1.1 主机设备主程序流程图

主机设备程序包括了 IO Port 配置，LIN 模块的初始化以及各阶段的处理。

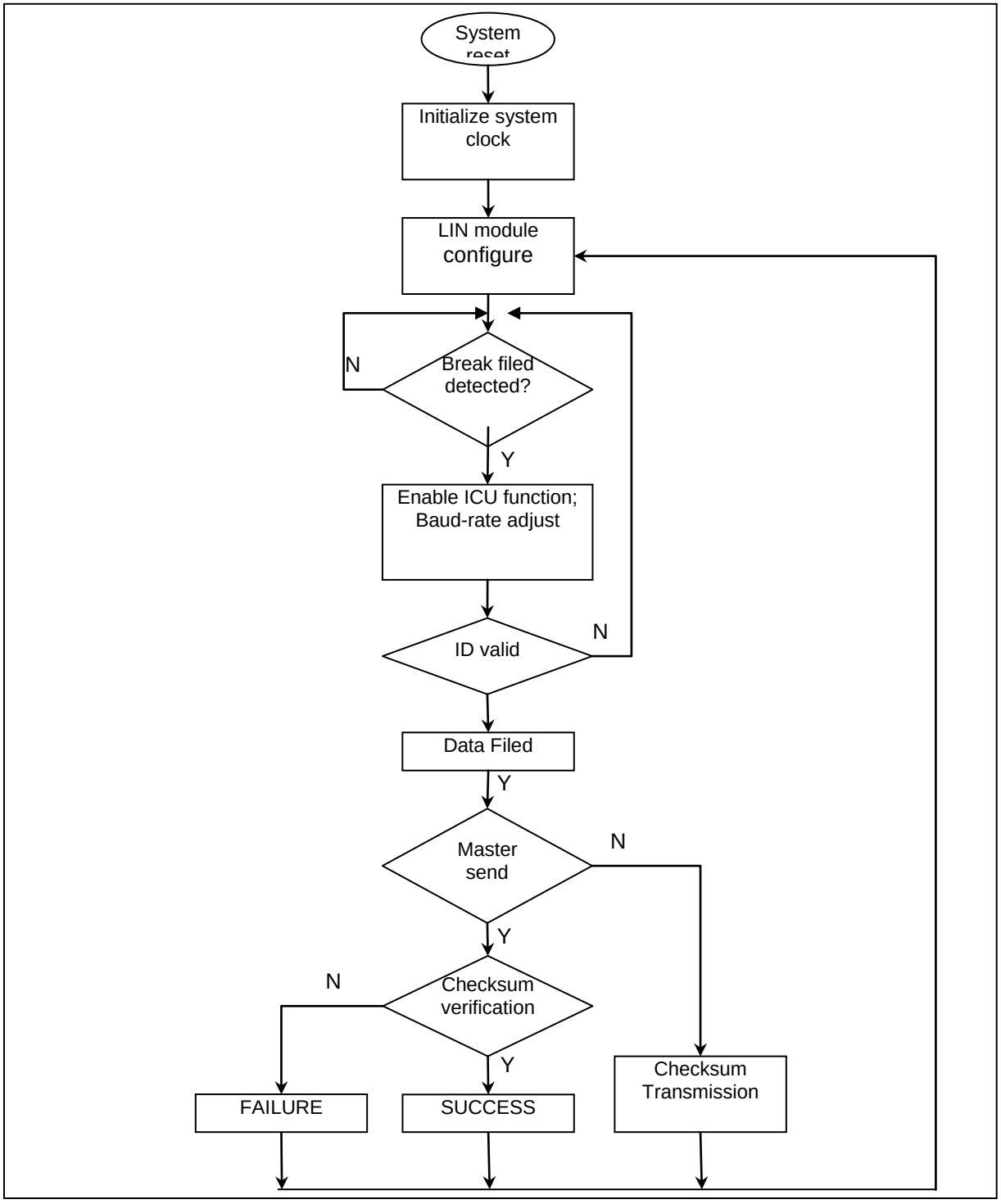
图 5. 主机设备主程序流程图



3.1.2 从机设备主程序流程图

从机设备程序包括了 IO Port 配置，同步场检测，波特率自动调整等功能。

图 6. 从机设备主程序流程图



3.2 LIN 驱动说明

LIN 驱动层函数包括了 LIN 模块所有寄存器的读写操作以及部分寄存器的配置操作。

- LIN API
  - uint16_t MFS_LINConfigBaudrate(uint8_t Ch, uint32_t baudrate);
  - void MFS_LINConfigMode(uint8_t Ch, MFS_LINModeConfigT *pModeConfig);
  - void MFS_LINSetMode(uint8_t Ch, uint8_t MS_Mode);
  - void MFS_LINSetBreakField(uint8_t Ch);
  - void MFS_LINConfigISRCallback(uint8_t Ch, MFS_LINISRCallbackT *pCallback);
  - void MFS_LINConfigFIFO(uint8_t Ch, MFS_LINFIFOConfigT *pFIFOConfig);
  - void MFS_LINResetFDRQ(uint8_t Ch);
  - void MFS_LINTXOneData(uint8_t Ch, uint16_t Data);
  - uint16_t MFS_LINRXOneData(uint8_t Ch);
  - void MFS_LINTXEnable(uint8_t Ch);
  - void MFS_LINRXEnable(uint8_t Ch);
  - void MFS_LINFIFOEnable(uint8_t Ch, uint8_t nfifo);
  - void MFS_LINTXDisable(uint8_t Ch);
  - void MFS_LINRXDisable(uint8_t Ch);
  - void MFS_LINFIFODisable(uint8_t Ch, uint8_t nfifo);
  - void MFS_LINIntTXBusIdleEnable(uint8_t Ch);
  - void MFS_LINIntTXEnable(uint8_t Ch);
  - void MFS_LINIntLBIEEnable(uint8_t Ch);
  - void MFS_LINIntTXFIFOEmptyEnable(uint8_t Ch);
  - void MFS_LINIntRXEnable(uint8_t Ch);
  - void MFS_LINIntTXBusIdleDisable(uint8_t Ch);
  - void MFS_LINIntTXDisable(uint8_t Ch);
  - void MFS_LINWakeUpDisable(uint8_t Ch);
  - void MFS_LINWakeUpEnable(uint8_t Ch);
  - void MFS_LINSOEEnable(uint8_t Ch);
  - void MFS_LINSOEDisable(uint8_t Ch);
  - void MFS_LINIntLBIEDisable(uint8_t Ch);
  - void MFS_LINIntTXFIFOEmptyDisable(uint8_t Ch);
  - void MFS_LINIntRXDisable(uint8_t Ch);
  - FlagStatusT MFS_LINStatusGetOE(uint8_t Ch);
  - FlagStatusT MFS_LINStatusGetLBD(uint8_t Ch);
  - FlagStatusT MFS_LINStatusGetFE(uint8_t Ch);
  - FlagStatusT MFS_LINStatusGetRXRegFull(uint8_t Ch);
  - FlagStatusT MFS_LINStatusGetTXRegEmpty(uint8_t Ch);

- ☐ FlagStatusT MFS_LINStatusGetTXBusIdle(uint8_t Ch);
- ☐ void MFS_LINGetFDRQ(uint8_t Ch);
- ☐ uint8_t MFS_LINStatusFBYTE1(uint8_t Ch);
- ☐ uint8_t MFS_LINStatusFBYTE2(uint8_t Ch);
- ☐ void MFS_LINErrorClr(uint8_t Ch);
- ☐ void MFS_LINProgramClr(uint8_t Ch);
- ☐ void MFS_LINStatusSetLBD(uint8_t Ch, FlagStatusT val);

3.2.1 各 API 功能介绍

3.2.1.1 MFS_LINConfigBaudrate

返回 BRG 寄存器写入值。

函数原型

uint16_t

MFS_LINConfigBaudrate(uint8_t Ch, uint32_t baudrate)

参数

baudrate：目标波特率

返回值

tBGR

描述

传入目标波特率，通过计算返回需要写入到 BGR 寄存器的值。

3.2.1.2 MFS_LINConfigMode

配置 LIN 模块基本功能。

函数原型

void

MFS_LINConfigMode(uint8_t Ch, MFS_LINModeConfigT *pModeConfig)

参数

*pModeConfig：结构体指针，包含波特率，停止位长度，Break Field 位长，LIN Delimiter 位长。

- Baudrate：目标波特率
- BreakFieldbits：LIN Break Filed 位长
- BreakDelimiter：LIN Break Delimiter 位长
- Stopbits：停止位长

返回值

无

描述

配置 LIN 模块的波特率，LIN Break 域位长，LIN Delimiter 位长，停止位长度。

3.2.1.3 MFS_LINSetMode

设置 LIN 模块为主机或者从机。

函数原型

void

MFS_LINSetMode(uint8_t Ch, uint8_t MS_Mode)

参数

Ch：目标通道

MS_Mode：　　0 –> 主机；1 –> 从机

返回值

无

描述

设置 LIN 模块模式

3.2.1.4 MFS_LINSetBreakField

发送 LIN Break Field

函数原型

void

MFS_LINSetBreakField(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

设置 SRC:LBR 位，发送 LIN Break Field，仅针对于主机设备使用。

3.2.1.5 MFS_LINConfigISRCallback

配置回调函数入口地址

函数原型

void

MFS_LINConfigISRCallback(uint8_t Ch, MFS_LINISRCallbackT *pCallback)

参数

Ch：目标通道

pCallback：回调函数入口

返回值

无

描述

配置 LIN 模块中断回调函数入口。

3.2.1.6 MFS_LINConfigFIFO

配置 FIFO。

函数原型

void

MFS_LINConfigFIFO(uint8_t Ch, MFS_LINFIFOConfigT *pFIFOConfig)

参数

Ch：目标通道

pFIFOConfig：结构体指针，包含 FIFO1，FIFO2 的配置

- FIFOSel：FIFO 模式选择，0 -> FIFO1 作为发送；1 -> FIFO2 作为发送
- Bytecount1：FIFO1 字节数
- Bytecount2：FIFO2 字节数

返回值

无

描述

为 LIN 模块配置 FIFO，FIFO1 发送，FIFO2 接收或 FIFO1 接收 FIFO2 发送，以及各 FIFO 中数据的长度。

3.2.1.7 MFS_LINResetFDRQ

清零传输 FIFO 数据请求位。

函数原型

void

MFS_LINResetFDRQ(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

将目标通道寄存器 FCR1:FDRQ 位清零，从而清除传输 FIFO 数据请求位。

3.2.1.8 MFS_LINTXOneData

发送一字节数据

函数原型

void

MFS_LINTXOneData(uint8_t Ch, uint8_t Data)

参数

Ch：目标通道

Data：需要发送的数据

返回值

无

描述

发送 TDR 寄存器里面的数据

3.2.1.9 MFS_LINRXOneData

接收一字节数据

函数原型

uint8_t

MFS_LINRXOneData(uint8_t Ch)

参数

Ch：目标通道

返回值

RDR：接收到的数据

描述

接收一字节数据到数据接收寄存器（RDR）

3.2.1.10 MFS_LINTXEnable

使能数据发送

函数原型

void

MFS_LINTXEnable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

使能数据发送

3.2.1.11 MFS_LINRXEnable

使能数据接收

函数原型

void

MFS_LINRXEnable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

使能数据接收

3.2.1.12 MFS_LINFIFOEnable

使能 FIFO 功能

函数原型

void

MFS_LINFIFOEnable(uint8_t Ch, uint8_t nfifo)

参数

Ch：目标通道

nfifo：FIFO 编号

返回值

无

描述

使能 FIFO 功能

3.2.1.13 MFS_LINTXDisable

禁止数据发送

函数原型

void

MFS_LINTXDisable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

禁止数据发送

3.2.1.14 MFS_LINRXDisable

禁止数据接收

函数原型

void

MFS_LINRXDisable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

禁止数据接收

3.2.1.15 MFS_LINFIFODisable

禁止 FIFO 功能

函数原型

void

MFS_LINFIFODisable(uint8_t Ch, uint8_t nfifo)

参数

Ch：目标通道

nfifo：FIFO 编号

返回值

无

描述

禁止 FIFO 功能

3.2.1.16 MFS_LINIntTXBusIdleEnable

使能传输空闲中断

函数原型

void

MFS_LINIntTXBusIdleEnable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

使能传输总线空闲中断

3.2.1.17 MFS_LINIntTXEnable

使能发送数据中断

函数原型

void

MFS_LINIntTXEnable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

当传输数据寄存器空标志（SSR:TDRE）置位时，产生一个发送中断

3.2.1.18 MFS_LINIntLBIEEnable

使能 LIN Break Field 侦测中断

函数原型

void

MFS_LINIntLBIEEnable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

当 LIN Break Field 被检测到时，如果 LIN Break Field 侦测中断已经使能，将会产生一个状态中断（Status Interrupt）

3.2.1.19 MFS_LINIntTXFIFOEmptyEnable

使能传输 FIFO 中断

函数原型

void

MFS_LINIntTXFIFOEmptyEnable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

使能传输 FIFO 中断

3.2.1.20 MFS_LINIntRXEnable

使能发送数据中断

函数原型

void

MFS_LINIntRXEnable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

使能发送数据中断。除数据接收完成外，任何接收错误标志（SSR:FRE， ORE）也会产生该中断输出

3.2.1.21 MFS_LINIntTXBusIdleDisable

禁止发送空闲中断请求

函数原型

void

MFS_LINIntTXBusIdleDisable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

禁止发送空闲中断请求

3.2.1.22 MFS_LINIntTXDisable

禁止数据发送中断请求

函数原型

void

MFS_LINIntTXDisable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

禁止数据发送中断请求

3.2.1.23 MFS_LINWakeUpDisable

禁止唤醒功能

函数原型

void

MFS_LINWakeUpDisable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

禁止唤醒功能，INT 作为外部中断引脚

3.2.1.24 MFS_LINWakeUpEnable

使能唤醒功能

函数原型

void

MFS_LINWakeUpEnable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

使能唤醒功能，SIN 作为外部中断引脚

3.2.1.25 MFS_LINSOEEnable

使能串行数据输出

函数原型

void

MFS_LINSOEEnable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

使能串行数据输出

3.2.1.26 MFS_LINSOEDisable

禁止串行数据输出

函数原型

void

MFS_LINSOEDisable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

禁止串行数据输出

3.2.1.27 MFS_LINIntLBIEDisable

禁止 LIN Break Field 侦测中断

函数原型

void

MFS_LINIntLBIEDisable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

禁止 LIN Break Filed 侦测中断

3.2.1.28 MFS_LINIntTXFIFOEmptyDisable

禁止传输数据 FIFO 空中断

函数原型

void

MFS_LINIntTXFIFOEmptyDisable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

禁止 LIN 传输数据 FIFO 空中断

3.2.1.29 MFS_LINIntRXDisable

禁止接收中断

函数原型

void

MFS_LINIntRXDisable(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

禁止 LIN 数据接收中断

3.2.1.30 MFS_LINStatusGetOE

检测是否接收数据溢出

函数原型

FlagStatusT

MFS_LINStatusGetOE(uint8_t Ch)

参数

Ch：目标通道

返回值

SET：1 -> 产生数据接收溢出错误

RESET：0 -> 无数据接收溢出错误

描述

通过查看 SSR:ORE 位状态，判断是否产生接收数据溢出错误

3.2.1.31 MFS_LINStatusGetLBD

是否侦测到 LIN Break Filed

函数原型

FlagStatusT

MFS_LINStatusGetLBD(uint8_t Ch)

参数

Ch：目标通道

返回值

SET：1 -> 已侦测到 LIN Break Filed

RESET：0 -> 未侦测到 LIN Break Filed

描述

通过查看 SSR:LBD 位状态，判断是否侦测到 LIN Break Field，当侦测到 LIN Break Field 后，将进行 LIN Sync Field 的操作

3.2.1.32 MFS_LINStatusGetFE

检测是否接收数据帧错误

函数原型

FlagStatusT

MFS_LINStatusGetFE(uint8_t Ch)

参数

Ch：目标通道

返回值

SET：1 -> 产生数据接收帧错误

RESET：0 -> 无数据接收帧错误

描述

通过查看 SSR:FRE 位状态，判断是否产生数据接收帧错误

3.2.1.33 MFS_LINStatusGetRXRegFull

检测接收数据寄存器是否满

函数原型

FlagStatusT

MFS_LINStatusGetRXRegFull(uint8_t Ch)

参数

Ch：目标通道

返回值

SET：1 -> 接收数据寄存器有数据

RESET：0 -> 接收数据寄存器空

描述

通过查看 SSR:RDRF 位状态，判断接收数据寄存器（RDR）内是否有数据

3.2.1.34 MFS_LINStatusGetTXRegEmpty

检测发送数据寄存器是否为空

函数原型

FlagStatusT

MFS_LINStatusGetTXRegEmpty(uint8_t Ch)

参数

Ch：目标通道

返回值

SET：1 -> 传输数据寄存器空

RESET：0 -> 传输数据寄存器内有数据

描述

通过查看 SSR:TDRE 位状态，判断发送数据寄存器（TDR）是否为空

3.2.1.35 MFS_LINStatusGetTXBusIdle

检测数据是否正在传输

函数原型

FlagStatusT

MFS_LINStatusGetTXBusIdle(uint8_t Ch)

参数

Ch：目标通道

返回值

SET：1 -> 无数据传输

RESET：0 -> 数据正在传输

描述

通过查看 SSR:TBI 位状态，判断数据是否正常传输。当有数据被写入数据寄存器（TDR）后，SSR:TBI 被清零；当数据寄存器为空时，SSR:TBI 被置位

3.2.1.36 MFS_LINGetFDRQ

获取传输 FIFO 数据请求位状态

函数原型

FlagStatusT

MFS_LINGetFDRQ(uint8_t Ch)

参数

Ch：目标通道

返回值

SET：1，传输 FIFO 数据请求产生

RESET：0，无传输 FIFO 数据请求产生

描述

返回 FCR:FDRQ 值，取得传输 FIFO 数据请求状态

3.2.1.37 MFS_LINStatusFBYTE1

获取 FIFO1 数据长度

函数原型

uint8_t

MFS_LINStatusFBYTE1(uint8_t Ch)

参数

Ch：目标通道

返回值

FBYTE1：FIFO1 数据长度

描述

获取 FIFO1 数据长度

3.2.1.38 MFS_LINStatusFBYTE2

获取 FIFO2 数据长度

函数原型

uint8

MFS_LINStatusFBYTE2(uint8_t Ch)

参数

Ch：目标通道

返回值

FBYTE2：FIFO2 数据长度

描述

获取 FIFO2 数据长度

3.2.1.39 MFS_LINErrorClr

清零接收错误标志

函数原型

void

MFS_LINErrorClr(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

清零帧错误（FER）、溢出错误（ORE）标志位

3.2.1.40 MFS_LINProgramClr

复位 LIN 模块

函数原型

void

MFS_LINProgramClr(uint8_t Ch)

参数

Ch：目标通道

返回值

无

描述

通过置位 SCR:UPCL，软件复位 LIN 模块；重载波特率计数器；初始化接收发送的中断源（SSR:TDRE, TBI, RDRF, FRE, ORE, LBD）

3.2.1.41 MFS_LINStatusSetLBD

清零 SSR:LBD 位

函数原型

void

MFS_LINStatusSetLBD(uint8_t Ch, FlagStatusT val)

参数

Ch：目标通道

val： SET，1 -> 写入无效

RESET，0 –> 清零

返回值

无

描述

清零 LIN Break Field 检测标志位

3.2.2 样例代码

3.2.2.1 LIN 接口 GPIO 初始化设置

以下样例代码给出了如何初始化 LIN 接口为主机设备或从机设备，以通道 4 举例。

将 GPIO 作为 LIN 接口使用。如果是从机设备的话，还需要有 FRT 以及 ICU 的配置。

图 7. 主机设备 GPIO 初始化设置

```
/* Init I/O Port */
/*Set P0A/P0B as Func*/
FM3_GPIO->PFR0 = FM3_GPIO->PFR0 | 0x0C00;
/*Select SOT4_0/SIN4_0*/
/*Use SIN4_0*/
bFM3_GPIO_EPFR08_SIN4S0 = 0;
bFM3_GPIO_EPFR08_SIN4S1 = 0;
/*Use SOT4_0*/
bFM3_GPIO_EPFR08_SOT4B0 = 1;
bFM3_GPIO_EPFR08_SOT4B1 = 0;
/* Pull up GPIO */
bFM3_GPIO_PCR0_PA = 1;
bFM3_GPIO_PCR0_PB = 1;
```

图 8. 从机设备 GPIO 初始化设置

```
/* Init I/O Port */
/*Set P0A/P0B as Func*/
FM3_GPIO->PFR0 = FM3_GPIO->PFR0 | 0x0C00;
/*Select SOT4_0/SIN4_0*/
/*Use SIN4_0*/
bFM3_GPIO_EPFR08_SIN4S0 = 0;
bFM3_GPIO_EPFR08_SIN4S1 = 0;
/*Use SOT4_0*/
bFM3_GPIO_EPFR08_SOT4B0 = 1;
bFM3_GPIO_EPFR08_SOT4B1 = 0;
/* Pull up GPIO */
bFM3_GPIO_PCR0_PA = 1;
bFM3_GPIO_PCR0_PB = 1;
/* connect MFS ch.2 to ICU input IC02 */
FM3_GPIO->EPFR01 = FM3_GPIO->EPFR01 & 0xecffffff;
FM3_GPIO->EPFR01 = FM3_GPIO->EPFR01 | 0x10000000;
/* connect MFS ch.4 to ICU input IC00 */
bFM3_GPIO_EPFR01_IC00S0 = 1;
bFM3_GPIO_EPFR01_IC00S1 = 0;
bFM3_GPIO_EPFR01_IC00S2 = 1;
```

3.2.2.2 LIN 接口参数配置

图 9. LIN 接口参数设置

```
static MFS_LINModeConfigT tLINModeConfigT =
{
        19200,                                  /* Baudrate */
        LIN_BREAKFIELDBITS_14,      /* LIN Break Field Length */
        LIN_BREAKDELIMITER_4,       /* LIN Break Delimiter Length */
        LIN_STOPBITS_1                  /* Stop bit Length */
};
MFS_LINConfigMode(Ch, &tLINModeConfigT);
```

3.2.2.3 LIN 接口中断函数入口配置

图 10. LIN 接口中断函数入口配置

```
/* Set the callback func address */
tLINCallback.pISRRXCallback = LINIrqRXCallback;
tLINCallback.pISRTXCallback = LINIrqTXCallback;
MFS_LINConfigISRCallback(Ch, &tLINCallback);
NVIC_EnableIRQ((IRQn_Type)(MFS0TX_IRQn + (Ch*2)));
```

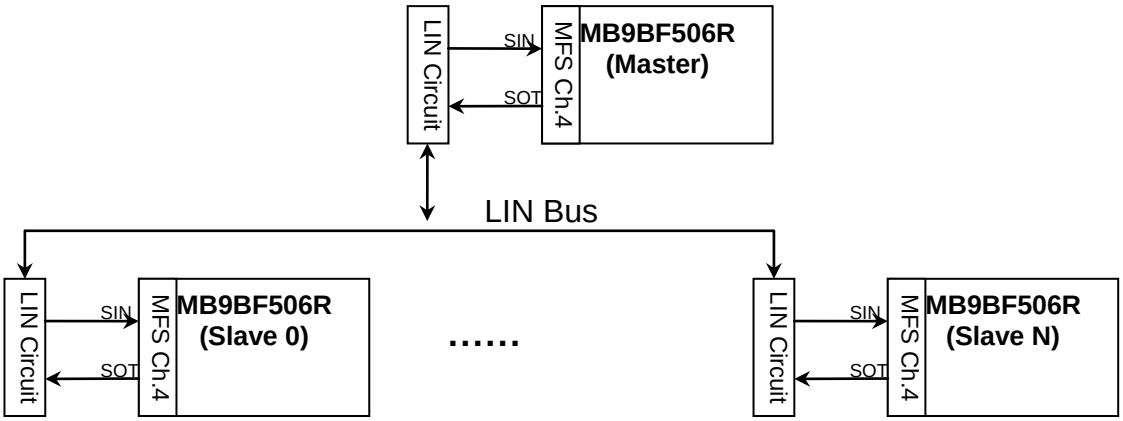
用户可以在中断回调函数里面添加自己的处理程序。

4 样例程序演示

4.1.1 硬件连接

使用该样例代码需要至少两颗或以上的 MB9BF506R，一颗作为主机设备，剩余的作为从机设备；除此之外，还需要与之对应数量的 LIN 接口电路。

图 11. LIN 接口硬件连接示意图



4.1.2 样例执行

该样例程序实现了 LIN 接口作为主机设备，从机设备使用中断和查询方式实现数据的传输。

使用样例代码时，需先运行从机设备程序，然后再运行主机设备程序。

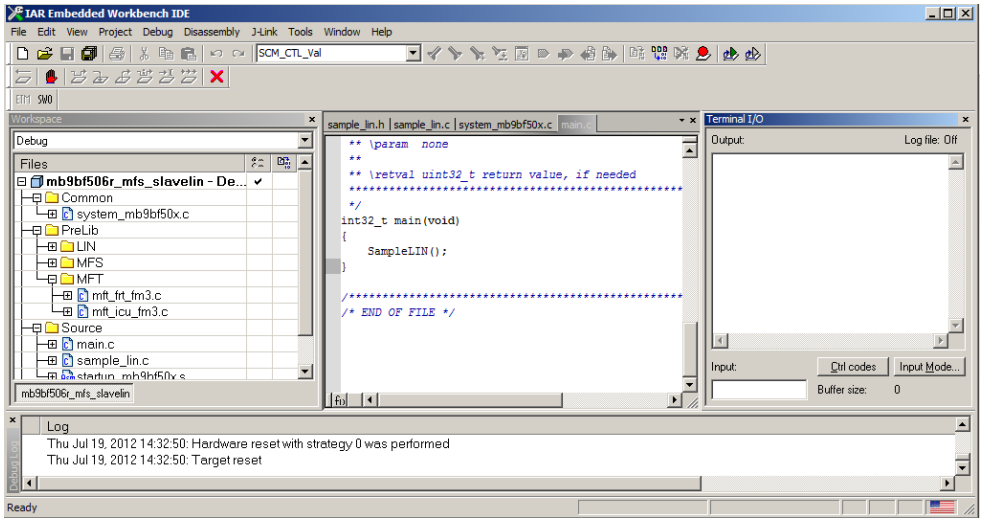
数据传输状态将由 IAR 的 Terminal I/O 窗体显示。

- 步骤 1：设置为主机发送或主机接收

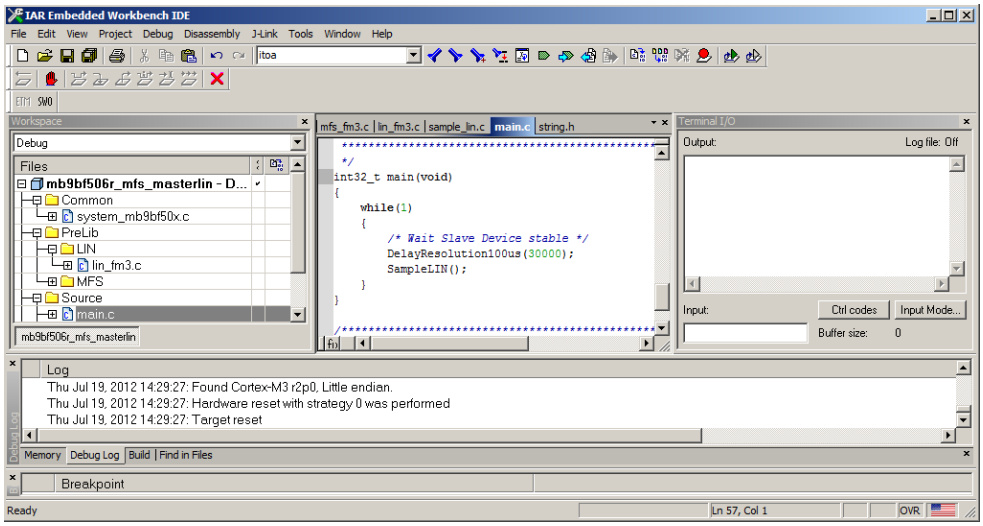
在主机设备工程的 sample_lin.c 文件的 InitVar(void)函数里面，设置主机设备发送或接收数据。

```
static void InitVar(void)
{
        tLIN_Info.LIN_Header = MASTERSend;      // master device send
        //tLIN_Info.LIN_Header = SLAVESend;     // master device receive
        ...
}
```

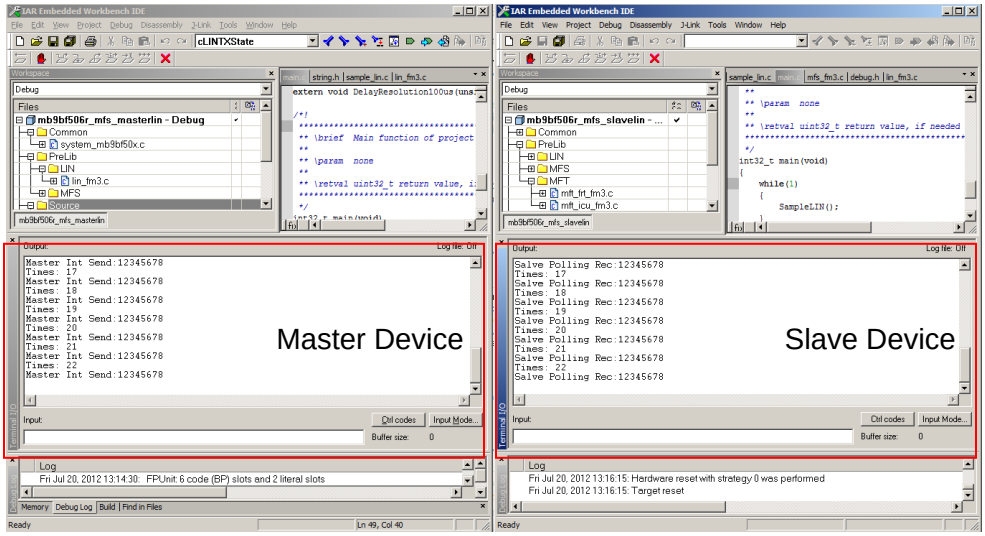
- 步骤 2：运行从机设备工程程序



- 步骤 3：运行主机设备程序



- 步骤 4：通过 IAR 的 Terminal IO 来查看当前 LIN 通信状态



5 附加信息

关于 Cypress 半导体更多的产品信息，请访问以下网站：

http://www.cypress.com/cypress-microcontrollers

http://www.cypress.com/cypress-mcu-product-softwareexamples

修改记录

文档标题：AN205754 - FM3 MB9BF506R 系列微控制器，带 LIN 模块

文档编号：002-05754

修订版	ECN	变更者	提交日期	变更说明
**	-	HUAL	07/16/2012	初稿
*A	5591617	HUAL	01/18/2017	将 Spansion 应用手册“MCU-AN-510109-Z-10”转换为 Cypress 格式。

全球销售和设计支持

赛普拉斯公司拥有一个由办事处、解决方案中心、厂商代表和经销商组成的全球性网络。如果想要查找离您最近的办事处，请访问赛普拉斯所在地。

产品

ARM® Cortex®微控制器	cypress.com/arm
汽车级产品	cypress.com/automotive
时钟与缓冲器	cypress.com/clocks
接口	cypress.com/interface
物联网	cypress.com/iot
存储器	cypress.com/memory
微控制器	cypress.com/mcu
PSoC	cypress.com/psoc
电源管理 IC	cypress.com/pmic
触摸感应	cypress.com/touch
USB 控制器	cypress.com/usb
无线连接	cypress.com/wireless

PSoC® 解决方案

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

赛普拉斯开发者社区

论坛 | WICED IoT 论坛| 项目 | 视频 | 博客 | 培训 | 组件

技术支持

cypress.com/support

PSoC 是赛普拉斯半导体公司的注册商标。PSoC Creator 是赛普拉斯半导体公司的商标。 此处引用的所有其他商标或注册商标都归其各自所有者所有。

赛普拉斯半导体
198 Champion Court
San Jose, CA 95134-1709

电话 :408-943-2600
传真 :408-943-4730
网站地址 :www.cypress.com

©赛普拉斯半导体公司，2012-2017 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属人性质的、非独家且不可转让的如下许可（无再许可权）（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码形式的向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。

在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的限度内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的主张，并使之免受损失。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，WICED，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。