

32 位微控制器

HC32L15 系列的同步传输通信模块

适用对象

系列	产品型号
HC32L15	HC32L150KATA
	HC32L150JATA
	HC32L150FAUA
	HC32L156KATA
	HC32L156JATA

目 录

1	摘要	3
2	简介	3
3	HC32L15 系列的同步传输通信模块	4
3.1	简介	4
3.2	说明	4
3.3	功能	5
3.3.1	物理连接	5
3.3.2	传输模式	5
3.3.3	物理连线配置	6
3.3.4	主机发送	7
3.3.5	主机接收	8
3.3.6	从机发送	10
3.3.7	从机接收	11
3.3.8	定时发送	12
3.3.9	同步传输片选功能	14
4	样例代码	16
4.1	代码介绍	16
4.2	代码运行	17
5	总结	19
6	版本信息 & 联系方式	20

1 摘要

本篇应用笔记主要介绍如何使用 HC32L15 系列芯片中的同步传输模块，包含模块特性和配置使用方法。

2 简介

HC32L15 系列 MCU 包含一个同步传输通信模块。此模块和异步传输、I2C 传输共享硬件资源，可选择其一使用。对于同步传输模式，都有独立的 6 路通道使用，6 通道的功能完全一致。此外，还内置定时器，用于定时发送数据。可搭配片选功能，输出片选信号。

3 HC32L15 系列的同步传输通信模块

3.1 简介

华大 HC32L15 系列单片机内部集成同步传输模块，用于实现和外部装置同步通信的通用串行数据通信接口。

3.2 说明

- 全双工双缓冲
- 支持主/从模式
- 支持无片选信号模式(时钟同步模式)
- 专用波特率发生器(使用 15 位重载计数器配置；主控操作时)
- 可输入外部时钟(从动操作时)
- 数据长度 5~16 位
- 接收溢出错误检测
- 多种中断请求
- 串行片选
- 建立/保持/释放时间可设定
- 选中有效电平可以设置
- 数据按照定时器设定的周期自动发送
- 搭载 16 位定时器

3.3 功能

3.3.1 物理连接

在不使用片选功能下，同步通信使用 3PIN：

1. SCK

时钟控制信号

2. SOT

数据发送

3. SIN

数据接收

在增加片选功能后，

4. SCS

片选信号

可以同时使用 1-4 个片选信号来控制多个设备

以上这些引脚使用，可以通过配置 GPIO 模块，切换 IO 到同步传输功能上。

3.3.2 传输模式

同步传输模式实现了 SPI 的功能，作为 SPI 功能使用时，可以配置 SPI 的时钟极性和时钟相位。

通过寄存器位 SPI 和 CINV 来控制，对应于通常的 CPHA, CPOL。

CINV (CPOL)	SPI (CPHA)	
0 (1)	0 (1)	SCK 空闲 H, 上升沿采样
0 (1)	1 (0)	SCK 空闲 H, 下降沿采样
1 (0)	0 (1)	SCK 空闲 L, 下降沿采样
1 (0)	1 (0)	SCK 空闲 L, 上升沿采样

详细的描述可参考 https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

3.3.3 物理连线配置

要使特定的 GPIO 工作在同步传输模式下，需配置相应的 GPIO 口。

可以使用 API 完成操作。

例如：使用通道 1 的 SCK，SOT，SIN 时，初始化代码如下：

```
Gpio_SetFunc_SCK1_0();  
Gpio_SetFunc_SOT1_0();  
Gpio_SetFunc_SIN1_0();
```

更详细的设定请参考 MCU 数据手册。

连线图：

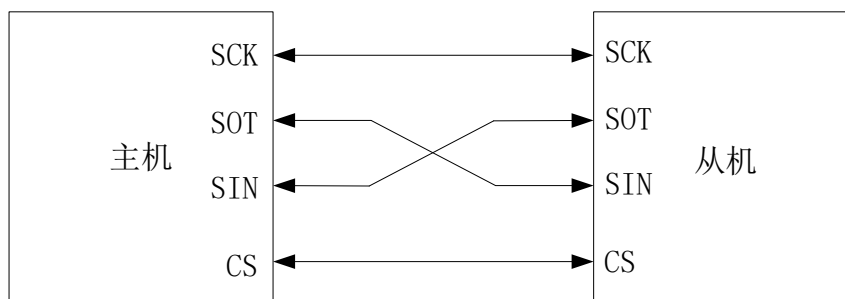


图 1 物理连线图（CS 可选）

3.3.4 主机发送

对于主机模式，初始化设置如下：

- | | |
|---------------------------|---|
| 1. 设置 CR.MSS = 0 | 主机模式 |
| 2. 设置 CR.SPIMODE, MR.CINV | 配置传输模式 |
| 3. 设置 CR.DOE = 1 | 使能数据输出（对于接收数据，可关闭此位，只发送数据 SCK 信号，让从机发送数据） |
| 4. 其他配置项 | 例如发送数据位数，大小端，等其他可选配置 |
| 5. 设置 CR.TXE = 1 | 使能发送主机模式的发送数据流程如下 |
| 6. 等待 SR.TDEF == 1 | 发送寄存器为空，可以发送数据 |
| 7. TXDR = data | 将数据写入发送寄存器 |
| 8. 下一次发送时，重复 6-7 步骤 | |

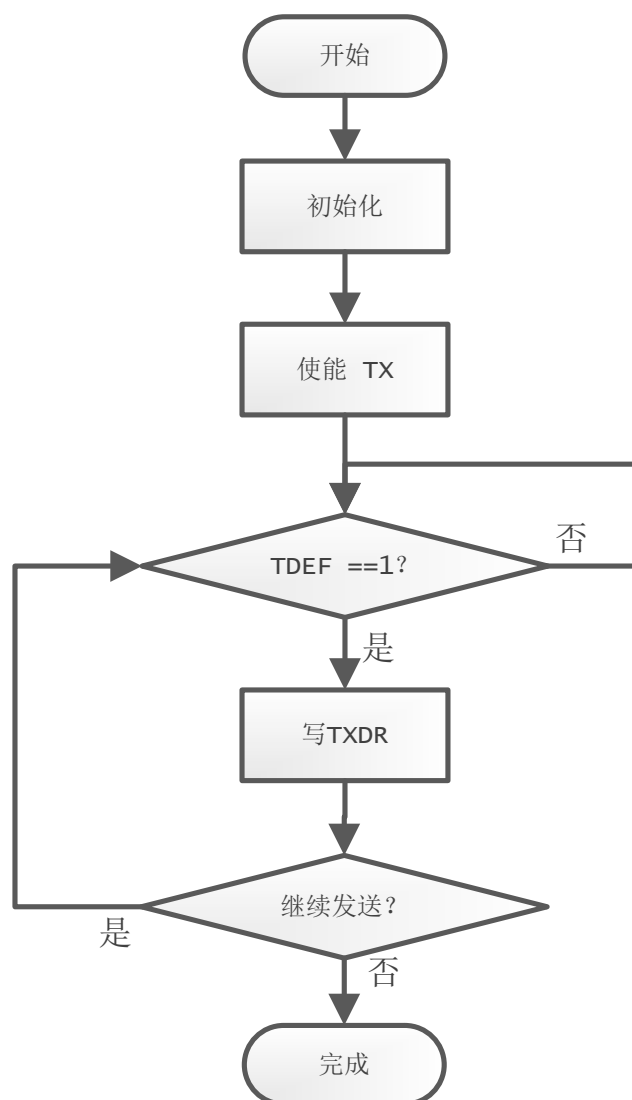


图 1 主机发送流程

3.3.5 主机接收

- | | |
|---------------------------|------------------------|
| 1. 设置 CR.MSS = 0 | 主机模式 |
| 2. 设置 CR.SPIMODE, MR.CINV | 配置传输模式 |
| 3. 设置 CR.DOE = 0 | 只发送 SCK 信号, 让从机发送数据 |
| 4. 其他配置项 | 例如发送数据位数, 大小端, 等其他可选配置 |
| 5. 设置 CR.RXE = 1 | 使能发送主机接收数据流程: |
| 6. 设置 CR.DOE = 0 | |
| 7. TXDR = 0x0 | 任何数据都可, 使控制器发送 SCK 信号 |
| 8. 等待 SR.RDFF == 1 | 说明接收到数据 |
| 9. Data = RXDR | 读取数据 |
| 10. 如果继续读取数据, 则重复 7-9 步骤 | |

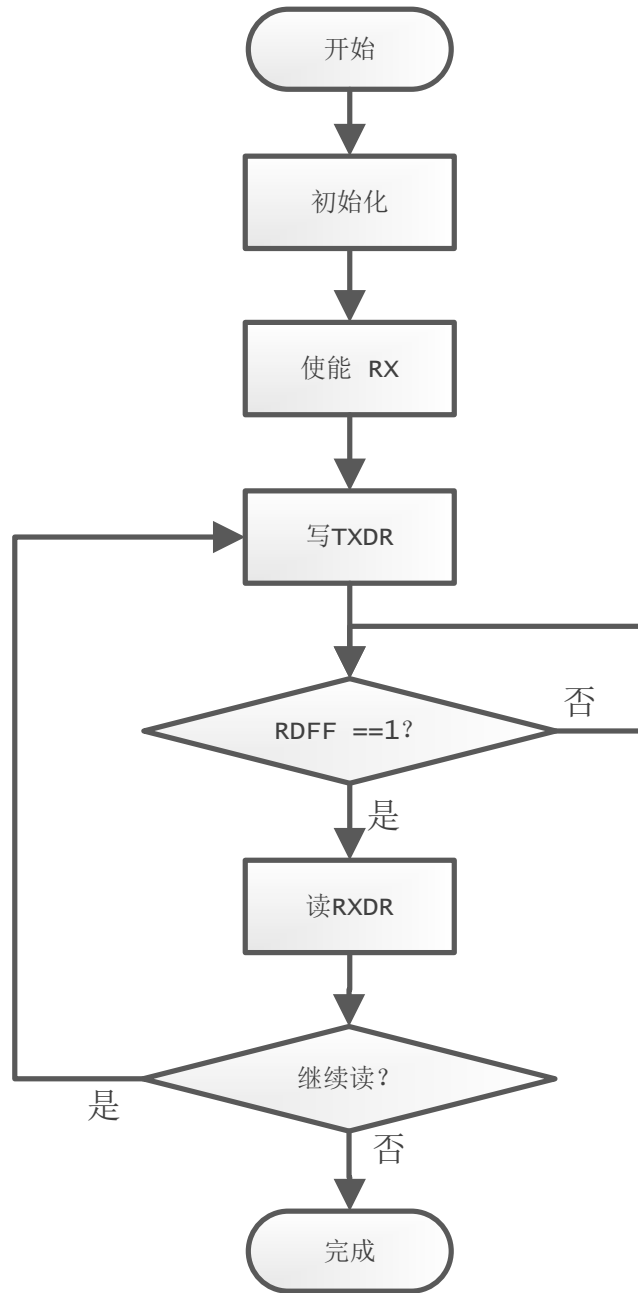


图 3 主机读取数据流程

3.3.6 从机发送

1. 设置 CR.MSS = 1 从机模式
2. 设置 CR.SPIMODE, MR.CINV 配置传输模式
3. 设置 CR.DOE = 1 使能数据输出
4. 配置数据位数
5. 配置大小端
6. 设置 CR.TXE = 1 使能发送

发送数据流程如下：

从机模式下，等待发送寄存器标志位为空，将发送的数据写入寄存器。当 MCU 收到主机的 SCK 时，自动发送数据到主机，并清空标志位后，可继续发送数据。

7. 等待 SR.TDEF == 1 发送寄存器为空，可以发送数据
8. TXDR = data 将数据写入发送寄存器
9. 下一次发送时，重复 7-8 步骤

流程图同主机发送模式。

3.3.7 从机接收

- | | |
|---------------------------|------------------------|
| 1. 设置 CR.MSS = 1 | 从机模式 |
| 2. 设置 CR.SPIMODE, MR.CINV | 配置传输模式 |
| 3. 设置 CR.DOE = 0 | 只发送 SCK 信号, 让从机发送数据 |
| 4. 其他配置项 | 例如发送数据位数, 大小端, 等其他可选配置 |
| 5. 设置 CR.RXE = 1 | 使能发送 |

主机接收数据流程:

- | | |
|--------------------|---------|
| 6. 等待 SR.RDFF == 1 | 说明接收到数据 |
| 7. Data = RXDR | 读取数据 |

如果继续读取数据, 则重复 6-7 步骤, 流程图如下:

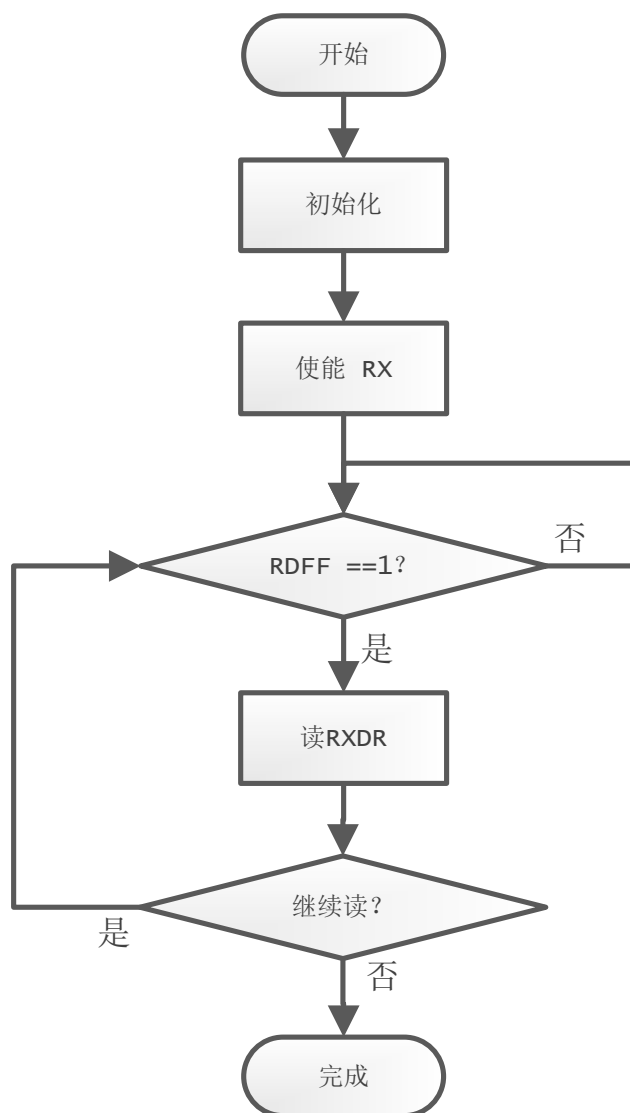


图 2 从机接收数据流程

3.3.8 定时发送

同步传输模式内置一个定时器，启用此定时器后，发送数据的最大个数可设置，超过最大个数后，下一数据将等待此定时器间溢出后再发送，以此类推。

定时发送数据设置：

对于主机模式，初始化设置如下：

1. 设置 $CR.MSS = 0$ 主机模式
2. 设置 $CR.SPIMODE$, $MR.CINV$ 配置传输模式
3. 设置 $CR.DOE = 1$ 使能数据输出（对于接收数据，可关闭此位，只发送 SCK 信号，让从机发送数据）
4. 其他配置项 例如发送数据位数，大小端，等其他可选配置
5. 配置定时器分频系数 $EACR.CLKDIV$ 和 定时器计数值 TMC ，这两个参数决定了发送时间间隔。
6. 使能同步发送 $EACR.SYNTE = 1$
7. 设置最大发送数据个数 $TDTCNT0$ ，不使用 CS 功能时，使用此寄存器。
8. 使能计数器 $EACR.TMRE = 1$
9. 设置 $CR.TXE = 1$ 使能发送

主机模式的发送数据流程如下：

10. 等待 $SR.TDEF == 1$ 发送寄存器为空，可以发送数据
11. $TXDR = data$ 将数据写入发送寄存器
12. 下一次发送时，重复 10-11 步骤，当连续发送数据时间小于定时器值时，则需要等待定时器溢出后，才能再次发送。

定时发送流程如下：

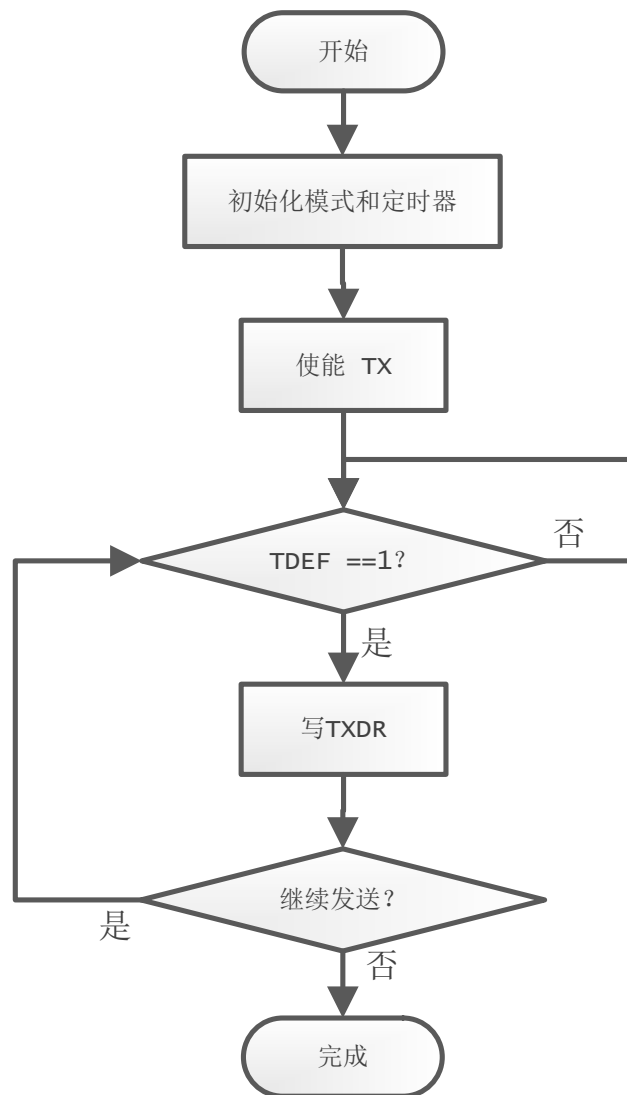


图 5 同步传输定时发送

3.3.9 同步传输片选功能

同步传输的每个通道，可使其相应的片选信号组，输出带片选信号的同步传输。

片选信号组有多个信号线输出，可以连接多个设备。通过设定片选信号的起始和结束编号，使能相应的片选 **PIN**。不再此编号范围内的，不工作。

片选信号同时受发送计数器控制，当有多个片选 **PIN** 使能后，每个片选 **PIN** 上发送指定个数的数据后，片选信号有效输出自动切换到下一个片选 **PIN** 上。

例如：

1. 使能 3 个 CS Pin，分别为 0,1,2，则起始 Pin 为 0，结束 Pin 为 2。
2. 设定 0,1,2 的发送数据个数分别为 2,3,1。
3. 设定 CS Pin 低电平有效。

发送数据时，CS 切换如下图所示。例如当 CS0 处发送 2 字节后，发送第 3 个字节时，CS0 失效，CS1 有效，模块自动完成此切换。后续数据以此类推。

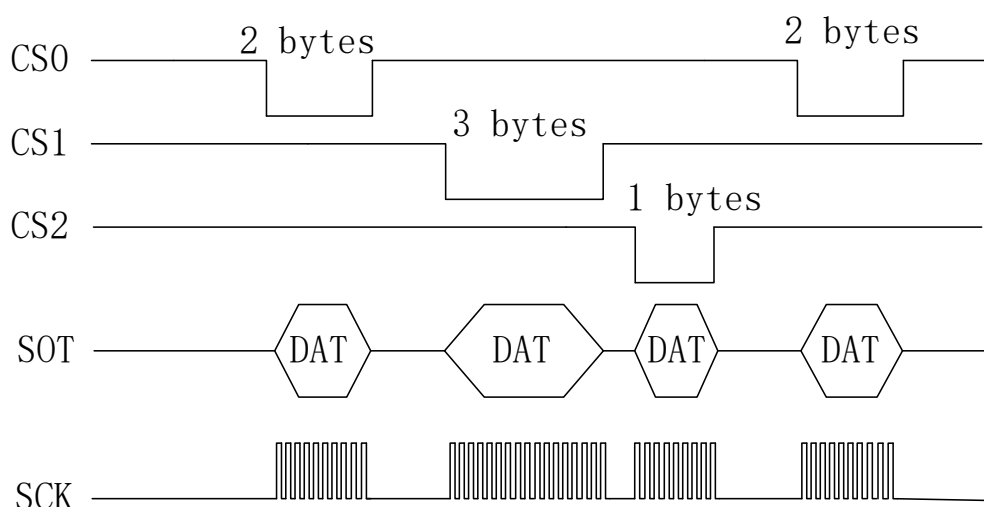


图 3 CS 片选切换示意图

片选发送数据设置：

对于主机模式，初始化设置如下：

- | | |
|---------------------------|--------|
| 1. 设置 CR.MSS = 0 | 主机模式 |
| 2. 设置 CR.SPIMODE, MR.CINV | 配置传输模式 |

- | | |
|------------------|---|
| 3. 设置 CR.DOE = 1 | 使能数据输出（对于接收数据，可关闭此位，只发送数据 SCK 信号，让从机发送数据） |
|------------------|---|

4. 其他配置项

例如发送数据位数，大小端等其他可选配置。

5. 配置 CS 功能

- | | |
|------------------|-----------------|
| a)设置 CSCR.CSAS | 片选起始 PIN |
| b) 设置 CSCR.CSAE | 片选结束 PIN |
| c)ECR.CSFE = 1 | 使能片选功能 |
| d) CSCR.CSTODIV2 | 控制片选的时钟分频 |
| e)CSCR.CSLVS | 片选空闲时电平 |
| f) CSEN 3-0 | 每个 CS Pin 的使能开关 |

其他配置:

- | | |
|------------------|---------------------------------------|
| g) CSTMG | 设定 CS 的保持 时间和等待时间 |
| h) CSFR | 每个 CS 的 CINV,SPIMODE, 数据长度等, 可覆盖之前的设置 |
| 6. 设置 TBYTE 3-0 | 每个通道发送数据的最大个数 |
| 7. 设置 CR.TXE = 1 | 使能发送 |

主机模式的发送数据流程如下:

- | | |
|------------------------|-----------------|
| 8. 等待 SR.TDEF = 1 | 发送寄存器为空, 可以发送数据 |
| 9. TXDR = data | 将数据写入发送寄存器 |
| 10. 下一次发送时, 重复 8-9 步骤。 | |

4 样例代码

4.1 代码介绍

用户可以根据上述的工作流程编写自己的代码来学习验证该模块，也可以直接通过华大半导体的网站下载 DDL 的样例代码并使用其中的 SPI 的 Example 进行验证。

以下部分简要介绍 spi_trx_polling 代码(example\spi\spi_trx_polling)的各个部分的功能：

1) SPI 的功能引脚配置：

```
Gpio_SetFunc_SCK1_0();
Gpio_SetFunc_SOT1_0();
Gpio_SetFunc_SCK4_0();
Gpio_SetFunc_SIN4_0();
```

2) 配置接收端口（按同样配置对发送端口进行设置，该处代码省略）：

```
stcConfigRx.bInvertClk = FALSE;
stcConfigRx.bMSB = TRUE;
stcConfigRx.bOutputData = TRUE;
stcConfigRx.bOutputSCK = FALSE;
stcConfigRx.bSOTHigh = FALSE;
stcConfigRx.bSPIMode = TRUE;
stcConfigRx.enDataLen = SpiDataLen8;
stcConfigRx.bSlaveMode = TRUE;
stcConfigRx.u32BaudRate = 100*1000;
stcConfigRx.u8DataOutputDly = 3;
```

3) 轮询发送判断：

```
while (FALSE == Spi_GetStatus(SPI_TX_CH,SpiTxEmpty));
Spi_SendData(SPI_TX_CH,u8TxBuff[u32offset]);
```

4) 轮询接收判断：

```
while (FALSE == Spi_GetStatus(SPI_RX_CH,SpiRxFull));
u8RxBuff[u32offset] = Spi_ReceiveData(SPI_RX_CH,FALSE);
```

5) 完成后处理：

```
Spi_DeInit(SPI_TX_CH);
Spi_DeInit(SPI_RX_CH);
```

通过以上代码即可完成 SPI 的发送和接收。

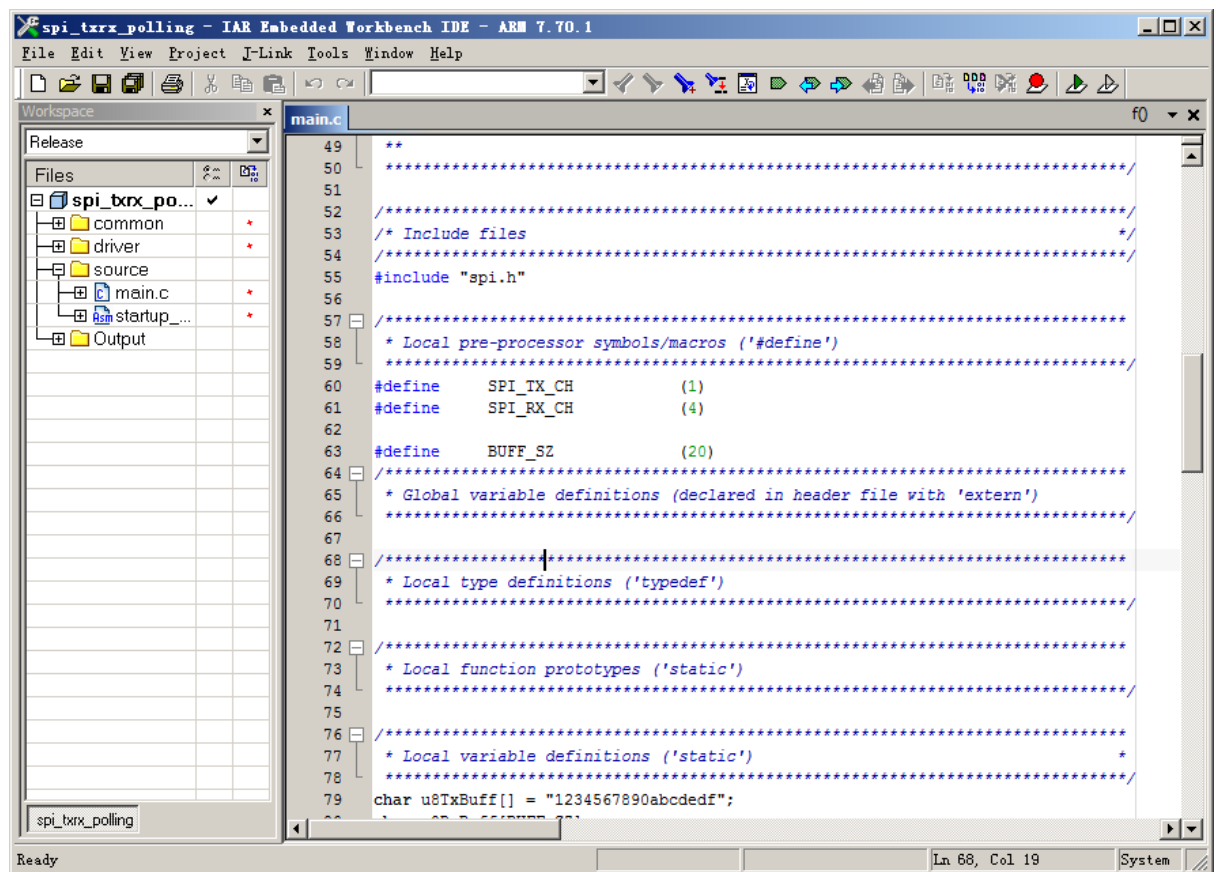
4.2 代码运行

用户可以通过华大半导体的网站下载到同步传输的样例代码，并配合学习板（比如‘SK-HC32L156-64L V10’）运行相关代码学习使用同步传输模块。

以下部分主要介绍如何在‘SK-HC32L156-64L V10’学习板上运行同步传输样例代码并观察结果：

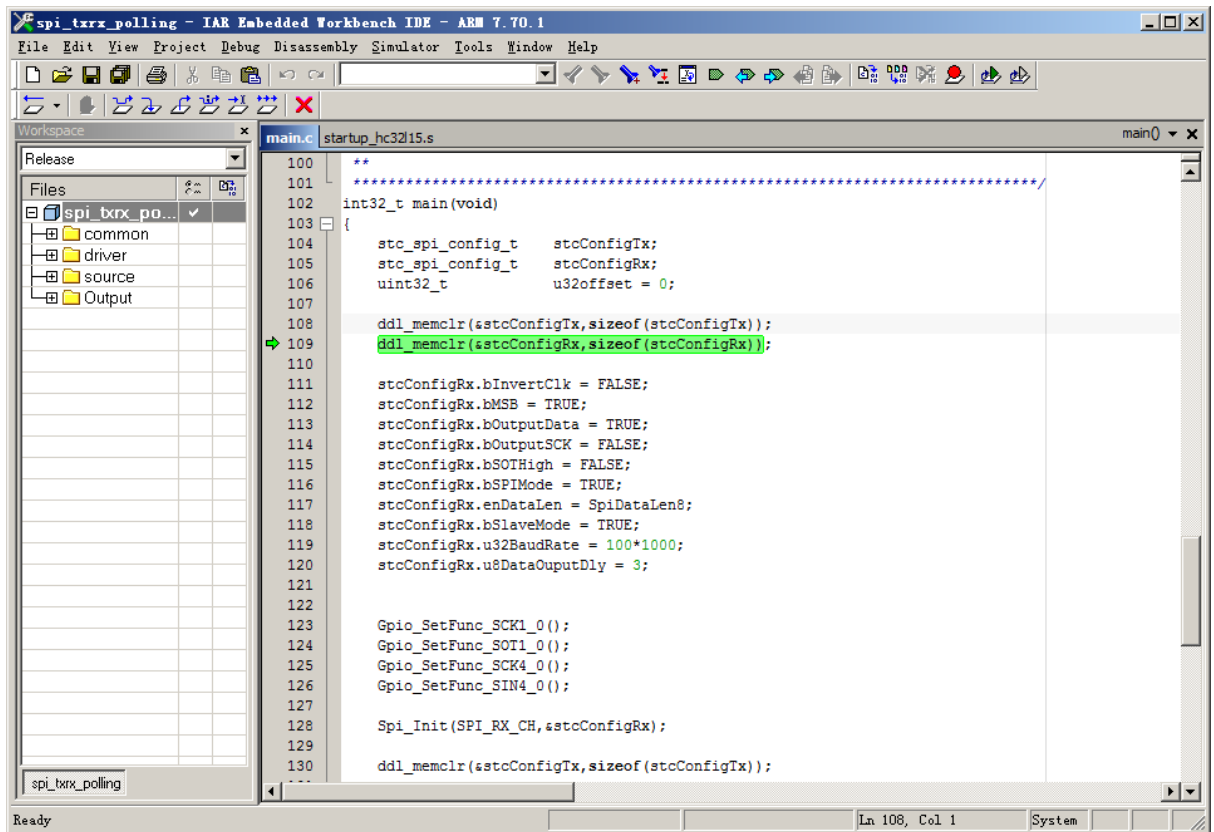
- 确认安装正确的 IAR EWARM V7.70 工具（请从 IAR 官网自行下载并安装）。
- 获取‘SK-HC32L156-64L V10’学习板。
- 从华大半导体网站下载 DDL 样例代码。
- 下载并运行样例代码（DDL→example\spi\spi_txrx_polling）：

1) 打开 Polling 项目，并打开‘main.c’如下视图：

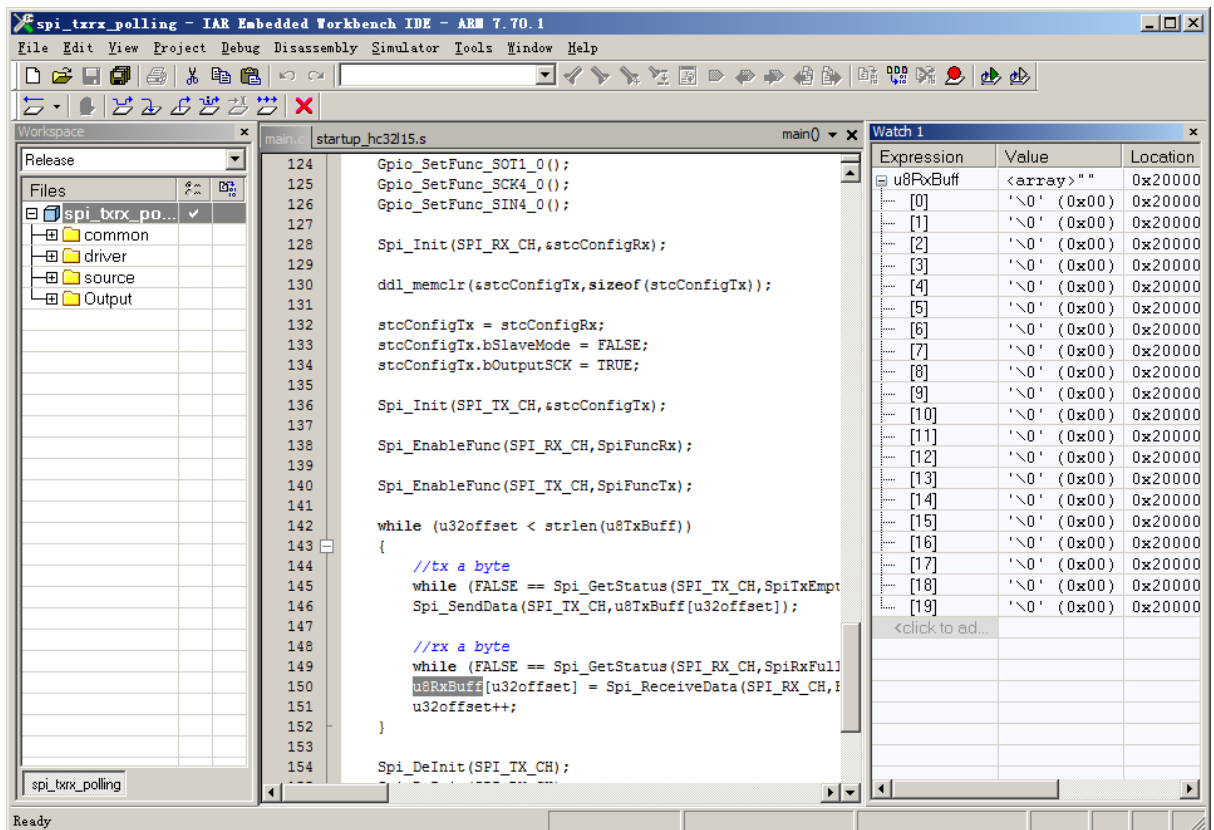


2) 点击  重新编译整个项目并将代码下载到学习板上。


3) 可以看见类似如下的视图:

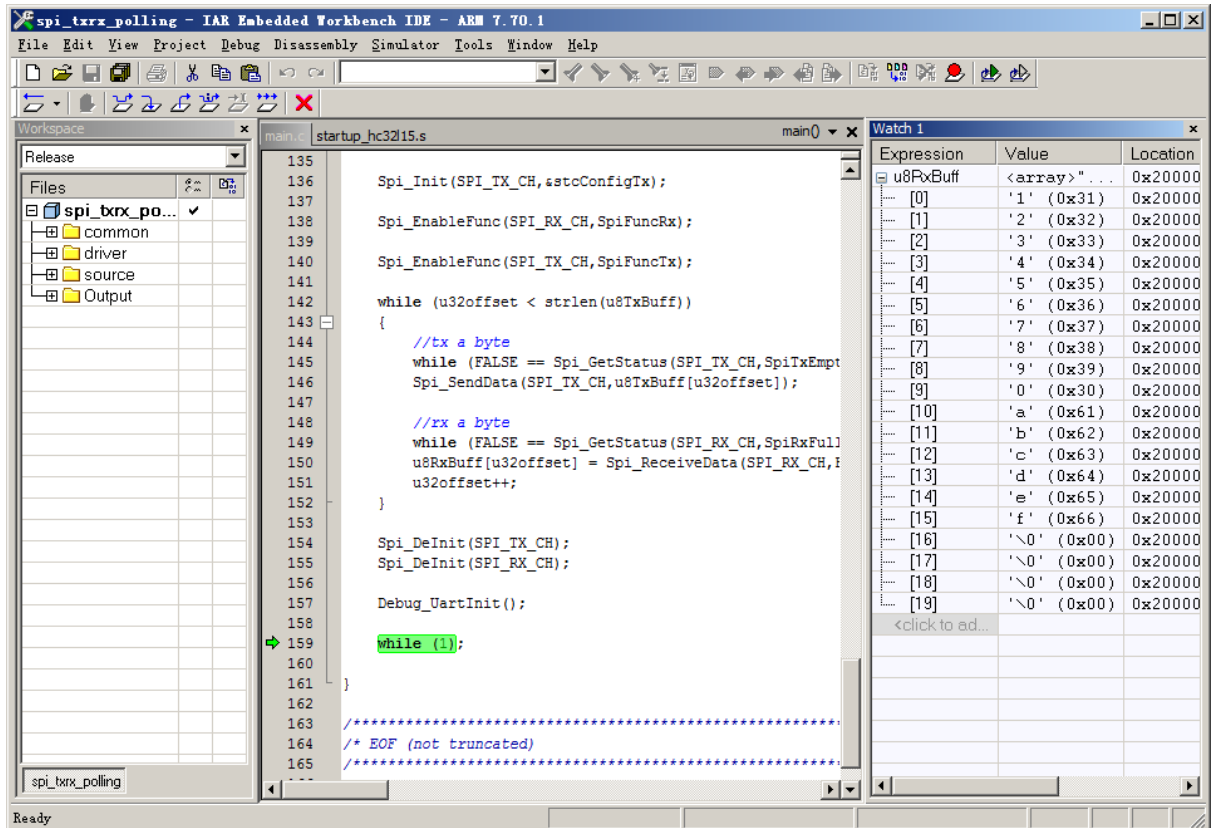


4) 打开 view->watch->watch1 窗口, 在 expression 中添加 `u8RxBuff` 来观测接收值。



5) 点击  运行。

6) 运行后按  停止，观察并记录 u8RxBuff 的值。



7) 运行完毕后可以关闭项目文件。

5 总结

以上章节简要介绍了 HC32L15 系列 MCU 的同步传输模块，详细说明了模块并且演示了如何使用相关的样例代码进行数据的收发，在开发中用户可以根据自己的实际需要使用该模块。

6 版本信息 & 联系方式

日期	版本	修改记录
2018/8/9	Rev1.0	初版发布。



如果您在购买与使用过程中有任何意见或建议，请随时与我们联系。

Email: mcu@hdsc.com.cn

网址: www.hdsc.com.cn

通信地址: 上海市张江高科园区碧波路 572 弄 39 号

邮编: 201203

