南京凌鸥创芯电子有限公司

LKS08x I2C 模块
应用笔记

目 录

1、I2C 概述

I2C 总线接口连接微控制器和串行 I2C 总线，多用于主控制器和从器件间的主从通讯，在小数据场合使用，传输距离短，任意时刻只能有一个主机等特性。I2C 总线提供多主机功能，控制所有 I2C 总线特定的时序、协议、仲裁和定时。支持标准和快速两种模式。根据特定设备的需要，可以使用 DMA 以减轻 MCU 的负担。

1.1、I2C 主要特性：

- 多主机功能：该模块既可做主设备也可做从设备。

  I2C 主设备功能：产生时钟、 START 和 STOP 事件。

  I2C 从设备功能：可编程的 I2C 硬件地址比较（仅支持 7 位硬件地址）、停止位检测。

- 根据系统分频，实现不同的通讯速度。

- 状态标志：发送器/接收器模式标志、字节发送结束标志、 I2C 总线忙标志。

- 错误标志：主模式时的仲裁丢失、地址/数据传输后的应答(ACK)错误、检测到错位的起始或停止条件。

- 一个中断向量，包含五个中断源：总线错误中断源、完成中断源、 NACK 中断源、硬件地址匹配中断源和传输完成中断源。

- 具单字节缓冲器的 DMA。

2、I2C 功能模式

I2C 模块接收和发送数据，并将数据从串行转换成并行，或并行转换成串行。可以开启或禁止中断。接口通过数据引脚(SDA)和时钟引脚(SCL)连接到 I2C 总线，与外界通讯只需要两根信号线。可以选择运行在主或从模式。

2.1、功能框图

本接口采用同步串行设计，实现 MCU 同外部设备之间的 I2C 传输。支持轮询和中断方式获得传输状态信息。本接口的主要功能模块如下图 1 所示。

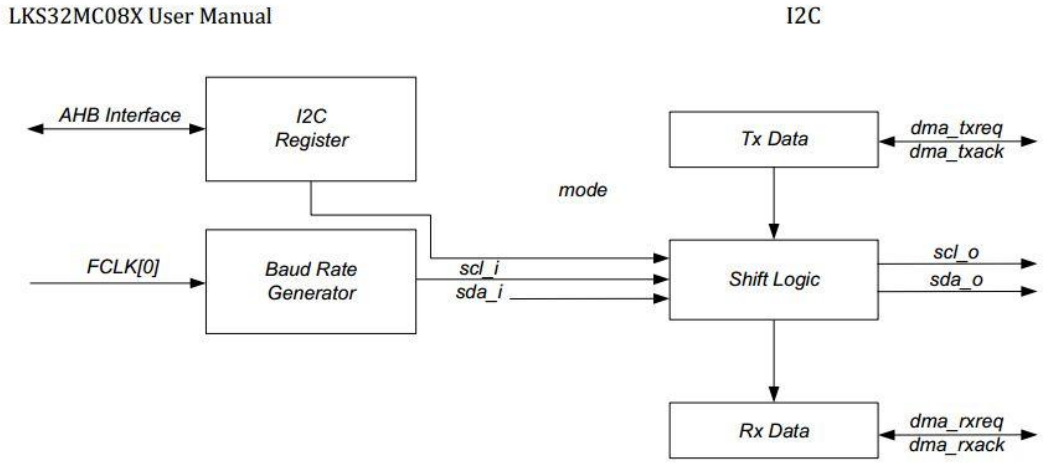


图 1、I2C 模块顶层功能框图

I2C 接口同外界通讯只有 SCL 和 SDA 两根信号线。SDA 为双向复用信号线，受到 sda_oe 控制。模块级，I2C 接口信号包括，scl_i，sda_i， scl_o， sda_o 和 sda_oe。I2C 接口信号共 5 个分别为：

scl_i：时钟信号。当 I2C 接口配置为从模式时，此为 I2C 总线的时钟输入信号。

sda_i：数据信号。当 I2C 接口接收数据时（无论主模式还是从模式），此为 I2C 总线的数据输入信号。

scl_o：时钟信号。当 I2C 接口配置为主模式时，此为 I2C 总线的时钟输出信号。

sda_o：数据信号。当 I2C 接口发送数据时（无论主模式还是从模式），此为 I2C 总线的数据输出信号。

sda_oe：数据使能信号。当 sda_o 输出时，sda_oe 有效；当 sda_i 输入时，sda_oe 无效。

2.2、I2C 模式选择

I2C 接口默认主从均不使能。接口根据配置情况，进入主模式或者从模式。当仲裁丢失或产生停止信号时，主模式自动释放总线并产生相应异常中断。允许多主机功能。I2C 接口实现发送和接收功能，因此结合主从选择，共可以分成四

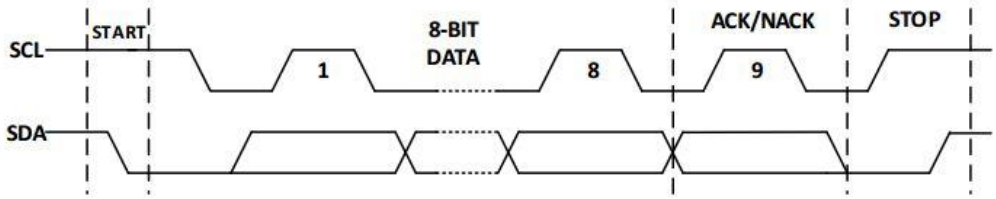
模式选择，分别是：从发送模式，从接收模式，主发送模式，主接收模式。

主模式时，I2C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。起始条件和停止条件都是在主模式下由软件控制产生。

从模式时， I2C 接口能识别它自己的地址(7 位)。软件能够控制开启或禁止硬件地址比较功能，硬件地址比较功能可降低 MCU 的负担。只有地址匹配才通知 MCU 进行相关处理。

I2C 接口没有 FIFO，若一次性发送大量数据，为了降低 MCU 的负担，需 DMA 配合。 I2C 接口支持 DMA 传输（多字节传输）和非 DMA 传输（单字节传输）。上述四种传输模式进一步扩展为：从模式单字节发送，从模式 DMA 发送；从模式单字节接收，从模式 DMA 接收；主模式单字节发送，主模式 DMA 发送；主模式单字节接收，主模式 DMA 接收。

数据和地址按 8 位/字节进行传输，高位在前。跟在起始条件后的 1 个字节是地址。地址只在主模式发送。在一个字节传输的 8 个时钟后的第 9 个时钟期间，接收器必须回送一个应答位(ACK)给发送器。软件可以开启或禁止应答(ACK)，并可以设置 I2C 接口的地址。基本的 I2C 传输时序图如下：



基本的 I2C 传输时序图

一般情况下，非 DMA 方式时，一次传输一个字节（可反复单次传输，需软件介入提供数据）。DMA 方式，一次连续传输可以多字节（最大不超过 16 字节，极端情况一次传输一个字节，因无 FIFO，每次 DMA 请求，仅传输一个字节，多轮完成本次数据传输）。

上述所有模式，遵循如下基本原则：

- 单字节发送，中断将在 8-bit 数据发送完毕且收到响应后（ACK/NACK 均可）产生。
- 单字节接收，中断将在 8-bit 数据接收完毕后产生。
- DMA 发送，正常情况下，中断将在数据发送完毕且收到响应后(ACK/NACK 均可)产生。
- DMA 接收，正常情况下，中断将在数据接收完毕后产生。
- 当 I2C 接口配置为主模式时，检测到错误后， I2C 接口会主动释放总线，恢复到起始状态并产生中断信号。

2.3、I2C 接口从模式

默认情况下，I2C 接口主模式和从模式均关闭。若工作在从模式，需使能从模式。为了产生正确的时序，必须通过系统寄存器 SYS_CLK_DIV0 设定 I2C 接口的工作时钟频率，I2C 接口时钟基于系统高速主时钟进行分频，SYS_CLK_DIV0 是 I2C 接口工作时钟的分频系数。

- 从模式下，I2C 接口时刻在监控总线上的信号。一旦检测到起始条件，其将保存地址位数据和读写位数据。
- 从模式下，若硬件地址匹配功能开启，只有地址匹配的情况下，才会产生中断，通知 MCU 进行后续处理。若没有开启，每次收到地址及读写位数据，都将产生中断。
- 从模式下，单字节接收模式。每次收到一个字节的数据后，产生中断，此时 I2C 接口可拉低 SCL，直至中断完成，继续后续操作。
- 从模式下，单字节发送模式。每次发送一个字节完毕后且收到响应（ACK/NACK），产生中断，此时 I2C 接口可拉低 SCL，直至中断完成，继续后续操作。
- 从模式下，DMA 接收模式。每次收到 SIZE 约定后的数据，产生中断，此时 I2C 接口可拉低 SCL，直至中断完成。
- 从模式下，DMA 发送模式。每次发送 SIZE 约定后的数据并收到响应(ACK/NACK)，产生中断，此时 I2C 接口可拉低 SCL，直至中断完成。其中上述从模式下的单字节发送和接收，DMA 发送和接收模式的具体传输流程，请参见《LKS32MC08x_User_Manual_v1.16》中 16.3.2.2 章节 I2C 接口从模式。

2.3.1、从模式 DMA 传输

DMA 传输，其含义为每次传输多个字节的数据后，将产生中断判断是否还要继续传输。 DMA 传输的极端情况是仅传输一个字节的数据。 DMA 传输，一般建议开启硬件地址比较功能，NACK 中断，传输完成中断。一般 DMA 传输的流程如下：

1、配置 I2C 从地址，使能 I2C 中断(可使能硬件地址比较中断)。地址匹配，产生 I2C 地址匹配中断， 在中断处理函数中，配置 DMA，准备好发送数据或者准备好接收地址。然后写 I2C_SCR，准备开始传输或者停止本次传输。

2、若是接收模式，I2C_BCR.BURST_SIZE 约定的字节接收完毕后，产生中断，软件判断是否继续接收，返回 ACK/NACK 响应。

3、若是发送模式， I2C_BCR.BURST_SIZE 约定的字节发送完毕后，等待响应（ACK/NACK），产生中断，根据响应判断后续操作。

4、 获得总线完成标志，本次传输完成。

I2C 模块从模式下 DMA 发送， 基本传输过程：

1、配置 I2C 模块时钟(SYS_CLK_FEN&SYS_CLK_DIV0)，从模块的硬件地址 I2C_ADDR 、开启硬件地址比较(I2C_ADDR[7])， I2C 相关中断（I2C_CFG&I2C_BCR）， 特别是硬件地址比较中断需开启）。

2、触发 START（I2C_MSCR[0]），开始传输地址+读。 若从设备地址匹配上， 产生中断。从设备若没准备好数据， 配置 SCR[4]为 0， 返回 NACK。 若从设备准备好要发送的数据， 配置 DMA， BCR[4]为 1（软件协助硬件预取第一发送的字节）， 配置好 SCR[4]为 1（接收主设备请求），传输方向 SCR[2]为 1（发送）。 开始传输。

3、发送完毕本批次数据， 产生完成中断。

4、接收 STOP， 完成本次传输。 无论是否响应本次传输， 主设备均要发出 STOP。

从模式下 DMA 发送。 一般， 从设备准备好接收 START 即可。 硬件地址匹配成功后， 根据主设备的需求， 决定从设备是接收还是发送。 若是发送，从设备也需要预取，此时可以通过软件协助硬件完成预取第一个发送的数据。若从设备暂时无法实现传输， 返回 NACK 结束本次传输。

从模式下 DMA 发送。 DMA 必定先把最后一个字节发给从设备， 然后从设备发送给主设备。那么，传输结束的标志是 I2C 传输完毕。 因此，I2C 的 STOP 中断可作为本次传输的结束标志。

I2C 模块从模式下 DMA 接收，基本传输过程：

1、配置 I2C 模块时钟（SYS_CLK_FEN&SYS_CLK_DIV0）， 从模块的硬件地址（I2C_ADDR）、 开启硬件地址比较（I2C_ADDR[7]），I2C 相关中断（I2C_CFG&I2C_BCR， 特别是硬件地址比较中断需开启）。

2、触发 START（I2C_MSCR[0]），开始传输地址+写。若从设备地址匹配上，产生中断。从设备若没准备好接收， 配置 SCR[4]为 0， 返回 NACK。 若从设备准备好，配置 DMA，配置好 SCR[4]为 1(接收主设备请求),传输方向 SCR[2]为 0（接收）。 开始传输。

3、接收完毕本批次数据， 产生完成中断（SCR[0]会被置 1）。

4、接收 STOP，完成本次传输。 无论是否响应本次传输， 主设备均要发出 STOP。

从模式下 DMA 接收。 一般， 从设备准备好接收 START 即可。 硬件地址匹配成功后，根据主设备的需求，决定从设备是接收还是发送。 若是接收， 配置好 DMA 等即可。若从设备暂时无法实现传输，返回 NACK 结束本次传输。

从模式下 DMA 接收。从设备接收主设备的数据， 然后 DMA 才开始搬移数据。 那么，传输结束的标志是 DMA 传输完毕。 因此， DMA 中断可作为本次传输的结束标志。 因为， DMA 传输一个字节耗时很短，且跟随在 I2C 的 STOP 中断后面。方便软件统一处理， 可以用 I2C 的 STOP 中断做完成标志，但建议在 I2C 的 STOP 中断函数中，检测 DMA 的完成标志位.

2.4、I2C 接口主模式

默认情况下，I2C 接口主模式和从模式均关闭。若工作在主模式，需使能主模式。为了产生正确的时序，必须在系统寄存器 CLK_DIV0 中设定 I2C 接口的工作时钟。I2C 接口执行主模式传输之前，需要判断总线是否空闲。可读取 I2C_MSCR 寄存器的 BIT3，查询当前总线状态。若总线处于忙的状态，可以开启 I2C 中断，通过收到 STOP 中断事件判断总线是否空闲下来。只有空闲状态下，才能正常发送 START 状态，以及后续的数据。

主模式单字节传输，主模式单字节发送，主模式单字节接收的详细介绍，请参阅《LKS32MC08x_User_Manual_v1.16》中 16.3.2.4 章节 I2C 接口主模式。

2.4.1、主模式 DMA 传输

主模式 DMA 传输发送的基本传输过程如下：

- 配置 I2C 模块时钟（SYS_CLK_FEN&SYS_CLK_DIV0），从模块的硬件地址（I2C_ADDR）、读写控制（I2C_SCR[2]）， 传输控制（SCR[2]），I2C 相关中断（I2C_CFG&I2C_BCR），DMA 传输使能（I2C_BCR[5]），数据大小（I2C_BCR[3:0]） 以及 DMA 寄存器。
- 触发 START（I2C_MSCR[0]），开始传输地址+写。 若返回 ACK，表明此时从设备响应本次传输； 若返回 NACK， 表明无对应地址的从设备/从设备没有准备好，无法完成本次传输。
- 响应本次传输时，开始发送数据。每发送一个字节，等待从设备反馈。若是 ACK，表明可以继续传输；若是 NACK，表明从无法继续接收。此时，将产生 NACK 中断，根据 SCR[0]可以判断本批次数据传输是否完成，同时也可以检查 DMA 寄存器的值判断。
- 触发 STOP，完成本次传输。 无论是否响应本次传输， 主设备均要发出 STOP。

主模式下 DMA 发送。 触发 START， 无论从设备的状态如何， 硬件都将预取第一个字节。 若从设备无法完成本次传输， 那么主设备需停止本次传输，同时关闭对应 DMA 通道并重新开启， 其它 DMA 配置也需重新配置。

主模式下 DMA 发送。DMA 必定先把最后一个字节发给主设备， 然后主设备发送给从设备。那么，传输结束的标志是 I2C 传输完毕。因此，I2C 的 STOP 中断可作为本次传输的结束标志。

主模式 DMA 传输接收的基本传输过程如下：

- 配置 I2C 模块时钟（SYS_CLK_FEN&SYS_CLK_DIV0），从模块的硬件地址（I2C_ADDR）、读写控制（I2C_SCR[2]），传输控制（SCR[2]），I2C 相关中断（I2C_CFG&I2C_BCR），DMA 传输使能（I2C_BCR[5]），反馈控制（SCR[4]），数据大小（I2C_BCR[3:0]）以及 DMA 寄存器。

- 触发 START（I2C_MSCR[0]），开始传输地址+读。若返回 ACK，表明此时从设备响应本次传输；若返回 NACK，表明无对应地址的从设备/从设备没有准备好，无法完成本次传输。
- 响应本次传输时，开始接收数据。 每接收一个字节，主设备将根据 SCR[4]的设置（此时都应是 ACK），反馈给从设备。 完全接收完毕后。 产生完成中断。
- 触发 STOP，完成本次传输。无论是否响应本次传输，主设备均要发出 STOP。

主模式下 DMA 接收。 触发 START，若从设备无响应。主设备不会发送 DMA 预取。后续，主设备可再次触发 START，无需重置 DMA 寄存器。

主模式下 DMA 接收。 主设备接收从设备的数据， 然后 DMA 才开始搬移数据。 那么，传输结束的标志是 DMA 传输完毕。 因此，DMA 中断可作为本次传输的结束标志。因为，DMA 传输一个字节耗时很短，且跟随在 I2C 的 STOP 中断后面。方便软件统一处理，可以用 I2C 的 STOP 中断做完成标志， 但建议在 I2C 的 STOP 中断函数中，检测 DMA 的完成标志位。

2.5、I2C 总线异常处理

本 I2C 接口。主模式下，总线错误可被检测到同时总线错误中断也会产生；从模式下，总线错误将触发地址数据被接收，同时让 I2C 接口恢复空闲状态并产生中断。

2.6、DMA 传输软件配置流程

因 I2C 接口支持 DMA 传输，也支持 MCU 传输。两者区别在于 DMA 传输，发送的数据来自 DMA 的搬移；MCU 传输，发送的数据来自 MCU 的搬移。

DMA 传输，推荐软件配置流程如下：

1、初始化 DMA 模块，将本次发送的数据来源，接收的数据去向配置好， 传输长度配置完毕。

2、初始化 GPIO 模块，将 I2C 复用的 GPIO 配置完毕。

3、初始化 I2C 接口，I2C_CFG/I2C_BCR 等寄存器配置完毕。

4、主模式下，触发 I2C 接口，进入发送状态；从模式下，等待主发送传输请求。

2.7、MCU 传输软件配置流程

MCU 传输，一次只能发送/接收 1 个字节，每次完成后需要通过中断或者轮询的方式判断传输是否完成。

MCU 传输，推荐软件配置流程如下：

1、初始化 GPIO 模块，将 I2C 复用的 GPIO 配置完毕。

2、初始化 I2C 接口，IE/CFG 等寄存器配置完毕。

3、 MCU 触发 I2C 接口进入发送流程，发送的数据来自 MCU 对 I2C_DATA 写入值。

2.8、中断处理

I2C 接口包含三种类型的中断事件，分别是：数据传输完成事件，总线错误事件、 STOP 事件、NACK 事件和硬件地址匹配事件。

- 数据完成事件。当前数据传输完成，高电平有效，对 I2C_SCR 的 BIT0 写 0 清除。
- 总线错误事件。传输过程中，总线产生错误的 START 事件/STOP 事件，高电平有效，对 I2C_SCR 的 BIT7 写 0 清除。
- STOP 事件。当前数据传输完成，主设备发送 STOP 事件，从设备收到 STOP 事件并产生相应中断。高电平有效，对 I2C_SCR 的 BIT5 写 0 清除。
- NACK 事件。发送端接收到 NACK 响应，表明接收端无法继续后续传输。高电平有效，对 I2C_SCR 的 BIT1 写 0 清除。
- 硬件地址匹配事件。从模式下接收到的地址同本设备地址匹配，产生相应中断。高电平有效，对 I2C_SCR 的 BIT3 写 0 清除。
- 使用 DMA 协助数据传输。若本模块为接收模式，I2C 收到数据后还需要通过 DMA 搬移到 RAM，此时数据最终完成是要看 DMA 是否搬移完成，若使用 I2C 的完成中断作为判断依据的话，推荐在中断处理函数中查询下 DMA 的状态。若本模块为发送模式，无此问题。直接使用 I2C 的完成中断作为判断依据即可。

2.9、通讯速度设置

I2C 接口的工作时钟来自系统时钟的分频，分频寄存器 SYS 模块的 CLK_DIV0。I2C 接口采用同步设计，需要对外部设备的信号进行同步采样，同步时钟为 I2C 接口工作时钟。数据和时钟信号的时钟频率为接口工作时钟/16。

- I2C 模块工作时钟频率= 系统频率/ (CLK_DIV0 + 1)。
- I2C 波特率 = I2C 模块工作时钟频率/ 17。

3、相关寄存器

地址分配：

I2C 模块寄存器的基地址是：0x4001_1400，寄存器列表如下：

名称	偏移	说明
I2C0_ADDR	0x00	I2C 地址寄存器
I2C0_CFG	0x04	I2C 配置寄存器
I2C0_SCR	0x08	I2C 状态寄存器
I2C0_DATA	0x0C	I2C 数据寄存器
I2C0_MSCR	0x10	I2C 主模式寄存器
I2C0_BCR	0x14	I2C DMA 传输控制寄存器

详细寄存器信息请参阅《LKS32MC08x_User_Manual_v1.16》16. 4 寄存器。

4、I2C 应用举例

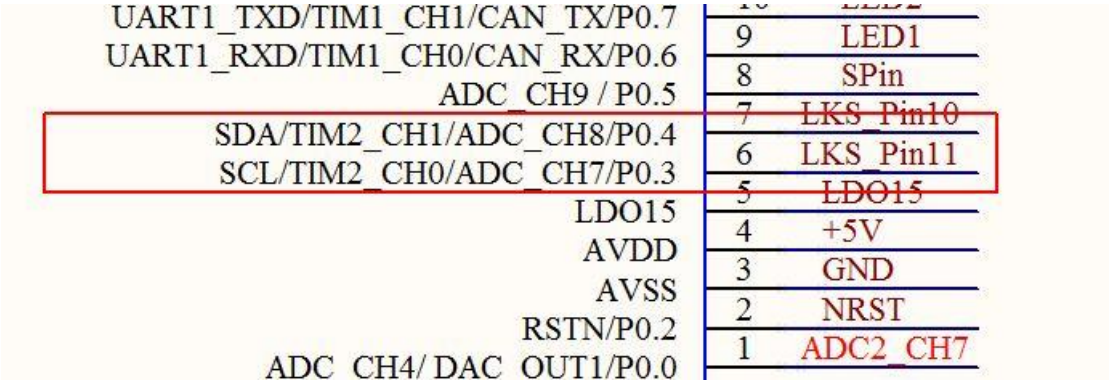
例程实现一个 I2C 主机发送 MDA 传输数据的过程，运行程序观察 SYS_AFE_REG0(地址：0x400000020 配置寄存器 0 的值，若 0xDADE 则表示 OK 标志，若 0xFA11 则表示 NOTOK 标志。本例使用的硬件平台是 LKS_DEMO_081_V2.0 电路板。

4.1、I2C 硬件接口配置

芯片 GPIO 一般有多组可以复用为 SPI 功能的接口，可根据实际需求自行分配，本例选择的 I2C 功能接口如下表：

GPIO　PIN 脚	复用 SPI 功能接口	说明
P0.3	SDA	I2C 时钟信号
P0.4	SCL	I2C 数据信号

对应原理图如下所示：



4.2、I2C 初始化及发送数据

本例包括程文件中的 lks32mc08x_i2c.h 中定义了 I2C 初始化相关的结构体，包括 I2C_TypeDef 寄存器结构体和 I2C_InitTypeDef 结构体，I2C_InitTypeDef 在初始化 I2C 结构体函数 I2C_StructInit(I2C_InitTypeDef* I2C_InitStruct)中进行展示。对应的程序及注释如下图：

lks32mc08x_i2c.c | test_i2c.c | lks32mc08x_i2c.h | main.c | hardware_config.h | hardware

```
32 #include "basic.h"
33
34 typedef struct
35 {
36     __IO uint32_t ADDR;   //I2C地址寄存器
37     __IO uint32_t CFG;    //I2C配置寄存器
38     __IO uint32_t SCR;    //I2C状态寄存器
39     __IO uint32_t DATA;   //I2C数据寄存器
40     __IO uint32_t MSCR;   //I2C主模式寄存器
41     __IO uint32_t BCR;    //I2C DMA传输控制寄存器
42 }I2C_TypeDef;
```

lks32mc08x_i2c.c | test_i2c.c | lks32mc08x_i2c.h | main.c | hardware_config.h | hardware_init.c

```
78  void I2C_StructInit(I2C_InitTypeDef* I2C_InitStruct)
79  {
80      //I2C硬件地址比较使能开关，只有在DMA有效；关闭
81      I2C_InitStruct->ADRCMP = 0;
82      //从模式下，I2C设备硬件地址
83      I2C_InitStruct->ADDR   = 0;
84
85      //I2C中断使能信号；禁止
86      I2C_InitStruct->INTEN      = 0;
87      //I2C数据传输完成中断使能信号；禁止
88      I2C_InitStruct->DONEINT    = 0;
89      //I2C总线错误事件中断使能信号；屏蔽
90      I2C_InitStruct->BUSERRINT = 0;
91     //I2C STOP事件中断使能信号；屏蔽
92      I2C_InitStruct->STOPINT    = 0;
93     //I2C主模式使能信号；关闭
94      I2C_InitStruct->MASTER     = 0;
95     //I2C从模式使能信号；关闭
96      I2C_InitStruct->SLAVE      = 0;
97
98      //I2C 总线错误状态标志位，仅用于主模式下
99      I2C_InitStruct->BUSERRFLAG     = 0;
100     //总线仲裁丢失状态标志位，仅用于主模式下，发生丢失此位置1
101     I2C_InitStruct->LOSTARBFLAG    = 0;
102     //stop 事件状态标志位
103     I2C_InitStruct->STOPFLAG       = 0;
104     //ACK发生控制位，主从模式均可使用；0：字节发送完，返回NACK回应
105     I2C_InitStruct->ACKCTRL        = 0;
106     //Address数据标志位，主从模式均可使用
107     I2C_InitStruct->ADDRFLAG       = 0;
```

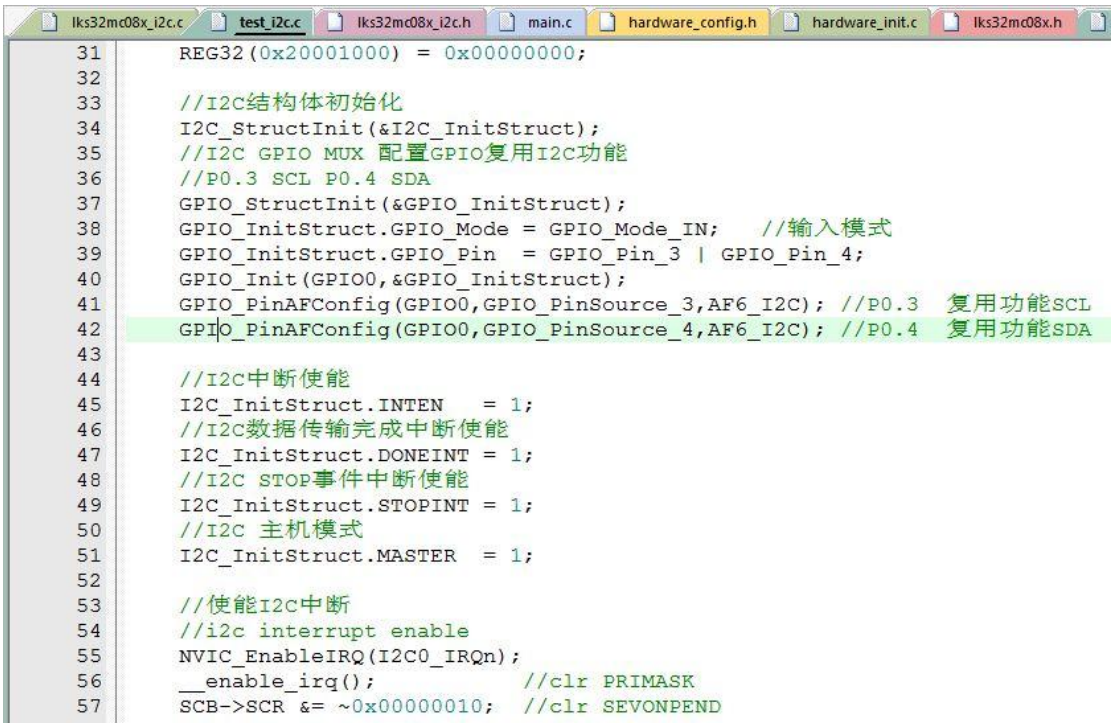
lks32mc08x_i2c.c | test_i2c.c | lks32mc08x_i2c.h | main.c | hardware_config.h | hardware_init.c | lks32mc08x.h | lks32mc08x_

```
107     I2C_InitStruct->ADDRFLAG       = 0;
108     //发送或者接收控制位，主从模式均可使用；0：接收
109     I2C_InitStruct->DIR            = 0;
110     //接收响应标志位，主从模式均可使用；0：本I2C接口发送数据，接收到ACK回应
111     I2C_InitStruct->RESPFLAG       = 0;
112     //传输完成状态标志位，主从模式均可使用；0：传输未完成
113     I2C_InitStruct->TRANSDONEFLAG = 0;
114
115     //I2C总线，闲忙状态；0：检测到STOP事件，空闲
116     I2C_InitStruct->BUSY       = 0;
117     //主模式争抢总线标志位。抢到总线，置1；释放总线，置0
118     I2C_InitStruct->ARBSTATUS = 0;
119     //再次触发START事件，写1有效。 发送START完毕，硬件清0,I2C_CFG[1]置1，才能实现写1操作。
120     I2C_InitStruct->RESTART    = 0;
121     //触发START事件并发送地址数据线至总线，写1有效。 I2C_CFG[1]置1，才能实现写1操作。
122     I2C_InitStruct->START      = 0;
123
124     //I2C传输，NACK事件中断使能信号；0：屏蔽
125     I2C_InitStruct->NACKINT     = 0;
126     //I2C传输，硬件地址匹配中断使能信号；0：屏蔽
127     I2C_InitStruct->ADRCMPINT   = 0;
128     //I2C多数据传输使能，需要采用DMA方式；0：关闭
129     I2C_InitStruct->DMA         = 0;
130     //I2C多数据传输.从模式执行DMA方式发送，触发硬件预取第一个字节。硬件自动清零。0：关闭
131     I2C_InitStruct->PREFETCH    = 0;
132     //I2C数据传输长度寄存器，用于多字节传输。实际传输字节数=B[3:0]+1.
133     I2C_InitStruct->ByteLength = 0;
134 }
135
```

本例的主要程序在 test_i2c.c 文件中的 i2c_master_tx_dma()函数中，I2C 具体配置步骤如下：

1、首先打开系统写保护，配置系统时钟，使能 I2C 时钟，配置 I2C 时钟分频。

2、复用 GPIO 配置成 I2C 功能接口，对应 4.1 节设置的硬件接口,然后使能 I2C 中断，使能传输完成事件、STOP 事件中断；设置为主机模式。具体配置如下图：



3、burst config 配置涉及从模式下，I2C 设备硬件地址、I2C 配置成发送模式，NACK 事件中断，使能 DMA 传输方式，确定实际传输字节数。具体如下图：

```
58
59      //burst config
60      //从模式下，I2C设备硬件地址
61      I2C_InitStruct.ADDR = 0x19;//I2C Slave Address is 0x19
62      //发送或者接收控制位，1：发送
63      I2C_InitStruct.DIR  = 1;   //Transmit
64      //NACK事件中断使能信号 使能
65      I2C_InitStruct.NACKINT = 1;
66      //I2C多数据传输使能，需要采用DMA方式：1：使能
67      I2C_InitStruct.DMA     = 1;
68      //I2C多数据传输长度寄存器，用于多字节传输，实际传输字节数=B[3:0]+1=16
69      I2C_InitStruct.ByteLength = 0x0f;
70      I2C_Init(I2C,  &I2C_InitStruct);
```

4、DMA 寄存器相关配置，以及配置 I2C->MSCR=1；触发 START 事件并发送地址数据至总线，开始传输数据。具体配置如下图：

```
72      // DMA config
73      // use DMA to write data into I2C FIFO
74      //DMA通道3采样轮数为16  每轮数据搬运1次
75      DMA_CTMS3          = 0x00100001;
76      //DMA通道3 外设地址
77      DMA_CPAR3          = 0x4001040C;
78      //DMA通道3 内存地址
79      DMA_CMAR3          = ( UINT32 )i2c_txdma;
80      // enable chn2 at last, or chn regs cannot be written
81      // MINC = 0, PINC = 0
82      //1、通道3使能，2：传输完成中断使能；3、外设访问位宽half-word;
83      //4、内存访问位宽word;5、通道3三个硬件DMA请求使能
84      DMA_CCR3           = 0x00004093;
85      // enable DMA, CPU must has higher priority
86      //DMA使能
87      DMA_CTRL           = 0x00000001;
88      //触发START事件并发送地址数据至总线，写1有效。
89      //I2C_CFG[1]置1，才能实现写1操作
90      I2C->MSCR          = 0x00000001;
91      while(1){if(REG32(0x20001000) == 0x00112233)break;}
92      OK;
93  }
94
```

4.2、I2C 中断服务子程序

中断服务子程序中根据中断标志位进行情况，判断 I2C 传输的状态，其中当有 STOP 事件触发中断时候，设置 REG32(0x20001000) = 0x00112233;从而使 i2c_master_tx_dma()函数中 while(1){if(REG32(0x20001000) == 0x00112233)break;} 跳出死循环，测试程序执行 OK 状态的配置。其中 OK 宏定义如下：

#define OK　　SYS_PROT = PSW_PROT, SIMDONE = 0xDADE

#define NOTOK　　SYS_PROT = PSW_PROT, SIMDONE = 0xFA11。

I2C 中断服务子程序如下图，此处仅显示了本例主机发送 DMA 传输对应的函数部分：

interrupt.c | test_i2c.h | basic.h | main.c | hardware_config.h | hardware_init.c | lks32mc08x.h | lks32mc08x_cmp.

```
73  /*I2C中断服务子函数*/
74  void I2C0_IRQHandler(void)
75  {
76      if ((I2C->CFG &0x02) && (I2C->BCR & 0x20))
77    {    //主模式 采用DMA传输方式时候
78      switch (I2C->SCR){
79
80        case 0x01://byte complete  传输已完成
81                  I2C->SCR       = 0x00;//NACK means i2c will receive last byte, then flow done
82                  break;
83
84        case 0x05://transmit and byte complete 触发发送，传输已完成。
85                  I2C->SCR     = 0x00;
86                  break;
87
88        case 0x06://transmit and nack  触发发送，接收到NACK响应
89                  I2C->SCR     = 0x00;
90                  break;
91
92        case 0x20://stop;        有STOP事件
93                  I2C->SCR = 0x00;
94                  REG32(0x20001000) = 0x00112233;
95                  break;
96        default:
97                  I2C->SCR = 0x00;
98                  NOTOK;
99                  break;
100     }
101   }
```

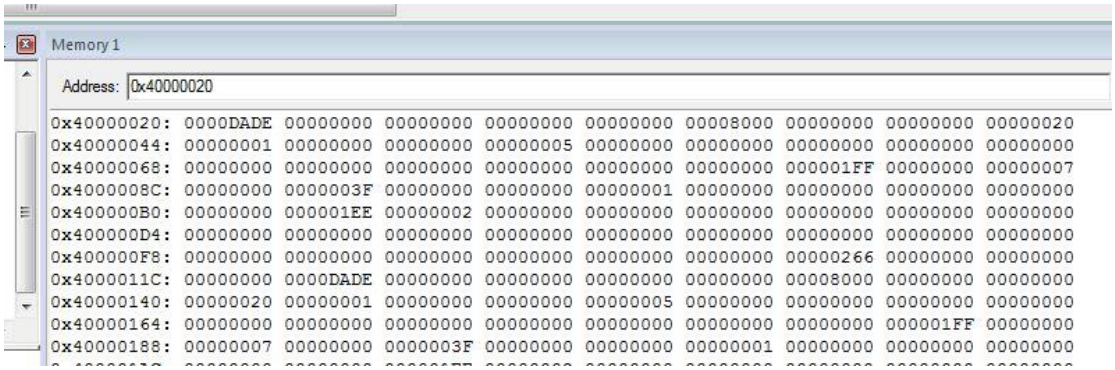
4.3、程序运行结果

对于本例 I2C 主机发送 DMA 传输方式。软件调试，观察 0x40000020 寄存器的数值是否出现 OK 对应的 0xDADE。

OK: 配置 SYS_AFE_REG0(地址：0x40000020) 模拟配置寄存器 0 =0xDADE

NOTOK: 配置 SYS_AFE_REG0(地址：0x40000020) 模拟配置寄存器 0 =0xFA11

运行的一组结果如下图：显示 0000DADE



5、主要相关文档

1、LKS32MC081C8T8 数据手册 V1.8
2、LKS32MC08x_User_Manual_v1.16
3、LKS08X I2C 模块采用 DMA 传输使用技巧