ST7 ROUTINE FOR I2C SLAVE MODE MANAGEMENT

by Microcontroller Division Application Team

INTRODUCTION

The goal of this application note is to present a useful example of communication using the I2C peripheral of the ST7. The ST7 microcontroller is used as a slave and can communicate with any master. This slave, through the I2C interface, receives data bytes from the master implementing error management and returns them. This application has been realized with a ST72264,7-bit, general call and 10-bit addressing mode.

1 ST7 I2C INTERFACE

The ST7 I2C peripheral allows multi master and slave communication with bus error management. In this application, only the single slave mode is used (with error management).

The I2C synchronous communication needs only two signals: SCL (Serial clock line) and SDA (Serial data line). The corresponding port pins have to be configured as floating inputs (here PA4 and PA6).

Please refer to the ST7 datasheet for more details.

1.1 COMMUNICATION SPEED

The ST7 I2C peripheral allows to communicate at different speeds. It is able to work in standard and fast I2C modes up to 400kHz.

It's the master which imposes the communication speed. For more details, please refer to the AN971.

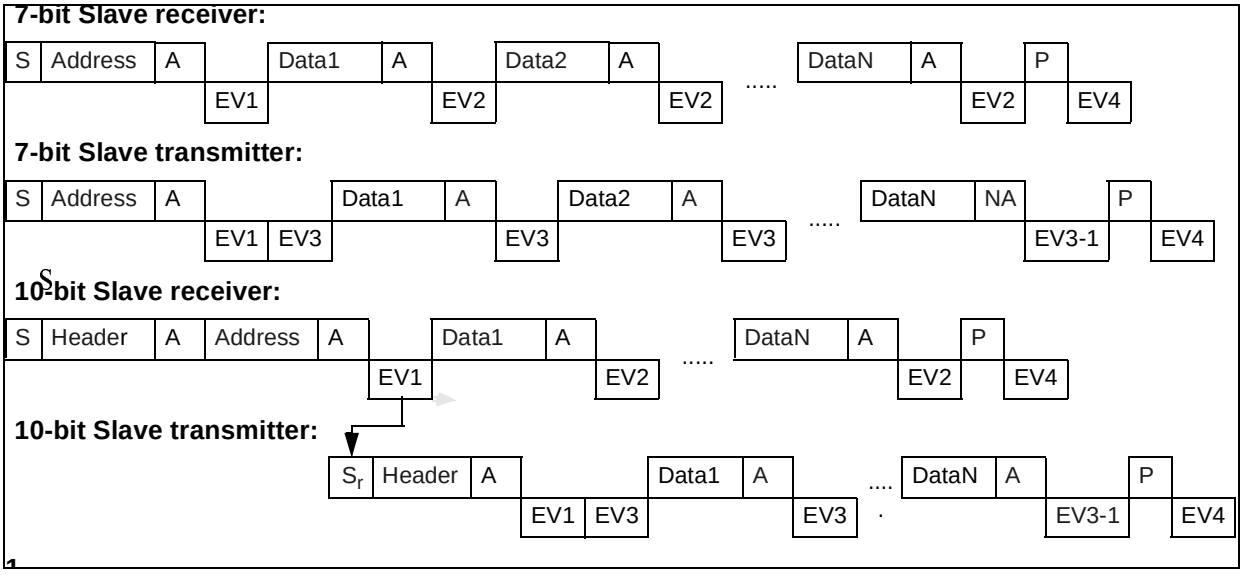
1.2 START, STOP CONDITION AND ACKNOWLEDGE GENERATION

The Start and Stop conditions are sent by the master.

An Acknowledge is sent after an address or a data byte is received when the ACK bit is set in the Control register (CR).

1.3 TRANSFER SEQUENCING

Figure 1. I2C Slave Transfer Sequencing Diagram



Legend: S=Start, P=Stop, A=Acknowledge, NA=Non-Acknowledge, EVx=Event (with interruptif ITE=1).

EV1: EVF=1, ADSL=1, cleared by reading SR1 register.

EV2: EVF=1, BTF=1, cleared by reading SR1 register followed by reading DR register.

EV3: EVF=1, BTF=1, cleared by reading SR1 register followed by writing DR register.

EV3_1: EVF=1, AF=1, cleared by reading SR1 register.

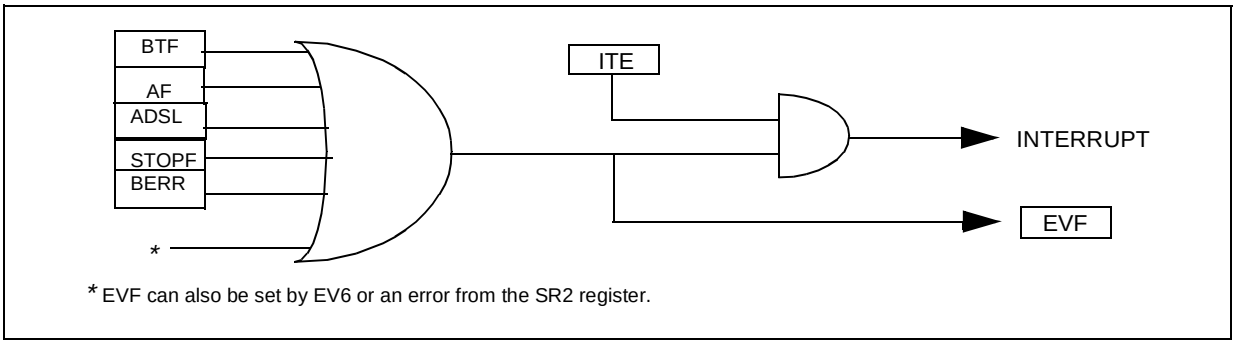
EV4: EVF=1, STOPF=1, cleared by reading SR2 register.

In blue are events sent by the master whereas blank ones are events sent by the slave.

These frames are sequential and as the SDA line is bidirectional, this line is held sometimes by the master and sometimes by the slave.

1.4 SLAVE EVENT FLAGS AND INTERRUPT GENERATION

Figure 2. Slave Interrupt Events Diagram



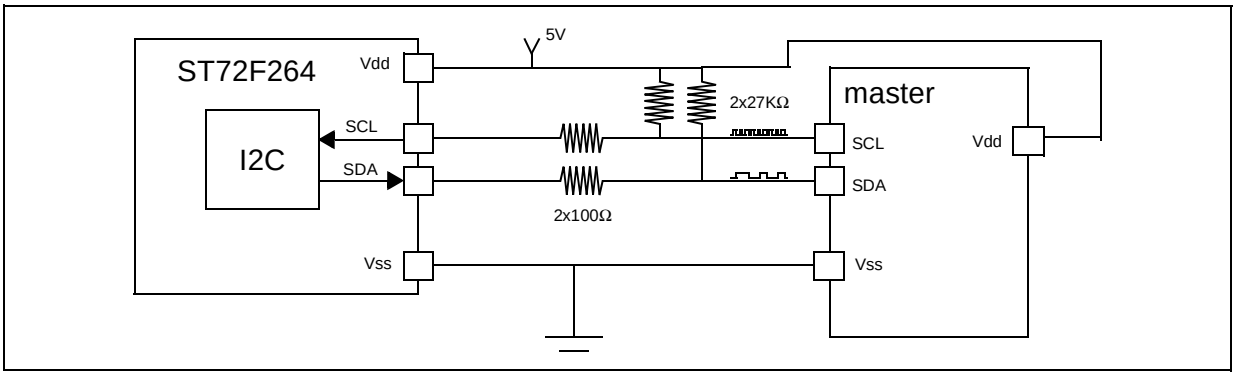
* EVF can also be set by EV6 or an error from the SR2 register.

2 ST7 I2C COMMUNICATION APPLICATION

2.1 HARDWARE CONFIGURATION

The ST7 I2C communication application hardware is composed by a ST72264 microcontroller used as a slave which communicates with any master through an I2C bus interface.

Figure 3. ST7 I2C Communication Application



2.2 ST7 I2C PERIPHERAL BASIC DRIVERS

In this chapter all registers refer to the ST7 I2C peripheral ones (otherwise specified).

2.2.1 Initialize the I2C peripheral

In this application the initialization of the ST7 I2C peripheral is done completely by software.

First the Control register (CR) is cleared and the Data (DR) and Status (SR1,SR2) registers are touched to clear all possible pending event.

Then, the peripheral is enabled through the Control register (CR). This action needs to write twice in the register due to the fact that the Control register (CR) bits can be set only when the PE enable bit is already set. To allow the peripheral to acknowledge the received data the ACK bit of the Control register (CR) is set.

2.2.2 Slave Communication on the I2C Bus

First the interface frequency must be configured using the FRi bits in the OAR2 register.For 10 bit addressing mode, user must configure the slave address using the ADD9 and ADD8 bits of the OAR2 register.

To initiate a I2C communication, first a start condition has to be generated and then the selected slave address has to be sent, both by the master. As soon as a start condition is detected, the address is received from the SDA line and sent to the shift register; then it is compared with the address of the interface (OAR1) or the General Call address (if selected by software). If the GCAL option is to be implemented, user must enable it by selecting the GCAL_Add in user customizable part of the code.

When the address received matches, the interface generates in sequence:

- Acknowledge pulse if the ACK bit is set.

- EVF and ADSL are set with an interrupt if ITE bit is set.

Then the interface waits for a read of SR1 register, holding the SCL line low. Then it determines from the TRA bit of the SR1 register if the slave has to be receiver or transmitter.

- TRA = 0: Data byte received (if BTF=1)

- TRA=1: Data byte transmitted

2.2.3 Receiving Data on the I2C Bus

The slave then receives bytes from the SDA line into the DR register via the internal shift interface. After each byte, the interface generates in sequence:

- Acknowledge pulse if the ACK bit is set.

- EVF and BTF bits are set with an interrupt if ITE bit is set.

Then the interface waits for a read of the SR1 register followed by a read of the DR register (see Transfer Sequencing).

After each reception, the data bytes received are placed in a table. If an error occurs, the interface is re-initialized (I2CCR cleared and put to its initial value after having removed pending interrupts) and the slave waits for next start condition.

2.2.4 Sending Data on the I2C Bus

When the slave receives start condition and matching slave address (with LSB set) from the master, it becomes transmitter and then sends the master back data it stored into a table. It sends data from DR register to the SDA line via the internal shift register.

The slave waits for a read of the SR1 register followed by a write of next data into the DR register (see Transfer Sequencing).

When the acknowledge pulse is received:

- the EVF and BTF bits are set by hardware with an interrupt if ITE bit is set.

After the last data byte is transferred, a Stop condition is generated by the master. The interface detects this condition and sets:

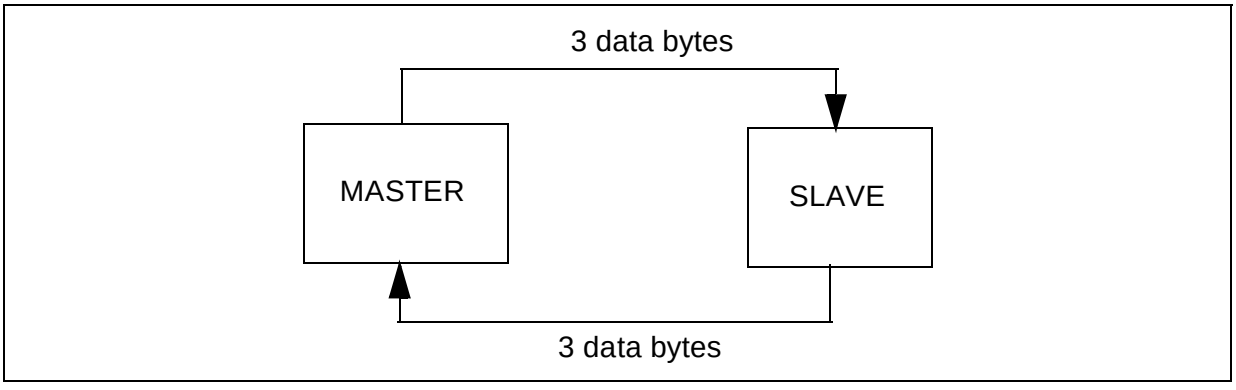
- EVF and STOPF bits with an interrupt if ITE bit is set.

Then the interface waits for a read of the SR2 register.

In this application, the TRA bit of the SR1 register is used as a flag to determine the mode (receiver or transmitter) of the microcontroller.

2.3 COMMUNICATION

Figure 4. Communication Flow

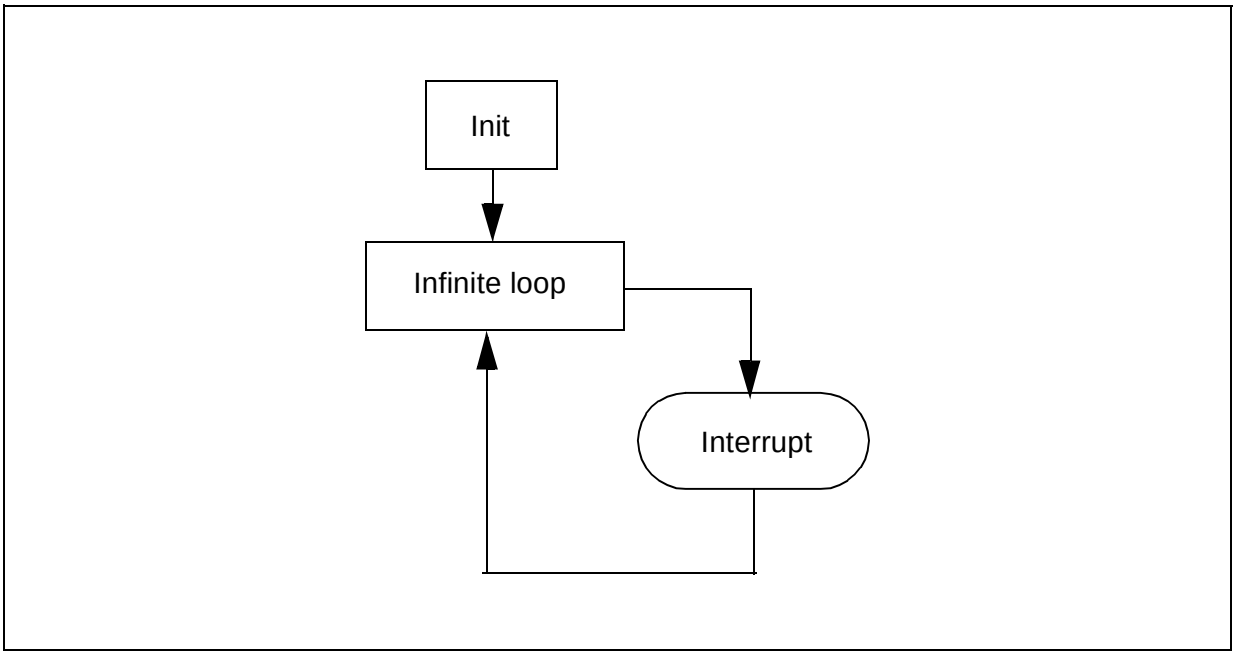


The ST7 slave mode management application is based on two steps driven by the master:

– a reception of 3 data bytes from any master.

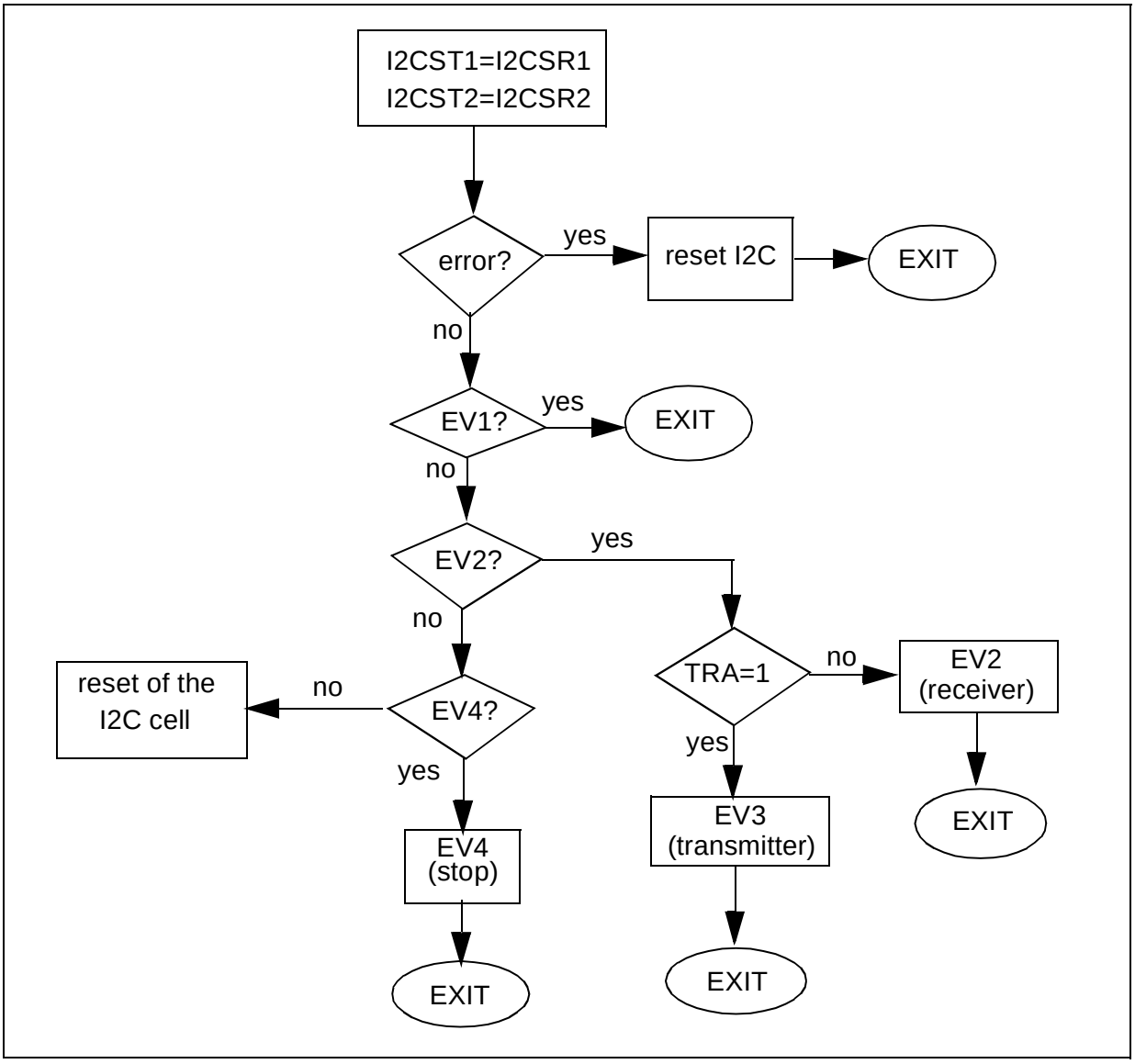
– a transmission of received data bytes to the master.

Figure 5. Main Program Flowchart



Events in the following flowchart refer to events defined into the paragraph “Transfer sequencing” on page 2.

Figure 6. Interrupt Flowchart



3 SOFTWARE

All the source files in assembly code are given in the zip file with this application note.

The source files are for guidance only. STMicroelectronics shall not be held liable for any direct, indirect or consequential damages with respect to any claims arising from use of this software.

THE PRESENT NOTE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH A NOTE AND/OR THE USE MADE BY CUSTOMERS OF THE INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.