AN1525
APPLICATION NOTE

I²C Communication between ST52x520 and EEPROM

Author: C.Ruggieri

1 INTRODUCTION

This AN shows an example of how to interface style I²C EEPROM with an ST52x520 microcontroller using an I²C protocol. An ST24C04 (4Kbit) memory is used for this example, although similar thoughts are also true for any other type of EEPROM that uses an I²C bus.

The software that is proposed uses a read/write routine that performs random access to a certain number of user defined EEPROM memory locations with the ST52x520 microcontroller configured in master mode.

2 COMMUNICATION PROTOCOL

The I²C standard protocol defines every device that sends data in the bus as a “transmitter” and every device that reads data as “receiver”.

The device that controls data transfer is defined as “master”, while all the others are defined as “slave”.

The standard I²C protocol uses a bidirectional data line (SDA) and a clock line (SCL) generated by the master device, which in this specific case has to necessarily be the microcontroller, since EEPROM only works as slave.

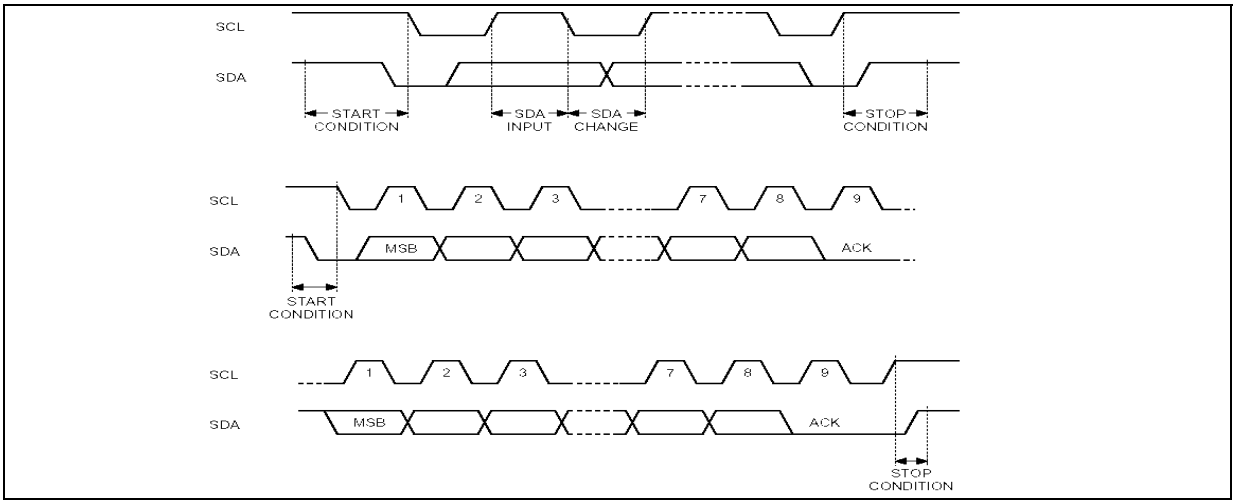
Start Conditon: Start is recognized when the SDA line switches from HIGH to LOW, while the SCL line is stable in HIGH state. START has to precede every data transfer command.

Stop Condition: Stop is recognized when the SDA line switches from HIGH to LOW during a stable phase of the SCL line in HIGH state. The stop condition ends communication.

Acknowledge Bit (ACK): The ACK signal is used to indicate that the data transfer was successful. The “bus transmitter’” (both master and slave) will release the SDA line after having sent an 8 bit byte. During the 9th clock cycles, the “receiver” places the SDA line in LOW in order to indicate that it has received the 8 bits of data.

EEPROM Addressing: In order to begin the communication between the “bus master” and the “slave” memory, the master has to begin with a START condition. Afterwards, the master serially sends 8 bits to the SDA line (MSB the first), which corresponds to the Device Select Code (7 bits) and to one READ or WRITE bit. The Device Select Code is composed of the first four bits equal to 1010 (unique device identification code), which correspond to the specifications of the I²C protocol. The three successive bits identify the chip enable inputs that allow the same bus to interface with other memories.

Figure 1. I²C BUS PROTOCOL

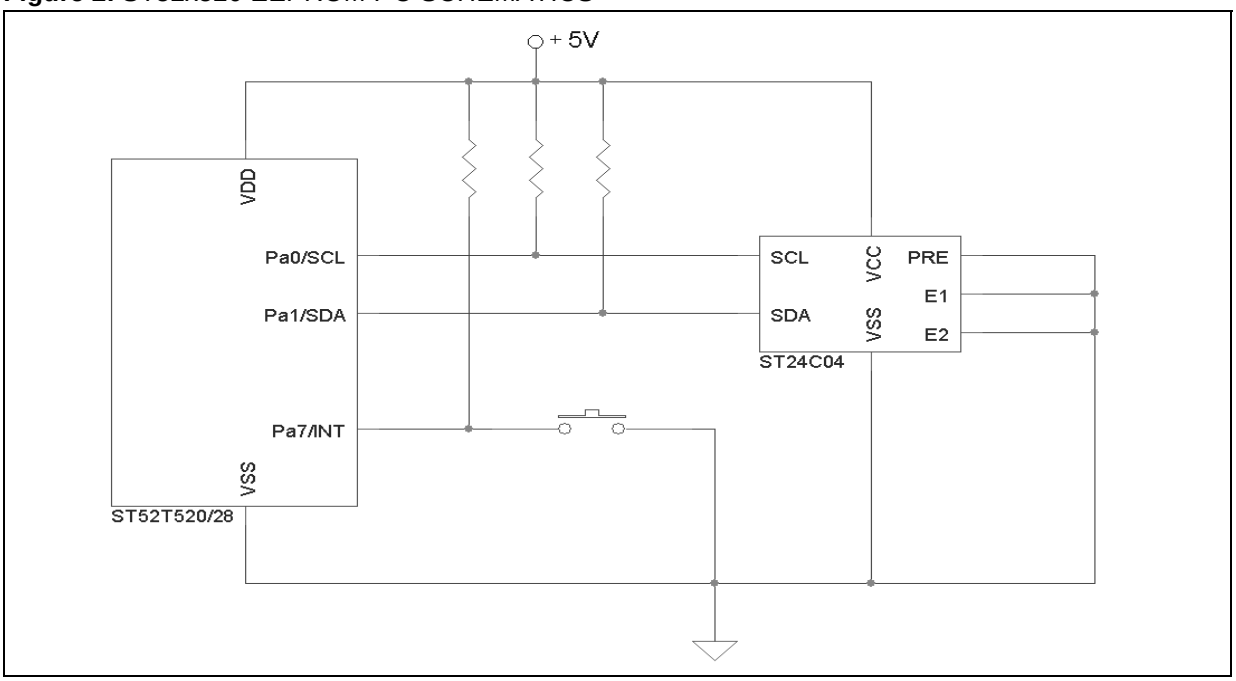


3 HARDWARE DESCRIPTION

The electrical schematic of the circuit, which the proposed program refers to is shown in Figure 2. The Pa0/SCL pin of the microcontroller is connected to the SCL line and the PA1/SDA pin is connected with the SDA line for bus synchronization. The pushbutton is connected with pin PA7 (external interrupt, INT) in order to select the reading and writing phase.

In this scheme, the E1 and E2 enable pins are set as Vss. In order to address up to a maximum of 4 memories, ST24C04 is required to dynamically drive the two E1 and E2 pins from other I/O pins of the microcontroller. The other devices will be addressed by setting the corresponding bits of the Device Select Code, which will be A0h for writing and A1h for reading. The first time the button is pressed, the microcontroller will start sending data in a sequence, beginning from the address that is indicated by the program user. Once the button is pushed again, the microcontroller will request the memory to send the data that was previously written.

Figure 2. ST52x520 EEPROM I^2C SCHEMATICS

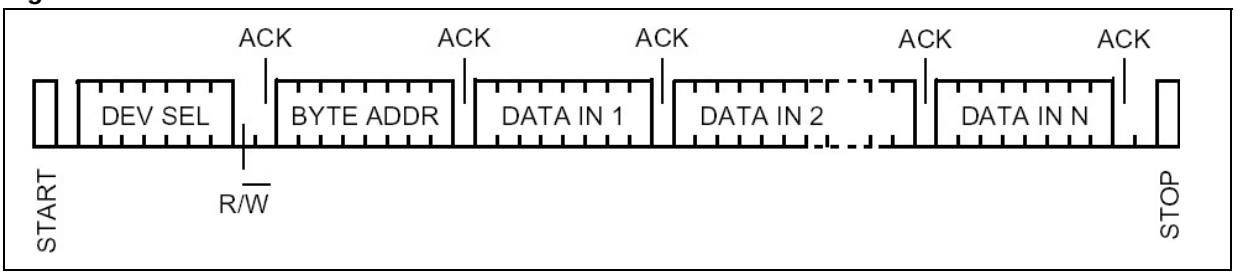


3.1 Page Write Mode

The program performs 8 byte memory writing, equal to the dimensions of the sequential writing page for ST24C04. In PAGE WRITE mode, after having sent the START condition, the I^2C peripheral of ST52x520 sends the Device Select Code with the RW bit reset to '0' (see Figure 3) in the bus. The memory sends an ACK and waits for the initial address writing location to be sent. Upon receipt of the address, the memory sends another ACK.

After the memory sends the ACK, ST52x520 will send the 8 bytes of data (maximum limit) to be written on the memory. The memory will send an ACK after every byte and will automatically increment the last three bits of the internal address counter. After the last byte has been sent and after having waited for the last ACK, the microcontroller will send the STOP condition.

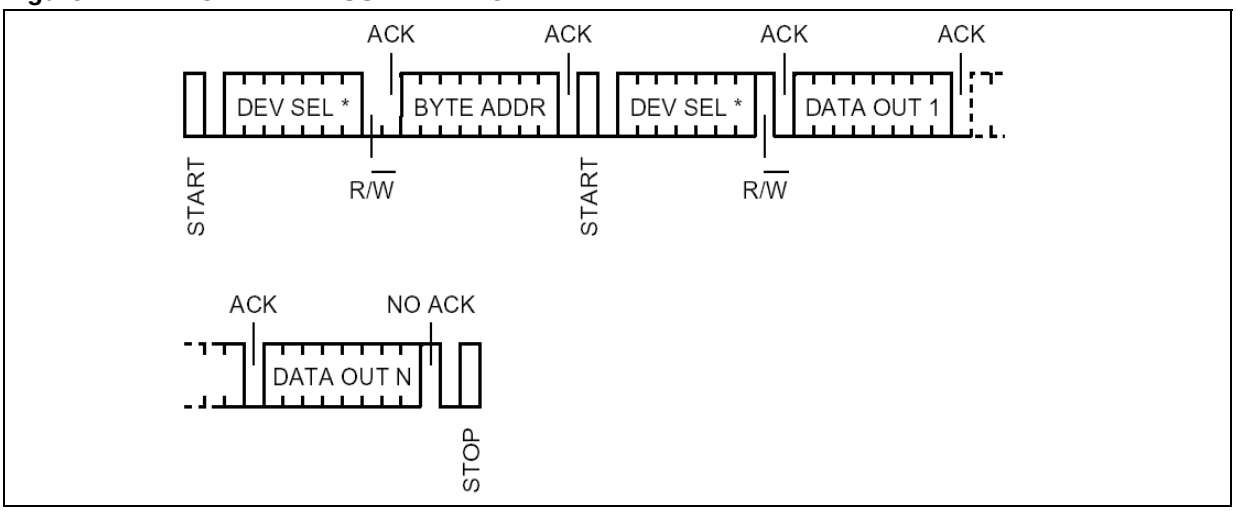
Figure 3. PAGE WRITE MODE



3.2 Random Address Read Mode

In order to read a series of data from a memory address, empty writing (dummy write) must be performed to set the memory address location where to start reading. As shown in Figure 4, after the START condition, the microcontroller sends the Device Select Code with the RW bit reset to '0' (as in the writing operation). After receiving the ACK, the microcontroller sends the address of the memory location where to begin reading. Then, after receiving the ACK from the memory, the microcontroller sends another START condition followed by a new Device Select Code, but this time with the RW bit set to '1'. After receiving an ACK, the memory begins transmitting data that will be received sequentially by the ST52x520, which is each followed by an ACK. The microcontroller wont have to send an ACK after receiving the last byte. Receiving will end with a STOP condition.

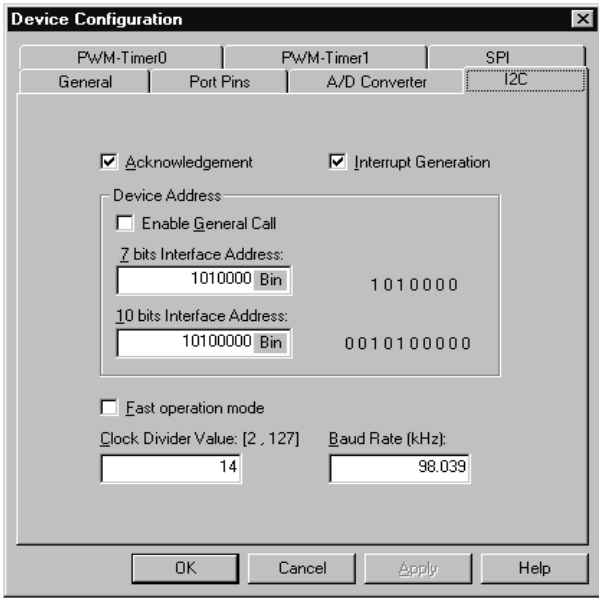
Figure 4. RANDOM ADDRESS READ MODE



4 SOFTWARE DESCRIPTION

The following section describes the software to communicate with the ST52x520 via an I^2C type EEPROM ST24C04. The software is developed with Visual FIVE and allows sequential writing of a maximum of 8 bytes (for ST24C04) to a certain memory address and the reading, afterwards, of the same amount of bytes. The software that is proposed can be found in file I2CGP.FS4 (see file AN1525.FS4 in the application note page at the following web site: http://www.st.com/five).

Figure 5. I^2C DEVICE CONFIGURATION WINDOW



In the example that has been proposed, bit b1 (that represents bit A8 of the memory address) wasn't used in the Device Select Code. The user can select only one range of addresses, which range from 0 to 255. In order to address memory locations that exceed the quantity indicated, the program has to be modified.

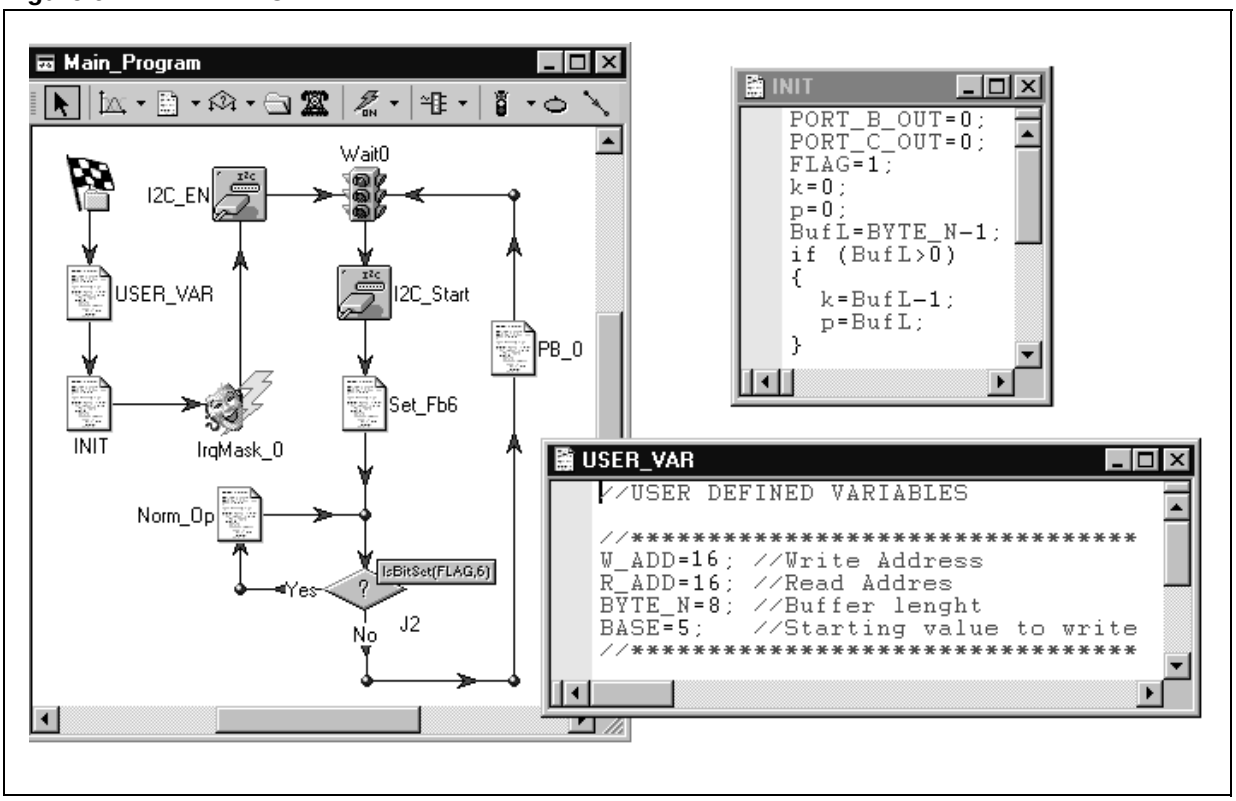
The reading and writing phases are alternated based on the pressure exerted on the pushbutton (see Figure 2). During the reading phase, data read is seen in port B of the ST52x520 microcontroller. Seeing that this is a demonstration program, the user will be responsible for adequately managing the program in the data reception phase. In the example, the micro clock frequency is set to 5 MHZ, all port B pins are configured as OUT, PA7 as INT and pin PA0 and PA1, respectively as SCL and SDA (pins of the I^2C peripheral of the ST52x520 microcontroller).

Figure 5 shows the Device configuration window for the configuration of the I^2C peripheral of the microcontroller, where both the Acknowledgment generation and the Interrupt generation are set, in order to allow the program to synchronize various events. Moreover, the Clock Divider Value (prescaler) is set to 14 in order to obtain a transmission frequency of approximately 98 KHz. The ST52x520 micro can also be configured for communications in Fast Mode (400KHz).

4.1 Main Program Window

The window of the main program flow is shown in Figure 6. In the USER_VAR block, which is shown beside it, the user will be able to define the apposite variable memory address where to begin data writing (W_ADD), reading (R_ADD), as well as the number of data to write and read (BYTE_N) and the initial value to be written in the first location (BASE). BASE will be incremented automatically by the program before writing in the next location.

Figure 6. MAIN WINDOW



The ST52x520 ports and the indexes k and p are initialized in the INIT block. These indexes are used to determine the inhibition of the last ACK during the reading phase and generate the STOP condition correctly, in accordance with the specifications of the I^2C interface of the ST52x520 microcontroller. In case the number of bytes to write and read (BYTE_N) is equal to one, the read/write operations in EEPROM will be respectively that of Byte Write Operation and Random Address Read. The FLAG variable is used to establish exactly which phases the micro ST52x520 has to read or write each time. All of these phases are performed in the interrupt routine of the I^2C described further ahead. The meaning of the single bits of the FLAG variable is shown in Figure 7.

Figure 7. PROGRAM FLAG DEFINITION

FLAG		
	B7	B7=0 : No pushbutton is pressed B7=1: Pushbutton is already pressed and the operation R and W isn't terminated
	B6	B6: This bit is set to "1" when transmission begins and is reset to "0" when it ends
	B5	B5 – Not used
	B4	B4 – Not used
	B3	B3=1: The START bit was sent after the address was written in the Random Address Read routine
	B2	B2=1: The address was sent to the memory before reading or writing on it
	B1	B1=1: The address of the location where to begin reading has been sent. Reading begins right afterwards.
	B0	B0=0: Write – data will be written in the memory – B0=1: Read – data will be read within the memory. The state of this bit changes each time pressure is exerted on the pushbutton at the end of the routines.

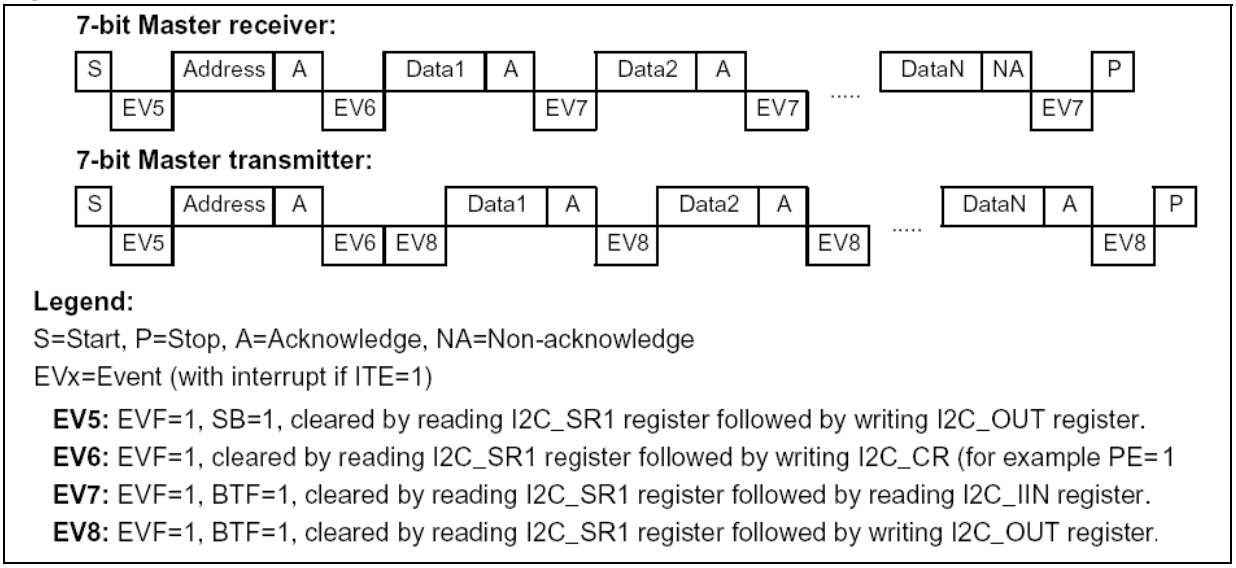
The use of a register with various flags is necessary in this program, seeing that within a sole routine (I^2C interrupt) the program manages all phases of sequential writing, random reading and all interrupt events generated during the various steps of every phase (see Figure 8).

The program flow begins by enabling the interrupt of the I^2C peripheral and afterwards enables the peripheral iteself (IrqMask_0, I2C_EN controls). The program flow is interrupted immediately after the Wait0 block. Once the pushbutton is pressed, the status of bit b0 of the FLAG register is inverted. This occurs via an XOR operation on the b0 bit of the FLAG register. Initially, as FLAG = 1, b0 will be reset to '0'. This last operation is performed by the external interrupt routine, which sets bit 7 of the FLAG register to '1'. In this manner, the External Interrupt Routine will be executed again only at the end of reading or writing to the memory.

After exiting the routine via the Ret10 control, the START condition will be activated through the I^2C peripheral of the ST52x520 micocontroller and bit 6 will be set to '1' in the FLAG register. This flag bit is necessary since the START condition begins at the Main Program level, while the communication management occurs after each event generated by the I^2C interrupt routine.

In the Norm_Op block, a code can be inserted to perform operations while the microcontroller communicates with the memory. Finally, the PB_0 block resets the value of port B to 0. Figure 8 represents the sequence to be followed to receive and transmit data to an I^2C peripheral, configuring the ST52x520 microcontroller as master (receiver and transmitter) with the addressing mode of 7 bits (refer to the ST52x520 datasheet for the complete diagram of the sequences, relative to the I^2C peripheral of the microcontroller).

Figure 8. 7 BIT MASTER TRANSFER SEQUENCING



Legend:

S=Start, P=Stop, A=Acknowledge, NA=Non-acknowledge

EVx=Event (with interrupt if ITE=1)

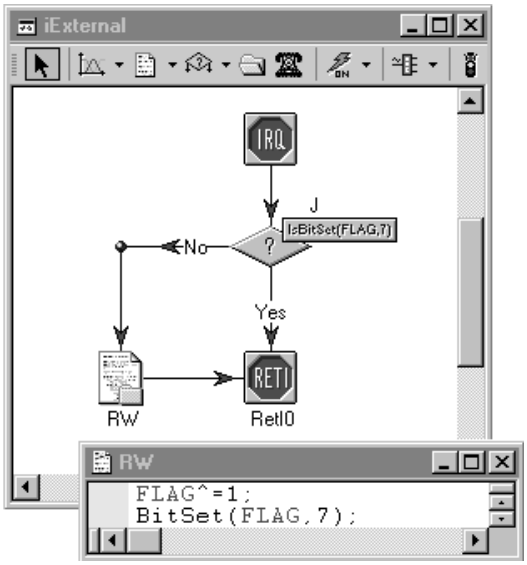
EV5: EVF=1, SB=1, cleared by reading I2C_SR1 register followed by writing I2C_OUT register.

EV6: EVF=1, cleared by reading I2C_SR1 register followed by writing I2C_CR (for example PE=1

EV7: EVF=1, BTF=1, cleared by reading I2C_SR1 register followed by reading I2C_IIN register.

EV8: EVF=1, BTF=1, cleared by reading I2C_SR1 register followed by writing I2C_OUT register.

Figure 9. EXTERNAL INTERRUPT WINDOW



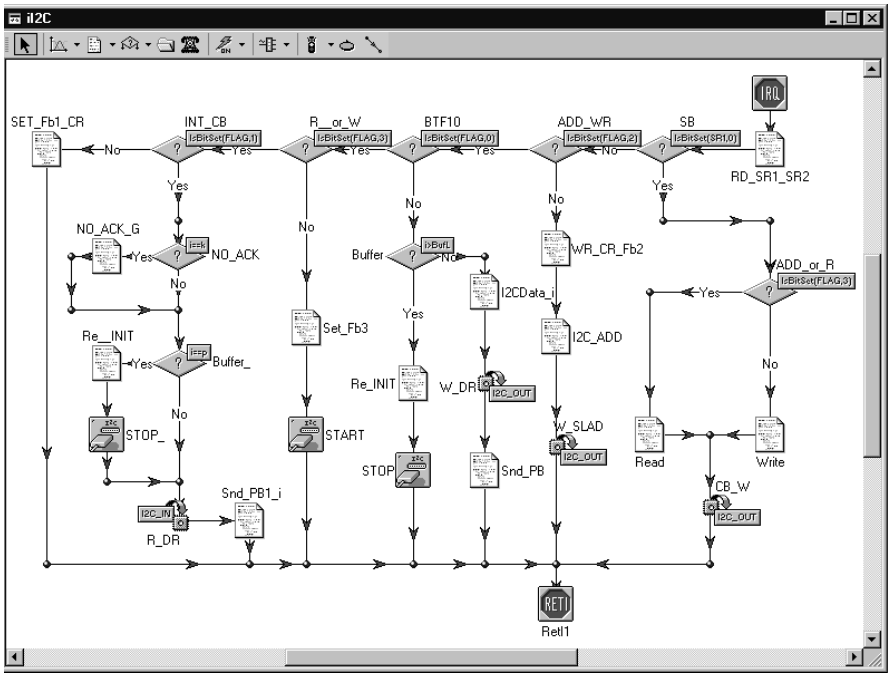
4.2 I²C Interrupt Window

The routine of the interrupt window of the I²C peripheral (represented in figure 10) manages all the phases that are necessary for the correct reading and writing sequence. As can be seen in Figure 8, each time a START condition begins or data begins, the I²C peripheral of the microcontroller sends an interrupt request that will have to be managed by the user (EVx events).

4.2.1 Common steps in reading and writing.

When the I²C peripheral of the microcontroller has sent the START condition, the I²C routine will be called. Block RD_SR1_SR2 is the first block that will be executed (this will occur at every call of the routine). The Status Register of the I²C peripheral is read in this block (SR1-other than the Status Register SR2, which can be used to manage eventual transmission or reception erros, errors which are not managed in this example) via the instruction SR1 = _inpReg_07 (and SR2=_inpReg_08).

Figure 10. I²C INTERRUPT WINDOW



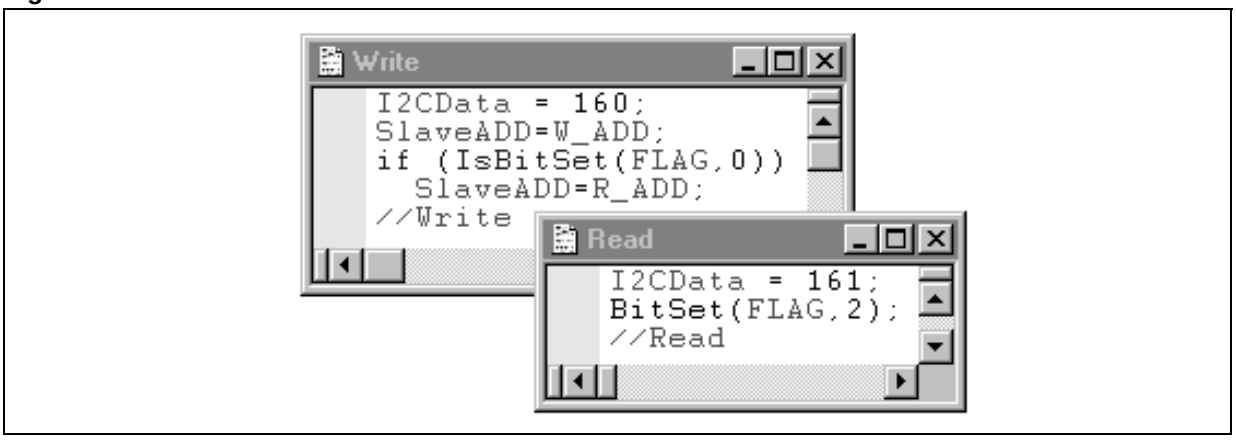
This operation is indispensable, seeing that the I^2C peripheral will not continue operations requested if SR1 is not read. In fact, only by reading this register the microcontroller resets the Event Flag bit (EVF) to '0'.

The first conditional block (SB) verifies if the START condition was just generated. In fact, bit 0 of register SR1 is set to '1' only after a START condition.

When an interrupt is generated following a START condition, the program verifies if START is the initial one, or the one that follows the address location where to begin reading (Random Address Read Mode, see Figure 4). In this last case, bit 3 of the FLAG register will be set to '1', otherwise it will be reset to '0'.

When the first START occurs, the value transferred to the I2CData (used by the program to transfer data to/from memory via the Send and Receive controls of Visual FIVE) will be 160 (corresponding to the value of the Select Code with bit 7 - RW - reset to '0' for writing operations) if bit 3 of the register FLAG is '0'. The address transferred to the variable SlaveADD will be that of writing, W_ADD. If, on the other hand, bit 3 of FLAG register is '1' (value that is modified after the second START has been sent), the value transferred to the I2CData register will be 161 (RW bit = '1'). Control is performed through the conditional block ADD_or_R.

Figure 11. WRITE AND READ BLOCK WINDOWS



In case the START condition occurs after the initial address has been sent (in the Random Address Read Mode), the program sets bit 2 of the FLAG register to '1'. This indicates that the Device Select Code for reading has already been sent to the memory and from this point on, sequential reading may be performed. Lastly, the CB_W control transfers the I2CData variable to the I^2C peripheral of the microcontroller and the routine ends via the Retl1 control.

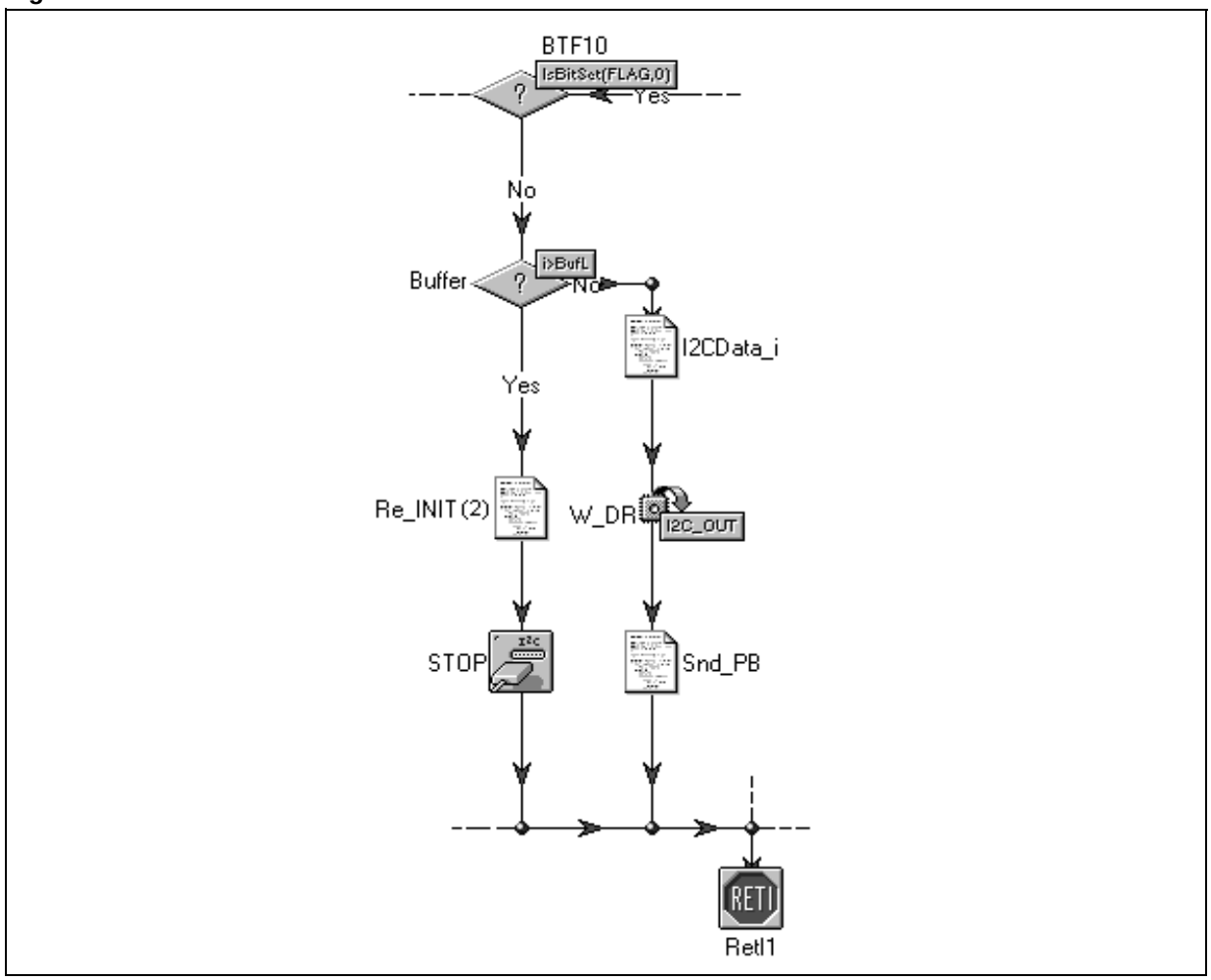
At the end of the process in which the contents of the I2CData have been sent to the memory, the microcontroller will generate a new interrupt. However, bit 0 of register SR1 will be reset to '0'. The program will examine the condition of the ADD_WR block, which verifies bit 2 of the FLAG register.

If this bit is still reset to '0', the memory address to begin writing/reading data must be sent to the EEPROM. The I2C_ADD block transfers the value of the SlaveADD, which contains this address (address previously transferred by the Write block, Figure 11) in the I2CData variable. In this phase, other than reading the SR1 register, which is guaranteed by the RD_SR1_SR2 block performed at every entry of the interrupt of the I^2C routine, writing on the Control Register (CR) of the I2C peripheral (see Figure 8, EV6) is indespensable. This operation is performed by the WR_CR_Fb2 block via the CR=cnfReg_16 and_cnfReg_16=CR instructions, which simply read, then rewrite the CR register without modifications. In this block, bit 2 of the FLAG register is set to '1' in order to inform the program that the memory address where to read or write data was sent to EEPROM. After the address has been sent to the I^2C peripheral of the microcontroller (control W_SLAD), the routine terminates via the Retl1 control. At the end of the transfer process, the micro will generate a new interrupt and a new I^2C routine will be called. This time, due to a '1' value of the bit 2 of the FLAG register, control will pass to the conditional block BTF10, which determines if the operation to be performed is that of reading or writing (operation which is established by the value of bit 0 of the FLAG register, modified with every exterted pressure on the pushbutton).

4.2.2 Writing Block.

If writing is to be performed, the program will execute the instructions of the block shown in Figure 12 (bit 0 of the FLAG register = 0). The conditional Buffer block verifies when the number of bytes sent from the I^2C peripheral of the ST52x520 microcontroller to the memory has achieved the number of bytes indicated by the user in the BYTE_N variable.

Figure 12. WRITE FLOW BLOCKS

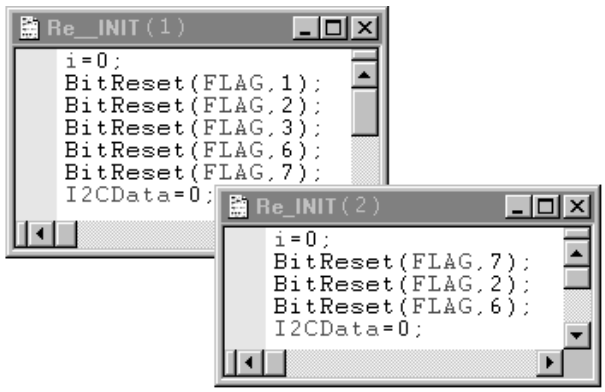


In case the result is positive (i>BufL[=BYTE_N-1]) the STOP condition is sent, ending communication (see figure 3) and the Re_INIT (2) block is executed, which sets all the bits in the FLAG register to the initial conditions once again (see Figure 13).

If the number of bytes sent still hasn't reached the limit, the value of the variable index i in the I2CData_i block is incremented by one each time. In the same block the initial data defined by the user in the BASE variable is assigned to the I2CData_i variable. The BASE variable is incremented by the instructions I2CData = BASE+i and i++ contained in the I2CData_i block. In this manner, the function of the writing routine is that of sending the growing numeric values that begin with the value specified by the BASE variable, written in the memory locations that begin with the address specified by the W_ADD variable. The Snd_PB block simply sends the byte transferred to the I^2C peripheral of the microcontroller, from the W_SLAD control to port B of the microcontroller.

After each byte is sent (since there is no variation of the value of the FLAG register), the program returns the control to the Main Program Window through the Retl1 control, and each time an interrupt is generated the I^2C routine will return in the writing block (see Figure 12).

Figure 13. INITIALIZATION BLOCKS



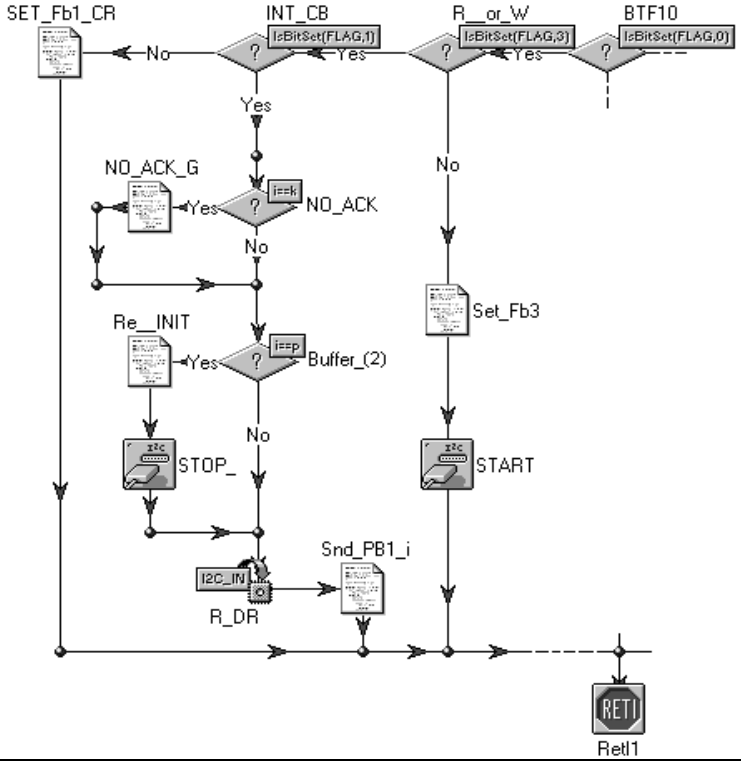
4.2.3 Reading Block.

If, on the other hand, the operation set by the pushbutton is that of reading (FLAG, bit b0=1) the program flow is different: before reading a memory location, as represented in Figure 4, a dummy write must be performed. Meaning that, writing mode must be set with the RW bit of the Device Select Code reset to '0', the address where to begin reading must be sent, a START condition must be sent, the Device Select Code (this time with the RW bit set to '1') must be resent and reading may begin.

After the START condition has been sent in the Main Program Window, the microcontroller generates an interrupt that will service the I^2C routine. After each START, bit 0 of the SR1 register is set to '1' and the conditional expression specified in the SB block will become true. The operations performed by the program from this point on are those described in paragraph 4.2.1: common steps in reading and writing where a description is provided on how the program sends to the memory the initial address where to begin reading.

Once the BTF10 block condition is performed, seeing that bit0 of register FLAG is set to '1', the program will perform the reading block represented by Figure 14.

Figure 14. READ FLOW BLOCKS



The first conditional block (R_ _or_W) verifies if bit 3 of the FLAG register has already been set to ‘1’. In case it is negative, the routine sets this bit to ‘1’ and sends the START condition, exiting immediately after the I^2C routine via the Retl1 control.

When the START condition has been generated by the peripheral, the microcontroller will generate an interrupt, which will allow the program to perform the ADD_or_R conditional block, seeing that at every START bit, b0 of the SR1 register is set to ‘1’.

This time, seeing that bit 3 of the FLAG register is set to ‘1’, the program will perform the Read block, which via the CB_W control will send the Device Select Code to the memory with the RW bit set to ‘1’ (see Figure 10). At the end of the transmission of the I^2C peripheral, the microcontroller will generate an interrupt. At this point, two operations will have to be performed: reading the SR1 register (operation performed everytime the I^2C is called by the RD_SR1_SR2 block) and writing on the CR register (for example PE=1), as per the specifications of the I2C peripheral that are found on the ST52x520 microcontroller datasheet.

It’s important to notice that as soon as the SET_Fb1_CR block writes on the CR register, the I^2C peripheral immediately begins to generate the clock signals on the SCL line to receive data, generating a second interrupt. Only at this time may the receiving register be read through the R_DR control. If reading occurs immediately after writing on the CR register (before the peripheral completes reading data), the micro would return the value previously memorized in the Data Register (DR) of the I^2C peripheral, which in this example, is the value of the Device Select Code with the RW bit set to ‘1’ (A1h).

This is the reason why bit 1 of the register FLAG was used: when this bit is reset to ‘0’, it means that the routine is serving an interrupt generated, following the completion of the Device Select Code being sent with the RW bit set to ‘1’. The INT_CB condition becomes true and the program follows the SET_Fb1_CR block, exiting the I^2C routine via the Retl1 control, right after having set bit 1 of the FLAG register to ‘1’. As soon as the peripheral has finished receiving the first data and the data is ready in the reading register DR of the I^2C peripheral of the microcontroller, a second interrupt is generated. The program within the I2C Window will restart from the conditional block NO_ACK, since at this point bit 1 of the FLAG register is set to ‘1’.

Two conditional blocks are present in the program flow of data reception (as shown in Figure 14), which verify the transmission both of the next to last and of the last data. This is necessary seeing that the microcontroller wont have to generate the ACK after receiving the last data, in order to allow the memory to close communication. This control is performed by the conditional block NO_ACK which, when index i (this index specifies the number of bytes that have already been sent) achieves the value of variable k (=BufL-1), it inhibits the generation of the Acknowledgement via the instructions CR =_cnfReg_16 and _cnfReg_16 = CR & 251 of the NO_ACK_G block, resetting bit 2 (=ACK) of the Control Register (CR) of the I^2C peripheral of the ST52x520 microcontroller to ‘0’.

Instead, the second conditional Buffer block (2) verifies reading of the last byte by performing a confrontation of the index with variable p, and in case it is positive it generates a STOP condition before reading the last data byte that is sent, via the R_DR control.

The use of the two variables k and p is necessary in this program in order to anticipate if the user wants to write or read only one data, which in this case, ACK inhibition and the STOP condition must be sent contemporaneously. If the value of BYTE_N is equal to 1, both the variable p and k will have the value 0 (see Figure 6) and both conditions of the two conditional blocks NO_ACK and Buffer (2) will be true.

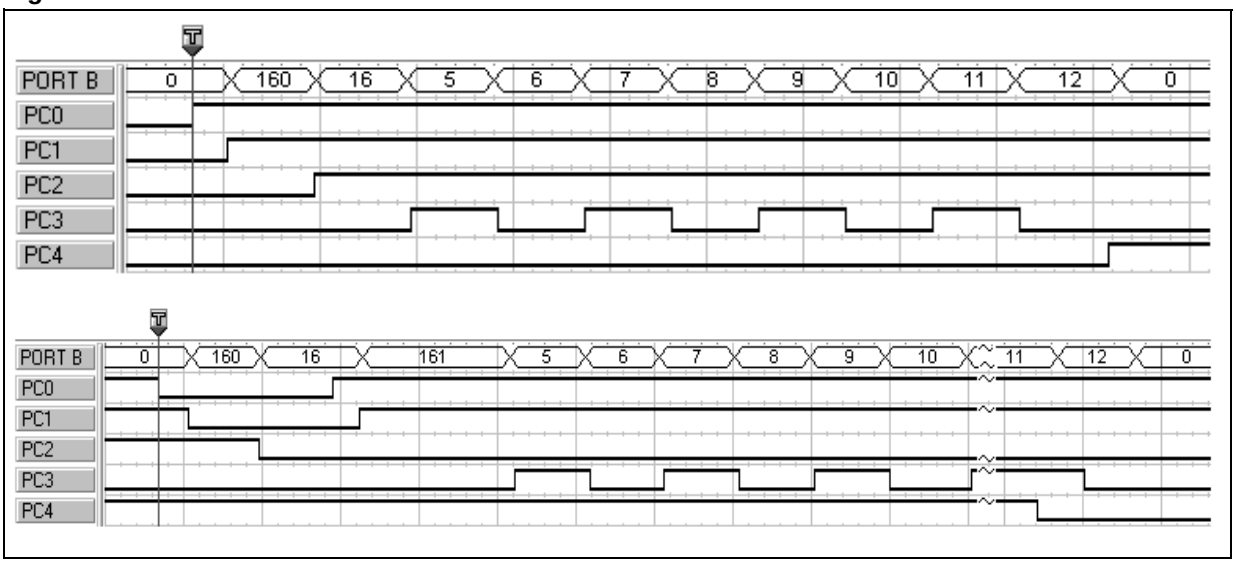
Finally, the Snd_PB1_i block allows data read by the memory to be sent to port B of the ST52x520 microcontroller and increments the value of the index variable i.

After receiving the last byte and before exiting the interrupt routine via the Retl1 condtion, the program returns to the initial conditions of the bit in the FLAG register in order to retransmit or be able to receive once again via the Re-INIT (1) block (as represented in Figure 13).

5 TEST

The graphics represented in Figure 15 are those obtained by the logical states analyzer and they regard the writing and reading phase, obtained by slightly modifying the program in order to introduce the reference signals indicated (PC0->PC4), to view the synchronization of the data of port B with the various phases.

Figure 15. WRITE AND READ BLOCKS



In this test, the initial memory location address where reading and writing will begin is 16. The number of bytes written in the memory is 8 and the initial value written is 5. This value will be incremented every time writing in the memory occurs. This data is specified in the USER_VAR block (see Figure 6).

The PC0 signal changes state everytime the START condition is sent. On the other hand, the PC1 signal changes state every time the CB_W control is executed, which sends the Device Select Code to the memory. The PC2 signal changes its state of performance of the W_SLAD control, which sends to the memory (EEPROM) the address of the location where reading or writing will be performed. The PC3 signal changes state before sending or receiving every byte of data from the memory. Lastly, the PC4 signal changes state everytime the STOP condition is sent.

REFERENCES

[1] ST52x520 - Datasheet, STMicroelectronics, 2001

[2] ST24C0Z1 - Datasheet, STMicroelectronics, 1999

[3] Visual FIVE 5.0 - User Manual, STMicroelectronics, 2002