Application Report
SLAA330–September 2006

Software I²C Slave Using the MSP430

Thomas Kot *MSP430*

ABSTRACT

This application report describes the design of a software I²C slave that can run up to 100-kbps using an MSP430 MCU. This software I²C uses a small amount of MCU resources so that it can be implemented on lower-end MSP430 devices. The interface to the I²C bus uses two I/O pins; and the code size, including flash memory and RAM, is small. Low power is built into the software design. The application of this software I²C is not just confined to low-end products.

Contents

List of Figures

List of Tables

1 Introduction

I²C is a common communication protocol in the electronics industry. The two-wire configuration can attach up to 127 devices using 7-bit addressing.

Texas Instruments MSP430 is an ultralow-power 16-bit RISC MCU. With its fast processing speed, the MSP430 can emulate data transfer through an I/O pin at high speed. In this application, the MCU accommodates an infrared receiver and an I²C slave software routine using a clock speed of 8 MHz.

The device chosen for this application is the MSP430F1121. The peripherals used are as follows:

- Two I/Os with interrupt for emulating I²C
- One I/O with interrupt for receiving infrared data input
- Timer A

This application demonstrates the implementation of a software 100-kbps I²C slave. The fast processing capability of the MSP430 allows additional software modules to be implemented as well. For demonstration purposes, infrared communication was also implemented.

This application note contains a brief section on I²C slave and infrared reception, followed by a more detailed description of the system design using hardware and software.

2 I²C Operation

The full function of I²C is described in its specification documents (I²C specification, version 2.1). In this application, only the slave mode with sequential read and write is implemented. This mode is enough to cover normal use of I²C slave communication.

There are four major states when implementing an I²C.

The following are typical I²C signals:

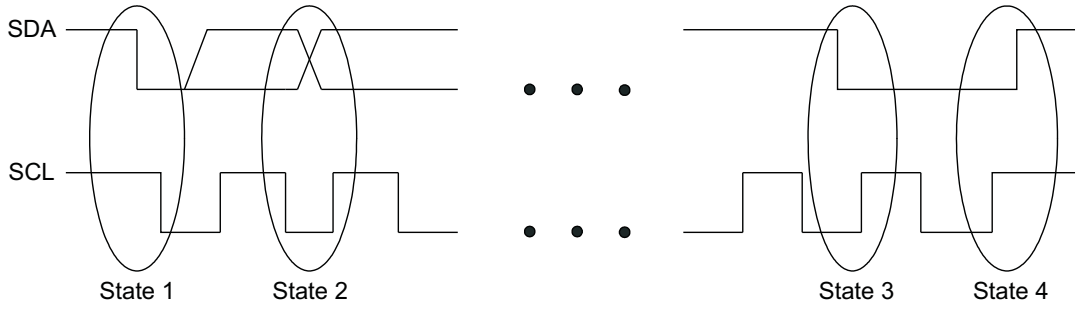


Figure 1. I²C Signals

State 1

Start Condition. A high-to-low transition on the SDA line while SCL is high is one unique case. This situation indicates a start condition.

State 2

After the start condition, data is transmitted bit by bit. When SCL goes low, SDA is free to set its state to high or low. When SCL goes high, state of SDA is latched into the MCU buffer; SDA cannot make any change at this time, because it is treated as error.

State 3

On every 9th SCL clock, an acknowledge bit is issued. To indicate an acknowledgment, the MCU on the receiving side needs to set SDA to low before the 9th rising edge of SCL.

State 4

Stop condition. A low-to-high transition on the SDA line while SCL is high defines a stop condition.

For 100-kbps communication, SCL must run at 100 kHz. Assuming, the mark space ratio is 50%, the clock cycle of SCL is 10 µs, with 5 µs at level high and 5 µs at level low. When the MCU is running at 8 MHz, one instruction cycle (CPU cycle) is 125 ns. 10 µs can accommodate 80 instruction cycles while 5 µs can accommodate 40 instruction cycles.

To prevent infinite waiting loops and unpredictable errors, a time-out timer is started whenever the start condition is triggered. When stop condition is detected or there is a time out, the I^2C slave routine stops running.

3 Firmware Overview

The firmware can be divided into three modules: key scan routine, IR detection routine, and I^2C routine. To avoid conflict among routines, as they all use the same RAM buffer, only one interrupt service routine is enabled during data transfer. After the IR routine or key pad routine returns valid data, P1.1 is set high to notify the master, and the I^2C routine is enabled. The following sections briefly describe how the three modules work. For the I^2C module, a detailed explanation and some working examples are discussed.

3.1 Key Scan

An infinite loop is used for scanning key input. Five inputs in port 1 and three inputs in port 2 are directly connected to the MCU (not in matrix form). This is a simple example of key scanning. Each key is checked sequentially in a software loop. In this loop, all other interrupt services are enabled. However, when a key press is being checked, the MCU must also check if an I^2C or IR routine is running. If not, the global interrupt enable bit is reset, the command corresponding to the key pressed is stored in the RAM buffer and P1.1 is set high, waiting for the master unit to fetch this command through I^2C. It is important to note that if the master does not check the status of the I^2C and IR routines, the RAM buffer will be corrupted. During the process that the command is stored in the buffer, other interrupts must be disabled to avoid corrupting the buffer. See the Appendix C for a flow chart of the key scan routine.

3.2 Infrared Receive

When receiving an infrared signal, the I^2C interrupt must be disabled. There are two reasons:

- The IR routine uses Timer_A's external input to measure the IR pulse width, but Timer_A is also used by the I^2C routine.
- The RAM buffer is shared between two routines.

If both the IR and I^2C routines are enabled at the same time, the received data will be corrupted. The following is the basic flow of IR routine.

1. Run Timer_A in continuous mode. Disable P1.0 interrupt to disable I^2C function.
2. Receive IR signal and check its pulse width to determine a logic 0 or 1.
3. Customer code checking (see Figure B-1)
4. Customer code checking (see Figure B-1)
5. If the last IR signal is received, store the received data in the RAM buffer. Set P1.1 high and enable I^2C interrupt on P1.0

3.3 I²C Data Fetching

When the I^2C routine is enabled, the IR routine must be disabled. This is because Timer_A is used as a time-out timer, and the I^2C routine needs access to the RAM buffer. Therefore, there is a conflict between the IR and the key pad routines. After an I^2C data frame has completed and a stop condition is generated, both Timer_A and the RAM buffer are freed. P1.1 is reset, and the system is ready for the next event.

4 I²C Software Flow

When a valid start condition is detected, the software enables the SCL rising-edge interrupt and resets the software state-machine pointers. Then, the subsequent bits are processed by the SCL interrupt service routine (ISR). For a more detailed explanation, please see Appendix A.

4.1 Data Writing into MCU

When the master sends the address to the slave and the R/W bit is clear, the I²C ISR executes the corresponding sequence of states to process the master write request. The same software routine is used to process both data and the address sent by the master. Figure 2 and Table 1 show an overview of a master write operation. The ISR column contains the labels listed in the source code of the software state machine.

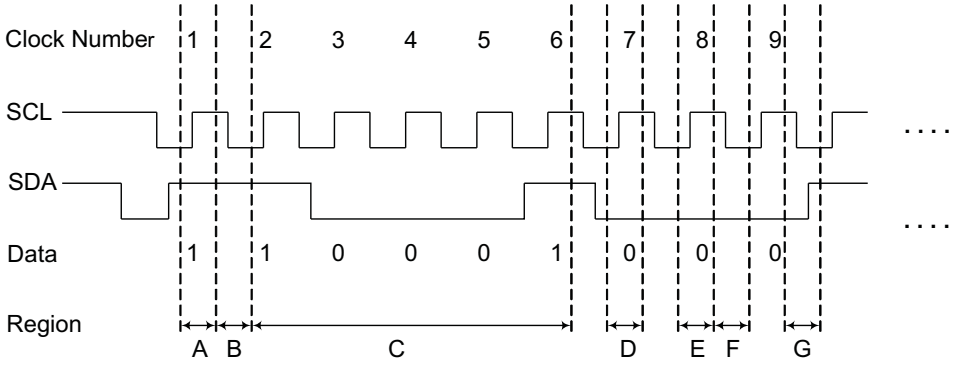


Figure 2. Writing Data

Table 1. Writing Data

ISR	SCL Edge Triggering	Region	Description
SCL_W1LH	↑	A	The most-significant bit, bit 7, is detected; enable SDA rising edge INT, which enables stop condition detection.
SCL_W1HL	↓	B	Disable SDA INT, which in turn disables stop condition detection. If an SCL rising edge is detected, instead of SDA, Label22 is triggered and the SDA INT is disabled.
SCL_W2to6LH	↑	C	Bit 6 to bit 2 detect
SCL_W7LH	↑	D	Bit 1 detect and address detect
SCL_W8LH	↑	E	Bit 0 detect; R/W bit detect; normal write-in data.
SCL_W8HL	↓	F	Set SDA output for ACK; if a write command is received, record data in RAM. If a read command is received, R4 ISR address pointer is set to the address for reading data.
SCL_W9HL	↓	G	Reset SDA, reset R4 for write-in data ISR address pointer.

4.2 Data Reading From MCU

When a read command is detected, the ISR for master read is used. Figure 3 and Table 2 show the overview of a master read. The ISR column contains the labels listed in the source code of the software state machine.

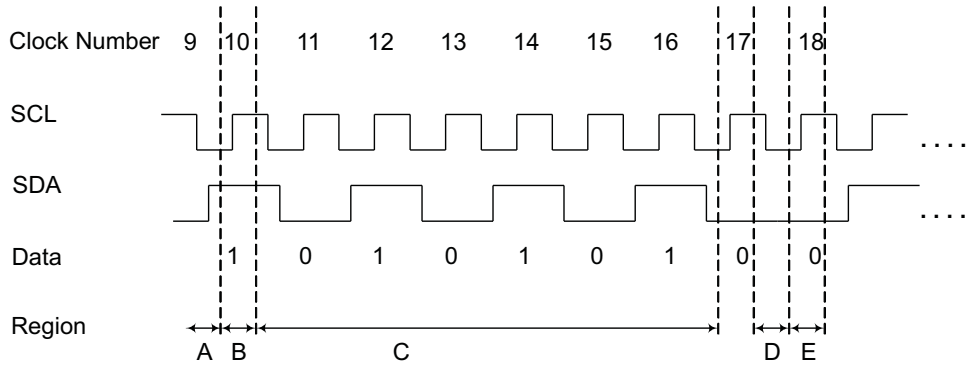


Figure 3. Reading Data

Table 2. Reading Data

ISR	SCL Edge Triggering	Region	Description
SCL_R1HL	↓	A	Set the following SCL ISR trigger to be rising edge. The remaining instructions are the same as SCL_R2to8HL.
SCL_R1LH	↑	B	SDA rising edge INT is enabled, which in turn enables the detection of a stop condition. The SDA INT is enabled until SCL sends out a falling edge. The routine SCL_R2to8HL disables the SDA INT and continues to read out data. This routine provides an alternative way to stop communication with ACK at the end.
SCL_R2to8HL	↓	C	Rotate one bit out of a data byte; set the SDA output according to the bit.
SCL_R9HL	↓	D	Release the control of SDA. Set the rising edge trigger of SCL, prepare for detection of the ACK bit.
SCLR9LH	↑	E	Detect the ACK/NACK bit. If ACK is received, falling edge of SCL is set. Next data is loaded. R4 is set to SCL_R1HL. If NACK is received, indicating the end of data reading, SCL triggering is set to rising edge, and R4 address pointer is set to SCL_W1LH which enables detection of a stop condition.

5 Timing of I²C

5.1 Writing Process

Figure 4 shows an example for writing 0xAA into the slave MCU with device address 0x62 (seven bits).

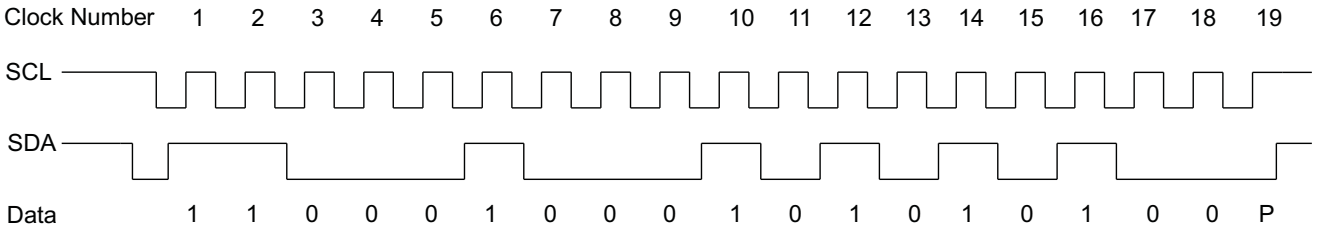


Figure 4. Writing Process Timing Diagram

Note: The data in Table 3 are measured with an operating clock frequency of 8 MHz. One instruction cycle is 0.125 µs. The number of instruction cycles is equal to the total time divided by 0.125 µs. Integer clock numbers indicate the number of rising edges of SCL after a start condition; half clock numbers indicate the falling edges of SCL. ISRs from clock numbers 1 to 9.5 are used for detection of device address, while clock numbers 10 to 18.5 are the ISRs for writing data into MCU. In Table 3, the Total Time column shows the total amount of time taken to complete the corresponding interrupt service routine. The Latch In/Out Data column shows the time taken for the corresponding interrupt service routine to reach the instruction of latch in/out and execute it.

Table 3. Writing Process Timing

Clock Number	ISR Label	Total Time (µs)	Latch In/Out Data (µs)	Remarks
Start Condition	SDA	9.375	NA	Start condition initialization. Time-out timer starts running.
1	SCL_W1LH	4.875	2	Capture the first bit and prepare for detection of stop condition.
1.5	SCL_W1HL	3.125	2	Set rising edge detect for SCL. Disable the SDA signal detect, hence disable detection of stop condition. The 2 µs is the time taken from the falling edge to the instruction that clears the SCL interrupt flag.
2	SCL_W2to6LH	3.375	2	Latch in 2nd data bit
3	SCL_W2to6LH	3.375	2	Latch in 3rd data bit
4	SCL_W2to6LH	3.375	2	Latch in 4th data bit
5	SCL_W2to6LH	3.375	2	Latch in 5th data bit
6	SCL_W2to6LH	3.375	2	Latch in 6th data bit
7	SCL_W7LH	5.75	2	Latch in 7th data bit
8	SCL_W8LH	4.375	2	Latch in 8th data bit
8.5	SCL_W8HL	4.125	1.5	Set SDA to low for ACK bit
9.5	SCL_W9HL	3.375	1.5	Release SDA for input
10	SCL_W1LH	4.875	2	Capture the first bit and prepare for detection of stop condition
10.5	SCL_W1HL	3.125	2	Set rising-edge detect for SCL. Disable the SDA signal detect, hence disable detection of stop condition. The 2 µs is the time taken from the falling edge to the instruction that clears the SCL interrupt flag.
11	SCL_W2to6LH	3.375	2	Latch in 2nd data bit
12	SCL_W2to6LH	3.375	2	Latch in 3rd data bit
13	SCL_W2to6LH	3.375	2	Latch in 4th data bit
14	SCL_W2to6LH	3.375	2	Latch in 5th data bit
15	SCL_W2to6LH	3.375	2	Latch in 6th data bit
16	SCL_W7LH	4.875	2	Latch in 7th data bit
17	SCL_W8LH	4.25	2	Latch in 8th data bit
17.5	SCL_W8HL	4.125	1.5	Set SDA to low for ACK bit
18.5	SCL_W9HL	3.375	1.5	Release SDA
19	SCL_W1LH	4.875	3	Enable SDA INT, preparing for detection of stop condition. The 3 µs is the time taken from the rising edge to the instruction that clears SDA interrupt flag.
Stop Condition	SDA	12	NA	Stop condition detected. Reset time out time.

5.2 Bit Rate Versus MCU Clock Speed in Writing Process

In the I²C master write process, the timing bottleneck begins at the start condition and continues to the rising edge of the 2nd clock. Figure 5 shows the timing diagram. The data for calculation is based on an 8-MHz MCU clock.

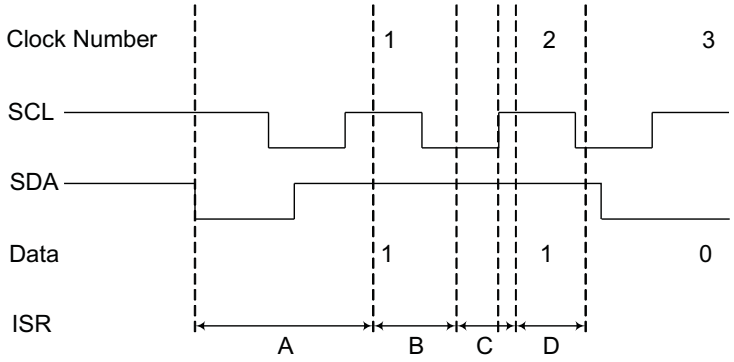


Figure 5. Master Write Timing

Table 4. Master Write Timing

Label	ISR	Time (μs)	Description
A	SDA	9.375	The time spent on this ISR is longer than the sum of hold time of start condition, which is 4 μs (see I²C specification), and the low period of SCL clock.
B	SCL_W1LH	4.875	Immediately after the SDA ISR, run the SCL_1LH, as interrupt flag is set during running SDA ISR.
C	SCL_W1HL	3.125	After completing SCL_W1LH, this routine follows. The instruction of clearing SCL interrupt flag limits the timing of SCL. The rising edge of next clock can only occur after the instruction or at least 2 μs after starting this routine.
D	SCL_W2to6LH	3.375	After SCL_W1HL, this routine follows. The next routine starts after a rising edge.

The time of one cycle for the SCL clock is expressed as follows:

Time = {Total time (SDA + SCL_W1LH) Hold time (start cond) + Rising-edge time limit (SCL_W1HL)} / 1.5 cycles of SCL
= (9.375 + 4.875 4 + 2) / 1.5
= 8.17 ms

At 100 kbps, the time taken to transfer one bit is 10 μs, therefore, when the CPU is running at 8 MHz, the MCU has enough processing power to deal with this bottleneck. The minimum clock speed required for running 100-kbps I²C slave write process is 6.84 MHz.

From Table 3, ISR SCL_W7LH requires more time in clock number 7 than it does in clock 6. This is because the ISR must check the device address and run more instructions in clock 7. In clock 16, only writing in a data bit requires fewer instructions. ISR SCL_W8LH requires more time in clock 8 than it does in clock 17. This is because it must check the bit for read or write and run more instructions in clock 8. In clock 17, it only needs to write in a data bit.

Normally, the hold time for a start condition is longer than 4 μs. If a longer hold time can be set, or the user adds extra instructions to extend the SCL low in this interrupt service routine, this bottleneck is resolved. However, in the firmware of this application, there is no extended SCL low feature. The second timing bottleneck is located between the rising edge of the 8th clock to the rising edge of the 9th clock.

Time = 4.375 + 1.5 + 1 ms timing clearance
= 6.875 ms

The minimum clock speed required for running 100-kbps I²C slave with this setting is 5.22 MHz.

5.3 Number of MIPS Used in Writing Process

In the I²C master write process, the device address is received and followed by a sequence of data. The same interrupt routines are used for receiving device address and data sent from the master.

To calculate the number of MIPS used, the total number of instructions for receiving 8 bits of data and an acknowledge bit is divided by the time period. The number of instructions for start and stop condition can be ignored as it is relatively small compared to the total number since they are used only once in transmission.

The number of MIPS is equal to the number of instruction cycles used from clock 10 to clock 18 divided by 9 clock periods for a 100-kbps transmission.

= 332/90 MIPS

= 3.69 MIPS

Where one instruction cycle equals one clock cycle for the CPU clock. If running at 8 MHz, one clock cycle is 0.125 µs.

5.4 Reading Process

Figure 6 shows the timing for reading 0xAA from a slave MCU with device address 0x62 (seven bits).

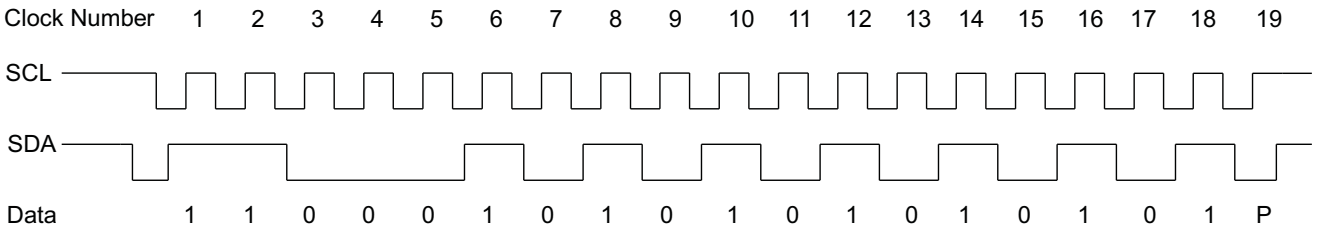


Figure 6. Reading Process Timing Diagram

Note: The data in the tables are measured with an operating clock frequency of 8 MHz. Integer clock numbers indicate the number of rising edges of SCL after start condition; half clock numbers indicate the falling edges of SCL. The first 8 bits in the sequence are the device address sent by master unit of I²C, while second 8 bits are the data (0xAA) read out from the MCU. ISRs starting from clock numbers 1 to 8.5 are used for detection of device address, while clock numbers 9.5 to 18 are the ISRs for reading data.

Table 5. Reading Process Timing

Clock Number	ISR Label	Total Time (µs)	Latch In/Out Data (µs)	Remarks
Start Condition	SDA	9.375	NA	Start condition initialization. Time-out timer starts running.
1	SCL_W1LH	4.875	2	Capture the first bit and prepare for detection of stop condition
1.5	SCL_W1HL	3.125	2	Set rising edge detect for SCL. Disable the SDA signal detect, hence disable detection of stop condition. The 2 µs is the time taken from the falling edge to the instruction that clears the SCL interrupt flag.
2	SCL_W2to6LH	3.375	2	Latch in 2nd data bit
3	SCL_W2to6LH	3.375	2	Latch in 3rd data bit
4	SCL_W2to6LH	3.375	2	Latch in 4th data bit
5	SCL_W2to6LH	3.375	2	Latch in 5th data bit
6	SCL_W2to6LH	3.375	2	Latch in 6th data bit
7	SCL_W7LH	5.75	2	Latch in 7th data bit
8	SCL_W8LH	4.375	2	Latch in 8th data bit
8.5	SCL_W8HL	4.875	1.5	Set SDA to low for ACK bit

Table 5. Reading Process Timing (continued)

Clock Number	ISR Label	Total Time (μs)	Latch In/Out Data (μs)	Remarks
9.5	SCL_R1HL	4.625	3	Output 1
10	SCL_R1LH	3.625	2	Enable SDA INT to detect stop condition. After that, the INT is disabled in SCL_R2to8HL. The 2 μs is the time taken from the falling edge to the instruction that clears the SCL interrupt.
10.5	SCL_R2to8HL	4.375	2.5	Output 0
11.5	SCL_R2to8HL	4.125	2.5	Output 1
12.5	SCL_R2to8HL	4.375	2.5	Output 0
13.5	SCL_R2to8HL	4.125	2.5	Output 1
14.5	SCL_R2to8HL	4.375	2.5	Output 0
15.5	SCL_R2to8HL	4.125	2.5	Output 1
16.5	SCL_R2to8HL	4.375	2.5	Output 0
17.5	SCL_R9HL	3.125	1.5	Release SDA
18	SCL_R9HL	4.25	1.5	Latch in NACK bit
19	SCL_W1LH	4.875	3	Enable SDA INT, preparing fro detection of stop condition. The 3 μs is the time taken from the rising edge to the instruction that clears SDA interrupt flag.
Stop Condition	SDA	12	NA	Stop condition detected. Time-out timer stops running.

5.5 Bit Rate Versus MCU Clock Speed in Reading Process

In the reading process, the timing bottleneck is the same as that in writing process, 8.17 μs, starting from the start condition to the rising edge of the 2nd clock. The minimum clock speed required for running 100-kbps I²C slave read process is 6.84 MHz.

Another bottleneck is at clock 9.5. It takes 3 μs to output a data bit in half cycle. A 1-μs timing clearance is required for rise time of rising edge, in total 4 μs in half cycle. Therefore an 8-μs period of I²C SCL clock is required. A technique of adding extra instructions to extend SCL low can resolve this bottleneck, which is left to the user to design.

The minimum clock frequency required to run a 100-kbps I²C read process with this bottleneck is 6 MHz.

5.6 Number of MIPS Used in Read-Out Process

In the read-out process, after the device address has been received, data is sent out to the master unit. The resources required for the most frequently used ISRs are used to calculate the overall number of MIPS required. These are:

- Data bit output ISR (SCL_R2to8HL)
- ACK detection ISR (SCL_R9HL of ACCRETE)

The others, such as start and stop detection, only contribute to a small part of this calculation and can be ignored. This is especially true for multi-byte reads.

The number of MIPS is equal to number of instructions from clock 9.5 to clock 18.5 (ACK) divided by 9 clock periods for 100-kbps transmission.

= 368/90 MIPS

= 4.09 MIPS

5.7 Clock Setting

From the previous sections, a 6.84-MHz crystal frequency is the minimum required to operate the I²C at a transfer rate of 100 kbps.

When the high-frequency clock is provided by the DO, a nominal frequency 7.42 MHz is suggested. As the minimum and maximum working frequencies are 6.84 MHz and 8 MHz, respectively, there is a ±8% timing tolerance. An operating frequency of 8 MHz gives ±15% timing tolerance for I²C transactions. When using MSP4301xx devices, if the DO frequency is set above 5 MHz, an external resistor is required to connect the RISC pin to V_{CC}. When using the MSP430F2xx devices, the DO can be set to run at a maximum of 16 MHz, and its variation due to temperature drift is limited to only 1%. Be sure to check the device datasheet for V_{CC} requirements for any given operating frequency.

5.8 Running With Other Interrupt Routines

In an MSP430, when processing an interrupt service routine, the global interrupt enable bit (GIE) is cleared to disable other interrupts until the present interrupt routine is finished. For some applications, it may be necessary to nest interrupt routines. In this case, an EINT instruction can be inserted into an interrupt service routine to enable other interrupts. The I²C interrupt service routine can run on top of other interrupt routines using this method.

6 Testing the Software I²C

Figure 7 illustrates how to set up a test environment for the I²C slave software. A more detailed I²C slave circuit diagram is contained in Section 7.

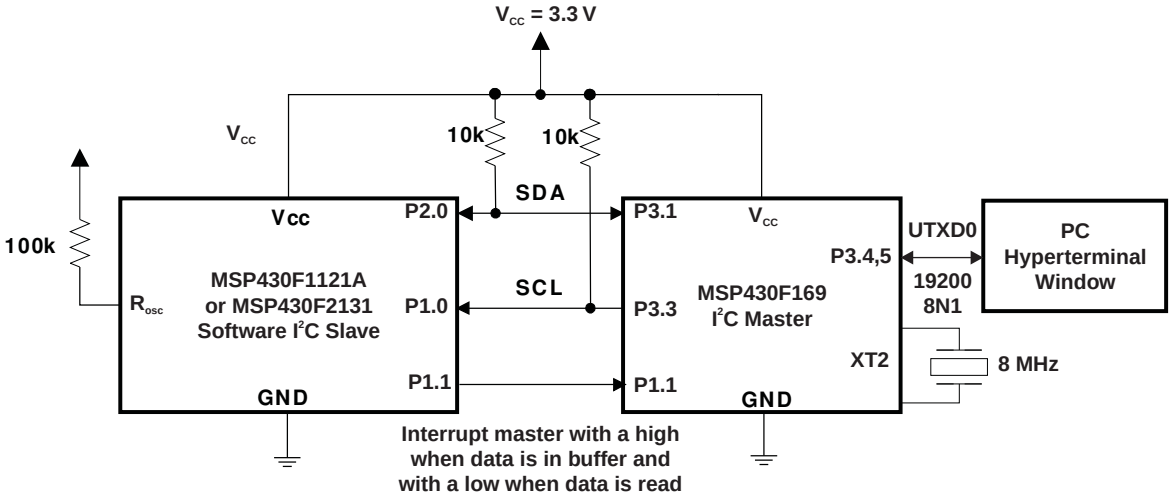


Figure 7. Software I²C System With Hardware I²C Master Block Diagram

The master I²C device is an MSP430F169, which uses its hardware I²C module. The software slave device is an MSP430F1121A on an MSP-FET430P120 target board.

To test the system, the user enters a command on the PC's hyperterminal window. The setting of the RS232 port is 19200-8-N-1. The I²C speed is set to 108 kbps. The master source code can also be downloaded with this application report.

In the master, the following commands have been implemented which can be entered at the hyperterminal window, followed by pressing the <Enter> key:

`clr` `<Enter>` : Clears the screen

`A5` `<Enter>` : Performs a loop-back test with the slave unit using characters: AA 05 AA ...

`rand` `<Enter>` : Performs a loop-back test with the slave unit using characters: 01 02 03 ...

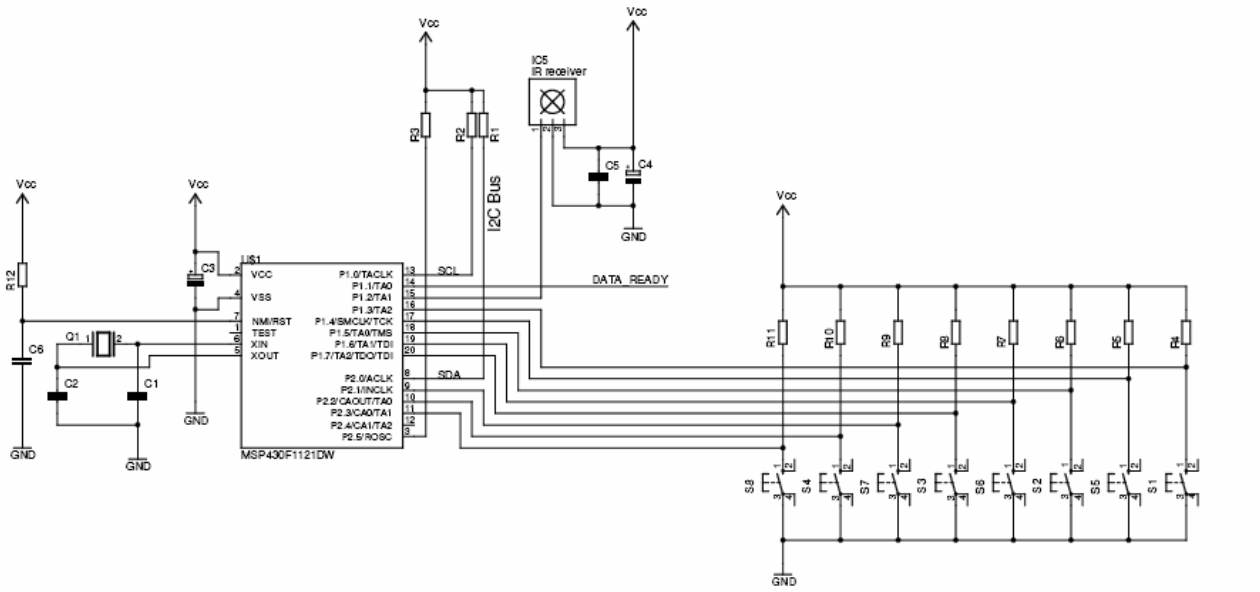
`RX` `<Enter>` : Reads 15 stored characters from slave

`<any key>` `<Enter>`: sends the character to slave and store in its buffer

Note: Before entering a command in hyperterminal, it may be necessary to insert a new line by pressing <Enter> before typing the commands listed above.

This circuit can be used as a front panel with IR receiver in many portable devices or home appliances, such as televisions, etc.

7 Schematics



8 Hardware Implementation

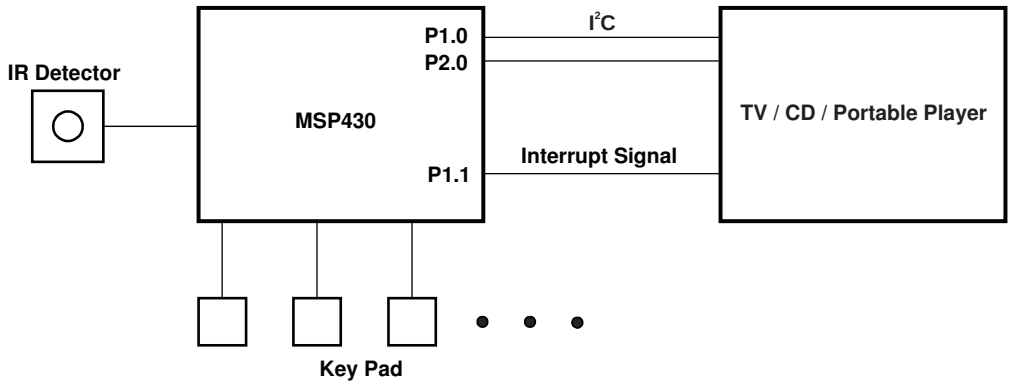


Figure 8. Infrared Receiver With I²C System Block Diagram

The circuit consists of an IR detector, two pullup resistors for I²C, and eight key buttons. When an IR signal is received and decoded, the received data is stored in a buffer. The slave unit, an MSP430F1121A, then generate a high signal on P1.1, indicating that its buffer is filled with new data and is waiting for it to be fetched. The master unit then fetches the data through the I²C interface. In addition, an 8-key pad is implemented. Similarl to the process when an IR signal is received, when a key button is pressed, a corresponding command is stored in the same buffer, and P1.1 is set high. The master unit can fetch the command through I²C.

This circuit can be used as a front panel for a car audio set or in many portable devices like cameras, portable CD players, or home appliances like TVs, mini-combos, etc.

9 References

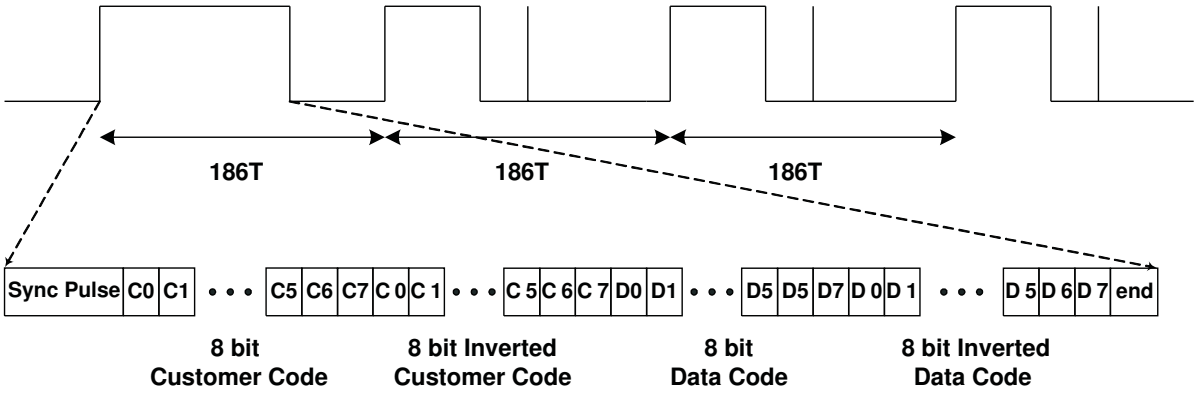
1. I²C Specification
2. *MSP430F1xx User's Guide* (literature number SLAU049)

Appendix A Sample Code

The ISR SCL_W1LH is shown below as an example. When SCL generates the first rising edge, it triggers the ISR_SCL interrupt routine, which, in turn, branches to the appropriate ISR pointed by address pointer R4. In this example, it branches to SCL_W1LH. Hence, each ISR has an overhead of 8 instruction cycles, of which 6 cycles are for the MCU to process interrupt input, and 2 cycles are for branching instructions. In SCL_W1LH, the instruction to latch-in SDA data, taking 4 instruction cycles, is `BIT.B #SDA,&P2IN`. Before that, it has one more instruction, which takes 4 cycles. In this way, the number of instruction cycles taken for SCL_W1LH to reach and operate the latch-in instruction is 8 + 4 + 4 = 16 instruction cycles. The time taken to latch-in data is $16 \times 0.125\ \mu s = 2\ \mu s$. However, completing the whole SCL_W1LH ISR takes more instructions, including RETI. In total, it needs 39 cycles, which equates to 4.875 μs.

```
ISR_SCL
            BR      @R4+
SCL_WRT     DW      SCL_W1LH              ; Bit 1 (first bit rising edge)
            DW      SCL_W1HL              ; Bit 1 (first bit falling edge)
            DW      SCL_W2to6LH           ; Bit 2
.
.
.
.

SCL_W1LH    BIS.B   #SCL,&P1IES           ; FALLING EDGE FOR CLEAR SDA INT
            BIT.B   #SDA,&P2IN            ; Test if data line is high
            BIC.B   #SCL,&P1IFG           ; Reset interrupt flag
            BIC.B   #SDA,&P2IFG           ; Clear flag, FOR STP_CON DETECT
            BIS.B   #SDA,&P2IE            ; ENABLE SDA INT
            RLC.B   IICDATA
            RETI

SCL_W1HL
.
.
.
.
            RETI

SCL_W2to6LH
.
.
.
.

            RETI
.
.
.
.
```

Appendix B Infrared Transmission



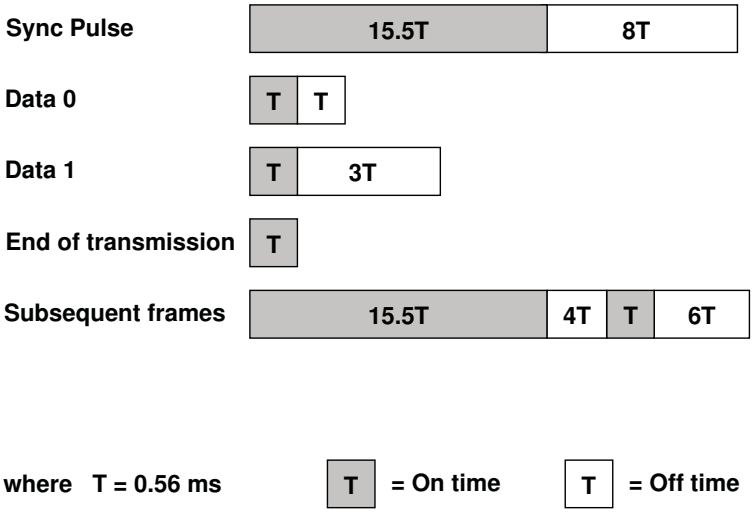


Figure B-1. Infrared Transmission Format

To transmit a 0, a 38-kHz square pulse is turned on for a period of T (T is 0.58 ms) and off for a period of T. To transmitting a 1, the 38-kHz pulse is turned on for a period of T and off for 3T.

The timing to the receiver may vary due to its clock tolerance, propagation delay, etc., in this application, a timing tolerance of 0.3 ms is set.

Appendix C Key Scan Flow Chart

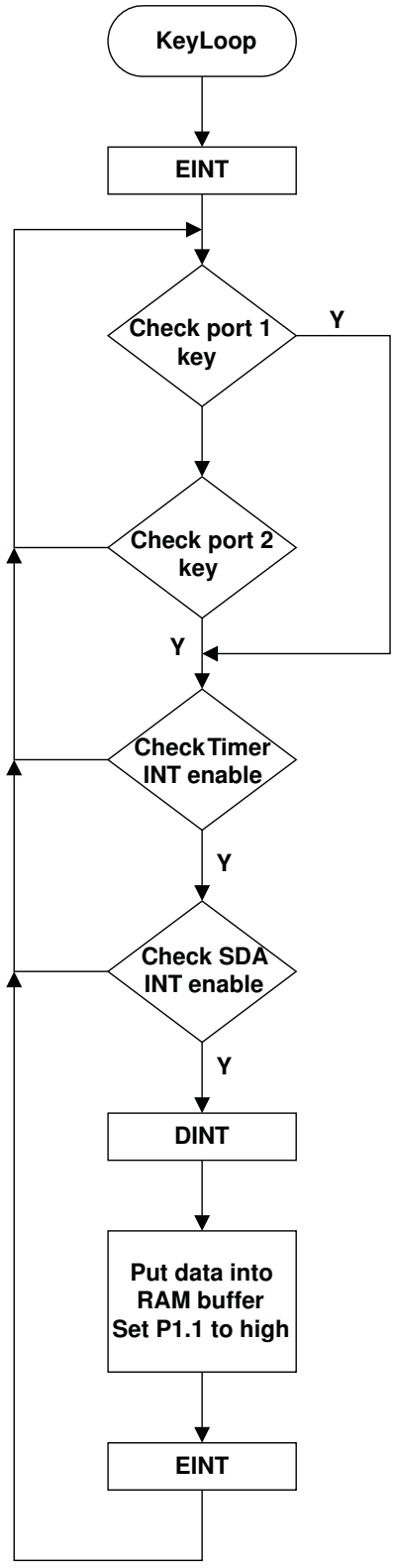


Figure C-1. Flow Chart of Key Scanning

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265