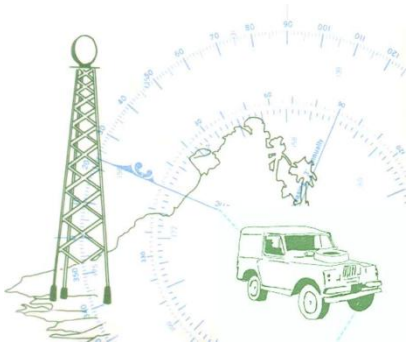
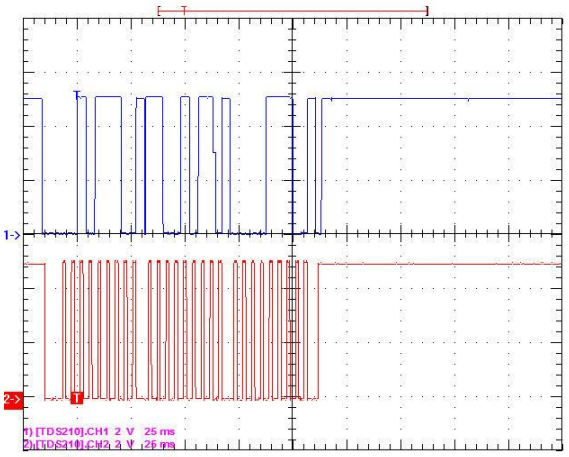
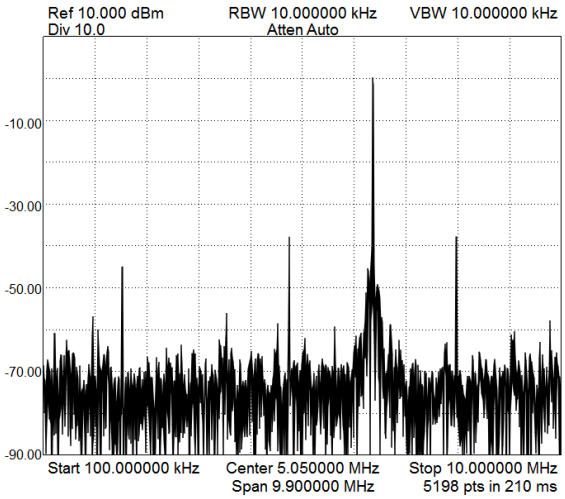


Programming the Linear LTC6904 1KHz/68MHz I2C Oscillator with Stamp/Arduino/Rpi





Jeremy Clark VE3PKC

Copyright Information

Clark Telecommunications
Jeremy Clark
500 Duplex Suite 506
Toronto M4R-1V6, Ontario, Canada
416-488-5382
jclark@clarktelecommunications.com
www.clarktelecommunications.com

Table of Contents

1 - Introduction 1

1.1 - LTC6903/6904 Programmable Oscillator 1
1.2 - LTC6904 Block Diagram 1
1.3 - LTC6904 DAC & OCT Register Settings 3
1.4 - LTC6904 I2C Write Word Protocol 5
1.5 - I2C Specification NXP UM10204 6
1.6 - I2C Bus Programming Approach 7

2 - Parallax Basic Stamp BS2-IC 8

2.1 - BS2 Programming Platform 8
2.2 - BS2 Programming Environment 9
2.3 - BS2 Program Structure 10
2.4 - BS2 Programming Waveforms for Fout = 6.5MHz 11
2.5 - BS2 Memory Map 15

3 - Arduino Uno Rev.3 17

3.1 - Arduino Programming Platform 17
3.2 - Arduino Programming with IDE 1.6.5 20
3.3 - Arduino Programming in C with Atmel Studio 6.2 25

4 - Raspberry Pi 2B 28

4.1 - Raspberry Programming Platform 28
4.2 - Raspberry Programming with Python 30

Appendix A - Instrumentation Setup 32

Appendix B - Stamp Basic BS2-IC Code 34

Appendix C - Stamp Basic BS2-IC Code Subroutine Structure 36

Appendix D - Arduino C Code Atmel Studio 6.2 37

Appendix E - Arduino Uno Rev.3 Setup 40

Appendix F - Raspberry Pi Configuration 46

Appendix G - Raspberry Pi Python Code 48

Appendix H - ScicosLab Code 50

Glossary 51

References 52

1 - Introduction

1.1 - LTC6903/6904 Programmable Oscillator

The LTC6903/6904 (Ref.1) is a compact programmable oscillator. It has many features that make it attractive for various applications such as: Microcontroller Clock Source, Clock Source for a Switched Capacitor Filter or general replacement for a DAC/VCO combination. Listed below are some of the important features. Although it comes in a small package, you can still prototype with it using an MSOP8 adapter board as shown in Figure 1.1. The LTC6903 is programmed using the SPI bus and the LTC6904 is programmed using the I2C bus. This publication is based on the LTC6904 using the I2C bus. Note that it will work both at 3.3 and 5VDC logic levels.

LTC6904 Applications	**LTC6904 Important Features**
Microcontroller Clock	Vcc = 2.7 - 5.5VDC, Icc = 7mA Typical
Clock for Switched Cap Filter (LTC1068)	Frequency Accuracy <= 1.6% (Vcc=+5.5V)
Replacement for DAC/VCO	Spurs <= -30dBc Typical
VCO for HF Amateur Applications	Min.Ext.Comps = 2xPullupRs + 2xBypassC
	Cost Approx $5.90Cdn (Digikey)

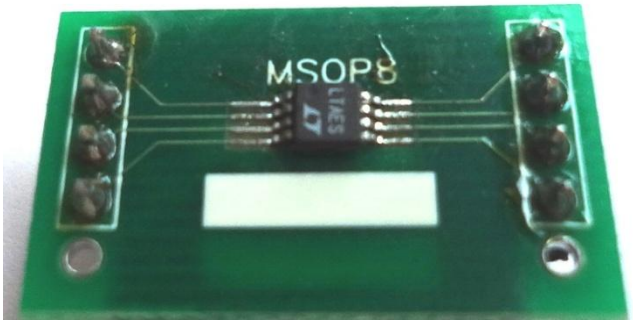


Figure 1.1 LTC6904 Mounted on MSOP8 Adapter Board

1.2 - LTC6904 Block Diagram

Figure 1.2 shows the block diagram and pin out of the LTC6904 (Ref.2). A resistor DAC feeds a square wave VCO in such a way that the frequency ranges from 34MHz to 68MHz. R_DAC varies from R to 2R. Ref.3 Chapter 4 explains the theory and design of synthesizers and VCOs.

$$Freq_{vco} = \frac{68MHz \times R}{R_{DAC}}$$

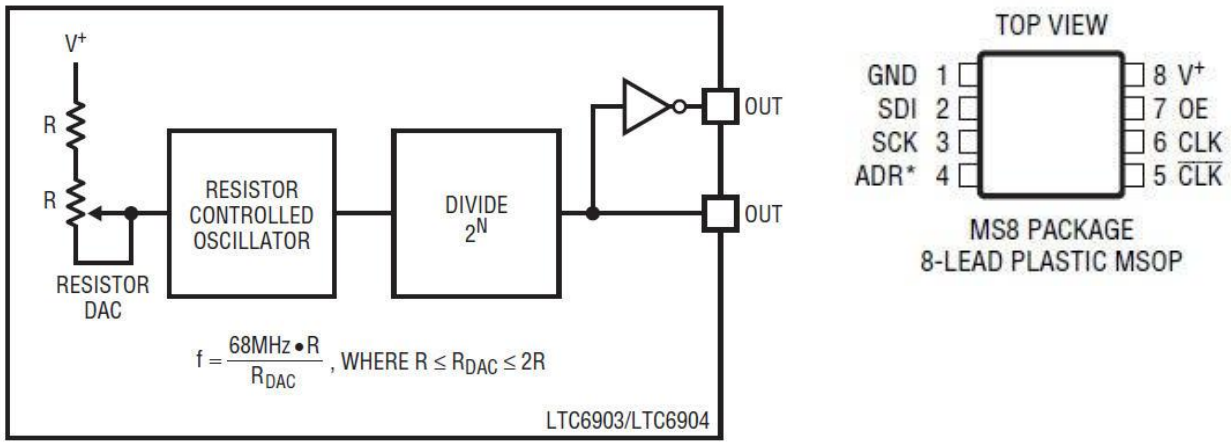


Figure 1.2 Block Diagram & Pin Out of LTC6904

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
OCT3	OCT2	OCT1	OCT0	DAC9	DAC8	DAC7	DAC6	DAC5	DAC4	DAC3	DAC2	DAC1	DAC0	MODE1	MODE0
4-Bit Control Divider 2^N				10-Bit Control DAC										2-Bit Control OUT and $\overline{OUT}$	

Figure 1.3 Digital Control Word

The resistor DAC is assigned 10bits. The output of the VCO is divided down by a programmable divider N which is assigned 4bits. 2 bits are assigned to the control of the outputs. Figure 1.3 shows the 16 bit LTC6904 control word. Bits D0 and D1 (Mode0, Mode1) are for output control. Bits D2-D11 are the 10 bits assigned to the resistor DAC (DAC0-DAC9). Bits D12-D15 are the four bits assigned to the output divider (OCT0 to OCT3).

The frequency output is given by the following formula:

$$F_{out} = 2^{OCT} \times \frac{2078(Hz)}{\left[2 - \frac{DAC}{1024}\right]}$$

DAC = 10 bits (0,1023) and OCT = 4 bits (0,15)

Given the output frequency required, the DAC and OCT words can be determined by:

$$DAC = 2048 - \left[\frac{2078(Hz) \times 2^{(10+OCT)}}{F_{out}} \right]$$ Round to nearest integer

$$OCT = 3.322 \times \log_{10}\left[\frac{F_{out}}{1039} \right]$$ Round down to nearest integer

1.3 - LTC6904 DAC & OCT Register Settings

Figure 1.4 shows ScicosLab program ltc6904.sce used to determine the DAC, OCT, Wdata1 and Wdata2 registers. ScicosLab/Scicos is a C language like mathematics program. An introduction can be found in Ref.4.

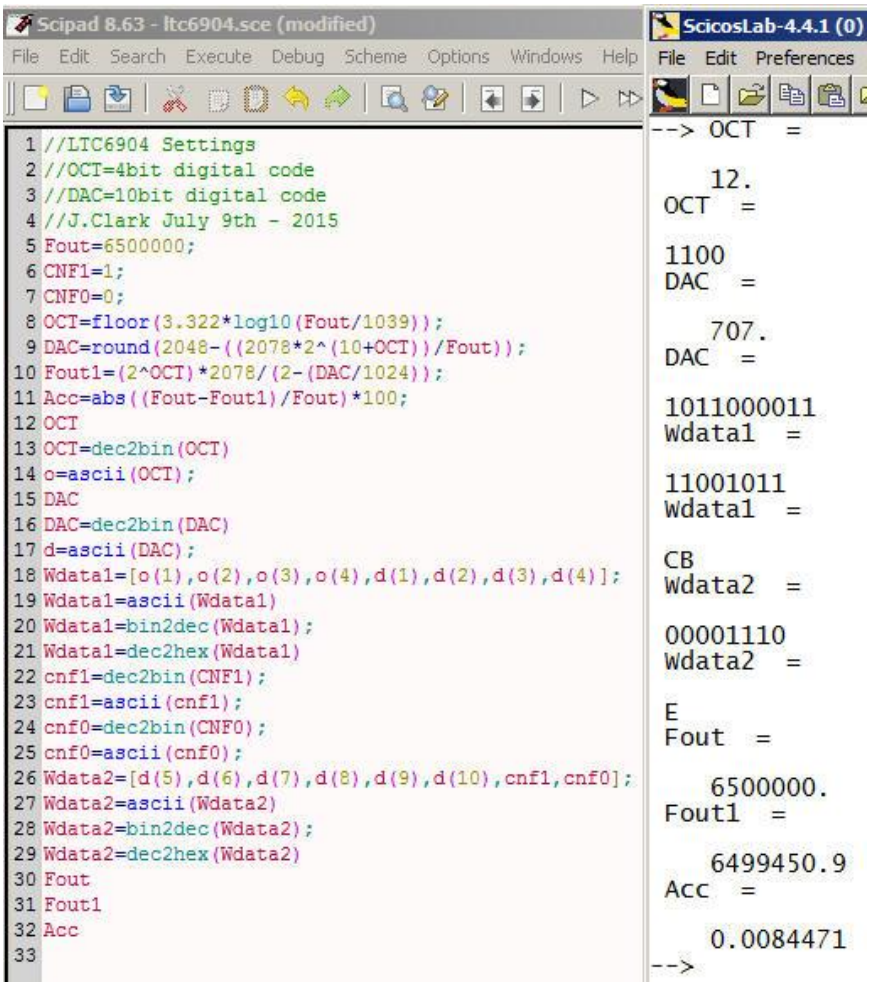


Figure 1.4 DAC & OCT Control Registers ltc6904.sce

Consider a required oscillator output frequency of 6.5MHz. Enter this value in Hz into the script file as Fout = 6500000;

Executing the program, we find that the OCT value = 12(decimal) = 1100(binary) and DAC = 707(decimal) = 1011000011(binary).

Note that the actual output frequency is not quite 6.5MHz, but actually 6.49945MHz. This represents an error of approx. 0.01% in this case. This program can be used to determine the correct DAC and OCT combinations for any required oscillator frequency between 1KHz and 68MHz. This program also determines the 2 bytes of the Write Control Word Wdata1 = $CB and Wdata2 = $0E.

Note that when the LTC6904 first powers up, all register values (all 16bits of the control word) are reset to zero which generates an output frequency of 1039Hz on both outputs. Figure 1.5 shows ScicosLab program ltc6904a.sce that can be used to calculate the output frequency given the values of OCT/DAC (given in decimal format).

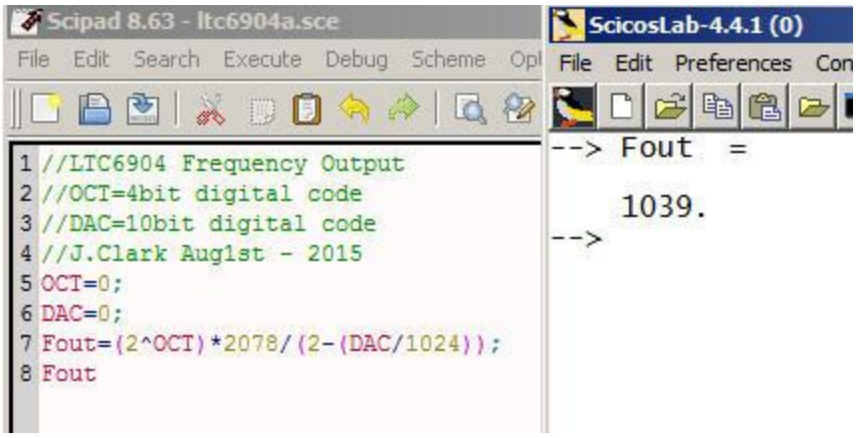


Figure 1.5 Determination Fout Given OCT/DAC ltc6904a.sce

Figure 1.6 shows the output control bits CNF1(Mode1) and CNF0(Mode0). When both bits are set to 1, the outputs CLK and $\overline{CLK}$ are turned off for power saving. When both bits are set to 0, then we have CLK on and $\overline{CLK}$ = CLK shifted by 180deg.

Pin out connections are as follows. Vcc is connected to pin 8, ground to pin 1, SDA to pin 2, SCL to pin 3, $\overline{ADR}$ to pin 4, OE to pin 7, Fout to pin 6 and $\overline{Fout}$ to pin 5. Note the Data Sheet uses SDI and SCK but the I2C International Standard UM10204 uses SDA and SCL which is used in this document.

Table 2. Output Configuration

CNF1	CNF0	CLK	$\overline{CLK}$
0	0	ON	CLK + 180°
0	1	OFF	ON
1	0	ON	OFF
1	1	Powered-Down*	

*Powered-Down: When in this mode, the chip is in a low power state and will require approximately 100µs to recover. This is not the same effect as the OE pin, which is fast, but uses more power supply current.

Figure 1.6 Output Control Bits CNF1(Mode1) & CNF0(Mode0)

1.4 - LTC6904 I2C Write Word Protocol

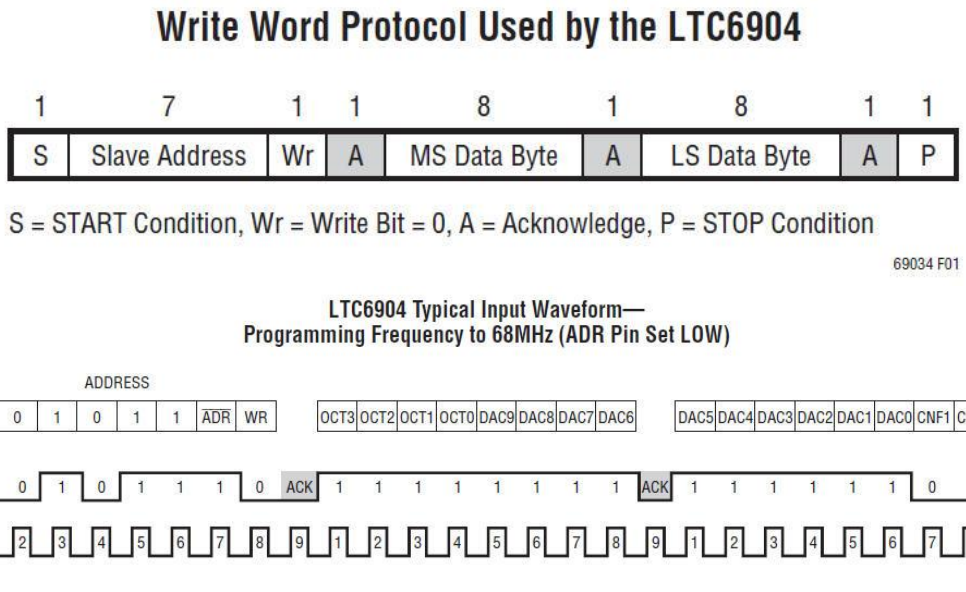


Figure 1.7 I2C Write Waveform

Figure 1.7 shows the I2C write waveform required from the master programmer to write the 16 bits of the control word plus the slave address. The master (host) communicates over a two wire bus. The data signal is denoted as SDA and the clock signal is denoted as SCL. Factory default slave addresses are either 0010111 (ADR=pin4=0) or 0010110 (ADR=pin4=1). A 'START' is first sent, followed by the 7bit address + Wr bit. Note that the LTC6904 can be written to, but not read from (Wr = 0 = write/Wr = 1 = read).

When an 'ACK is received correctly (SDA pulled low by the slave during 9th clock pulse), then the next 8 bits (OCT3--DAC6) are sent. When these bits are 'ACK'd correctly, then the last 8bits (DAC5-CNF0) are sent, 'ACK'd and finally a 'STOP' is sent.

Figure 1.8 shows a typical schematic of a microcontroller master writing to the LTC6904 slave. The SDA and SCL lines are open collector, so they both require pull up resistors of 4.7K to Vcc. Bypass capacitors of 1uF in parallel with 0.1uF are placed directly on the Vcc pin 8 of the LTC6904. ADR pin4 is grounded for the address 0010111 as mentioned above. OE pin7 the output enable pin is connected directly to Vcc. In the example shown, only the CLK output is enabled (CNF1=1/CNF0=0).

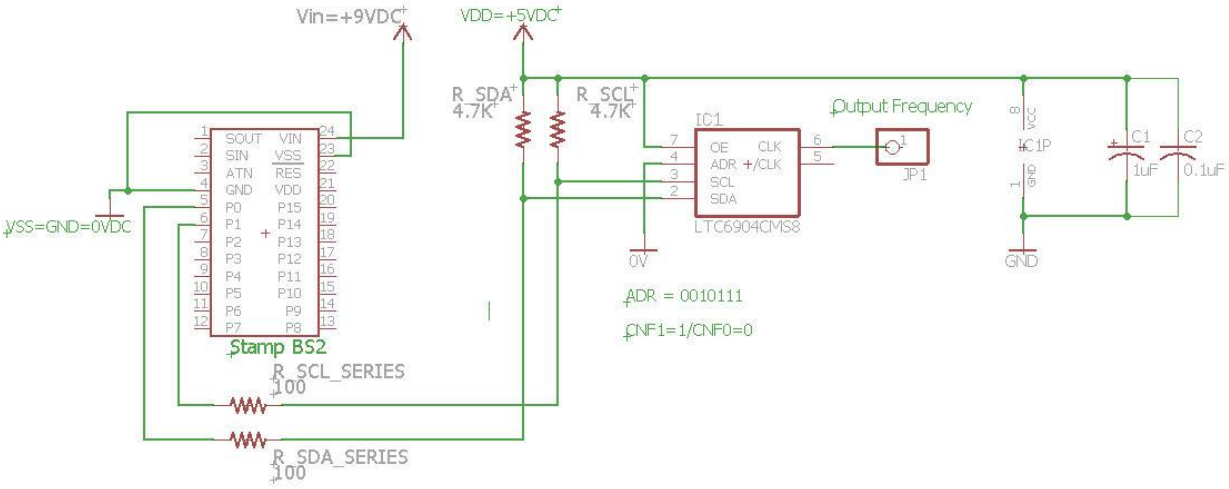


Figure 1.8 Microcontroller Master to LTC6904 Slave Schematic ltc6904_bs2.sch

1.5 - I2C Specification NXP UM10204

Before discussing how to implement the I2C Write Waveform of Figure 1.7, let's review some I2C basics from the design specification from Philips Semiconductors now NXP-UM10204 Reference 5. Figure 1.9 shows the START & STOP conditions sent by the Master, as described in the specification.

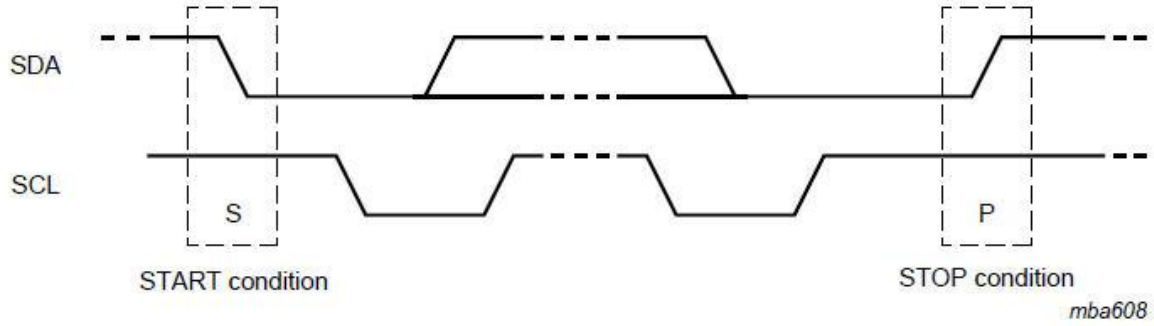


Figure 1.9 I2C START/STOP Waveforms

For the START condition, the SDA goes from high to low while the SCL is high. For the STOP condition, the SDA goes from low to high while SCL is high. Figure 1.10 shows the data transfer on the bus between Master and Slave. A START condition is first established, then 1 byte of data is sent MSB first and an ACK is received from the slave. For an ACK, the Master releases the SDA line and during the 9th clock pulse, the slave pulls the SDA low. If the SDA stays high, this means a NAK is received, or the byte was not received correctly. The SCL line is then held low by the Master to service any interrupts. Following this, the next byte is sent, followed by an ACK from the Slave and finally a STOP is sent.

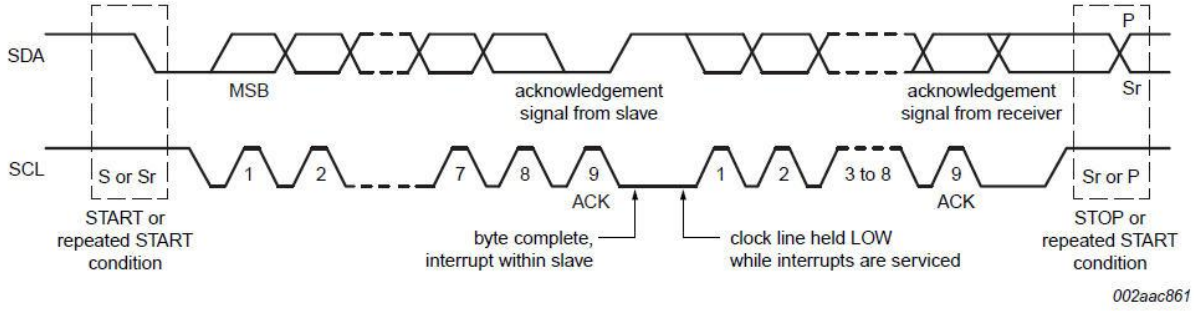


Figure 1.10 I2C Bus Data Transfer

1.6 - I2C Bus Programming Approach

In the next chapters we will try several different approaches to program the LTC6904 using popular devices. The first approach will be to use the Parallax Basic Stamp BS2 (PIC 16F57c Microcontroller). The next approach will be to use the Arduino Uno Rev.3 (Atmel ATmega328 Microcontroller) using the Arduino IDE and also using C with Atmel Studio. Finally we will use the ever popular Raspberry Pi 2B using the Linux environment with Python.

Note that all code is available in printed form in the Appendices as well as a downloadable zip file.