Freescale Semiconductor
Application Note

Document Number: AN3895
Rev. 2, 02/2013

MPR121 Serial Communication

INTRODUCTION

The MPR121 uses an I^{2}C Serial Interface. The I^{2}C protocol implementation and the specifics of communicating with the Touch Sensor Controller are detailed in this application note.

SERIAL-ADDRESSING

The MPR121 operates as a slave that sends and receives data through an I^{2}C two-wire interface. The interface uses a Serial Data Line (SDA) and a Serial Clock Line (SCL) to achieve bidirectional communication between master(s) and slave(s). A master (typically a microcontroller) initiates all data transfers to and from the MPR121, and it generates the SCL clock that synchronizes the data transfer.

The MPR121 SDA line operates as both an input and an open-drain output. A pullup resistor, typically 4.7 kΩ, is required on SDA. The MPR121 SCL line operates only as an input. A pullup resistor, typically 4.7 kΩ, is required on SCL if there are multiple masters on the two-wire interface, or if the master in a single-master system has an open-drain SCL output.

Each transmission consists of a START condition (Figure 1) sent by a master, followed by the MPR121's 7-bit slave address plus R/$\overline{W}$ bit, a register address byte, one or more data bytes, and finally a STOP condition.

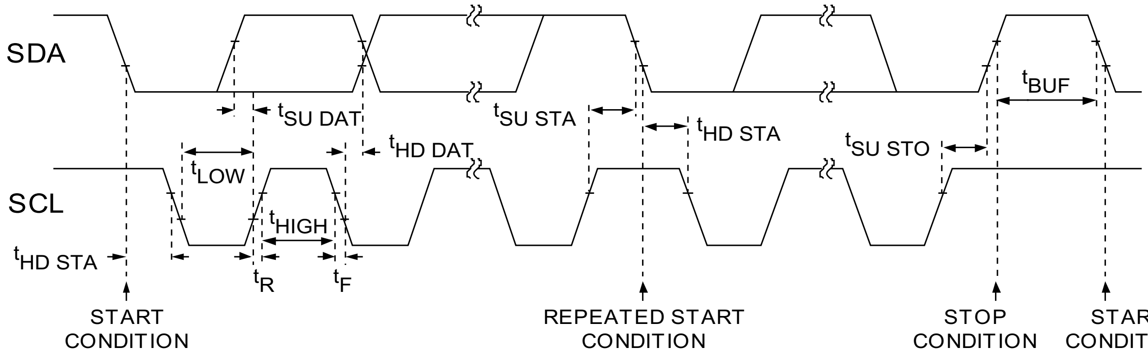

Figure 1. Wire Serial Interface Timing Details

START AND STOP CONDITIONS

Both SCL and SDA remain high when the interface is not busy. A master signals the beginning of a transmission with a START (S) condition by transitioning SDA from high to low while SCL is high. When the master has finished communicating with the slave, it issues a STOP (P) condition by transitioning SDA from low to high while SCL is high. The bus is then free for another transmission.

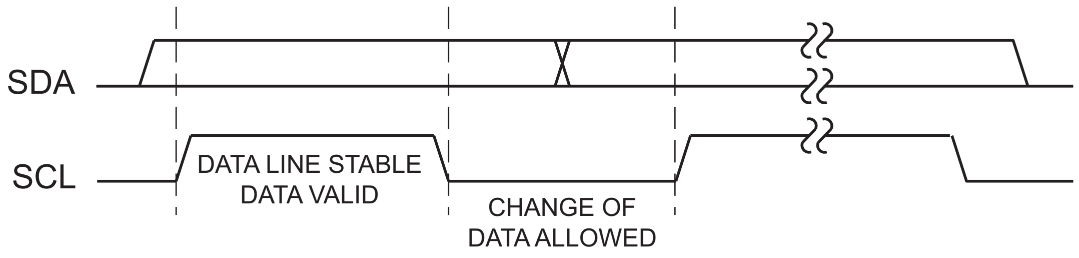

Figure 2. Start and Stop Conditions

BIT TRANSFER

One data bit is transferred during each clock pulse (Figure 3). The data on SDA must remain stable while SCL is high.

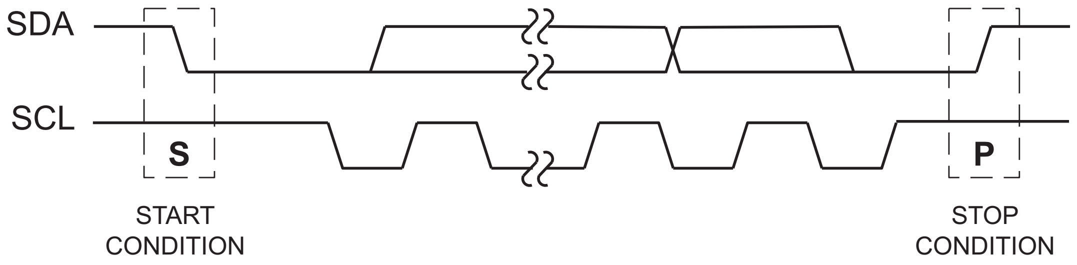

Figure 3. Bit Transfer

ACKNOWLEDGE

The acknowledge bit is a clocked 9th bit (Figure 4) which the recipient uses to handshake receipt of each byte of data. Thus each byte transferred effectively requires nine bits. The master generates the 9th clock pulse, and the recipient pulls down SDA during the acknowledge clock pulse, such that the SDA line is stable low during the high period of the clock pulse. When the master is transmitting to the MPR121, the MPR121 generates the acknowledge bit, since the MPR121 is the recipient. When the MPR121 is transmitting to the master, the master generates the acknowledge bit, since the master is the recipient.

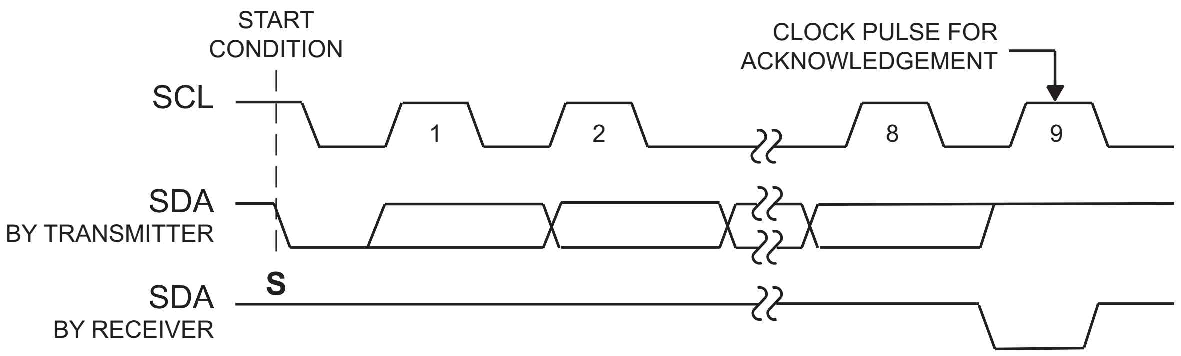

Figure 4. Acknowledge

SLAVE ADDRESS

The MPR121 has selectable slave addresses listed by different ADDR pin connections. This also makes it possible for multiple MPR121 devices to be used together for channel expansions in a single system.

Table 1. MPR121 Slave Address

ADDR Pin Connection	I^2C Address
VSS	0x5A
VDD	0x5B
SDA	0x5C
SCL	0x5D

MESSAGE FORMAT FOR WRITING THE MPR121

A write to the MPR121 comprises the transmission of the MPR121's keyscan slave address with the R/$\overline{W}$ bit set to 0, followed by at least one byte of information. The first byte of information is the command byte. The command byte determines which register of the MPR121 is to be written by the next byte, if received. If a STOP condition is detected after the command byte is received, the MPR121 takes no further action (Figure 5) beyond storing the command byte. Any bytes received after the command byte are data bytes.

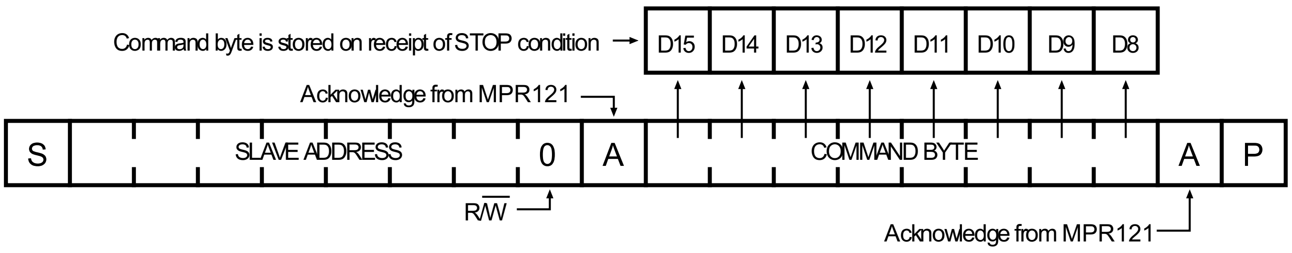

Figure 5. Command Byte Received

Any bytes received after the command byte are data bytes. The first data byte goes into the internal register of the MPR121 selected by the command byte (Figure 6).

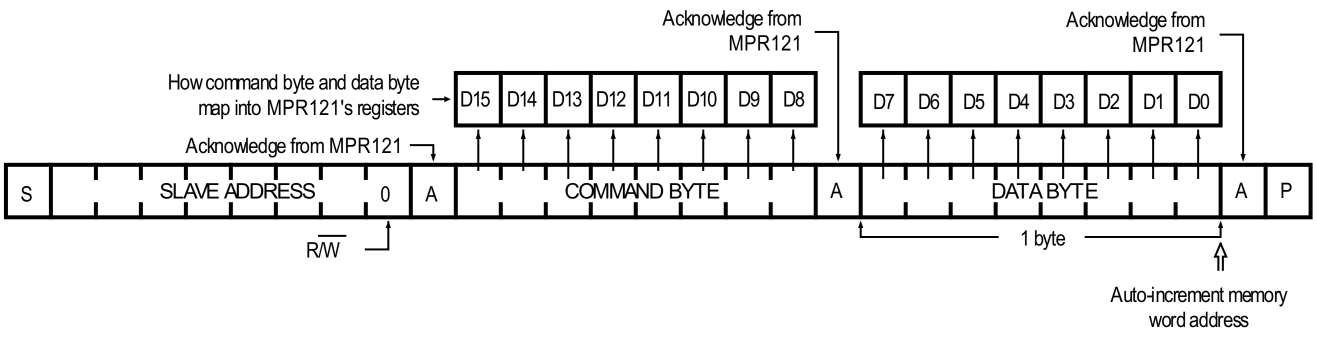

Figure 6. Command and Single Data Byte Received

If multiple data bytes are transmitted before a STOP condition is detected, these bytes are generally stored in subsequent MPR121 internal registers because the command byte address generally auto-increments.

MESSAGE FORMAT FOR READING THE MPR121

MPR121 is read using MPR121's internally stored register address as address pointer, the same way the stored register address is used as address pointer for a write. The pointer generally auto-increments after each data byte is read using the same rules as for a write. Thus, a read is initiated by first configuring MPR121's register address by performing a write (Figure 5) followed by a repeated start. The master can now read 'n' consecutive bytes from MPR121, with first data byte being read from the register addressed by the initialized register address.

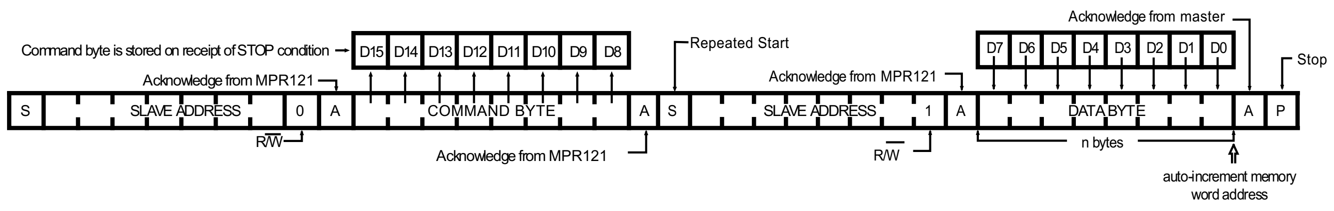

Figure 7. Reading MPR121

OPERATION WITH MULTIPLE MASTER

The application should use repeated starts to address the MPR121 to avoid bus confusion between I^{2}C masters.On a I^{2}C bus, once a master issues a start/repeated start condition, that master owns the bus until a stop condition occurs. If a master that does not own the bus attempts to take control of that bus, then improper addressing may occur. An address may always be rewritten to fix this problem. Follow I^{2}C protocol for multiple master configurations.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN3895
Rev. 2
02/2013